BLUETOOTH TECHNOLOGY

Chandra Shekhar Rajora
Dr. Giresha AS

Bluetooth Technology

Bluetooth Technology

M. Sashilal Singh
G. Ezhilarasan

BOOKS ARCADE
KRISHNA NAGAR, DELHI

Bluetooth Technology

Chandra Shekhar Rajora
Dr. Giresha AS

BOOKS ARCADE

Regd. Office:
F-10/24, East Krishna Nagar, Near Vijay Chowk, Delhi-110051
Ph. No: +91-11-79669196, +91-9899073222
E-mail: info@booksarcade.co.in, booksarcade.pub@gmail.com
Website: www.booksarcade.co.in

Year of Publication 2023

International Standard Book Number-13: 978-81-962948-2-3

CONTENTS

CHAPTER 1

INTRODUCTION TO BLUETOOTH

Chandra Shekhar Rajora, Assistant Professor,
Department of Electronics and Communication, School of Engineering & Technology, Jaipur National University, Jaipur, India
Email Id- chandra.shekhar@jnujaipur.ac.in

Mobile phones, PCs, and other peripherals may send data or audio wirelessly over short distances thanks to Bluetooth, a simple wireless communication technology. The goal of Bluetooth is to eliminate the need for connections while maintaining the security of communications between devices. The term "Bluetooth" is derived from Harald Bluetooth, a Danish monarch who is credited with bringing together several, feuding regional groups in the tenth century. Similar to its designation, Zigbee uses a common communication standard to connect a wide variety of products from several sectors.

The Bluetooth Protocol

The purpose of Bluetooth, which was created in 1994, was to eliminate the need for cords. It makes use of another 2.4GHz frequency as a few other wireless technologies used in homes and businesses, such as WiFi routers and cordless phones. It establishes a personal area network (PAN) or ligand bound, a wireless network with a 10-meter (33-foot) radius that may connect two to eight devices. You may transmit a page to your printer, for instance, avoiding having run the ugly connection thanks to this short-range network.

Compared to Wi-Fi, Bluetooth consumes less energy and is less expensive to deploy. Due to its reduced power, it is also far less likely to encounter or create problems with other smart networks operating in just the same 2.4GHz frequency channel. Wi-Fi often has a greater range and faster transfer rates than Bluetooth. The maximum data rate supported by Bluetooth v3.0 + HS Bluetooth high-speed technology devices is 24 Mbps, which is faster than the WiFi 802.11b standard but slower than the blue tooth or wireless-g standards. Bluetooth rates, however, have accelerated with technological advancement. The Bluetooth 4.0 specification's most distinguishing improvement is its decreased power consumption. Devices employing Bluetooth v4.0 are designed for low charging current and can function with tiny coin-cell batteries, creating new possibilities for wireless technology. For instance, you may keep a Headset v4.0 smartphone always linked to your last Bluetooth gadgets without worrying that putting Bluetooth on would deplete the battery of your cell phone.

Bluetooth radios are built into a lot of mobile gadgets. By installing a Bluetooth adapter, for instance, PCs and other devices without built-in radios may be Bluetooth-enabled. In general, devices advertise their existence to each other as well, and the user picks the Bluetooth connection they communicate to when its name or ID shows on their device. This process of pairing two Peripherals is known as "pairing." There could be a code you need to input to make sure you're going to the right device since as Bluetooth-enabled gadgets grow more prevalent, it becomes necessary to know whenever and from which device you're connected.

The devices involved will determine how this pairing procedure will go. For instance, the procedures required to pair a Bluetooth connection with your automobile may be different than those required to link a Bluetooth device with your iPad.

Bluetooth Capabilities

The disadvantages of Bluetooth are many. The first is that it may be a battery drain for portable wireless devices like cellphones however, this issue is less important now than it once was due to advancements in technology particularly battery technology. Additionally, as with other wireless technologies, obstructions like walls, floors, or ceilings may further diminish the range, which is typically only around 30 feet. According to the network elements, the designers, and perhaps other variables that may all lead to annoyance while trying to connect, the pairing procedure may also be challenging.

The Security of Bluetooth

When used properly, Bluetooth is regarded as a relatively secure wireless technology. Connections are encrypted to avoid casual listening by neighbouring devices. Additionally, Bluetooth devices often switch radio frequencies when coupled, preventing an unintentional invasion. Additionally, devices have a range of options that let users restrict Bluetooth connections. When a Bluetooth device is "trusted," connections are limited to only that one particular device. You may limit the sorts of actions your device is allowed to do when connected to Bluetooth by using service-level security settings.

So there is always some credible threat when using wireless technologies. Bluetooth networking is used in a wide range of harmful exploits that hackers have created. For instance, "bluesnarfing" is the term for when a hacker uses Bluetooth to get allowed access to data on a device, and "blue bugging" is when an adversary gains control of your mobile phone and all of its features. When utilised with safety in mind, Bluetooth doesn't pose a significant security risk for the ordinary user while out in public or not using Mobile data, you may entirely turn it off for optimal protection.

On a microchip, Bluetooth is an always-on, short-range radio connection. It was first created in 1994 as a technique to enable laptop computers to make calls via a mobile phone by Swedish mobile phone manufacturer Ericsson. Since then, a large number of businesses have endorsed Bluetooth, making it the low-power, short-range wireless standard for a variety of gadgets. Industry watchers predict that by 2005, billions of gadgets will have Bluetooth installed. The Bluetooth SIG, a business collaboration, is responsible for publishing Bluetooth specifications to to special interest groups.

Bluetooth's idea is to provide a widely applicable short-range wireless capability. Two Bluetooth devices within 10 metres of one another may share up to 720 Kbps of capacity using the 2.4 GHz band, which is accessible worldwide for unlicensed low-power usage. The applications that Bluetooth is designed to enable are many and include data such as timetables and phone numbers, music, graphics, and even video. For instance, audio devices might include digital MP3 players, cordless and regular phones, home stereos, and headsets. The features that Bluetooth may provide

users include the following make calls using a wireless headset that is distantly linked to a mobile phone. Get rid of the connections that connect computers to keyboards, mice, and printers; Connect wirelessly an MP3 player to a computer to get the music, Create home networks so a couch potato can remotely watch the oven, air conditioner, and kids' Internet use; To activate and deactivate appliances, set an alarm, and keep an eye on activities, dial your house from a distance.

Applications for Bluetooth

Bluetooth is designed to function in a setting with lots of users. A piconet is a compact network that supports up to eight devices. Within the Bluetooth radio's service area, ten of these Pico nets may coexist. Each connection is encrypted and secured against eavesdropping and interference to offer security. Three general short-range wireless communication application areas are supported by Bluetooth:

1. **Data and voice access points:** Bluetooth enables seamless wireless connection of mobile and stationary communications devices, facilitating real-time voice and data exchanges;
2. **Replacement of cables:** Bluetooth does away with the need for a wide variety of, often proprietary cable attachments for connection of almost any kind of communications equipment.

Instant connections are made and maintained even when devices are not in direct line of sight. Each radio has a range of around 10 metres, but with an extra amplifier, that range may be increased to 100 metres; Ad hoc networking: As soon as a Bluetooth radio-equipped device enters range, it may instantly connect to another Bluetooth radio.

Bluetooth Specifications Documents

The Bluetooth standards come in two sections called core and profile, totaling more than 1,500 pages. The Bluetooth protocol architecture's different levels, from the radio interface to link control, are described in depth in the core specifications. Interoperability with related technologies, testing specifications, and a description of multiple Bluetooth timers and their associated values are only a few of the connected subjects that are addressed. The usage of Bluetooth technology to support diverse applications is covered by the profile requirements. Each profile definition describes how to implement a certain use model using the technology described in the core standards.

Which parts of the core standards are required, optional, and not applicable are all described in the profile specification. A profile specification's goal is to establish an interoperability standard so that goods from various suppliers who claim to support a certain use model may cooperate. In general, profile characteristics may be classified as either wireless audio or cable replacement. With the help of the cable replacement profiles, it is simple to logically link objects that are close to one another and to transfer data between them. For instance, two devices may automatically ask each other for a shared profile when they are first in a range of one another. This might then trigger an automated data exchange or result in an alarm being sent to the device's end users. The establishment of close-range voice links is the focus of wireless audio profiles.

The many papers must be sorted through by the Bluetooth developer while keeping an application in mind. The overall access profile and a few key core requirements are covered at the start of the reading list. This profile is one of many that provide a framework for other profiles but do not define independently usable functionality. The Bluetooth baseband architecture, outlined in the core specifications, is used between devices that support one or more profiles, according to the general access profile. The reading list divides into two lines after a basic set of documents, depending on whether the reader is more interested in wireless audio or cable replacement.

Architecture for Protocols

According to its definition, Bluetooth is a layered protocol architecture made up of core protocols, accepted protocols, and protocols for replacing cables and controlling telephones. The following components make up the five-layer stack that the core protocols form:

1. **Radio:** Specifies information about the air interface, such as transmission power, modulation type, frequency, and frequency hopping if applicable.
2. **Baseband:** Addressing, packet format, timing, power management, and connection formation inside a piconet.

The Link Manager Protocol (LMP) is in charge of initializing and maintaining Bluetooth links between devices. This covers baseband packet size negotiation and control, along with security features like authentication and encryption.L2CAP, or the Logical Link Control and Adaptation Protocol, adapts baseband-layer protocols to upper-layer protocols. Both connectionless and connection-oriented services are offered by L2CAP. The construction of a connection between two or more Bluetooth devices is made possible via the Service Discovery Protocol (SDP), which allows for the querying of device data, services, and the properties of the services.

Protocol Architecture According to its definition, Bluetooth has a layered protocol architecture made up of core protocols, accepted protocols, and protocols for replacing cables and controlling telephones. The following components make up the five-layer stack that the core protocols form: Radio - Specifies information about the air interface, such as transmission power, modulation type, frequency, and frequency hopping if applicable. Baseband: Addressing, packet format, timing, power management, and connection formation inside a piconet. The Link Manager Protocol (LMP) is in charge of initializing and maintaining Bluetooth links between devices. This covers baseband packet size negotiation and control, along with security features like authentication and encryption.L2CAP, or the Logical Link Control and Adaptation Protocol, adapts baseband-layer protocols to upper-layer protocols. Both connectionless and connection-oriented services are offered by L2CAP.

The construction of a connection between two or more Bluetooth devices is made possible via the Service Discovery Protocol (SDP), which allows for the querying of device data, services, and the properties of the services. The cable replacement protocol described in the Bluetooth standard is called RFCOMM. To make the change in cable technology as transparent as possible, RFCOMM offers a virtual serial port. One of the most popular forms of communication interfaces used with computers and communication equipment is serial ports. As a result, RFCOMM makes it possible to replace serial port cables with the least amount of equipment modification. Over the Bluetooth

baseband layer, RFCOMM offers binary data transmission and simulates EIA-232 control signals. RS-232, often known as EIA-232, is a serial port interface standard that is extensively used.

Bluetooth provides a standard for controlling phone calls. A bit-oriented protocol called TCS BIN telephony control specification binary specifies the call control signals for the formation of voice and data conversations between Bluetooth devices. It also specifies mobility-management practices for managing teams of Bluetooth TCS devices.

The accepted protocols are included in the overall Bluetooth architecture and are described in specifications published by various standards-making bodies. The Bluetooth approach is to only create protocols that are essential and to utilise current standards if feasible. The adopted protocols are as follows: PPP, or the point-to-point protocol, is a protocol that is used on the Internet to transport IP datagrams over a point-to-point connection; the core protocols of the TCP/IP protocol suite are TCP/UDP/IP;

The Infrared Data Association (IrDA) created OBEX, or the object exchange protocol, as a session-level protocol for the sharing of objects. OBEX offers features comparable to HTTP's, but in a more user-friendly way. A model for representing objects and operations is also provided. Examples of content formats exchanged by OBEX include vCard and calendar, which, respectively, offer the format of a digital business card and entries from a personal calendar and scheduling data;

WAE/WAP Bluetooth's design takes into account both the wireless application environment and the wireless application protocol. The Bluetooth profile papers specify a variety of use models. A used model is essentially a group of protocols that carry out a certain Bluetooth-based application. Every profile outlines the protocols and features that enable a certain use model. The following use models are of the highest priority:

Transfer of directories, files, documents, pictures, and streaming media formats is supported under the file transfer use model. This use paradigm additionally offers the option to view distant devices' directories; Internet bridge: In this use paradigm, a computer is wirelessly linked to a mobile device or cordless modem to enable dial-up networking and faxing. Dial-up networking uses another protocol stack (such PPP over RFCOMM) for data transport while AT instructions are used to operate the mobile phone or modem. The fax programme uses RFCOMM directly for fax transmission;

This use model allows devices connected to a piconet to access a LAN. Once connected, a device performs as if it were hooked directly to the LAN; a device-to-device synchronization of PIM personal information management data, including phone book, calendar, message, and note information, is offered by this approach. An IrDA protocol called IrMC (Ir mobile communications) offers client/server functionality for sending updated PIM data from one device to another; tri-purpose phone - This usage model's phone handsets can function as a cordless phone that connects to a voice base station, an intercom device that connects to other phones, and a cell phone;

Headset - The headset may serve as the audio input and output interface for a distant device. A brand-new technology called Bluetooth low energy was developed to complement standard Bluetooth while also being the most energy-efficient wireless technology that is now conceivable. Bluetooth low energy should be seen as a distinct technology that targets various design objectives and market niches while using the Bluetooth name and using a lot of technology from its father.

Traditional Bluetooth was created to connect mobile phones and computers, bringing together the formerly disparate worlds of computing and communications. However, its most effective use has been as an audio conduit from a mobile phone to a headset worn on or near the ear. More and more use cases were added as the technology developed, including stereo music streaming, phone book downloads to your vehicle from your phone, wireless printing, and file sharing. Since each of these new use cases needed more bandwidth, the Bluetooth ecosystem has gradually introduced newer and faster radios as a result. Beginning with a Basic Rate (BR) and a maximum Physical Layer data rate of 1 megabit per second, Bluetooth was first introduced (Mbps). In Bluetooth version 2.0, Enhanced Data Rate (EDR) was introduced to boost Physical Layer data rates to 3Mbps; in Bluetooth version 3.0, Alternate MAC 1 PHY2 (AMP) was added to leverage IEEE 3 802.11 to give Physical Layer data rates of up to hundreds of megabits per second.

The approach used by Bluetooth low energy is quite different. It has been tuned for very low power consumption rather than merely boosting the data speeds that are accessible. As a result, it's likely that you won't experience high data rates or even want to maintain a connection for many hours or days. This is an intriguing development since the majority of wired and wireless communications technologies are continually accelerating their speeds.

Speeds Almost Always Increase

The realisation that traditional Bluetooth technology cannot meet the low power requirements necessary for devices powered by button-cell batteries has led to this new route. Another factor has to be taken into account, too, to completely comprehend the requirements for low power. In addition, Bluetooth low energy is designed to be used in devices that lack wireless capability today in very large quantities. Being cheap is one way to reach extraordinarily large quantities. Radiofrequency identification (RFID) tags, for instance, may be used in extremely large quantities since they are relatively inexpensive and eventually function by scavenging power from a more costly scanner. It is essential to consider the Bluetooth low-energy system design from the standpoint of low-cost needs. This design's three main components all indicate extremely cheap cost:

ISM Band

It is not a good idea to create or utilise wireless equipment in the 2.4GHz ISM band. Given that water makes up the majority of the human body and that radio energy is easily absorbed by anything, including water, it has poor propagation qualities. The fact that the radio spectrum is accessible internationally and there are no licencing restrictions offsets these very considerable drawbacks. Naturally, the presence of this Free Rent sign indicates that other technologies, such as the majority of Wi-Fi radios, will also utilise this area. However, the absence of a licence does not imply that something is legal. There are still several regulations, mostly about restricting the

range and power output of devices that utilise the spectrum. These restrictions are still more appealing than paying a high price for licenced airwaves, however. As a result, using the ISM band is less expensive.

IP License

Nokia could have submitted the Wibree technology to any existing wireless standards organisation when it was ready to be integrated. It may have brought it to the Wi-Fi Alliance, which standardizes technologies in the same 2.4GHz ISM band, as an example. However, they decided on the Bluetooth Special Interest Group (SIG) because of its stellar reputation and liberal licencing practices. When compared to a technology created in another SIG or organisation that has a FRAND4 policy, these regulations essentially indicate that the costs associated with patent licencing are much lower for a Bluetooth device. Bluetooth offers relatively affordable licencing fees, which considerably lowers the cost per device.

Small Power

The greatest method to create an inexpensive gadget is to use fewer components, such as batteries, in its construction. Bigger batteries need larger battery casings, which raises the price once further. The cost of replacing a battery comprises the potential costs of not having that item accessible as well as the cost of the user needing to buy a new battery. There are extra labour charges to replace this battery if this device is maintained by a third party, maybe because it is a component of a monitored home alarm system.

Consequently, including low power consumption in technological design also lowers expenses. Larger batteries might be used in many gadgets. Although AA batteries may readily power a keyboard or mouse, manufacturers choose to use AAA batteries instead since they lower the device's cost of production and bill of materials. Therefore, button-cell batteries the smallest, least expensive, and most widely accessible form of battery are the basis for the basic low-energy design. This implies that for use cases requiring big data transfers or data streaming, high data speeds or low energy cannot be achieved. The key distinction between Bluetooth's traditional and low-energy versions lies in this one aspect. The next section goes into further detail about this.

Types of Devices

Dual-mode and single-mode devices are two categories of devices that may be constructed using Bluetooth low energy. A Bluetooth device that supports both Bluetooth classic and Bluetooth low energy is known as a dual-mode device. A Bluetooth device that only supports Bluetooth low energy is known as a single-mode device. A third kind of gadget is a Bluetooth classic-only gadget. A dual-mode device can communicate with the billions of Bluetooth devices already in existence since it supports Bluetooth classic. The use of dual modes is novel. They call for new host software, controller hardware, and firmware. As a result, upgrading an existing Bluetooth Classic controller or host to enable low energy is not feasible. However, the majority of dual-mode controllers are straightforward upgrades for Bluetooth classic controllers that already exist. This makes it possible for makers of smartphones, PCs, and other devices to rapidly swap out their current Bluetooth classic controllers with dual-mode controllers.

A Bluetooth low-energy single-mode device cannot communicate with older Bluetooth devices since it does not support Bluetooth classic, but it can still communicate with other single-mode and dual-mode devices. These brand-new single-mode components, which are designed to fit into button-cell battery-powered components, are highly optimised for ultra-low power consumption. Because single-mode Bluetooth low energy does not allow audio for headsets and stereo music or high data speeds for file transfers, single-mode devices will also be unable to be utilised in the majority of use cases for which Bluetooth classic is now employed lists the different device categories that may communicate with one another as well as the Bluetooth radio technology that is used when they link up. Single-mode devices may communicate with one another using little power. Dual-mode devices and single-mode devices may communicate using little energy. Dual-mode devices may communicate with conventional devices or other dual-mode devices via BR/EDR.

Design Objectives

The first thing to consider when examining any technology is how the creators optimised it. The majority of technologies excel at one or two things extremely well while failing miserably at many others. A deeper comprehension of that technology may be attained by figuring out what these one or two items are. This is quite easy using Bluetooth low energy. It was designed to use very little electricity. The Bluetooth SIG has a distinctive organisational structure in that it designs and manages all aspects of the technology, from the Physical Layer through the application. The SIG has perfected the process of developing wireless specifications over more than 10 years that are not only functional at the moment of release but also interoperable, resilient, and of very high quality. It achieves this in a cooperative and open yet commercially driven standards approach.

It was intended to develop the lowest-power short-range wireless technology conceivable when the low-energy work began. To do this, each layer of the design has been tuned to lower the amount of power needed to complete a particular activity. For instance, the radio might consume less power while sending or receiving data thanks to the Physical Layer's relaxation of the radio parameters in comparison to a Bluetooth classic radio. Connections may not even be required thanks to the link layer's optimization for very quick reconnections and effective data broadcasting. The host's protocols are improved to decrease the amount of time needed once a link layer connection has been established before the application data can be transferred. All of this is only achievable if the same set of individuals designs every component of the system at the same time. The original Bluetooth radio's design objectives have not been forgotten.

Global operation

An international wireless band is necessary for global operation. The 2.45GHz spectrum is the only one that can currently be used thanks to low-cost, high-volume production technologies. This is accessible because neither astronomers nor mobile phone companies nor other commercial interests are interested in it. Unfortunately, everyone wants to participate in it since it's free, which leads to congestion. Other wireless bands are also available, such as the 60GHz ISM band, which is not practicable from a low-cost standpoint, or the 800/900MHz bands, which vary in frequency and regulations depending on where you are in the world.

It's noteworthy that the low-cost design objective indicates that the system should be maintained as compact and effective as feasible. Although it could be feasible to include full-mesh networking or scatter net functionality into Bluetooth low energy, the cost would go up since more memory and processing power would be needed to sustain this network. As a result, fascinating research-based networking topologies were sacrificed in favour of system optimization for a cheap cost.

Bluetooth low energy operates in the already extremely congested 2.45GHz frequency. Just looking at standards-based technologies, consists of IEEE 802.11, IEEE 802.11b, IEEE 802.11g, IEEE 802.11n, and IEEE 802.15.4. Bluetooth classic and Bluetooth low energy are also included. In addition, a variety of specialised radios, such as X10 video repeaters, wireless alarms, keyboards, and mice, are also using the band. Other sources of noise in the band include microwave ovens and streetlights. Figure 1.1 discloses the Bluetooth flow with the power management.

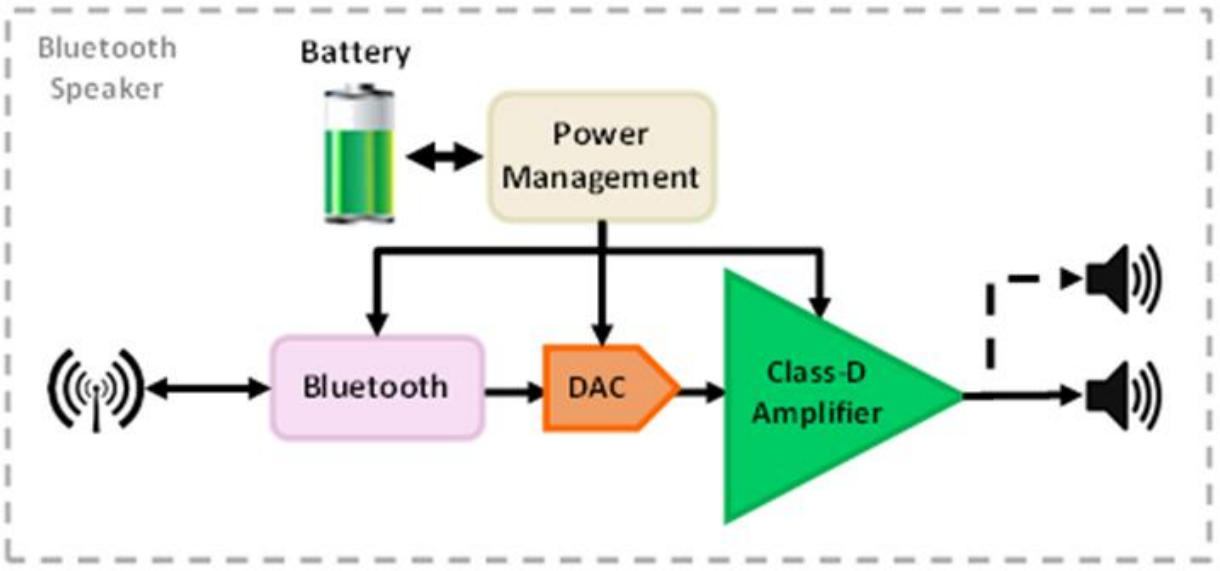

Figure 1.1: Discloses the Bluetooth flow with the power management.

Therefore, excluding the usage of adaptive frequency hopping, which Bluetooth Classic invented, it is very difficult to create a radio that will always function with all potential interference. In addition to immediately identifying sources of interference, adaptive frequency hopping also helps by adaptively avoiding them in the future. Additionally, it bounces back rapidly from the unavoidable lost packets brought by radio interference from other devices. Any wireless system must possess this durability to succeed in the most crowded radio band possible. Robustness also includes the capacity to recognise and correct bit mistakes brought on by ambient noise. Although some employ very lengthy checks, the majority of short-range wireless protocols compromise by adopting a short cyclic redundancy check (CRC). The strength of the checks and the amount of time it takes to provide this information will be compromised in a good design.

There is a little issue with the short range. Keep the transmitted power as low as possible to minimize the amount of energy required to send the signal if you want a low-power system. Similarly to this, you must maintain a receiver sensitivity that is quite high to lower the power needed to detect other devices' radio signals through the background noise. How brief a range

indicates that it is not focused on a cellular base station system in this situation. Bluetooth low energy should be a personal area network due to its short range.

Except for a one- or two-order-of-magnitude reduction in power consumption, the original Bluetooth design aim of low power hasn't altered all that much. While Bluetooth low energy had a design objective of a few years for a sensor detecting the temperature or how far you've walked, Bluetooth classic has a design goal of a few days of standby and a few hours of talk time for a headset. The professionals that work in Bluetooth low energy utilise their vocabulary to explain the features and technologies with the standard, much as many high technology fields. Each unique term is included in this section along with its definition.

Frequency Hopping with Adaptation (AFH) is a technique that uses a limited range of frequencies. As a result, devices may stay away from the frequencies that other non-adaptive technologies such as a Wi-Fi access point are utilising. Architecture the architecture of Bluetooth low energy is another name for its design. The band look up radio band changing frequencies the communication between two devices using several frequencies. Each frequency is employed in a certain order, one frequency at a time. Layer is a component of the system that performs a certain function. The Physical Layer, for instance, deals with radio functioning. The layers above and below each layer in a system are abstracted from one another. The Logical Link Control Layer and the Adaptation Layer do not need to understand every aspect of how the radio works, and neither does the Link Layer need to understand every aspect of how it operates. To maintain the complexity of the system at acceptable levels, this abstraction is crucial. Figure 1.2 discloses the Micro Controller and the Antenna system.

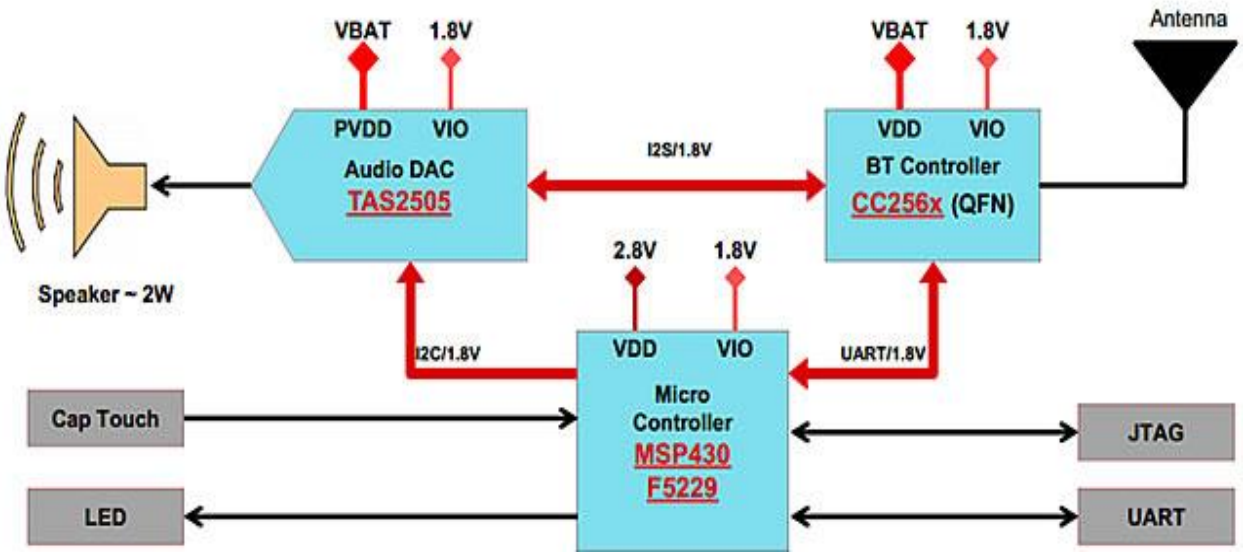

Figure 1.2: Discloses the Micro Controller and the Antenna system.

Master a sophisticated gadget that controls how other piconet devices behave. Piconet, the terms pico and network are combined to form this word. The SI 5 prefix for 10 through 12 is pico. This word is a translation of the Italian word for small, piccolo. A piconet is an extremely tiny network as a result. The activity of all other devices slaves and one or more slaves inside a piconet is

coordinated by a single master device band of radio the frequency or wavelength of radio waves is used to define them. Then, various regulations and purposes are assigned to various radio wavelengths. A radio band is a collection of radio frequencies that have been grouped according to the same regulations.

A slave is a basic tool that collaborates with a master. These gadgets often serve a single function. Wi-Fi An additional wireless technology that is intended to link computers and other highly complicated equipment to the Internet at high data speeds. Understanding Bluetooth Low Energy is understanding how low power consumption in a short-range wireless system can be accomplished. All of the most fundamental design choices are made to enable low power consumption in typical use cases.

Instead of attempting to improve Bluetooth Classic, Bluetooth Low Energy focuses on new market segments that haven't previously used open wireless standards. These market segments concern products that must transmit a few octets of data anywhere from once every second to a few days. These are monitoring and control programmes that carry out operations like switching to a different TV channel, detecting whether windows are open or closed for Smart Home heating programmes, and turning on and off appliances in response to changes in electricity prices.

Button-Cell Batteries

The main goal of Bluetooth low energy design is button-cell batteries. There are a lot of restrictions on how these batteries can be used. Other battery sizes are also available, though the, one depicts a CR2032. Lithium manganese dioxide was used to create the 3-volt battery, as indicated by the "CR" portion of the battery label. The numbers "20" and "32" indicate that the battery is 3.2mm tall and 20mm in diameter, respectively.

A button-cell battery

The maximum energy that may be stored in a battery from one manufacturer is quite comparable to that of any other for such tiny batteries. The stated energy capacity of a common CR2032 is 230mAh at 3 volts. To put this in perspective, this is about equivalent to the energy needed to sustain a human person for slightly over 20 seconds. As a result, by the time you finish reading this, you will have used more energy than a standard CR2032 battery can store to operate a Bluetooth low-energy device for a few years. A gadget will never be able to use all 230mAh of energy, even though it is a very little number. First, the battery's temperature affects the amount of energy that is accessible. There will be less energy available when the battery becomes cooler. Only 80% of the energy that is available at room temperature could be supplied by a button-cell battery.

Second, excessive usage of the battery will dramatically lower the overall amount of energy available. Most button-cell batteries typically have a peak current that shouldn't be exceeded without the battery being harmed. Typically, 15mA is used. The overall amount of energy available would decrease if this high-level current were to be pulled from the battery continuously for a lengthy period. Any effective radio design would thus need to take care of this and provide the battery time to recharge after a significant or prolonged current drain. Finally, it is important to

consider the internal leakage current of the battery itself. Batteries continue to lose charge even when they are not being used by any devices. The leakage current will start to have a major impact on the overall energy budget when the battery is utilised rarely.

Time Is Energy

The idea that time equals energy is another fundamental idea that permeates Bluetooth's low-energy design. The radio uses energy whenever it operates, even if it just checks to see whether it needs to broadcast or receive anything. Therefore, it's critical to cut down on the amount of time needed to do any productive task. It is necessary to optimise several significant and repeated tasks. These include transferring data, connecting to devices, and reliably finding gadgets. The energy consumption is decreased by shortening the time needed for these tasks, extending the battery's life. A device that is searching for other devices and one or more discoverable devices is necessary for robust device discovery, which calls for a minimum of two devices. For a device to be discoverable in Bluetooth low energy, it must broadcast a very brief message three times every few seconds. If the device wishes to hear from other devices, it must listen right away after broadcasting its message. A gadget that is searching for other devices turns on its receiver and scans the air for transmitting objects.

Because three frequencies are needed for robustness, three transmitters are made. Three is selected as a middle ground between reliability and efficiency. If there were just one frequency, like in many other technologies, the system would cease to function as soon as that frequency was blocked. If there were, say, 16 frequencies, the gadget would have to spend so much time broadcasting that it would no longer qualify as low power. It is also quite purposeful which equipment sends and receives. You need to listen for a long time to find a device that is transmitting, which consumes a lot of energy. As a result, you should perform this on a device with a larger energy budget or if there is another solid reason to utilise a low-energy device. The discoverable devices in Bluetooth Low Energy send, while the devices seeking for the other devices receive.

The actual package is rather brief. Three things make short packets beneficial. First, small packets may transport the same amount of data quicker and with less energy by utilising efficient encoding. Second, the need to continuously calibrate the radio in the controller as the packet is broadcast is reduced by limiting the devices to only utilise brief packets. When sending or receiving, radios heat up, affecting the silicon chip's properties and, as a result, the frequency of the broadcasts. The chip doesn't have time to heat up if the packets are kept brief, therefore this energy-intensive process may be disregarded. The necessity for a brief packet also somewhat lowers the chip's peak power consumption. Last but not least, using current from a button-cell battery intermittently rather than continuously over an extended period will provide more energy from it. As a result, utilising many short packets spaced apart enough to give the battery time to recharge is better for the battery than using a single longer packet.

Memory Costs a Lot

Everyone is aware that a computer costs more the more memory it has. However, a computer's memory consumes energy in addition to being expensive. Dynamic refreshing is often needed for

memory, which means that the chip's memory has to be updated periodically. Energy is required for this dynamic refreshing. Therefore, a gadget takes more energy to power the more memory it needs. To lower the amount of memory needed in each layer, Bluetooth low energy as a whole has been devised.

For instance, keeping the Link Layer packets short help since it lowers the amount of memory needed by the radio to broadcast and receive packets. For instance, no packets greater than 23 octets must be handled by the Attribute Protocol Layer. Additionally, no state data has to be stored in between transactions. All of this lowers the amount of RAM required to do relevant tasks. The several protocols that must be active when a gadget may do various tasks is another memory stress. Consider a headset that has hands-free operation, remote control, and battery status reporting, for instance. If each of these use cases had its protocol, the total amount of RAM needed would have to be totalled up. There is just one protocol for Bluetooth low energy. The Attribute Protocol is used to read and write the data necessary to implement a specific use case, as well as for name and service discovery. The overheads of several protocols are greatly minimised by using a single protocol.

Asymmetric Design

Once the architecture of Bluetooth low energy is understood, one of the immediately apparent design principles is the asymmetry that is present at every layer. Because the gadget with the smaller energy source is given less to accomplish, this asymmetric design is crucial. Transmitters and receivers are the two different categories of radios found at the Physical Layer. Both a transmitter and a receiver may be present on a device. A gadget, however, can only use a transmitter or a receiver. This is an asymmetric network if one device only has a transmitter and the other only has a receiver.

The underlying premise behind this asymmetric design is that all other devices will be optimised for the device with the fewest resources. Advertisers, scanners, slaves, and masters are the different categories of devices at the Link Layer. A scanner is a device that accepts packets from an advertiser, which is a device that broadcasts packets. Even when a slave and master are related, the asymmetry is still there. A master must control the piconet time, the adaptive frequency hopping set, encryption, and many other sophisticated operations, but a slave is unable to start any complex procedures. The slave doesn't have to carry out any complicated processing; it merely follows instructions. As a result, the slave remains relatively basic and operates at the lowest feasible power, cost, and memory requirements.

The two categories of devices are referred to as client and server at the Attribute Protocol Layer. The client makes requests for data from the server, which is where the data is stored. The server just follows instructions, much like the slave at the Link Layer. Understanding what data the server has and how to utilise it is the client's difficult task.

Even the low-energy security architecture is asymmetric. The master device receives a key from the slave device for it to remember, and this is how the security architecture works. This bonding information must be recalled by the master; the slave need not remember anything. This implies that although supporting security is easy for a slave device, it is more difficult for a master device.

All of this suggests that the devices with the fewest resources will want to be servers, slaves, and advertising. The asymmetric architecture is advantageous to the objective of very low power consumption on these sorts of devices since they have the lowest potential memory and processing load.

The resources available to the other device type's scanners, masters, and clients are plentiful. Larger batteries, sophisticated user interfaces, and potentially even an electric supply are often seen in these gadgets. Moving the load from the slave to the master, the advertiser to the scanner, and the server to the client is appropriate at this point. By doing this, the most resource-limited devices use less power at the expense of the most resource-rich ones.

Design for Success

So many wireless standards fail right away because even the best radio designs can't handle the congestion that comes with having so many radios in operation at once. Operating in a very crowded environment is one of the things Bluetooth performs best. The Bluetooth Special Interest Group (SIG) hosts Unplug Fest testing occasions three times a year when engineers from various rival firms get together to test their products before they are made available to the public. These incidents demonstrate how Bluetooth continues to function even in situations when hundreds or even thousands of wireless devices are gathered in a single hotel ballroom. Additionally, Bluetooth Low Energy has taken note of this.

Designing for success implies that every individual who travels on a packed bus or train attends a crowded sporting event, or attends a crowded concert should be able to use several low-energy gadgets. As a result, device detection and connectivity should work as anticipated even when thousands of devices are only a few metres away from a particular device. It also implies that there shouldn't be a restriction on how many other devices a particular device may communicate with simultaneously. A device shouldn't have to worry about connecting to more than seven slaves simultaneously, which is the restriction imposed by traditional Bluetooth, to communicate with another device.

Device density was merely one of the metrics employed in the controller's design. Security was another factor. Any well-liked radio system will attract attention from those looking to attempt to compromise its security. This has much more significance when money is involved. Therefore, cutting-edge security and encryption engines are required. Beyond security, the problem of privacy must also be addressed if a person is going to carry several resource-constrained gadgets that are all constantly advertising. The protection of privacy is a top design priority in Bluetooth low energy. Every connection employs a unique signature that is unrelated to either of the two devices' unique identifiers. By only listening to the packets being transferred during connections, it is not feasible to determine who is passing by on the sidewalk. Additionally, it is possible to use a private address while advertising. A private address is a resolvable address that enables a friendly device to resolve it if they know the identity resolving key but prevents enemy devices from tracking or resolving it.

The fact that even a single bit of mistake might become important when the radio is utilised everywhere was another consideration. You don't want a single bit of mistake to cause that sewage

outflow valve to open when you needed to be sure it was still closed, for example, if it prevents your pleasant public park from being flooded by effluent. All packets include a robust cyclic redundancy check (CRC) value that can guard against any 1, 2, 3, 4, 5, and all odd-bit mistakes to prevent this. Additionally, you may start encrypting if you desire extra security. A new message authentication code is then added to the data to confirm that it came from the device you believe it did; this prevents an attacker from responding to messages sent to the sewage outflow valve. And for the very cautious, there is the option to prepare a write into an attribute at the Attribute Protocol Layer and only carry out the execution of that write when the value to be written has been returned and confirmed. This implies that the data is protected by a total of 14 octets of CRC and authentication codes for this one-bit state of the valve. Strong Bluetooth low energy technology.

The idea that everything has a state is one of the fundamental ideas underpinning Bluetooth low energy. Utilizing an attribute server and the Attribute Protocol makes this state visible. The status might be anything, including the current temperature, the device's battery condition, its name, or a description of the location where the temperature is being monitored. The state may also be written; it isn't limited to being a readable state. The temperature at which this room should be heated or cooled may be set using a thermostat or any other mechanism. The state of a state machine may also be exposed if the state can. Explicit state machine characteristics allow for the clear exposure of the device's state. Customers now have the option to disconnect anytime they choose since, upon reconnecting, they can easily ascertain the current situation by just reading it.

Some states are unstable and subject to frequent change. Direct notifications of the state information from the server to the client are a possibility, enabling an effective transmission of state information from the server to the client. Because these alerts do not need the client to query the server, more effective application designs are possible. A device wouldn't have to worry about the battery status at all until the notification arrived since the battery condition could only be reported when anything interesting happened.

It is feasible to build a highly effective client-server architecture using this simple-state-based concept. Additionally, it makes it possible to develop applications using an object-oriented approach to state, complete with reusable data types and service behaviour. The amount of code that a device must contain is reduced as a result, and because less memory is required to store the reduced amount of code, the device consumes less power. Less code also has the important advantage of having fewer defects. Simpler systems are easier to design and more affordable. A simpler system is often more resilient since it has fewer faults. Simpler systems are also simpler to maintain. Less is more, as Robert Browning once stated.

Client-Server Design

One more important design component of the client-server architecture is essential to the development of Bluetooth low energy. The challenge of connecting gadgets to the Internet was taken into account while low energy was being built. It could have been conceivable to install an Internet Protocol (IP) stack on every device with little resources, and then simply expose every device to the Internet.

Sadly, even the most basic IP stacks use more memory and power than is ideal on the most basic of devices. Therefore, it was decided to prohibit the direct routing of any IP packets to slave devices. Smart gateways, on the other hand, provide connectivity between the Internet and very effective low-energy slaves.

The purely client-server design allows for this interaction. A server is only a data storage facility; it is unconcerned with the identity of its users. A client may connect to the server directly or indirectly via a gateway on the opposite side of the world's Internet.

People may use this to keep an eye on and manage their homes while they are away. It would also be possible to check that all the windows are locked on the way to the beach, record your favourite television show while you're lounging in the sand, and then turn the heat back up while flying home because low energy will be used for everything from security alarms to set-top boxes and heating systems [26]–[29].

Sports and fitness equipment that can link to gateways may also rapidly update the connected websites with the data they have gathered, even before the exerciser has had a chance to finish her water. Additionally, it would let elderly people be watched after, allowing them to remain in their own homes secure in the knowledge that assistance is there should they need it.

The client-server gateway approach also makes it possible to employ complete Internet security from the client to the gateway and lets the gateway execute access control, firewall, and client authorisation before giving access to resources outside of the gateway. These are tried-and-true innovations seen in several homes and companies today.

Modular Architecture

Future-proofing the building is a fundamental idea that is often disregarded. Most wireless standards are developed in a hurry to spread the technology as rapidly as possible, with little thought given to how it will work in 10 or 20 years.

This raises issues because subpar design choices made under the pressure of "speed to market" undermine the platform's long-term sustainability. To address this issue and assure a future-proof design, the Bluetooth SIG established a unique architectural working group specifically for the Generic Attribute Profile-based architecture.

The modular service architecture, which is built around the generic attribute profile, is the group's primary output.

This makes it possible to package atomically encapsulate chunks of behaviour into a single service and expose them on a device in a uniform manner. In this context, encapsulate refers to the ability to be detached from other services and wrapped up on its own. An atomic service is one that only performs one action.

If a battery contains a temperature sensor of its own, the battery service may relate to a temperature service; this temperature service may then be used for a home thermometer, a freezer temperature sensor, or a vehicle engine coolant temperature sensor.

The fact that the services offered on a device do not necessarily have to be explicitly tied to a certain profile is an intriguing side consequence of this design. There won't be much of a connection required since profiles will just need a certain set of services on a specific device. This implies that if a separate profile can be made to utilise a different set of services on a device, it will be able to mix the current services without any issues. This is true even if the profile was created after a device's services were developed and put into use.

With its very adaptable and modular framework, ecosystems may develop over time. For instance, smart metres might be installed in houses to provide consumption and pricing information in the present and the future. Later, smart appliances that can be remotely switched on and off may be introduced; employing the gateway model, might be done from outside the house. Even later, a smart energy broker may be installed to save the homeowners money by scheduling energy consumption that takes into account the price information from the meter and the information from smart appliances [30]–[33].

A billion is a small number

Any new technology has a significant barrier when attempting to gain momentum in the market. Technology must be affordable to be successful. You need volume to be inexpensive. You have to be successful to have volume. Since mobile phones are now the most popular consumer electronics item, any technology that is incorporated into them will be a success. A classic example of this is Bluetooth. The attach rate of Bluetooth in mobile devices is expanded upon by Bluetooth's low energy, creating an immediate market.

As mobile phone makers upgrade their platforms to enable Bluetooth low energy, the technology has the potential to have more than one billion devices in use within the first couple of years. The intriguing aspect of this is that it expands the market for phone accessories significantly. And because adding Bluetooth low energy to an existing Bluetooth system is so inexpensive, it is not just phones that can have it quickly integrated; it is also likely that computers, televisions, and other devices with Bluetooth classic will be updated to include Bluetooth low energy.

Lack of connection Model

The main focus of Bluetooth classic was cable substitution, including cables for file transfers, mice, and headsets. This suggests an architecture in which the expense of establishing a connection is less significant since the link will be maintained for a short period of a few minutes, hours, or even days. The connection's first odd-second latency is not very significant. All of this is altered by Bluetooth low energy.

The fundamental idea behind low energy is that connections are fleeting. You rapidly establish a connection, do the necessary tasks, and then disconnect when you need to do anything or check something. The radio would only need to be active for less than one second each day for a gadget that only alerts certain status information once every five minutes.

This indicates that the radio is off 99.999 percent of the time or 100 percent of the time to four significant numbers. Power usage will significantly rise with any delay in each connection setup. In less than three milliseconds, Bluetooth low energy may establish a connection, deliver data, and

then gracefully break it. As a result, many devices that previously couldn't afford to add wireless technology due to the expense of energy needs but had some status information now have the option to do so thanks to Bluetooth's low energy. A button, for example, may be activated even without a battery by employing potential scavenged power.

CHAPTER 2

BLUETOOTH CONNECTION AND HISTORY

Lokesh Lodha, Associate Professor,
Department of Electronics and Communication, School of Engineering & Technology, Jaipur National University, Jaipur, India
Email Id- lokesh.lodha@jnujaipur.ac.in

A renaissance in wireless connection has been made possible by Wireless connectivity. The various Bluetooth protocols use the very little right to transfer voice or data between two devices. It is created by the Beacon Special Industry Group, an association of businesses from the telephony, computer, and electrical equipment industries. Bluetooth uses a 2.4 GHz wireless technology to enable data transfer between two devices separated by a specific distance. Without using wires, we may utilise them to transmit or receive any kind of information, including music, data, and movies. Its development began in 1994 when the Ericsson business started looking into novel ways to deliver files through telephones to reduce costs and power use. It is designated after the Norse monarch Harald Blatand, sometimes known as Harold Bluetooth because of his propensity for clear communication. We often hear the phrase "Bluetooth." The fact that it aids in connecting various devices is well known. More than that, Bluetooth is often overlooked. However, the topic of Bluetooth is rather fascinating. Then it is likely that you have used Bluetooth before. This is a brief wireless connection that occurs between two electronic devices. In many cases, this connection is made between a mobile device and a set of earphones or between a workstation and a gaming adapter or mouse. Information is sent through various devices using Bluetooth. This very same frequency as a WIFI connection, 2.4 GHz, is used for radio waves used for communicating.

Headsets

The service's developers had the concept of developing a short-link framework for signal transmission as early as 1989. In 1999, the 1.0 version was released. Even though the early versions of Wireless headphones were known to contain faults, they still formed a crucial pillar of the future of the technology [4]–[6]. The technology's first iteration controlled communication at a frequency of approximately 721 bit/s. The maximum speed was raised to around 3 Mbit/s in the version released in 2004. Subsequent versions have gotten faster. Because once two units are linked, communication is created so that one unit may view the system in the other, and however there are security hazards. Therefore, it's crucial to confirm which device the mobile or computer is linked to, which is often as simple as looking up the division's name.

Later, in 1999, a dedicated organisation called the Special Interest Subgroup was established to improve Bluetooth technology (SIG). Leading firms in the transportation and telecommunications sectors, including Nokia, Intel, Ericsson, Toshiba, and IBM, as well as later participants Microsoft and Motorola, were brought together via this alliance. Initially, this technique was most effective when used with mobile phones since it offers a convenient method for exchanging data without

charge. These days, it is widely used and may be found in a variety of electrical products, including TVs, Bluetooth speakers and headphones, GPS navigators, and even the newest home appliances.

Range of connection and transmission speed

When Bluetooth first came out, it could send data transfer rates of 720 kbps, which was fantastic at the time but sounds laughable now. The various Bluetooth varieties now available may reach a value in addition to 24Mbs after many generations. Another area that has been improved is the connection range. They can now operate at distances of more than 100 metres after initially working at less than one meter. The majority of cutting-edge technology is based on certain paradigms, and Bluetooth low energy is no exception. Client-server architecture and service-oriented architecture are the two primary architectural paradigms used by Bluetooth low energy. Clients may communicate with servers across a network and get replies from them using the client-server architectural paradigm. The Internet, possibly the most successful networking technology ever deployed, is based on this concept. For instance, when you enter a URL into a web browser, the address is initially sent to a DNS server. This server replies with the server's IP address that was given that name. The client then makes an HTTP request to that server, and after it has established a connection, it asks the server to get the resource specified in the request. The relevant resource, often a text file containing markup (HTML) data on how to display the information, is then returned by the server. Figure 2.1 discloses the Block Diagram of the Micro USB port.

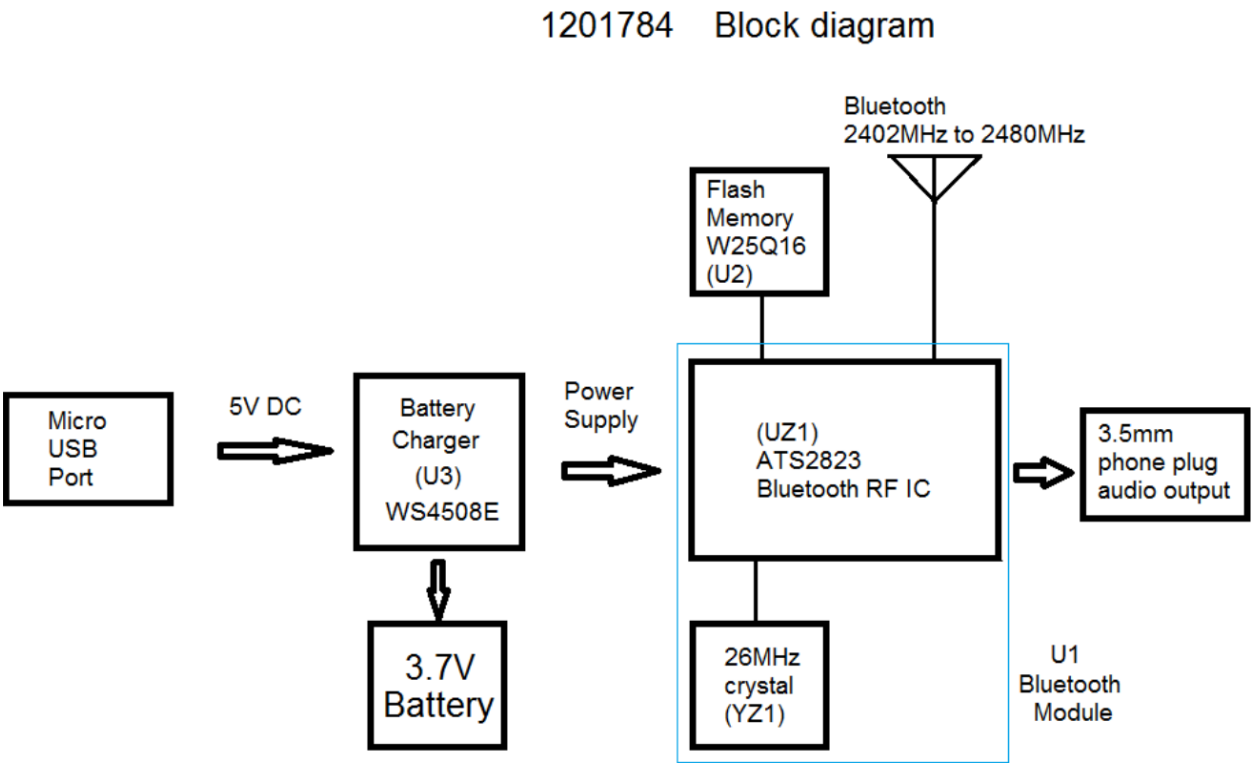

Figure 2.1: Discloses the Block Diagram of the Micro USB port.

This file may also include extra URLs that the client may use to obtain additional resources, such as images or other sites. The phrases "web page" and "web server" refer to the idea that HTML pages are connected to the web because of these extra linkages. There is a major difference between what a server and a client perform. The information is on the server, usually in an organized manner. The server exists primarily for this data. The weather in Kona, Hawaii right now, the departure time of the next train from Seoul to the airport, or even simply some idle conversation amongst friends, are all examples of this data. On the other side, the client is data-free. Just requests to servers are sent. Once it has received the responses from a server, it may carry out the duty it was given to perform, such as showing the user information or alerting them when someone they know has tweeted or put something on a wall.

This clearly defined division between the client and the server is the major advantage of the client-server design. When the various system components are on distinct devices, this divide is required. The explicit link between these two system components may be established by designating one of these components as a server and one as a client. This architecture's key advantage is its ability to scale. The only thing a client has to know to access a resource is the URL. Numerous customers may be present. Millions of requests each day from millions of users are made to certain websites on the Internet. The server just answers each request as it comes in without really caring what or where these clients are. Additionally, this server design is scaleable. A single computer may experience overload and begin to malfunction if it were to handle millions of queries every day. The answer is to set up several similar servers that can all access the same data. Giving customers various IP addresses for a single name helps with this by distributing the load across the servers equally. It is referred to as load balancing.

Service-Oriented Architecture

The service-oriented paradigm is another abstraction on top of the client-server architecture. This paradigm divides the data on a server into several services. These services may be found, utilised, and interacted with using established semantics. This indicates that the services have a set behaviour that, under identical circumstances, will always yield the same outcome. The most widely used Internet technologies, including SOAP, REST, COBRA, RPC, Web Services, and others, are built on this paradigm. This may be shown by using an actual example from the outside world. Let's say you have a shipment that has to be rapidly transported to another business. Call a courier service, scheduling a pickup of the box, and then pay for the service are likely the first steps you do. The idea that you know what will happen is crucial. The courier service will adhere to a predetermined set of characteristics. The courier service will always deliver the identical item, on schedule, to its destination if it is supplied on any given day. This service's established behaviour and well-known semantics lead to predictable outcomes.

The fact that you are communicating with two separate individuals at the courier service the person who answered the phone and received your order and the delivery driver who picked up your package is an intriguing aspect of this. Unknowingly, you also employed a third party to handle the financial transaction. Each of these individuals offers a supplementary service, which when combined yields the courier company's main line of business. Many different kinds of businesses might utilise some of these sub-services interchangeably since they are generic. In every business,

financial transactions are processed in mostly the same manner. Taxi firms might potentially serve the same purpose of receiving phone calls for picking things up at one area and dropping it off at another. Everything has to follow a set of guidelines and conventions that are described in the following sections for everything to function.

Formal Contract

A service must adhere to a formal definition of its exposed functionality and behaviour to be recognised as a service. For instance, the courier service driver drives the corporate car, wears the company uniform, and gives the clients a friendly greeting. Additionally, he will deliver the products undamaged and drive between destinations swiftly and securely. The agreement between the client and the courier service will be broken if any of these regulations are broken. Therefore, the majority of courier services also demand that clients accept this legal contract before picking up their products.

The ease with which one instance of a service may be replaced by another instance of the service is a side consequence of having a written contract. Only if the two instances of the service disclose identical functionality and behaviour is this feasible. For instance, it should be simple to identify a substitute who is familiar with the same accounting principles if the company's finance person quit. These official agreements are documented in service standards for Bluetooth low energy, which the Bluetooth SIG has officially accepted. To guarantee that an implementation's behaviour is correct, these standards also provide test specifications.

Loose Coupling

Each component of the system is intended to be created as a distinct object with no side effects in object-oriented software. Then, those component-to-component interactions that do take place may be formally specified and put to test. Each service implementation may be modified without fear of introducing or losing unanticipated side effects by minimising dependencies. The formal contract and its execution should be kept apart, to follow this argument to its logical conclusion. As long as the formal contract is upheld and not violated, this then permits the implementation to be altered at any time.

For instance, it is possible to increase the number of drivers working for the courier service, switching from a setup where a single driver handled everything to one where multiple drivers would pick up packages from a specific area of the city, transport them to a central warehouse, and then send them back out, perhaps with a different driver. The service is the same from the customer's perspective packages are gathered and delivered as needed but the execution is different. And all of this was accomplished without altering the order-taking or finance services.

Abstraction

Because of the ramifications, if this design principle is broken, service abstraction is a crucial one. A client may begin to utilise a service in a manner that limits how the service can grow if there were no abstractions and they were fully aware of how it was designed. It is well known that having more information is beneficial. The better service is in the context of a service-oriented architecture, however, the less a customer knows about how it is delivered. If the client has too

much information, they may prevent the service from being reused or redesigned since they are implicitly associated with the particular implementation. The client might malfunction if the service implementation changes. Only the bare minimum of the state must be revealed by the service to guarantee compliance with this requirement. Additionally, just the service behaviour's outward manifestation should be defined.

Reusability

Reusability is a design objective that has long been one of the promises of object-oriented techniques. Reusability, on the other hand, is the capacity of a service to be created such that it might be used for a variety of applications. Without thorough consideration, it is always simpler to create a service that does only one task. Services may be created with proper design to be independent of the actual technique employed. As a result, the service may be rapidly and readily utilised in another application. In response to this problem, the Bluetooth SIG established a working group with just one task: to search for generic functionality and then abstract the requirements to allow for considerable reuse.

Lack of state

The servers cannot save any client state information to scale services for multiple clients. A service might be created that keeps track of what the customer has told it, saving the customer from having to repeat everything when making future requests. The issue with this method is that it uses a lot of memory and depends on the shared state data being up to date on both the client and server. This results in the server being dependent on the client's proper operation, which is a poor assumption. Therefore, the statelessness design objective eliminates all states from the client-server interaction. While some state data will still be kept in the service, it will always be server-specific rather than client-specific. This implies that regardless of whose client submitted the request, any client may afterwards make any request at any time, and the server will always reply to the same request in precisely the same manner.

Compos ability

All of the aforementioned objectives suggest that services should be created to be both minimal and highly straightforward. But things are seldom that easy in the real world. Services in the real world are complicated. A service-oriented architecture promotes the aggregation of smaller services to allow higher service interfaces to reconcile this seeming contradiction. All of the aforementioned design objectives for services support their ability to be merged. For instance, it was shown that the courier service was made up of three distinct services. These may be integrated into a courier service as long as each of the constituent services adhered to the specified objectives. An executive vehicle service or a taxi firm might be created by combining some of these services. The actual delivery vehicle a family car, a delivery van, or an elite limo might change, but the services are fundamentally the same.

Independence

Services need to be dependable to be aggregated and reused. A service that depends on other system elements to carry out received requests won't be as dependable as one that exercises total

control over all it does. A self-sufficient service can operate independently and do its work no matter what is going on around it. There is virtually no problem in reusing these services in different applications. A non-autonomous service will likely need to bring in a lot of extra support services, which might infringe on the rights of other services. For instance, courier firm drivers work independently, collecting and delivering products as instructed. Even if there is a serious problem in the workplace, they will continue to operate independently.

Discoverability

Finally, services must be discoverable to be utilised. Although at first look it may seem simple, service discoverability is crucial for ad hoc networking. Without service discovery, all services would have to be statically designed, which is a difficult, time-consuming, and error-prone operation. These often use a different protocol than the one used to communicate with the service. For instance, instead of calling random phone numbers and hoping one of them is a courier business, someone may have utilised a phone book or the Internet to discover the courier firm. A different method is used by Bluetooth low energy, which employs a single protocol for both finding services and communicating with them. Service discoverability is covered via this protocol's profile, the Generic Attribute Profile, which goes by the name Attribute Protocol.

CHAPTER 3

TYPES OF CONNECTION IN BLUETOOTH

Dr.Sudhir Kumar Sharma, Professor,
Department of Electronics and Communication, School of Engineering & Technology, Jaipur National University, Jaipur, India
Email Id- hodece_sadtm@jnujaipur.ac.in

Variations of Bluetooth technology

The performance of Bluetooth technology has significantly increased with each new version:

V1 of Bluetooth

Versions 1.0 and 1.0B were included in the initial Bluetooth receivers. They are already out of date today since they had several issues with the device connection. Following this, Bluetooth 1.1 and 1.2 were accepted as the industry standard for wireless data transfer. The Bluetooth 1 variants had a transmission speed of around 721 kbps.

V2 of Bluetooth

These Bluetooth models made it possible for consumers to connect more quickly and easily. They included a menu that allowed users to pick and find additional nearby Bluetooth-capable equipment, which allowed them to automatically connect different mobile devices. They also increased the transmission rate, leading to the inclusion of the BR/EDR concept in various Bluetooth variants. While the transmission rate in BR might reach 1 Mbps, it could reach up to 2 Mbps in EDR Bluetooth and, in rare instances, even close to 3 Mbps. A user security-focused enhancement offered by the Bluetooth 2.1 version is Secure Simple Pairing (SSP). The process of analyzing is improved. Additionally, they made the Oler Subrating information available, which allowed for less energy use.

V3 of Bluetooth

Launched in 2009, Wireless 3.0 Wirelessly types included HS (High Speed) functionality, making it easier to transmit larger data packages, including films or music files. Bluetooth 3.0 altered the usage of the technology by enabling greater transmission rates of up to 24 Mbps. They also included the usage of wifi as an option for the transmission of extremely large data packages. This brand-new function was known as MAC/PHY alternative.

V4 Bluetooth

This version, which was made available in 2010, included Bluetooth 4.1 and 4.2 improvements. The newest of these Bluetooth varieties, Bluetooth Smart, incorporates Wireless Low Activity technology in addition to handling the capabilities that Bluetooth HS previously handled. This results in less power usage, which is beneficial for devices designed for prolonged use.

The range of Bluetooth wireless and Built-in Bluetooth technologies expanded with the addition of the BLE philosophy, which allows for longer synchronization, and now includes items like medical equipment and performance monitors for sports. Additionally, the transmission distance was enhanced and can now transport data at speeds of up to 38 Mbps.

Features of Bluetooth 4.2

Because it manages information more intelligently, Bluetooth Smart, a word created with Wireless 4.1 and Bluetooth 4.2, earned its moniker. The Bluetooth 4.0 classes prioritise transmitting brief data packets only when necessary and putting the connection into deep sleep if not used, as opposed to maintaining a steady flow of data.

V5 of Bluetooth

Because since the end of 2016, these most recent Bluetooth iterations have become accessible. Does Bluetooth 5.0 support? The majority of handsets on the market do not yet use them, but it is anticipated that most smartphones introduced in the last part of the year will do so. Bluetooth 4.2 vs. Bluetooth 5: The new version's data transmission rate, which is twice that of Android 4.1 and Wifi 4.2, has been significantly improved. Additionally, it increases connection by 4 times over prior iterations. Another oddity of this particular Bluetooth is that it allows for many simultaneous users for communications between various Bluetooth devices. Fundamentally, Bluetooth low-energy's design is fairly straightforward. It is divided into three fundamental components: controller, host, and applications, as shown in, 3-1. A physical device that can broadcast and receive radio signals and comprehends how these signals might be interpreted as packets containing information is often the controller. The host, which controls how two or more devices may connect and how several services can be offered simultaneously over the radios, is often a software stack. To allow a use case, the apps make use of the controller and the software stack.

The Physical Layer and Link Layer, as well as a Direct Test Mode and the lowest layer of the Host Controller Interface, are all included in the controller. Three protocols are present on the host: the Security Manager Protocol, the Attribute Protocol, and the Logical Link Control and Adaptation Protocol. The Generic Attribute Profile, the Generic Access Profile, and modes are also included in the host of the controller. The component that most people associate with the Bluetooth chip or radio is the controller. However, referring to the controller as a radio is oversimplified; the controller is made up of hardware to handle packet transmission and receiving, as well as analogue and digital radio frequency components. An antenna serves as the controller's interface with the outside world, and the host controller interface serves as its interface with the host.

Physical Layer

The bit that uses the 2.4GHz radio to send and receive bits is known as the Physical Layer. The Physical Layer is amazing to many individuals. It is just the transmission and receiving of electromagnetic energy, not magic in its purest form. Radio waves often transmit data by changing their amplitude, frequency, or phase within a certain frequency region. When utilising Bluetooth low energy, the radio waves' frequency is changed using a modulation technique known as Gaussian Frequency Shift Keying, allowing either a zero or a one to be disclosed (GFSK).

Ones and zeros are encoded into the radio using a technique called frequency shift keying, which involves slowly adjusting the frequency up and down. There is a pulse of energy that spreads out across a greater range of frequencies if the frequency is quickly switched to one side or the other at the time the frequency changes. To prevent the energy from moving too far into higher or lower frequencies, a filter is needed. In the case of GFSK, the filter in use has a Gaussian curve-like form. The Bluetooth low energy filter is less restrictive than the Bluetooth classic filter. This indicates that compared to conventional radio transmission, a low-energy radio signal dissipates a bit more widely.

This minor enlargement of the radio signal is advantageous since it indicates that the radio is subject to spread-spectrum radio laws as opposed to Bluetooth classic radio standards, which apply to frequency-hopping radio. Radios are permitted to broadcast on fewer frequencies under spread-spectrum radio restrictions than under frequency-hopping radio laws. The Bluetooth low-energy radio would not be able to broadcast advertisements on only three channels without the more relaxed filter shape; instead, it would have to employ a large number of channels, increasing the system's power requirements as discussed earlier. The modulation index describes this modest enlargement of the radio signal.

The modulation index measures how often utilised higher and lower frequencies are around a channel's fundamental frequency. A positive frequency deviation from the centre frequency of the radio signal of more than 185 kHz indicates a bit with the value 1; a negative frequency deviation of more than 185 kHz represents a bit with the value 0. The 2.4GHz spectrum is divided into 40 distinct RF channels, each spaced 2MHz away from the others, for the Physical Layer to function, particularly when several other radios are in the same region broadcasting at the same time. Every microsecond, one bit of application data is sent over the physical layer. Although this does not account for any packet overhead, it would take only 80 s to transfer the 80 bits of data for the text "low energy" encoded in UTF-8.

Direct Test Mode

An innovative method for evaluating the Physical Layer is Direct Test Mode. The majority of wireless standards don't specify how to have a device do common Physical Layer tests. This results in the issue of several businesses developing their own unique, proprietary techniques to evaluate just their Physical Layers. This raises prices across the board and makes it harder for a maker of finished goods to switch rapidly between silicon suppliers.

The Physical Layer of a controller may be instructed to broadcast or receive a series of test packets using the Direct Test Mode. The Physical Layer's functionality may then be ascertained by the tester by examining the packets received, or the number of packets that the test equipment received. To check if the Physical Layer complies with the RF specifications, the tester may also measure various RF properties from packets that have been received. The Direct Test Mode may be utilised for radio calibration and production line testing in addition to qualification testing. The radio may be tweaked to match what it should be doing, for instance, by simply ordering a Physical Layer to send on a certain radio frequency and monitoring the sent signal. Every unit normally requires this

kind of calibration, thus having test tools that can accomplish it quickly may help product producers save money.

Layer Link

The Link Layer is perhaps the Bluetooth Low Energy architecture's most complicated component. It is in charge of connection creation and maintenance, scanning, and advertising. Additionally, it is in charge of making sure packets are properly formatted, with precisely computed check values and encryption sequences. Channels, packets, and processes are identified as the three fundamental ideas needed for this.

Data channels and advertising channels are the two categories of Link Layer channels. Devices that are not connected to a network and are not delivering data utilise advertising channels. There are three channels for advertising; once again, this strikes a balance between strength and low power. These channels are used by devices to broadcast data, announce their connectivity and discoverability, and search for and make connections. Once a link has been established and data has to move, the data channels are only utilised then. To provide resilience, 37 data channels are used by an adaptive frequency-hopping engine. The data channels enable the sending, acknowledging, and retransmission of data from one device to another. On a per-packet basis, data channels may be encrypted and authenticated.

Small packets are specified to deliver data on either of these channels data or advertising. A packet is a little chunk of data that is transferred across a very short distance from a transmitter to a receiver. In addition to information identifying the intended recipient, packets also include a checksum to guarantee their validity. Each packet has at least 80 bits of addressing, header, and check information, and its fundamental packet structure is the same for both data channels and advertising channels.

The packet structure of the Link Layer

The packets are designed to increase their robustness by using a 32-bit access address that is fixed for advertising packets but completely random and private for data packets; an 8-bit header to describe the contents of the packet; an 8-bit length field to describe the payload length, though not all of these bits are used for length; and an 8-bit preamble to allow the receiver to synchronise bit timing and set the radio's automatic gain control. The smallest packet that may be transmitted is an empty data packet that is 80 seconds long, while the longest packet is an advertising packet that is filled and is 376 seconds long. Most data packets are 144 seconds in length, but the majority of advertising packets are just 128 seconds long.

A Host/Controller Interface (HCI), which is available for many devices, enables the communication between a host and a controller across a defined interface. Over 60% of all Bluetooth controllers employ the HCI interface for Bluetooth classic, where this architectural separation has proved to be quite successful. It enables data and instructions to be sent back and forth between the controller and the host. The logical interface and the physical interface are truly two distinct components.

The instructions, events, and related behaviour are defined by the logical interface. A local application programming interface (API) on the controller may be used to offer the logical interface, enabling the inclusion of an embedded host stack inside the controller, or it can be supplied through any of the physical transports. The orders, events, and data are transferred across various connection technologies according to the physical interface. A few of the stated hardware interfaces include USB, one SDIO, two, and two UART variations. They will only support one or maybe two interfaces for the majority of controllers. Additionally, a USB interface is not the lowest power interface and needs a lot of hardware to implement, therefore it is not often offered on a Bluetooth low-energy single-mode controller.

The host controller interface must exist on both the controller and the host; therefore, the portion of the interface that is inside the controller is frequently referred to as the lower-host controller interface and the portion of the interface that is inside the host is the upper-host controller interface. The host is the Bluetooth world's unsung hero. The host includes multiplexing layers, protocols, and processes for carrying out a variety of practical and fascinating tasks. The uppermost controller interface serves as the foundation for the host. The multiplexing layer known as the Logical Link Control and Adaptation Protocol is placed above this.

Two key system-building elements are added on top of this: the Attribute Protocol, which exposes device state data, and the Security Manager, which handles everything from authentication to setting up secure connections. The Generic Attribute Profile, which is based on the Attribute Protocol, outlines how to use it to enable reusable services that reveal a device's common characteristics. The Generic Access Profile defines the interoperable way that devices locate and connect. The host has no specified higher interface. The host APIs will be exposed differently by each operating system or environment, whether it be via a functional or object-oriented interface.

Adaptive Logical Link Control Protocol

The multiplexing layer for Bluetooth low energy is called the Logical Link Control and Adaptation Protocol (L2CAP). The L2CAP channel and the L2CAP signalling instructions are two fundamental notions that are defined by this layer. A single bidirectional data channel that terminates at a certain protocol or profile on the peer device is known as an L2CAP channel. Each channel is autonomous and can be con,d with its flow control and other configuration data. The majority of L2CAP's capabilities, such as dynamic channel identification, protocol service multiplexers, improved retransmission, and streaming modes, are used by Bluetooth classic. L2CAP is just required at the barest minimum for Bluetooth low energy.

Only fixed channels one for the signalling channel, one for the Security Manager, and one for the Attribute Protocol are used in Bluetooth low energy. There is only one frame format available, the B-frame, which contains a two-octet length field and a two-octet channel identification field. Up until the frame formats are negotiated to something more sophisticated, traditional L2CAP utilises this frame format for every channel. For instance, frame formats with extra frame sequencing and checks are feasible for Bluetooth classic. These are not required in Bluetooth low energy since the Link Layer checks are sufficient to prevent the need for further checks, and the straightforward

Attribute Protocol does not need the delivery of packets from different channels out of order. Figure 3.1 discloses the system infrastructure of the model.

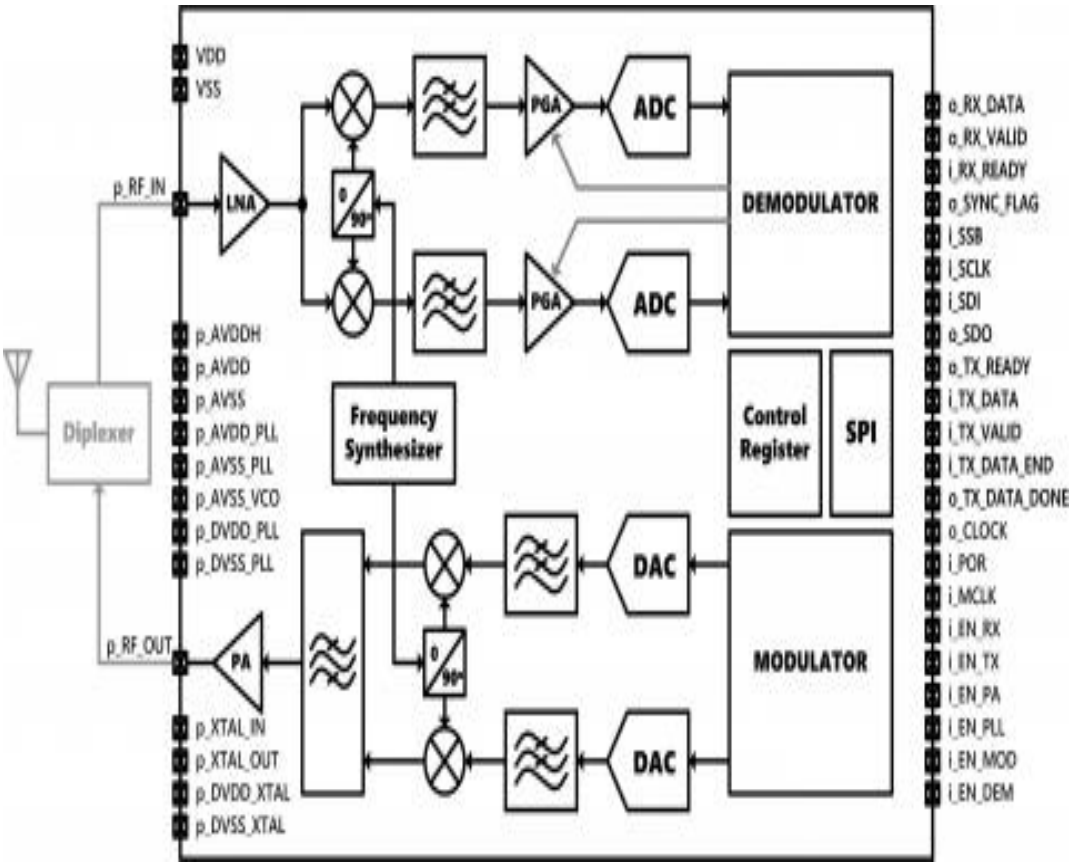

Figure 3.1: Discloses the system infrastructure of the model.

The Security Manager specifies a straightforward system for key distribution and pairing. By usually authenticating the other device, pairing is the process of trying to trust another device. Pairing is often followed by the encryption of the connection and the distribution of keys. Shared secrets may be sent from a slave to a master through key distribution so that when the two devices reconnect later, they can rapidly establish their validity by encrypting using the already transferred shared secrets. Additionally, the Security Manager offers a security toolkit for creating data hashes, confirmation values, and temporary keys for pairing.

The Attribute Protocol

A set of guidelines for getting access to data on a peer device is defined by the Attribute Protocol. An attribute client may read and write the "attributes" that contain the data on an attribute server. The server receives requests from the client and sends back response messages. These requests enable the client to discover all the server's characteristics and read and write them. There are six distinct message types in the Attribute Protocol Clients send requests to servers; Servers respond to requests by sending responses to clients; Clients send commands to servers that receive no

response; Servers send notifications to clients that receive no confirmation; Servers send indications to clients; and Clients send confirmations to servers in response to indications. Therefore, both the client and the server can start a conversation with messages that need a response or with messages that don't.

Data bits with addresses and labels are called attributes. Each attribute is identified by a distinct handle, a type that describes the type of data it stores, and a value. For instance, an attribute with the handle 0x01CE may house a temperature-related attribute with the value. While certain attributes may be grouped and their group meanings can be found using the Attribute Protocol, the Attribute Protocol does not specify any attribute types.

According to the Property Protocol, certain attributes may be read-only, write-only, or only allow access to the value of the attribute if the client has successfully authenticated themselves or has been granted permission by the server. The only way to directly determine an attribute's permissions is to issue a request and then get an error message explaining why the request could not be fulfilled.

The Attribute Protocol is mostly stateless in and of itself. The state is not kept on the server for each transaction, such as a single read request and read the response. This implies that relatively little memory is needed for the protocol itself. The prepare and execute write requests are the lone exception to this rule. These hold a collection of values that must be written to the server and then performed one at a time, in succession.

The Generic Attribute Profile

The Attribute Protocol is positioned under the Generic Attribute Profile. It outlines the many kinds of characteristics and their applications. Numerous ideas are introduced, such as "characteristics," "services," linkages between services, and characteristic "descriptors." Additionally, it specifies a variety of methods for reading and writing characteristic values as well as discovering services, traits, and connections between services.

An immutable encapsulation of some atomic device function is what is known as a service. Despite the lengthy string of difficult-to-understand jargon, the idea is rather straightforward. Once a service is published, it cannot be changed since it is immutable. This is essential since a service can never be altered and so cannot be reused. Version numbers and other cumbersome setup processes and settings become the antithesis of a connectionless model, one of the fundamental ideas underlying Bluetooth low energy, as soon as a service's behaviour changes.

Encapsulation is the concise expression of a thing's characteristics. When you understand the boundaries of service on an attribute server, you understand what information that service is encapsulating. Everything about a specific service is contained and represented via a collection of attributes in an attribute server. A single irreducible unit or component of a larger system is formed by an atomic method or formation. Atomic services are crucial because a smaller server is more likely to be reused in a different setting. The likelihood of them being repeated is drastically decreased if we developed complicated systems with many, potentially connected activities.

Behaviour refers to how something behaves in response to certain stimuli or environments. For services, the behaviour refers to what occurs when you read, write, or when the client is alerted of an attribute. Interoperability depends heavily on explicitly described behaviour. Each client might behave differently while communicating with the service if the behaviour of the service is not well described in the specification. Depending on whatever client is connected, the services may therefore behave differently, or, more importantly, the same service may behave differently across many devices. Interoperability is broken as soon as this is ingrained in the devices. Interoperability is thus supported by carefully described behaviour that is tested, even for incorrect interactions.

The complicated behaviours that gadgets disclose are mostly dependent on service interactions. Service is by definition atomic. It is not advisable to disclose complex actions in a single service. Consider a gadget that exposes a temperature service to measure the room's temperature. A battery service would be exposed if the gadget was powered by a battery. We should be able to uncover another instance of the temperature service on the device, however, if the battery includes a temperature sensor as well. For a customer to understand how this second temperature service relates to the battery, that link must be made clear.

Services are divided into two categories: major services and secondary services, to accommodate complex behaviours and interactions between services. The way a service is utilised in a device determines its type more often than not, not the service itself. A major service makes the functionality of the device visible from the user's viewpoint. A secondary service is one that a major service or another secondary service utilises for it to perform all of its functions. In the preceding example, the battery service and the first temperature service were both main services, but the second temperature service the battery temperature was a secondary service that was referred to by the battery service.

The Generic Access Profile

The Generic Access Profile outlines how gadgets locate one another and offer meaningful data to consumers. It also describes the bonding process, which describes how gadgets may establish a long-term connection. The profile specifies how devices may be discovered, connected, and bonded to achieve this. It also explains how devices may find other devices, connect to them, read their device names, and form bonds with them using processes.

By leveraging resolvable private addresses, this layer also introduces the idea of privacy. For gadgets that continually broadcast their existence so that other devices may find them and connect to them, privacy is crucial. However, to broadcast in secret, devices must use a random address that is continually changing. This prevents other devices from being able to identify the device by eavesdropping or by tracing the device's current random address over time. The private address must be able to be resolved for trustworthy devices to be able to find it and establish connections. Therefore, the Generic Access Profile specifies both how to connect to private devices as well as how private addresses may be resolved.

The Application Layer

The Application Layer is located in front of the Controller and the Host. Three categories of specifications characteristics, services, and profiles are established by the application layer. On top of the Generic Attribute Profile, each of these requirements is constructed. The applications specify the specifications that make use of these attribute groups, whereas the Generic Attribute Profile specifies how attributes for traits and services should be grouped.

Identifiers

A characteristic is a piece of data with a well-known format that is identified by a UUID. Since they are intended to be reused, characteristics lack behaviour. As soon as anything has behaviour, it restricts its ability to be used again. The most intriguing aspect of characteristic specifications is that, unlike human-readable language, they are described in a computer-readable manner. This enables computers to download this computer-readable specification and use it to present these characteristics to the user when they encounter a characteristic for the first time.

Services

A service is a human-readable definition of a group of properties and the behaviour that goes along with them. The service does not specify how clients should behave; it merely specifies how these features should behave on a server. For many services, the server behaviour of the service might indirectly influence the client's behaviour. However, for certain services, it could be necessary to describe more complicated client behaviour. The profiles, not the services, determine this client's behaviour.

Other services are one kind of service. The parent service is only permitted to specify the services that are included; it is not permitted to modify the traits or actions of the services that are included. However, the included service may explain how many included services communicate with one another. The main or secondary nature of a service may be declared in a service specification or can be left up to the profile or implementation, as indicated, Services come in two variants: primary and secondary. A primary service exemplifies what a certain gadget performs; it is via these services that the user would be made aware of the capabilities of the device. The main services and various secondary services are supported by secondary services.

The connection between devices necessary to locate and utilise services is not described by services. Services are limited to describing what occurs when a characteristic is read, written, or signalled. The qualities found in a service, how they are utilised by a client, or how Generic Attribute Profile processes are used to discover the service are not described in the services. The pinnacle of a use case or application is a profile. Profiles are descriptions of two or more devices with one or more services on each, according to their requirements. Additionally, profiles specify the discoverability and connectivity of the devices, establishing the topology required for each device.

To provide the functionality needed by the use case or application, profiles also specify how clients locate the service, learn about its features, and utilise it, there is a many-to-many mapping of profiles to services. Many profiles may utilise a service to allow a certain activity on a device.

Application stores may be provided with the list of services that a device supports and identify the set of apps from the store that utilises these services. The behaviour of a service is independent of which profile is utilising this service at this moment. The plug-and-play concept that has proven so successful for the universal serial bus is made possible by this flexibility.

Stack Splits

A Bluetooth low-energy product may be created by combining many distinct stack splits. There are numerous alternative stack splits that may be used, but the standard only specifies one that uses the host controller interface between the controller and the host.

Single-Chip Solutions

The single-chip option is the simplest stack split that is compatible with Bluetooth low energy. There are no stack splits in this; the entire product is crammed onto a single chip. The controller, host software, and applications are all present on this chip. This is the pinnacle of low-cost products because it only needs a power source, an antenna, some hardware to interface with like buttons and lights, and a few extra discrete parts.

Unfortunately, employing single-chip solutions has significant drawbacks. First, due to the chips' severe resource limitations, the development environments are more challenging to utilise. Second, to save money, the programme must be written to the chip's read-only memory (ROM). This necessitates the creation of a unique chip for a particular product. For extremely large production runs, the lower bill of materials may make up for this, but for smaller production runs, the technique can be highly costly.

A single mass-produced chip that integrates the controller and hosts up to the top may be utilised with a tiny nonvolatile memory chip to store the application in devices with short production runs or prototype items. This results in very affordable small manufacturing runs. The non-volatile memory's contents are read into and put to use by the single chip upon startup. As a result, you may produce a product in small quantities at a low cost while still having an excellent platform for prototyping.

The traditional approach for two-chip systems is as seen, where the controller is on one chip and the host and applications are on a different chip. Because they already have very strong processors capable of executing the whole host and application software stack, computers and mobile phones are the most common applications for this technology. A standard Host Controller Interface is commonly used in this solution's controller chips. For devices with a highly strong CPU already installed, this architectural divide is perfect, but it is not suitable for any other kind of device.

A different two-chip arrangement is one in which the applications are on a separate chip from the controller and host. Because the application chip doesn't need a lot of memory or other resources to execute the programme, this has the benefit of allowing the application chip to be a very compact, low-power microprocessor. Typically, a basic UART would be used as part of a bespoke interface between the two circuits. The benefit of this technique is that two typical mass-produced chips may be merged, and the application chip can be developed using typical tools.

Additionally, multiple-chip systems are a possibility. As illustrated it would be conceivable to merge a host chip, an application chip, and a standard controller on a single chip. Two different interfaces would be necessary for the host chip. These solutions are usually too expensive, so they are usually only used in developing systems where multiple interfaces are used to enable the independent instrumentation of each layer. Because the expense of integrating into fewer components is not negated by the savings on each end product if only a small number of items are ever to be created, short production runs may also be able to withstand this complexity. This architecture would never be economically feasible for mass-produced goods.

There isn't any more knowledge in the whole world's literature than is transmitted on television in just one major American city in a given year. Not all bits are created equal. Bluetooth low energy makes it possible to use wireless technology in a new manner. Presence detection, data broadcasting, and connectionless models are some of the primary new models that are centred on the advertising model. Gateways from gadgets to the Internet are also included.

Detection of Presence

Presence is the most intriguing new wireless technology made possible by Bluetooth low energy. The term "presence" refers to the fact or condition of being in a location or item. Devices may passively search for other broadcasting devices in the background using the advertising approach. The advertising devices could just be promoting their address or they might be promoting some presence-based information.

The Link Layer defines a new method of operation called advertising. Devices may use it to periodically communicate their identity and a little bit of data. This concept is feasible due to the radio's enhanced modulation index and the 2.4GHz band rules, which allow for the transmission of broader radio signals through a non-frequency-hopping radio. The radio doesn't utilise frequency hopping, hence there are fewer channels required for devices to be discoverable and connectable. This implies that advertising and scanning are substantially more effective.

There are two scanning modes available: active and passive. To gather more static data via active scanning, the scanner must ask marketers for more information. The only need for passive scanning is that the scanner listens to advertising packets. Ad packets may be delivered to the host after being received by the Link Layer. The host can pinpoint its location by using knowledge of the adjacent devices. For instance, if the host finds an automobile, it may recognise that it is inside or close to one and adapt its behaviour, possibly by connecting to it. Similar use scenarios may also arise at work, at home, or in a coffee shop. A gadget may automatically alter its behaviour depending on where it is thanks to this background judgement made by passive scanning.

As was previously said, presence is the process by which a mobile device ascertains its location. The ability of static devices to identify the devices in a certain place is an additional sort of presence. Finding someone or anything in a huge office building, for instance, would probably be the advantage of this that is most beneficial. Gadgets in each room keep track of the devices they can detect while the persons or devices that desire to be followed would advertise sparingly. A central device may then receive this information to identify the position. This may be used to

automatically route calls to the closest phone or to keep track of staff during a building's emergency evacuation.

Data for Broadcasting

A very little amount of data just a few tens of octets can also be broadcast using the advertising paradigm, but the ability to reach any nearby listening device with this tiny amount of data is highly useful. As was already said, figuring out a device's location based on what devices are broadcasting is a helpful feature in and of itself. The ability to pinpoint the transmitting equipment to a specific place is necessary for this, however. Data dissemination facilitates this mapping. Broadcasting may be used to send a wide range of important data. Initial connection setup, advertising, and broadcasting information are the three major ways that broadcasting data may improve user experience.

Devices may broadcast information about their kind and their desire to connect to another device with a complementary set of services or profiles to aid in the first connection setup. For instance, when a television is initially turned on after being removed from its packaging, it begins to look for the remote control. The remote control begins to announce that it is searching for the television as soon as the batteries are put in. This broadcast data is received by the television, which connects to the remote control and instantly couples with it before enabling secure communication. This implies that from the consumer's viewpoint, there are no connect buttons or pairing menus; instead, they just switch on the television for the first time, instal the batteries in the remote control, and then push buttons on the remote to operate the television.

Numerous groups find that advertising is a helpful tool. With it, customers may find real-world services from more than 100 metres away. International airports and train stations are two obvious locations where free wifi technology may be used for advertising. Travellers who don't want to spend a lot of money on roaming fees or Wi-Fi Internet connection now have another option for learning information thanks to the ability to advertise gate or track data for planes or trains. Simple bus stops may additionally display information about the next bus' arrival time and destination. It is also possible to broadcast data that a gadget has locally acquired. A temperature sensor, for instance, might broadcast the current temperature to any gadget that is actively listening for it. When information is felt that changes quickly and is valuable for various devices, this is most helpful.

CHAPTER 4

BLUETOOTH LOW ENERGY SYSTEM

Chandra Shekhar Rajora, Assistant Professor,
Department of Electronics and Communication, School of Engineering & Technology, Jaipur National University, Jaipur, India
Email Id- chandra.shekhar@jnujaipur.ac.in

A brand-new technology called Bluetooth low energy has been created as both a combination with the least amount of electricity and complementary technologies to conventional Bluetooth wireless technology that is programmable and constructible even if it makes advantage of Bluetooth Bluetooth Low Energy, a popular brand that heavily incorporates technologies from its parent, should be regarded as a separate technology, meeting various design objectives and market divisions. The purpose of classic Bluetooth was to connect the disparate domains of computation and communications that connect laptops and mobile devices. However, this game-changing feature has the mobile phone's audio connection to a headphone that was put on or nearby worked as an ear. Greater and greater use situations became created as the technology developed, including

Streaming stereo music, wireless phone book downloads to your automobile, and stereo transferring files and printing the bandwidth needed for each of these additional use cases increased, and as a result, Bluetooth has been regularly updated with faster radios environment through time. When Bluetooth was launched, Basic Rate (BR) had a maximum Data rate for the physical layer of one gigabits per second (Mbps). Superior Data Rate Current version of Bluetooth introduced (EDR) to improve the Physical Layer data rates to 3Mbps; version 3.0 of the protocol includes an Alternate MAC1 PHY2 (AMP).

Numerous measures have been made to reduce the likelihood of information theft using Bluetooth as a result of the endless discussions about how simple or difficult it should be to link two devices through Bluetooth. There must be a combination involving economy and assurance, as with most other situations. With our society becoming more digital, information security is becoming a rising concern. Most of us find that connecting two units makes our days simpler, but there are certain risks involved. To access the material on a computer or phone, one device may pass for a typewriter or a headset. While it might be challenging to protect oneself after a connection has been made, it is often simple to keep track of which Bluetooth-enabled devices are linked and to regulate each one individually. Security features need to be developed quickly, but as usual, common sense is needed.

Various power classifications

Three separate power classifications exist in the technology. Level 2 has a range of more than 100 meters and is used for devices with no power restriction. Class 2's range is around 10 metres, whilst Class 3's range is approximately 5 meters. Battery-powered gadgets often employ Class 2 or Class 3.

Various profiles

Different profiles include Bluetooth-enabled devices. These profiles list the functionalities that the device is capable of supporting. Various units might have various profiles. The Generic Connect Policy and Resource Scheduling File Profile, however, must be supported by everyone. The most popular profile is undoubtedly the headset profile (HSP). To communicate between headsets and often a mobile device, this profile is utilised. The Video Distribution Profile (VDP), which permits the transmission of a video stream, is another popular profile.

Using IEEE3 802.11, Bluetooth was able to transmit Physical Layer data speeds of up to many hundred megabits per second. The approach used by Bluetooth low energy is quite different rather than merely it has been designed to use very little power while improving the available data speeds consumption. The likelihood that you won't get high data speeds, or even want to maintain a relationship for many hours. Given that often wireless broadband transmission technologies are always advancing in speed, this is an exciting development.

Understanding that the traditional approach has allowed for this alternate course to be taken the minimal power requirements needed for devices cannot be met by Bluetooth technology. Button-cell batteries provide energy. To properly comprehend the demands, however, another aspect of limited power has to be taken into account. Low-energy Bluetooth is also intended for deployment on devices that do not already support or have no wireless technology at all. One strategy for achieving very large volumes is to affordable. Radiofrequency identification (RFID) tags, for example, are deployed in extremely huge quantities since they are ultimately very inexpensive they function by scavenging power from a costly scanner. It is vital to consider the Built-in Bluetooth system design as well. From the demand for affordability. This design's three main components point to being very affordable.

Connectivity-Free Model

The way a connectionless model has been developed and used from Bluetooth classic to Bluetooth low energy is one of the largest improvements. In a connectionless approach, devices may transmit relevant information rapidly without maintaining a connection. There is no cost to breaking and reestablishing a connection when data has to be transferred since the primary protocols never create a connection-oriented channel between devices. As a result, devices are encouraged to only connect when data needs to be transmitted rather than keeping a costly connection open in case data does need to be delivered. There are several intriguing design differences from conventional wireless protocols that are imposed by this connectionless architecture.

The protocol may be used to establish the state information over time in a connection-oriented channel. As a result, state information is often only accessible when a user remembers the implicit state that both devices have already produced. Delays occur during the first connection when the state information is found and negotiated since this state information takes a long time to establish. Implicit state-based protocols often contain negotiation and setup steps in addition to feature bits and version numbers. A state-full system may be more effective if a connection is likely to be active for a lengthy period and there is a lot of state information.

Unfortunately, many protocols lack complete definitions, with each state bit being implied rather than expressly declared. Because each device believes the connection is in a different state and thus has a different expectation of what might or should happen next, this causes interoperability issues. One of the main issues with connection-oriented systems is this. This problem may be resolved by clearly specifying the state and the operation of any state machines. An excellent illustration of this is the Logical Link Control and Adaptation Protocol (L2CAP) Layer of Bluetooth, which uses a simple state machine and configuration system to create connections for Bluetooth Classic devices. The connection's whole state is stated openly, and the connection state machine is completely explained. However, it has taken more than ten years for this to reach the excellent status it now has.

The connectionless approach addresses these issues by specifying the state of the device rather than the state of a connection. It is possible to connect at any moment and, upon reconnection, get the current state straight from the other device by exposing the state using a stateless protocol, such as Attribute Protocol. To force a state machine into various states as specified by a service, it is also feasible to explicitly create state machines with both an exposed state and an exposed control point. Additionally, if a device has registered to receive information about state changes, it may be able to re-establish a connection simply because some information in this state has changed.

It may be used, for instance, to indicate a device's battery level. When a battery-powered device is connected to a monitoring device, the battery level is read, the battery level is set up to receive notifications when it changes, and the monitoring device disconnects. The battery-powered gadget gradually makes itself connectable when the battery level does change, and the battery-monitoring device realises that it needs to communicate. The battery-powered gadget then joins the battery-tracking gadget.

The battery-powered gadget may inform the monitor of the updated battery level as soon as a connection is made, following which the link can be instantly cut off. All of this may occur in less than 3 milliseconds. This would take around 99 reports of 3 milliseconds each, for a total radio-active time of just under 300 milliseconds, for a fully charged gadget with a battery life of only one year. In comparison, it might take a comparable amount of time for each report in Bluetooth classic only to establish a connection-oriented channel.

Gateways

The development and ubiquity of the Internet over the last several years have brought about the most significant transformation in computer technology. Everything seems to be linked to it, including radios, TVs, and newspapers. Whether it be speech, moving images, or written text, all of them are gateways for media. Of course, there are a variety of additional applications for the Internet, including teleconferencing audio and video connections, social networking, and communications like email. For every gadget that is already linked to the Internet, connecting hundreds more devices will be the next major issue. This is a significant shift, and the existing infrastructure probably won't be able to initially handle all of the incoming data.

The Internet's reliance on a connection-oriented approach is its main drawback. A TCP 1 connection, which takes some time to create, is a session-oriented channel that is formed between two devices. Sessions are also predicated on the need that all devices connected to the Internet to have an address. A device may only have one address at a time, thus it must be programmed into it, or it must ask another device to give it an address for a certain time, which it can use until the time runs out, at which point it must request another address.

The fact that the Internet is now set up around a wired infrastructure is its worst flaw. Because the devices they link to are immobile, they may also be connected to other wires carrying power, which makes wires fantastic. This indicates that wired protocol energy efficiency is seldom taken into account. Internet devices often need to be listening continually because routers continuously check for the mapping of allotted or unallocated Internet addresses to devices, at random intervals. When gadgets need very low power consumption and are continually in motion, this is ineffective. There has to be a change in strategy.

Most houses with many computers linked to the Internet employ the concept that is used in Bluetooth low energy. This is how a network address translation gateway works (NAT). Your average house has a single Internet address that is assigned to the gateway or router from the outside. The gateway, however, assigns unique sets of addresses to each connected device in the house. The key is that the gateway hides the internal network's structure from the outside world by translating internal addresses to a single external address. The external environment only sees one device and is unconcerned with which device is transmitting or receiving the data.

The cost of Internet Protocol is high. A 128-bit source and destination address must be supplied in each packet sent over an IPv6-based network. This indicates that a packet must include at least 32 bytes before any further protocol overhead. The largest Bluetooth low-energy package is smaller than this. Therefore, even if we were to ignore the fact that they were created when everything was wired, it becomes extremely challenging to utilise the Internet Protocols directly over low energy. However, the gateway paradigm enables us to shield the internal addressing of devices from view. Any alternative transparent to the outside world addressing system or a different IPv6 address space might be used for this internal addressing of devices.

Tiny wireless devices can connect to the Internet and do so with the least amount of power feasible by utilising Bluetooth low-energy gateways. By shifting the complexity of the Internet to gateway devices with the necessary capacity, Bluetooth low energy achieves this. Additionally, it enables these gateway units to assign whatever mapping scheme they like to Internet addresses to devices. This might be accomplished by giving each device a unique IPv6 address or by assigning a port number from a single Internet address to each device.

Gateways are helpful since your refrigerator requires a means to communicate with the manufacturer's Internet server to let them know that the compressor pump is failing and needs to be replaced under warranty. The same procedures must be followed for all manufactured goods, including vacuum cleaners, autos, and washing machines, to mention a few. Additionally, the maker could wish to update the cleaning programmes by sending information to these devices.

Therefore, gateways provide a means for devices to communicate with the Internet without being burdened by the wired Internet's power-hungry protocols.

Albrecht Einstein

Background

Radio waves are used by two or more Bluetooth low-energy devices to communicate with one another. The history of radio goes back a long time, beginning with very basic spark-gap transmitters, progressing through amplitude modulation and frequency modulation, and more recently, phase-shift keying and other more sophisticated modulation systems. The parts that follow provide an introduction to radio operation, covering everything from the fundamentals to contemporary modulation techniques utilised by Bluetooth low energy.

Analog Modulation

The simplest radio you can make is probably the fundamental spark-gap model. It just only two components to make since it is so simple. Take a nine-volt battery with two battery connections at the top and an electrically conductive metal coin. But first, you need to set up a receiver. You may do this by using a radio that is tuned to the AM band but not to any specific radio station. Next, make short contact with the coin's terminals on the battery. When the coin is extremely near to the terminals at the top of the battery but not contacting them, electricity flashes across the gap and the radio should pick up the interference generated by the coin and the battery.

Radios with spark-gaps have two issues. They are ineffective, to begin with. To broadcast across a long distance, you normally require an electrical potential difference of several thousand of volts. Second, only one radio may broadcast simultaneously in the same region. This greatly limits the capacity to convey several messages simultaneously in a given area. Imagine, for instance, that there is only one television channel available to the citizens of the whole city.

Amplitude modulation, or AM radio, represented the following development in radio technology. It was discovered that radio waves may communicate a single frequency. Then, information might be modulated and sent using these carrier signals. Amplitude modulation included changing the carrier signal's amplitude or loudness. Fundamentally speaking, this was a significant advancement since many radio signals could now be broadcast simultaneously. A depiction of an analogue amplitude modulation signal.

Amplitude modulation in analogue

In the future, several radio stations would be established by governments and commercial enterprises. When only shortwave radio was available, nations would set up their radio station and purposely choose a section of the spectrum that would interfere with another country's station so that their populace could only listen to local propaganda. In the end, rational and non-conflicting international agreements were made to assign frequencies. Most radio frequency allocations subsequently have been based on these agreements.

Sometimes, while having relatively identical frequency band allocations, various uses may be made of them. For instance, the medium-wave radio bands used for AM radio in the US and the

EU are both between 530 kHz and 1620 kHz, but in the US, each station is given a frequency at intervals of 10 kHz, whereas in the EU, they are at intervals of 9 kHz. As a result, there might be more radio stations in the European Union, but radio receivers need to be able to handle two separate frequency bands.

Listening to an AM radio station makes the issues with amplitude modulation obvious. When a radio station's audio input is very quiet, the receiver may either entirely lose the signal or produce extra noise as it frantically tries to pick up anything usable. Background noise is the noise that is present all the time and is produced by the many electrical gadgets in our environment. Additionally, it is brought on by lightning and other atmospheric factors, such as solar radiation. By eliminating the impact of background noise from the signal, the next development in radio transmission would greatly improve sound quality. Frequency modulation, or FM radio, was used for this.

The carrier's frequency is modulated with the input rather than the amplitude, such that when the input is extremely weak, the output carrier is also very faint. Since audio serves as the input signal, a very quiet input would very certainly produce no carrier frequency departure, but a very loud input would result in a significant carrier frequency divergence. The carrier is always delivered at full strength with frequency modulation, allowing a receiver to latch onto the signal and subsequently demodulate the information out of the signal. Frequency modulation also allows for the placement of many additional carriers near one another. For instance, a contemporary FM broadcast might include pilot tones, a stereo signal (Left/Right), a mono signal (Left + Right), and digital station information (Radio Data System).

Analogue frequency modulation

Therefore, amplitude modulation and frequency modulation both address the primary issues with spark-gap radios. It is rather straightforward, can have much tighter grouped signals, and allows a receiver to latch on to the signal regardless of what that signal is. Phase modulation is one of many further sophisticated modulation techniques. Similar to frequency modulation, phase modulation alters the signal's frequency according to the input. The phase of the signal is altered during phase modulation. Only complicated digital systems, like digital radio, employ phase modulation because it is more difficult than straightforward frequency modulation. In addition, quadrature amplitude modulation, which employs amplitude modulation on two different carriers that are 90 degrees out of phase with one another, is another option. Most of the world transmits digital television using quadrature amplitude.

Digital Modulation

The distinction between chips, bits, and symbols should be quickly reviewed before discussing digital modulation. Bits, which may have a value of either zero or one, are used to convey the incoming data. A multiple-bit value may be created by adding one bit too many other bits. A symbol is an aggregate name for these concatenated bits. Therefore, a symbol is a single value that may stand in for many bits. The alternative method, which is almost ever used in reality, involves sending a bit using several chip codes. Since each chip is essentially a fractional bit, numerous

chips make up one bit. It is even conceivable to merge many chips into a single symbol that represents multiple bits at the very complicated end of the spectrum.

Numerous, s are often thrown about while comparing radio systems. The majority of comparisons are made between things that aren't similar. The application data rate is the most helpful tally. After accounting for any packet overhead and the maximum rate at which packets may be carried, this is the highest data rate at which application data can be delivered. Any additional symbols required to specify and manage the transmission itself in addition to the essential application data are referred to as packet overhead. This may include data for time synchronisation, addressing, headers for the application data, and checks to make sure the application data is legitimate when it is received.

The physical bit rate is the one that is most often cited. If the radio were to broadcast data nonstop for a whole second, this is the most bits that could be sent in that time. There are relatively few radios that can broadcast data continually, apart from television and radio stations. Instead, they must divide the data into several, independent packets. The symbol rate is another helpful tally. The maximum number of symbols that may be communicated in a second is known as the symbol rate, and it dictates how quickly the receiver must operate. The more symbols are processed to retrieve the information bits, the more energy is needed. This is true regardless of the symbol rate. The frame rate is a further factor that must be understood to determine how much application may be supplied. This is the maximum quantity of packets that may be transmitted in a certain amount of time. It is necessary to switch the radio from being a transmitter to a receiver. The radio takes some time to turn. The less time spent delivering application data, the longer this turnaround time must be. It is considerably easier to comprehend modulation strategies while conveying digital data.

On-off keying, or OOK, digital modulation is the most basic. The term "keying" refers to the process of modifying the carrier for a specific digital signal. On-off keying refers to the use of a carrier that is alternately on and off. This may be regarded as a very basic full-on or full-off amplitude-modulated signal.

Therefore, it is clear that it is feasible to communicate 8 bits of information by simply transmitting the right amplitude of the carrier at the proper time. If the lack of a carrier encoded the value zero and the presence of a carrier encoded the value one, respectively. Despite being fairly straightforward, this is susceptible to noise, particularly when the signal is very faint.

ASK, or amplitude-shift keying is the next sophisticated algorithm. It resembles amplitude modulation. ASK will degenerate to OOK if the input signal is binary (0 per cent and 100 per cent).

This can encode two bits of information for each of the four levels that may be represented 0 percent, 33 percent, 66 percent, and 100 percent. The 8 bits of data may thus be delivered in only 4 symbols, which would be twice as fast as an OOK encoded system. Low signal-to-noise ratios are a problem for both ASK and OOK, making it difficult to determine whether the symbol represents anything other than zero.

Amplitude shift keying:

The next level of complexity is frequency-shift keying, or FSK. Similar to frequency modulation, this. The key to figuring out the symbol's value is a change in frequency, as seen in, 5-5. Binary frequency-shift keying, for which an input bit of zero generates a negative frequency deviation and an input bit of one provides a positive frequency deviation of the carrier, is the most basic kind of frequency-shift keying. By altering the rate of frequency deviation, FSK allows for the use of additional levels. The benefit of frequency deviation is that it allows for a much wider operating range than the more straightforward ASK method because the carrier can always be received and, consequently, locked upon.

Frequency shift keying:

The modulation index (h) measures the magnitude of the deviation, where fm is the highest frequency in the source signal being modulated onto the carrier, and ff is the maximum deviation from the carrier frequency. The signal is referred to as a wide-band transmission if the modulation index is greater than one and the carrier oscillates at a higher frequency than the source signal. The signal is referred to be a narrow-band transmission if the modulation index is less than one and the carrier oscillates at a lower frequency than the source. Because minimum-shift keying, or MSK, uses a modulation index of 0.5, this, is regarded as being quite unusual. An FSK variation with high spectral efficiency is MSK.

Frequency Band

The 2.4GHz Industrial, Scientific, and Medical (ISM) band is used by Bluetooth low energy to transport data. Due to two factors, this frequency range is particularly unique. The majority of the radio spectrum is regulated, therefore obtaining a licence is required before transmitting anything. For instance, your local radio station could have enough cash to purchase a licence to broadcast music and commercials on the presumption that the money from the sale of the commercials would be enough to cover the costs of the music, staff salaries, and the licence. While certain frequency bands need a licence, none are charged. This contains bands for emergency services in aviation, the military, and the general public.

One of the frequency bands without a licence is the 2.4GHz ISM band. As long as you abide by the requirements, you may broadcast in this band without purchasing a licence. The requirements are straightforward: the band must be utilised for a local area network (LAN) or a personal area network (PAN) with a restricted range. Although the principles themselves are quite complex, using them for short-range applications is often free. The 2.4GHz ISM band is exceptional for a second reason it is the only license-free spectrum that is uniform across all nations. This implies that a product that utilises this band may be purchased anywhere and used in any other nation without the need for configuration. There are other ISM bands, such as those in the 900 MHz range, although they utilise various frequencies and bandwidths depending on where they are used; for example, the US uses 915 MHz whereas the EU uses 868 MHz. The 2.4GHz spectrum ranges from 2400MHz to 2483.5MHz and may be used everywhere. This results in an approximate 83.5MHz total accessible spectrum.

Modulation

Gaussian frequency-shift keying is used by the Bluetooth low-energy radio. By lengthening the time used to move the frequency from one value to another, a Gaussian filter improves the transition from one sign to the next. Without it, the frequency shift would occur very quickly and generate a lot of noise. This indicates that the shift from zero-bit to one-bit is quick and effective.

Bluetooth low energy sends data at a rate of one million bits per second (Mbps), using one bit for each symbol. With a modulation index of around 0.5, it is quite near to the ideal MSK. Since the modulation index may range from 0.45 to 0.55, Bluetooth low energy is not considered an MSK radio, while having many of the MSK characteristics, such as a reduced side-band power output, which allows for less costly filters to be used to meet regulatory standards.

A negative frequency deviation is used to communicate a zero. A positive frequency deviation is utilised to communicate a one. About 180 kHz is the minimal frequency variation. This implies that if a central frequency of 2402MHz is utilised, a transmission at 2401.820MHz would represent a zero, while a transmission at 2402.180MHz would indicate a one.

Modulation

Radio Stations

It is usually advisable to utilise more than one frequency to convey information if you want to construct the most durable, longest-range system. Some systems do this by using extremely broad transmissions; 802.11, for instance, sends data very fast using channels that are either 20MHz or 40MHz wide. While other systems achieve this through frequency hopping, Bluetooth Classic employs 79 constrained channels, all of which are utilised while sending data. Except for the fact that numerous narrow channels will be able to navigate through intricate multi-path settings that are continually changing far more effectively than fewer wide-band broadcasts, the decision is mostly arbitrary.

In other words, 2402 is the lowest frequency utilised in Bluetooth low energy, whereas 2480 is the maximum. A Bluetooth low energy channel and anything utilising the frequency band below it at the bottom of the spectrum is separated by 2MHz. A Bluetooth low energy channel and anything utilising the frequency band above it at the top of the spectrum is separated by 3.5MHz. Transmit Power A device may only utilise a certain amount of transmit power in the 2.4GHz ISM band to comply with license-free rules. The standard for Bluetooth low energy sets a maximum to transmit power of +10dBm. Devices cannot be made so quiet that no other devices can hear them, as required by the LE standard, which also stipulates that there must be a minimum transmit power of -20dBm. A transmit power of +10dBm indicates that it would transmit at 10mW, while a transmit power of -20dBm indicates that it would broadcast at merely 10W.

Tolerance

Every gadget is made with a certain tolerance. Generally speaking, the gadgets cost more the more exact the tolerance. The frequency accuracy is the main tolerance that may be set for radios. A radio may be transmitting at 2401.850MHz or 2402.150MHz even if its allocated frequency is

2402MHz. When sending a packet, this is the tolerance in the central frequency. The centre frequency tolerance for a Bluetooth low-energy packet is 150 kHz. Because it is often calculated by multiplying the frequency from a known frequency crystal, the centre frequency may be inaccurate. This crystal's natural frequency is 16 MHz, thus to increase it to 2 400 MHz, it must be multiplied by a factor of over 150. Any crystal-related errors would multiply and be incorporated into the transmission frequencies. The central frequency would be off by around 150 kHz, for instance, if the crystal was producing 16.0001MHz. The error rate for this crystal would be 62 parts per million (ppm). Usually, easily accessible low-cost, high-volume crystals with an error rate of around 50 ppm are available.

How far the radio deviates from its core frequency throughout the packet is another crucial metric. The heat that accumulates in a silicon chip as it is being utilised is what causes this drift. The internal radio frequencies will shift a little bit when the heat rises. During a packet, a Bluetooth low-energy radio cannot wander by more than 50 kHz. As a result, the radio would need to be between 2401.950MHz and 2402.050MHz after a packet if it had begun broadcasting flawlessly at 2402.000MHz at the beginning. Additionally, there is a 400Hz/s maximum drift rate.

Receiver Sensitivity

There is essentially just one factor that counts when creating a receiver: how effective is it? The receiver sensitivity, or how sensitive the radio is to pick up wireless communications from other devices, is used to quantify this. This is often a relatively tiny quantity and is measured in dBm. For Bluetooth low energy, a receiver sensitivity of -70dBm is necessary. In other words, for it to function, it has to be able to detect 0.0000001 mW of electromagnetic radiation. But there will always be noise. If you can't interpret a signal, it serves no use to be able to detect it. As a result, in reality, the sensitivity threshold is set at a level that allows for an acceptable bit error rate while decoding a signal (BER). This value of 0.1 percent BER has been selected for Bluetooth low energy. The majority of Bluetooth low energy controllers have a receiver sensitivity of around -90dBm, or 1pW. The amount of energy that can be separated from the band's background noise is extremely little, yet as will be shown in the next section, this results in some astonishing ranges.

Range

The connection budget of the system must be established to compute the Bluetooth low-energy radio's range. The power from the transmitter in a silicon chip is used by a variety of components that make up the link budget before it is received by a peer silicon chip. These components include the antenna and the gains and losses of the matching circuit. However, if the antenna and matching circuits are assumed to be insignificant, the route loss is what mostly affects the connection budget. Path loss is a measurement of how much the radio signal's strength has dropped between the transmitter's antenna and the receiver's antenna. Equation 5-3 calculates the necessary route loss for a certain distance.

The range is 2.5 metres when the transmit power is -20 dBm and the receiver sensitivity is -70 dBm, which results in a path loss of 50 dB. This is the maximum distance achievable while using the least amount of transmit power and the least amount of receiver sensitivity. The range is 40 metres when the transmit power is 0 dBm and the receiver sensitivity is -80 dBm, resulting in a

path loss of 80 dB. When a moderate transmit power and a moderate receiver sensitivity are employed, this distance is achievable. The range is 250 metres when the transmit power is 10 dBm, the receiver sensitivity is -90 dBm, and there is a 100 dB path loss. This is the greatest distance that can be achieved with the highest transmit power and the highest receiver sensitivity offered by contemporary processors.

Direct Test Mode

Knowledge must be gained by experience; the only non-fanciful test is trial and error. How to calibrate wireless systems and carry out certification and product line testing of their performance, particularly those that are intended for the lowest feasible cost of manufacturing, is one of the largest issues. This is particularly true if the device has been integrated into another module or product and there is no way to bypass the host stack for a brief period of testing at the beginning of the device's life. To drive this protocol even after a host stack and other components of the device have been included in the device, Direct Test Mode defines standard testing techniques and a hardware interface.

Shows the test setup

The controller, module, or finished product being tested is referred to as the DUT. An antenna and a Universal Asynchronous Receiver Transmitter (UART) or Host/Controller Interface (HCI) to the UT are required on the device. The UT can connect with and operate the LT and comes with software to drive the device under test via the UART or HCI interface. It is normally produced by a test-equipment company. The LT can send and receive packets, therefore it can successfully communicate with the equipment being tested by using its antenna.

The UT instructs the device under test to emit or receive packets. The LT is simultaneously instructed by the UT to perform the opposite, which is to either receive or send packets. This indicates that packets will be sent to or received from the LT by the testing device. The UT may utilise the information from both devices, such as a packet count from the device being tested or more detailed information from the lower tester, to decide if the DUT passed the tests after the test.

By instructing the device to emit packets at a specified frequency and detecting the actual frequency that the controller is transmitting, the UT may also calibrate the controller on a manufacturing line. This is usually necessary if the external crystals used as a timing reference are not precisely operating at their design frequency. While the controller is sending packets, this crystal trimming would take place, enabling extremely quick component calibration.

CHAPTER 5

LOW-COST POWER BLUETOOTH DESIGN

Anil Agarwal, Associate Professor,
Department of Electronics and Communication, School of Engineering & Technology, Jaipur National University, Jaipur, India
Email Id- anil.agarwal@jnujaipur.ac.in

Band ISM

It is not a good idea to create or utilise wireless equipment in the 2.4GHz ISM band. It has weak propagation properties, and the radio energy is easily absorbed. All are absorbed, but mainly water; take into account also that the natural body consists mostly of water. These relatively large drawbacks are more than offset by the availability of the rf spectrum everywhere and the absence of any prerequisites for a license. Naturally, the presence of this Free Rent sign indicates that other technologies, such as the majority of Wi-Fi radios, will also utilise this area. But the absence does not imply that anything is permissible. There are still a lot of regulations, especially concerning restricting the peak power of spectrum-using devices, limiting the range Nevertheless, these restrictions are still more appealing than purchasing licensed spectrum at great expense. Consequently, picking the ISM band reduces the price.

Many individuals think that Bluetooth and infrared radiation are interchangeable terms. This is not true at all, however. Radio waves are used as the connecting technique. Among both 2.4 and 2.485, GHz is the frequency of these radio waves. The frequency of a WiFi connection is comparable. Typically, you connect your phone to a speaker or a set of wireless headphones to play music wirelessly. If you've had a MacBook and linked it to a trackball mouse or wireless desk, that is another typical use.

Since the technology's introduction, many new gadgets continue to utilise an older version of Wireless headphones, which continue to operate without any issues at all. Bluetooth requires extremely little battery since the technology only needs 0.001-0.5 watts. As a result, you don't have to be concerned about the battery life of your Bluetooth device. This same Nobel Special Interest Group, or Bluetooth SIG, is advancing Bluetooth technology, although Ericsson was the true innovator and the first to commercialize it. Though there are now other participants in the group, Siemens has always had a lot of impact on recruits.

IP Permit

Nokia may have used the Wibree technology to any wireless standard when it was developed enough to be included in an existing wireless standards body such as a team. It might have, for instance, brought it to the Wi-Fi Partnership, which establishes technological standards for the 2.4GHz ISM band as well. But they decided due to its stellar reputation, the Headset Special Interest Group (SIG) and licensing practices that this company employs. These regulations essentially imply that for a Bluetooth device, the expenses associated with patent licensing are greatly reduced in comparison to a technology created by a different SIG or organization it uses

FRAND4 as its policy. Due to Bluetooth's very affordable licencing fees, additionally, the budget per device is greatly decreased.

Small Power

Keeping the materials needed to make a gadget as cheap as possible is the ideal method to design one create such a gadget using components like batteries. The more the battery size, the battery housing must be bigger, which raises the price once again. Changing a Not simply the customer who has to buy a new battery incurs expense battery but replacing also takes into account the lost opportunities gadget in hand. If a third company is responsible for maintaining this equipment, possibly because

There are added labor charges since it is a component of a controlled home alarm system replace this battery, please. Consequently, low-power technologies should be designed Consumption lowers expenses as well. What if we did a thought experiment and larger batteries might be used in many gadgets? Using a computer keyboard readily rely on AA batteries, but the makers prefer to utilise AAA batteries instead and it's not because they are larger but also because using them lowers the cost of materials and hence the device's price.

Transceiver Evaluation

The controller may be instructed to send or receive packets to test the transceiver. The tester specifies the duration during which the controller sends packets. The tester then attempts to receive these packets and analyses the received packets to extract different Physical Layer attributes. For instance, the LT can gauge the test device's transmission-related frequency drift. The transceiver waits for a different amount of time after each packet it receives, whose period is chosen by the tester. The DUT only counts the packets that were properly received after the tester transmits a certain number of packets. When the test is over, the UT may get this information. This allows the tester to evaluate the receiver's performance across a specified number of sent packets.

Test Packet Format

The access address is the bit-inverse of the access address used for advertising. This access address is not a protected property like the primary advertising access address since it is only ever used during testing and can never be utilised in a product during a regular connection. The advertising packet header format is used for the test packet format, therefore the test packet type is represented by the first four bits, and all other bits are set to zero. The test packet types include PRBS9, a 9-bit pseudo-random bit sequence, "11110000," "10101010," PRBS15, a 15-bit pseudo-random bit sequence, "00001111," and "01010101." The qualification test specification only uses the first three packet types. These tests, however, could be beneficial for testing manufacturing lines. All promotional envelopes have whitening turned off. To accurately estimate the frequency deviation for the nonrandom bit sequences, this must be done.

Transmitter tests

The results of the transmitter tests show how precisely the DUT's transmitter is operating. The transmitter tests may be used to find frequency drift, frequency deviation, and other required radio

characteristics. The UT sends an instruction to the DUT to initiate the transmitter test. Three arguments in this command specify the test that will be run. The frequency that will be examined is the first parameter. This is the radio channel number that Equation 5-2 specifies. The packet payload length is the following parameter; acceptable ranges are 0 to 37 bytes, inclusive. The kind of data that will be transferred is the last parameter.

Three different sorts of data may be sent in test packets:

1. PRBS9 is used for evaluating power transmission
2. 11110000 applied for measuring frequency deviation
3. Initial frequency and carrier testing is done using 10101010

A pseudo-random bit sequence using a repeating 9-bit sequence is called the PRBS9 packet sequence. It is produced using a shift register with linear feedback. The PRBS9 pattern is often used as a test pattern to evaluate the radio's functioning as rapidly as feasible. The PRBS9 sequence is mostly used because it closely matches the randomness of whitened packets used in connections. As a result, it may be used for both receiver sensitivity testing and studies of power transfer.

The repeating 11110000 packet sequence consists of four ones and four zeros. To measure the frequency variation, the same bit is constantly sent before switching to the other bit. A frequency variation of more than 225 kHz will be detected by the tester. This demonstrates the need of disabling the whitening in test mode for the radio to broadcast repeated bits. A one and a zero are repeated in the packet sequence 10101010. This is done to measure frequency variation during alternative bit transmission. This is used for carrier testing as well as for measuring the initial transmission frequencies.

The LT receives sufficient packets to provide an appropriate result, at which point the UT instructs the DUT to terminate the test. To signal completion, the DUT instantly stops transmitting and returns an event. The UT may then begin a new test utilising a different packet type, packet length, frequency, or a combination of the three.

Receiver Tests

The transmitter test is substantially more complicated than the reception exam. The bit error rate at different transmit power levels is calculated using receiver testing. To eliminate the uncertainty associated with the route loss over the air, the LT must broadcast packets to the DUT at a specified transmit power, often utilising a conducted antenna connection. After the test, the DUT counts the number of packets that were successfully received and communicates this information to the UT. The bit error rate of the receiver at the specified signal intensity may therefore be estimated by the UT by counting the resultant number of packet errors.

The UT sends an instruction to the DUT to initiate the receiver test. There is simply one argument in the command. As with transmitter testing, the parameter is the frequency that will be tested or the radio channel number. The DUT begins receiving after receiving this instruction and resets a packet counter to zero. The packet counter is increased when a valid packet is received by the receiver. Until the UT instructs the DUT to end the test, the DUT keeps increasing. The DUT instantly stops receiving after receiving the stop instruction and returns an event to signal that it

has done so. The number of valid packets the DUT received is included in the event. Based on the number of packets delivered and received, the UT may then calculate the receiver's performance. The UT may then begin a new test, perhaps with a different transmit power or frequency.

Hardware Interface

A standardised hardware interface that may be utilised by UTs is developed to allow extremely effective and device-independent testing of modules, perhaps from various controller manufacturers. The hardware interface consists of a straightforward two-wire UART with one line for bits to be sent from the UT to the DUT and another line for bits to be transmitted from the DUT to the UT. Two pins or pads that may be linked to the UT on a manufacturing line are anticipated to be exposed by module and end-product makers to enable calibration and certification of a specific product's capacity for transmission and reception. UART UARTs may normally be set up in a variety of ways. The Direct Test Mode UART only has one customizable parameter: baud rate, which helps to mitigate the issue of an incompatible interface. The DUT's baud rate may be any of the following numbers: 1200, 2400, 9600, 14400, 19200, \s38400, 57600, or 115200. The average rate is 38400 because it strikes a reasonable balance between cost-effective implementation and efficient command and event transmission.

The remaining UART settings are very typical. With no parity and only one stop bit, each byte is 8 bits long. Neither software nor hardware controls the flow of data. Since there are no hardware flow control connections, it is obvious that hardware flow control is not conceivable. A command or event may only be up to 2 bytes in length, and only one command can be transmitted at a time, therefore software flow control is not necessary. The common ground serves as the ultimate design criterion. It is presumed that the two devices share a common basis. A command or event must be issued with a minimum of 2 bytes and a maximum of 5 milliseconds between them. A command must return an event within 50 milliseconds after being delivered. This guarantees that the DUT begins and ends testing on time, making it simple to evaluate the accuracy of any counting.

Commands and Events

1. The UT may issue the following four instructions to the DUT:
2. Transmitter test, receiver test, reset, and test conclusion
3. There are two events that the DUT may send to the UT:
    a. Packet Reporting
    b. Test Status

The reset command has no side effects. The controller is terminated wherever it is; if the controller was sending or receiving test packets, it must halt. Additionally, it puts the controller in a well-known condition. To ensure that everything has returned to its quiescent state, it quickly returns a test status event. The reset command uses just two bits; the other bits are entirely ignored. Transmitter testing is initiated by the transmitter test command. This entails the three characteristics of frequency, length, and packet type. The DUT sends a test status event and begins delivering packets using the parameters as soon as this instruction is received. The DUT still returns the test status event but changes the status bit to indicate an error if the device cannot

transmit packets for whatever reason. The UT would need to reset the DUT to return it to a known good condition if the transmitter test command failed to initiate.

The bit structure of the Direct Test Mode command

The receiver test is initiated by the receiver test command. The frequency parameter is included in this; all other bits in this packet are disregarded. The test status event is returned upon receipt of this instruction, and the DUT begins to accept packets on the chosen frequency. Once again, the test status event is delivered with the status bit indicating an error if the device is unable to receive any packets.

The test may be terminated by the UT by sending the test end command after a successful transmitter or receiver test. The DUT instantly pauses what it is doing and returns the packet reporting event after receiving this instruction. This packet-reporting event contains the packet count for the successfully received packets if the receiver test was being performed. This field in the event is set to 0 if the transmitter test is running since no packets are received during the transmitter test. The UT may launch a new test with a new test command after receiving the packet-reporting event.

Direct Test Mode events reporting bit structure

One bit alone is defined by the test status event. The test command was successful if this bit is set to zero. The test command failed for some reason if this bit is set to one. Two things may go wrong with a test command:

1. The controller is in a situation that confuses it, therefore the UT's best course of action is probably to reset.
2. The controller is unable to perform the test, most likely due to the absence of a transmitter or receiver. However, a transmitter-only device may have its receiver tested.
3. A UT may only attempt each test and ascertain the status; it is not feasible to tell from this command set if the DUT supports a particular test.

The completion of the test is reported via the packet-reporting event. It contains a single, 15-bit packet count that indicates how many packets were successfully received. For transmitter testing, this packet count will always be 0, however, for reception tests, it may be any number between 0 and 32,767. The packet counter overflow is not a concern for the DUT because, even if 32,769 packets were received, a test that may have taken a very long time may reveal that only one packet had been received. Therefore, UTs should only execute tests for a length of time that prevents the packet counter from overflowing.

Direct testing using HCI

To exercise direct test mode on a controller, it is also feasible to reuse the current HCI transports and logical interface. This does need extra infrastructure, particularly if the host interface is intricate. This is a very viable strategy, nevertheless, for controllers that are being evaluated individually rather than as part of a highly optimised module design or final product. The only

difference between the test processes is that complete HCI instructions and events are transmitted, rather than 2-octet commands and events. The mapping between HCI commands and events and Direct Test Mode commands and events.

Direct Test Mode to HCI

When utilising HCI, there isn't a separate test status event or packet reporting event since the Command Complete already handles these tasks, has an opcode to identify the command that initiated it and has different arguments based on that command. With all this connectivity technology, the only thing we know more about is how invisible we are in the big picture.

Sherry Donahue

The Link Layer specifies how two devices may communicate with one another over the radio using this technology. This involves specifying a packet's specifics as well as the advertising and data channels. Additionally, it specifies how to find other devices, broadcast data, establish connections, control those connections, and eventually transfer data inside such connections. The difficulties of 2.4GHz ISM band wireless communication systems, such as interference, noise, and deep fades, make this worse.

The Link Layer State Machine

It's crucial to comprehend the fundamentals of the Link Layer state machine and how they affect Bluetooth low-energy architecture before talking about packets and how they're utilised. It should be remembered, nevertheless, that there are two substates of the scanning state: active scanning and passive scanning. The two substates of the connection state are master and slave. The separation of the broadcast, discovery and connection processes from the data transferred in a connection is another key design choice that Bluetooth low energy executed. The Link Layer state machine illustrates how devices may find and connect. This design was made in part to accommodate the advertising devices' very low power requirements. You may retain robustness while using less power by limiting the number of advertising channels to three. However, doing so calls for distinct advertising states and advertising packages. Three states in which advertising packets are transmitted or received and one state in which data packets are delivered and received make up the Link Layer state machine.

The Standby State

Link Layers begin in the standby state when they are switched on and stay there until the host layers instruct them to move out of it. You may transition from the standby state to the starting, scanning, or advertising phases. Every other state may likewise be transitioned into the standby state. The most significant, but inactive, the condition is the standby state, which serves as the centre.

The State of Advertising

The Link Layer may send advertising packets while it is in the advertising state. It can also deliver scan responses in response to scan requests from devices that are already scanning. If a device

wishes to be discoverable or connectable, it must be in the advertising state. If a device wishes to broadcast data to other nearby devices, the advertising state is also necessary.

The advertising state

A gadget needs a transmitter to advertise, but it might also include a receiver. The expense of the receiver on that chip might be avoided by creating a device that just supports the advertising state. It should be noted that in reality, a dual-purpose chip with a transmitter and a receiver may wind up costing less per unit than a low-volume transmit-only semiconductor. Stopping advertising enables the transition from the advertising state to the standby state. Additionally, if an initiating device sends this advertiser a to connect request packet, the advertiser may transition from the advertising state into the connection state.

The Scanning State

A device will take in packets from the advertising channel when it is in scanning mode. Simply listening to local advertisements for gadgets might be done with this. Passive scanning and active scanning are the two distinct substates that makeup scanning. Only advertisement packets are received via passive scanning. To acquire extra scan response data, active scanning also sends scan requests to advertising devices. Only the scanning state may be exited to enter the standby state. Scanning is stopped to do this.

Passive Scanning

Passive scanning involves no transmissions from the device; it just passively scans. Thus, a device with merely a receiver may accomplish passive scanning. Because a transmitter is not required, you may decrease the size and price of the controller by merely enabling passive scanning. However, as was already said, depending on how much you're creating, multipurpose devices may wind up being less expensive due to bulk pricing.

Active Scanning

In active scanning, a scan request is made to the advertising device whenever a new device is found by the Link Layer, and a scan response is anticipated in return. These scan packet requests and responses are both sent via the advertising channel. Since the scan response contains two extra packets that must be communicated or received to obtain the data, active scanning requires that the data in the scan response be mainly static to function effectively. However, as this will constantly be received, the information in the original advertising package may alter often.

The initiating state

The Link Layer must first be initiated to start a connection with another device. The receiver is used to search for the device that the initiator is trying to connect to while it is in the initiating stage. The Link Layer will submit a connect request to the advertiser and enter the connection state if it receives an advertisement packet from this device, assuming that the advertiser does the same. By ceasing to initiate a connection, it is also possible to exit the initiating state and return to the standby state.

The Connection State

The connection state is the last state of the Link Layer state machine. Either the beginning state or the advertising state may be used to access this. An initiating device sending a connect request packet to an advertising device is what initiates both of these transitions. Once again, there are only two substates: master or slave. Data channel packets are delivered and received between the two devices during the connection state. All other states utilise the advertising channels; only this state uses the data channels. Only entering the standby state allows one to exit the connection state. Cutting off the link, this is accomplished.

The Master Substrate

Only the initiating state may be used to access the master connection substrate. The connection to the peer device must be started by the device that becomes a master. A device that is the master must send packets to the slave regularly. The slave is then given the chance to respond and submit their information.

The Slave Substrate

Only the advertising state may be used to access the slave connection substrate. A slave device must have been reaching out to the peer device with advertising. When a device is a slave, it is unable to communicate anything until it properly receives a packet from the master. The slave may send their packet after receiving one from their master. The slave must wait for the master to deliver another packet of data back to it before sending new data. Slaves have the option to save power at any moment by ignoring the master. The slave device may save a lot of power by doing by remaining "asleep."

Multiple State Machines

It is possible to have numerous state machines in a Link Layer implementation; each state machine is distinct. A device may, for instance, simultaneously be a slave, advertise, and actively scan using this arrangement. Alternately, you might set up a device to simultaneously advertise, passively scan, establish a connection, and act as a master. Additionally, the device may simultaneously support several master connections to slaves. Recognize that various deployment-related limits are crucial to comprehend.

Not Master and Slave

A device cannot be a master and a slave at the same time in the Link Layer since it is an "autocracy". Similarly to this, a gadget cannot simultaneously be a master and a slave. According to this, a master device cannot promote using a connectable advertising packet but may instead do so using a no-connectable or discoverable advertising packet. Therefore, a device that is currently a slave cannot establish a connection to another device since doing so might make it the master of that device. By placing this constraint, it will be impossible for the device to operate in a nondeterministic manner at any moment.

Due to the Link Layer's deterministic nature, extremely effective scheduling techniques may be used to construct Bluetooth low-energy devices. The scheduling algorithms for any

nondeterministic system that must maintain synchronisation across various time domains must be quite sophisticated. Because they are non-deterministic, these algorithms will also have a substantial processing demand when implemented on general-purpose CPUs. The low-power objectives of the technology are not met by this. As a result, the deterministic architecture enables the use of discrete logic to create very efficient algorithms.

A gadget also cannot serve two masters simultaneously. A device cannot advertise using a connectable advertising packet if it is a slave. Having two masters as a slave is more difficult than having just one master. This is what Bluetooth Classic would refer to as scattered. Scatternets are not supported by Bluetooth low energy.

Packets

The basic unit of the Link Layer is the packet. A packet is just a designated data chunk that is sent from one device and received by one or more other devices. The label identifies the sending device as well as the receiver device. The prelude, a brief training sequence, appears at the beginning of the packet. The receiver uses the access address that follows to differentiate this packet from nearby radio noise. Header and length bytes come after the access address. The packet's payload comes next, followed by a cyclic redundancy check (CRC), which makes sure the payload is accurate.

Packet structure

Data packets and advertising

Advertising and data packets are the two kinds of packets used in Bluetooth low energy. There are two quite distinct uses for these packets. Advertising packets are used by devices to locate and connect to other devices. After establishing a connection, data packets are utilised. Advertising packets vary from data packets in that they are transmitted by a single device and may either be broadcast to all listening devices or targeted to a particular device. Only two devices, known as the master and slave devices, can interpret data packets. The channel on which a packet is sent determines whether it is a data packet or an advertisement packet. 37 data channels and 3 advertising channels are available. A packet is an advertisement if it is broadcast on one of the three advertising channels; otherwise, it is a data packet.

Bleaching

Receivers that use frequency-shift keying (FSK) are intriguing since they can't take in extremely lengthy streams of identical bits. For instance, if you send a series of bits like "000000000000," the receiver would think that the transmitter's centre frequency has shifted to the left and will no longer be able to maintain frequency lock. It subsequently fails to receive the subsequent "1" bit and the remainder of the packet. A whitener is employed to randomise the sent packets to guard against this. Typically, a whitener is a very brief random number generator that produces zeros and ones for a particular packet in a predefined sequence. The original bits may then be retrieved by a receiver using the same random number generator. The original data is mixed with the random number whitener using an exclusive operation to maintain the original information in the output sequence.

A string of identical bits in the original information will be changed into a sequence that is highly randomised by utilising a random whitener along with the original information in the packet. The receiver is less likely to lose frequency lock as a result. Any further randomization won't do any harm if the lengthy string of information bits was already random. Analogous to the one used to compute a CRC, a linear feedback shift register is utilised to create the whitening random number sequence.

The high bit in the shift register is first set to the Link Layer channel number that will be used to send this packet. This implies that even if a packet is whitened on one channel and the receiver loses the lock as a result, the packet will utilise a different whitening sequence when it is delivered on a separate channel and the receiver will be able to receive it. Although it happens extremely seldom, the whitener makes recovery possible.

The whitener is used to prevent the transmission of lengthy zero- or one-number sequences. In this illustration, transmission on channel 23 results in the first three bytes of data being whitened. This indicates that the binary pattern: input = 00000000: 00100000: 00010000 Output equals 11110101: 01100010: 11001110 when whitener = 11110101: 01000010: 11011110 is used. With the whitener, the data became randomised, as you can see. Long single-digit sequences may be found in the original input data: 10, 1, 8, 1, 4. These lengthy sequences are absent from the output data, and more importantly, the largest single-digit sequence has only four digits: 4,1,1,1,1,1,2,3,1,2,2,3,1.

Packet Structure

The packet structure is made up of several fields. The next subsections go into great depth on each of these areas. The sequence in which these bytes are sent as well as the bits included in these bytes must be examined since certain fields comprise multiple byte fields.

Bit Order and Bytes

Despite being transferred bit by bit, packets are made up of bytes of data. These bytes of data are transferred with the least important bit at the beginning. As a result, 0x01 is communicated as 10000000, whereas 0x80 is relayed as 00000001. The least significant octet is often sent first in multiple-byte fields. Consequently, the following would be broadcast in place of the value 0x010203: 110000000010000001000000

The Preamble

There are two possible sequences for the first eight bits of a packet that are transmitted: 01010101 or 10101010. A receiver may establish its automatic gain control and choose the frequencies utilised for the zero and one bits using this extremely basic alternating sequence. Due to the potential variety of input signal levels that a chip must be able to handle, this sequence is crucial. The radio must be capable of handling a signal at the antenna that is as strong as -90dBm and as weak as -10dBm. This dynamic range measures 80 dBm. This implies that a receiver may pick up a packet with either a power of 1pW or a power of 0.1mW. To get the signal into a range where the controller can readily operate, an automated gain control would need to detect the input power level and modify its gain.

The first bit of the sent access address serves as the determinant of whether the preamble is 01010101 or 10101010. If the access address's first bit is a "0," the 01010101 sequences are applied. The 10101010 sequences is utilised if the access address's first bit is "1". This ensures that either 101010101 or 010101010 alternately make up the first nine bits of a packet. The access address is contained in the next 32 bits of a packet. This comes in two varieties:

1. Access address for advertising
2. Address for data access

When broadcasting data, advertising, scanning, or opening connections, the advertising access address is used. After two devices have established a connection, the data access address is utilised in the connection. A controller always knows which access address it will be getting when it wishes to accept a packet. The receiver will begin to receive data as soon as it is switched on and adjusted to the proper frequency. The radio will pick up background radiation even if no other equipment is nearby broadcasting at this moment. The likelihood of getting a sequence of bits that fits the preamble is rather high, generally occurring once every several minutes for a low-energy device with its receiver always open, given simple odds of receiving pure random noise. As a result, the access address is used to lessen the likelihood that a pseudo-packet may be received due to random noise.

The Link Layer must maintain a copy of all possible bits received over the previous 40 seconds and check each time a new bit is shifted into this register to see if the sequence of bits now matches the anticipated preamble and access address. Because the Link Layer is unaware of the timing of the other device's packet transmissions, it must maintain a copy of all possible bits that have been received. The correlation of the access address is the name given to this process.

The access address for commercial channels is a set number: 0x8E89BED6. This is conveyed in binary as follows, going from left to right: 01101011011111011001000101110001 accordingly, the prologue for an advertising package would be 01010101. The high correlation properties of this value led to its selection. The fixed value allows any Bluetooth low-energy device, even one that has never before received a packet from this specific device, to correlate against this access address and know it is receiving an advertising packet. On every connection between two devices for data channels, a different random number serves as the access address. However, to maintain the access address's good whiteness, this random number must abide by several rules.

Radio transmissions must be whitened to make receiver construction as simple as possible, as on whitening. The simplest rule is that no one place in the access address may have more than six ones or zeros. Additionally, the packet must vary by at least 1 bit from the advertising access address. Each octet of the access address must be unique; it cannot include any repeated patterns. Stopping the usage of an alternating bit sequence, there should be no more than 24-bit transitions. Final but not least, the last 6 bits must have at least 2-bit transitions to guarantee that there are bit transitions shortly before the header begins, just in case the header lengthens to a lengthy series of bits.

It can be shown that there are around 231 potential uniquely valid random access addresses given the aforementioned requirements. Or, to put it another way, two billion Bluetooth low-energy

devices might potentially be in communication with one another at once. That was perhaps a little bit of design overkill, but keep in mind that Bluetooth low energy was created with success in mind. The fact that an attacker cannot identify the two devices in a connection just by getting this access address is another advantageous property of this random access address for data channels. This guarantees that devices' connections are private.

Header

The header is the next component of a packet. Whether the packet is a data packet or an advertisement determines what the header contains. The header for the advertising packet contains the kind of advertising packet as well as a few flag bits that indicate whether the packet contains random or public addresses. There are seven main kinds of advertising packets, each with a unique payload structure and behaviour: General advertising indicator codes include ADV IND, ADV DIRECT IND, ADV NONCONN IND, ADV SCAN IND, SCAN REQ, SCAN RSP, and CONNECT REQ. SCAN REQ stands for active scanning request, SCAN RSP for active scanning response, and CONNECT REQ stands for the connection request.

Encryption

Data in the payload may be encrypted when connected. The data's secrecy may be protected against attackers using this encryption. For anything to be confidential, a third party cannot. Since the "attacker" lacks the shared secret required to encrypt the connection, they cannot read the communications, decrypt them, or read their original contents. A message integrity check value that makes sure the data is authenticated is also included in encrypted packets. When data is encrypted and a common secret is used to calculate a signature, authentication implies that the sender's legitimacy can be verified. This prevents a third party from altering any of the packet's bits. By using authentication, a message's recipient may be sure that the data packet it just received came from a reliable source. A typical example of authentication is the use of a personal identification number (PIN) to confirm the identity of bank card users; the PIN ensures that the correct individual is using the bank card.

A packet counter is also included in encrypted packets to thwart replay attacks. An assault known as a replay occurs when an attacker intercepts a communication and repeats it later in the hopes of eliciting a reaction. Without replay attack defence, it could be conceivable, for instance, to search for a device that is transmitting a large number of packets, replay those packets, and see the results. The findings may be "interesting" if the receiving device was a sewage valve near a municipal park. Defending against replay assaults is crucial.

The Advanced Encryption System (AES) serves as the foundation for all Bluetooth low-energy encryption and authentication (AES). This encryption engine was first created as a part of a US government initiative to establish a future-proof encryption engine. Since then, other wired and wireless standards have embraced it, and so far, it has withstood efforts by security experts to uncover holes in its algorithm. Depending on the size of the blocks of data and keys that it can process at any one moment, AES may be constructed in a variety of ways. The 128-bit key size and 128-bit data blocks are utilised in Bluetooth low energy. As a result, all keys are 128 bits long, and a maximum of 16 octets of encrypted data may be generated simultaneously.

The AES encryption block just requires two inputs and produces one output, therefore it is quite straightforward. A 128-bit block of encrypted data is produced from two inputs: a 128-bit block of plain-text data and a 128-bit key value. The two inputs are designated as key and plain-text data because the encryption block must first process the key before using it, but the encryption block processes the plain text right away. In contrast to using a new key for each block, it is more efficient to create the key once and then send in many plain-text blocks to swiftly encrypt them. Thus, the function may be used to describe the encryption of some data, plaintext, with a key, key, using an algorithm, E, to create encrypted text, ciphertext:

The AES encryption engine serves four primary purposes in Bluetooth low energy:

1. Generating private addresses;
2. Calculating a message integrity check value;
3. Encrypting payload data

The Security Manager specifies data signing, and the Generic Access Profile specifies the creation of private addresses.

Encrypting Payload Data

The payload data is divided into 16-byte blocks for encryption, and a cypher bit stream is created for each block. The plain text is then Exclusive-Or'ed with the encrypted bit stream. The IETF RFC 3610 standard is used to describe this. The Counter with Cipher Block Chaining-Message Authentication Code Mode, or CCM, technique for encryption and authentication is described in this standard. This is a typical encryption technique for keys and messages of any size. The blocks used for Ax encryption in Bluetooth low energy. These are known-format initialised plain-text blocks that include a nonce made up of a packet counter, a direction bit, and an initialization vector (IV). The || symbol in the equation that follows denotes concatenation.

Packet counter || direction || nonce || IV

With each new non-empty packet that is sent, the packet counter, a 39-bit number, is increased. When encryption is enabled, the packet counter always begins at 0. Since empty packets are not encrypted, the packet counter does not need to be increased. Each device in the encrypted connection contributes 32 bits to the initialization vector, which is a random value of 64 bits in length. The nonce is 13 bytes long as a result. The Ax's other octets adhere to CCM specifications. The flags field in the first octet is always set to 0x01 to signify that this is an Ax block. The block counter is in the last two octets. When the first 16 octets of the payload are encrypted (CBlock1), the block counter is set to 0x0001, and when the second 11 octets of the payload are encrypted (CBlock2), it is set to 0x0002. The block counter is set to 0x0000 to use the CCM standard to encrypt the message integrity check value (MMIC).

0x01 | nonce | 0x0000 | CMIC = Ekey

Ekey = CBlock1 (0x01 | nonce | 0x0001)

Ekey = CBlock2 (0x01 | nonce | 0x0002)

The message's different components are then Exclusive-Or'd with the cypher blocks to create the encrypted payload.

Encoded is equal to CBlock1 Block1 || CBlock2 Block2 || CMIC MIC.

The peer device receives this encrypted payload next. The peer device can decode the message using the same packet counter, direction, and IV values since it is aware of the shared secret the key value. The CRC is computed over the encrypted payload and not the original payload blocks when the encrypted payload is transmitted. The data packet's header and length fields are never encrypted.

Message Integrity Check

The message integrity check (MIC) value, which is put between the data and the CRC, is used to authenticate the sender of the data packet. This guarantees that the peer device and not an outside attacker transmitted the encrypted message. The AES encryption engine is again employed to compute the MIC. This time, the blocks are chained together such that each bit in the original message is treated equally for determining the MIC. The output of one block is utilised as the input to the next block.

The format of encrypted data

The identical nonce that is used to encrypt the payload is also used to compute the MIC. It uses three or four B blocks. The nonce and the initial length of the data requiring authentication are both included in the first B0 block. Even though the largest payload that may be validated in low energy is just 27 octets, this length field contains 16 bits.

B0 = 0x49 || length || nonce

Additional information that should be authenticated with the payload but is not included in the payload is found in the following B1 block. This is used in Bluetooth low energy to authenticate a few bits in the packet's header. The logical link identification bits are the only bits that need authentication. All other bits in the header are set to zero to facilitate computation and enable the precalculation of blocks without requiring knowledge of values for SN, NESN, and other bits that are unimportant in terms of security.

B1 = 0x0001 | headermasked | 0x00000000000000000000

The actual payload data that is being authenticated is in the next block or blocks. Octets 0 through 15 make up the payload data in block B2. The payload data from octets 16 to 26 are included in B3. These blocks are then linked together using the same key that was used to encrypt the payload to determine the MIC. The packet only uses the payload's 32 most important bits.

Ekey (B0) = X0

(X0 B1) X1 = Ekey

B2 = Payload [0, where X2 = Ekey (X1 B2) .127]

MIC = Ekey (X2 B3 [128] .96] where Payload [128.] = B3 .215]

The same MIC computation is carried out on the receiving device whenever an encrypted packet is received to ensure that the MIC value computes to the same value given the same inputs. The connection is promptly severed if the value does not calculate properly. The peer device will stop communicating altogether and finally enter supervisory timeout. This may seem like a pretty severe approach at first, but if the CRC fails, the packet will simply be discarded and retransmitted by the peer without the MIC being examined.

A MIC can only fail if an attacker is actively trying to attack the connection or if the CRC mistakenly allowed many bit errors, which would then result in the MIC failing. The best course of action in the first scenario is to quickly cut off the connection since it could already be infected. In the second instance, the packet's contents have already been corrupted since the CRC mistook it for a valid packet. The safest course of action is to unplug the connection and, once again, expect the worst. The two devices may swiftly reestablish communication once the connection has been lost, choose a fresh initialization vector, and re-encrypt the link. The method of re-establishing a fresh initialization vector updates the connection's encryption and, as a result, the nonce.

If necessary, a new initialization vector may also be established while in a connection. The initialization vector must be refreshed regularly since the packet counter has a finite size and the encryption is only regarded as safe if nonce values are never repeated. This won't happen very often since it would take more than 12 years of continuous packet transmission to reach anywhere close to wrapping with the maximum number of 2 39 - 1 packets that can be transferred in a connection before the nonce repeats. However, Bluetooth low energy was created with success and long-lasting connections in mind. As a result, the procedure for refreshing the nonce is previously known.

Managing Connections

1. Once two devices are connected, they may control the connection and transmit and receive data.
2. Sending Link Layer control messages is a part of connection management. Only seven Link Layer control techniques are available:
3. Updating the adaptive frequency-hopping channel map and the connection parameters
4. Link encryption, link re-encryption, feature bit exchange, version information exchange, link termination

Update of Connection Parameters

The connection parameters are supplied in the connection request packet when the connection is established. The connection settings may no longer be appropriate for the services being utilised through this connection after a connection has been active for a while. To utilise the services effectively, the connection settings must be adjusted. It is feasible to change the connection settings inside the link, as opposed to disconnecting the link and rejoining it with new connection parameters. To do this, the master uses LL CONNECTION UPDATE REQ to send a connection update request to the slave along with the updated parameters. These settings are not negotiable; the slave is required to utilise them. There is just one course of action left for the slave if it rejects the specified parameters to disconnect the relationship. A subset of the parameters from the

connection request message provided previously during connection establishment as well as an extra parameter called the instant are included in the connection update request: Figure 5.1 discloses the Bluetooth system connection.

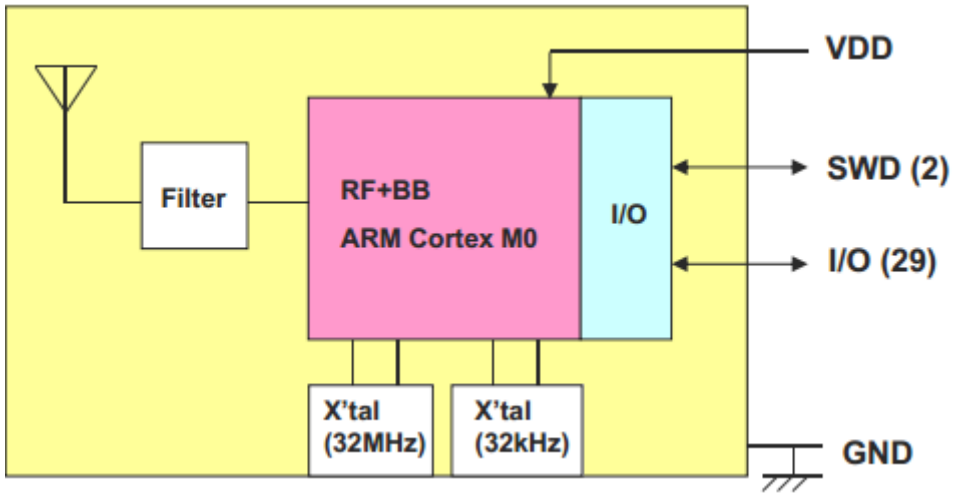

Figure 5.1: Discloses the Bluetooth system connection.

Slave latency, connection interval, transmit window size transmit window offset, supervision timeout, and instant. The parameter that controls when the connection update will begin is called the instant. The connection update is included in the message when the master transmits it by selecting a future time when it will be implemented. After receiving the message, the slave will keep track of this precise future time and wait the required amount of time before switching to the new connection settings.

Packet retransmission, one of the biggest issues with wireless networks, is aided by this. The process will be successful as long as the packet is retransmitted enough times and ultimately gets through before the instant expires. However, the connection will probably break if the packet is not delivered in time. Since Bluetooth low energy lacks a clock, counting connection events is the only method to define an instant. Since the initial connection event in the link is the one that was communicated in the first transmit window after the connection request was transmitted at time zero, each connection event is tallied individually.

Therefore, the connection event count is the moment at which the new settings will be applied. The slave should have ample time to receive this package from the master. This normally gives the master at least six tries to send the message to the slave, even at the highest latency. The instant would normally be at least three seconds away if the slave delay was 500 milliseconds. Similar to when establishing a connection, the slave waits for the instant before listening for the transmit window. This enables the master to change the timing of the slaves at both the gross and 1.25 millisecond levels. This makes it easier for the master device, which is also a Bluetooth classic device, to match the Bluetooth low energy slaves it has to do the rest of its functions. A fresh

connection interval, supervision timeout, and slave latency value are utilised when this process is finished.

Adaptive Frequency Hopping

Any radio system that operates in an open wireless band must successfully use adaptive frequency hopping. Unfortunately, certain systems are vulnerable to interference because they cannot do adaptive frequency hopping. The main issue with adaptive frequency hopping, particularly in model devices, is that it continually changes the list of channels that might be excellent or poor at any given moment. This implies that communication is required for devices to modify the channel map.

The process of updating the Channel map

A channel map request packet with the header LL CHANNEL MAP REQ contains the adaptive frequency-hopping updates. This is transmitted from the master to the slave and simply contains the two parameters listed below:

Updated channel map

Right now

The idea of the instant is the same as that of the connection update. It chooses a time when the new channel map will be applied. The new channel map is utilised immediately and moving forward for all connection events, at least until the next time the channel map is changed. Each data channel has its bit in the 37-bit channel map field. When a particular channel's bit is set to one, it is thought to be good and will be utilised; when it is set to zero, it is seen to be bad and won't be used.

Only once the instant has elapsed may the channel update request be resent. This limits how quickly the channel map of the connection may be updated. The channel map would typically only be modified if the connection wasn't functioning well with its existing set of channels or if the host found that the faulty channels were suddenly working well. A slave device cannot alter the channel map or even inform its master of the channel circumstances due to the lack of Link Layer control mechanisms.

Encryption is started

The connection has to be decrypted before encryption can begin. Both the nonce and a session key (SK) must be generated to encrypt the connection. Each device must provide 4 octets for the nonce and 8 octets for the session key for the transaction to be completed. Additionally, a second key known as the long-term key is needed (LTK). Encryption begins when the master sends the slave an encryption request message (LL ENC REQ), which establishes the shared secret. An encrypted response message (LL ENC RSP) is then sent back by the slave. The 4-byte initialization vector contribution from the master, the session key diversifier's 8 bytes, as well as some extra data that the slave sent to it when they first teamed up, are all included in the encryption request packet. The slave may utilise this additional information, which is static for a certain master, to identify the master with whom it is in communication and perhaps extrapolate the LTK for the master. The slave may not need to keep any information about bound devices if this is done. The slave's 4-byte

initialization vector contribution and 4-byte session key diversifier contribution are both included in the encryption response packet.

The slave will instantly transmit a reject signal to the master along with the reason it refused the encryption if the LTK is not present on the slave side. The slave will launch a three-way handshake to commence encryption if the LTK is available. The three-way handshake is necessary because, for instance, the slave has to be able to send an unencrypted packet to the master while also receiving an encrypted packet in return. As a result, during this handshake process, the two devices enter a completely encrypted connection in lockstep. A session key is used to encrypt the connection after encryption has begun. The LTK and the session key diversifier, which are contributions from both devices, are used to determine the session key.

An LTK may be used more than once thanks to the session key diversifier. This is accomplished by ensuring that a unique encryption key is generated from the session key each time a connection is encrypted. To make sure that even if one device is an attacker, the other device may force a different diversifier and therefore a different encryption key, the master and slave each contribute half of this diversifier. All of this is done primarily to guard against the one vulnerability of AES: A key cannot be used more than once, ever. Because of this, even though we have LTK as a shared secret, we are unable to encrypt the application data using it. We need to diversify this LTK into a session key, SK, instead. Concatenating the two session key diversifiers (SKD) SKDmaster, to which the master contributes, and SKDslave, to which the slave contributes provides the plaintext input for the AES encryption engine. The key input for the AES encryption engine is the LT K. The session key, which serves as the encryption key for the connection, is the product of the AES encryption engine.Figure 6: Discloses the block diagram of the RF radio and the Bluetooth

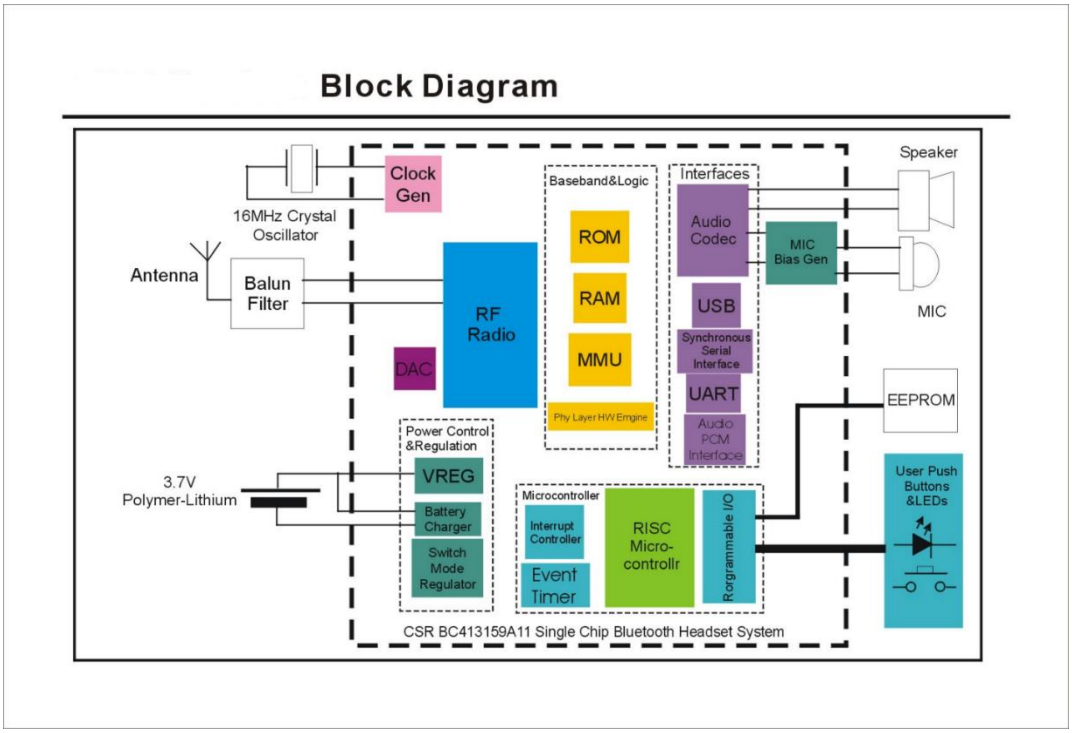

Figure 5.2: Discloses the block diagram of the RF radio and the Bluetooth.

Additionally, the initialization vector (IV) is constructed using the two values that each device supplied. The encryption process pauses all data exchanges. Before sending its encryption request packet, the master performs this action; similarly, the slave does this action before sending its encryption answer packet. This makes the three-way handshake work properly and guarantees that no data may be transferred while the encryption is set up without being encrypted.

The slave calculates the SK and IV, and transmits the start encryption request packet (LL START ENC REQ) without encryption, but then con,s the slave to accept an encrypted packet using the SK and IV values. Because all other data packets have been halted, the master will reply by requesting the same packet again if it doesn't get the slave's packet. The slave may choose to accept this packet or the master's subsequent encrypted packet since empty packets cannot be encrypted because there is no payload to encrypt.

The master sets up to receive an encrypted packet in response to the slave's packet by sending an encrypted packet (LL START RSP) using the SK and IV values it just determined. Because it was set up to accept encrypted packets, the slave may receive this encrypted packet. Now the slave will enable packet transfer with encryption. The slave will send back an encrypted packet (LL START RSP) in response to the master's encrypted packet. Because it has previously enabled the receipt of encrypted packets, the master may receive this encrypted packet.

The flow of application data, which is completely encrypted, may start after the master receives this last packet. The flow of application data, which is completely encrypted, may start after the slave has had the Link Layer acknowledgement of this last packet. On Bluetooth Classic, authentication must be completed before beginning encryption. Time and extra messages that must be sent are both expenses associated with this. This procedure is not required with Bluetooth low energy since authentication is part of every packet that is sent. As a result, rather than only once at the beginning of the encryption process, the connection is authenticated for every packet.

CHAPTER 6

BLUETOOTH DESIGN TYPE

Puneet Kalia, Associate Professor,
Department of Electronics and Communication, School of Engineering & Technology, Jaipur National University, Jaipur, India
Email id- puneet.kalia@jnujaipur.ac.in

The first thing to consider when analysing any technology is how the designers improvise on this technology. The majority of technologies are often one or two things many things they are not very good at but are extremely excellent at. Identifying one or more of these It is also possible to get a deeper grasp of that technology this is quite easy using Bluetooth low energy. It was intended for very low usage of electricity. The Bluetooth SIG has a distinctive organizational structure in that it designs and manages all aspects of the technology, from either the Network Layers through the application.

The SIG does everything in a collaborative, open, yet commercially motivated manner paradigm, and over more than 10 years, it has improved the procedure for developing wireless standards such that they are both functional at the time of release and interoperable. Sturdy and of the highest calibrate creation of the shortest transceivers wireless systems was the aim of the low energy work. To do something like this, each architectural layer has always been tailored to cut down on the amount of energy needed to complete a job. For comparison of the radio frequencies' loosening in the Physical Layer indicates that the radio may use less power while sending or receiving data when using a Bluetooth classic radio. The fullest sense is geared at very quick reconnections and effective data broadcasting, which may potentially eliminate the requirement for connections.

The host's protocols are improved to shorten the period after establishing a network protocol connection before the application data may be transmitted. Everything is conceivable only if the same team designs every component of the system at once many individuals. The original Bluetooth radio's design objectives have not been forgotten. These comprise the following

1. Global operation
2. Low price
3. Solid
4. Limited range
5. Low output
6. An international wireless band is necessary for global operation.

The 2.45GHz spectrum is the only one that can currently be used thanks to low-cost, high-volume production technologies. This is accessible since Astronomers, mobile phone companies, and other commercial interests have little interest in it. Unfortunately, everyone wants to participate in it since it's free, just like everything else producing a backup. There are other wireless bands, such as the 60GHz ISM spectrum, although from a cost-savings perspective this is impractical, or the 800/900MHz bands that vary in frequency and operation according to your location on the planet.

There is a little issue with the short range. A low-power system is what you need to minimise the amount of energy utilised, transmission power must be kept as low as feasible to send the signal out. Similarly, you must maintain a somewhat high receiver sensitivity to lower the power needed to detect radio signals from other devices among the clamour. In this instance, "short range" just means "not centred. “Surrounding a network of cell base stations. Bluetooth low energy indicates that the range is limited and ought to be a local area network. There hasn't been much of a shift in the initial Bluetooth design aim of low power, other than

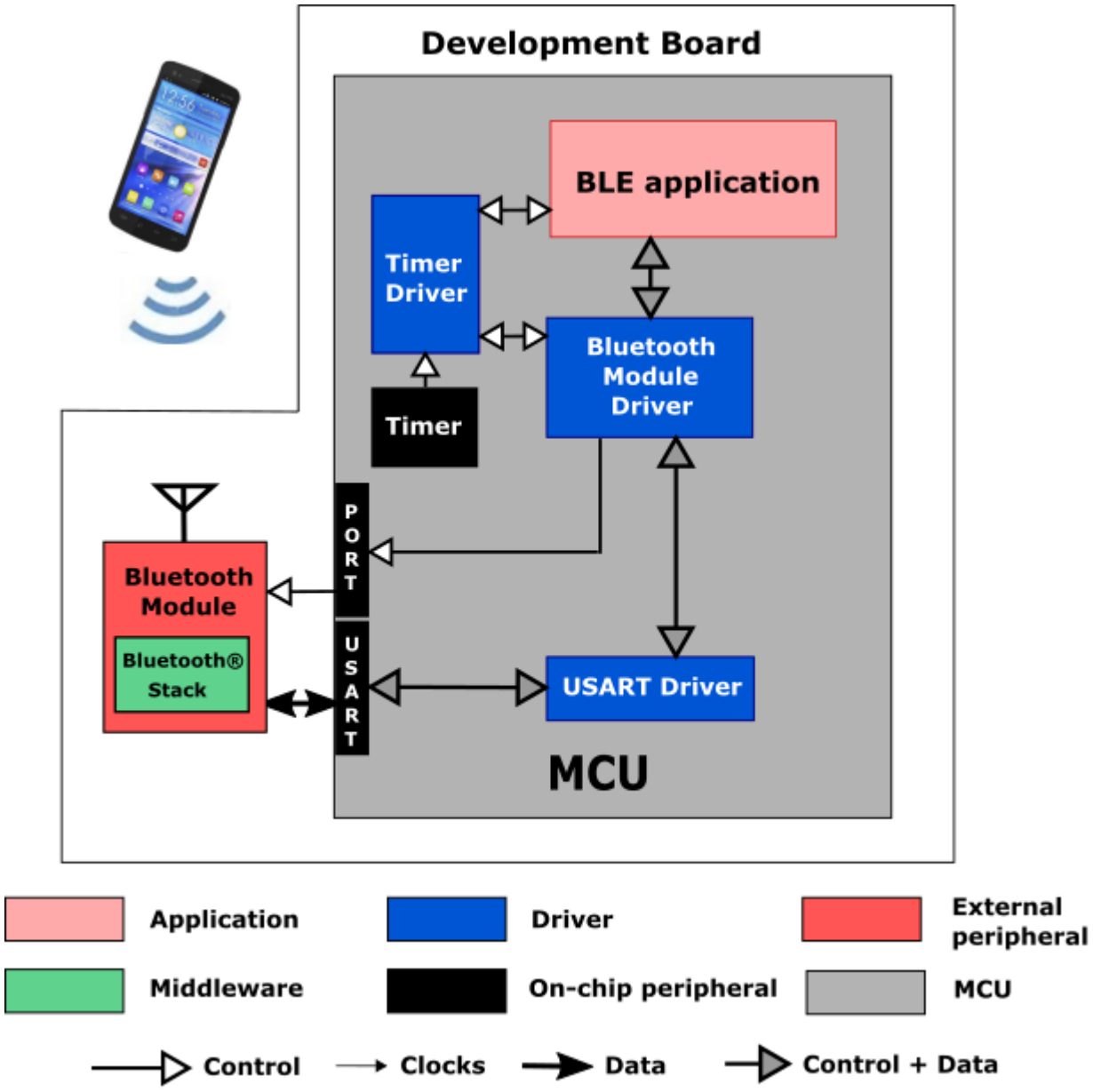

Figure 6.1: Discloses the Bluetooth Module Driver.

That the power consumption design targets have been lowered between one or multiple tens of nanometers in magnitude. Designing Bluetooth classic with a few days of standby but a few hours of use in mind speak lifespan for one headset, vs a few years for Bluetooth low energy for only a sensor that counts the number of steps you've taken or measures the climate. The low-cost design objective is intriguing since it suggests that the system needs to be maintained as compact and effective as doable. Although it's conceivable, Adding full-mesh networking or scatter net capability, for instance, to Bluetooth low Considering that additional memory and storage capability would be required, the price would rise would be necessary to keep this network running. Consequently, the system has been improved with very intriguing research-based networking topologies at a cheap cost.

Bluetooth low energy operates in the already extremely congested 2.45GHz frequency. When just considering standards-based technologies, they are Bluetooth Classic, and Bluetooth Low Energy Additionally, a handful of exclusive stations are also utilizing the band, including cordless alarms, computers, and mice as well as X10 video repeaters. Other vibrations in the area include microwaves and street lights ovens. Figure 7 discloses the Bluetooth Module Driver.

CHAPTER 7

DIFFERENCE BETWEEN BLUETOOTH AND UWB

Dr Giresha AS, Assistant professor
Department of Biochemistry, School of Science, Jain (Deemed to be University), Bangalore, India
Email Id- asgiresha@gmail.com

At the turn of the century, transactional Device but rather ultra-wideband (UWB) short-range broadband technological advances gained popularity. Their probable outcome paths have been influenced by the constant need to lower power costs and enhance power consumption for an incredible number of Bluetooth connection equipment. To remedy Bluetooth's initial power-consumption shortcomings, Neighboring Devices (BLE) was approved in 2006. Bluetooth 5.2 has introduced options to lower consumption for certain applications like music. These changes are, however, just minor ones. The Bluetooth design is fundamentally what limits power consumption decreases; a spectrum transceiver will always need a lot more power to start, stabilize, and power its RF oscillators.

In reality, a lot of current UWB technology implementations need to employ crystal synthesizers with greater frequencies than those needed for BLE. Improved UWB implementation, however, may use crystal oscillations with timing as low as 32 kHz. Second, performance at high data rates is penalized by the message signal overheard upper middle of the diagram above. It takes a lot of time and energy to transmit a lot of data across a channel with limited bandwidth, such as the one used by Bluetooth radios.

This is how UWB calculates measurement an impetus is delivered with one device to that other, and we clock how long it takes for the instinctive reaction to arrive from the other device. Accordingly, the proximity surrounding devices is calculated, because UWB chips can estimate this separation with picosecond precision. UWB "tagged" objects could perhaps be found with precision down by at least 10 cm by using embedded antennas to combine observations to calculate a sensor's angle of origin.

This accuracy is unmatched by Bluetooth technology, which measures spatial distance using received signal strength (RSS). The reason RSS is used so often is that it is a fairly simple implementation approach that can be utilised by any wireless transceiver. The reported distance between two immovable objects may fluctuate depending on obstructions in their direct route, thus BLE generally only provides locating fidelity almost within a few meters. Even while the placement technology allowed possible by UWB is incredibly precise, it is also tremendously power-hungry due to its great complexity in design. As a consequence, UWB chips used for detection and tracking today are up to 10 times less energy-efficient than Headset chips/radios. Therefore, even though UWB is specifically tailored for locating, it is an electricity application by virtue, and UWB does not provide any device-level power benefits.

With UWB, large volumes of data may be broadcast considerably more rapidly since they are spread over a wide bandwidth, requiring the transmitter to run for a lot less time and using a lot less power. This implies that UWB can communicate so much data for a given amount of power

(far upper right). This is due to the evening duality, which the Fourier transform effectively captures. Simply said, according to this duality, a periodic time signal with an endlessly long period will provide an infinitely tiny bandwidth. The bandwidth of an endlessly long impulse signal, on the other hand, will be infinitely short. In other phrases, you may exchange bandwidth for time.

Chosen to give its provision and sustain over a sizable portion of the radio spectrum, ultra-wideband enjoys a distinct inherent advantage over narrowband. Any transmission with a spectrum of more than 500 MHz is referred to as a UWB signal. The Federal Communications Commission (FCC) of the United States approved the unlicensed use of UWB in the frequency range of 3.1 to 10.6 GHz in 2002.UWB systems employ nanosecond-scale electromagnetic pulses with short durations for fast data transmission and reception across wide bandwidths. They now have a very fast duty cycle, which seems to be the proportion of transmission time that an excitation is accessible to. Figure 8 discloses the Microcontroller and the Bluetooth.

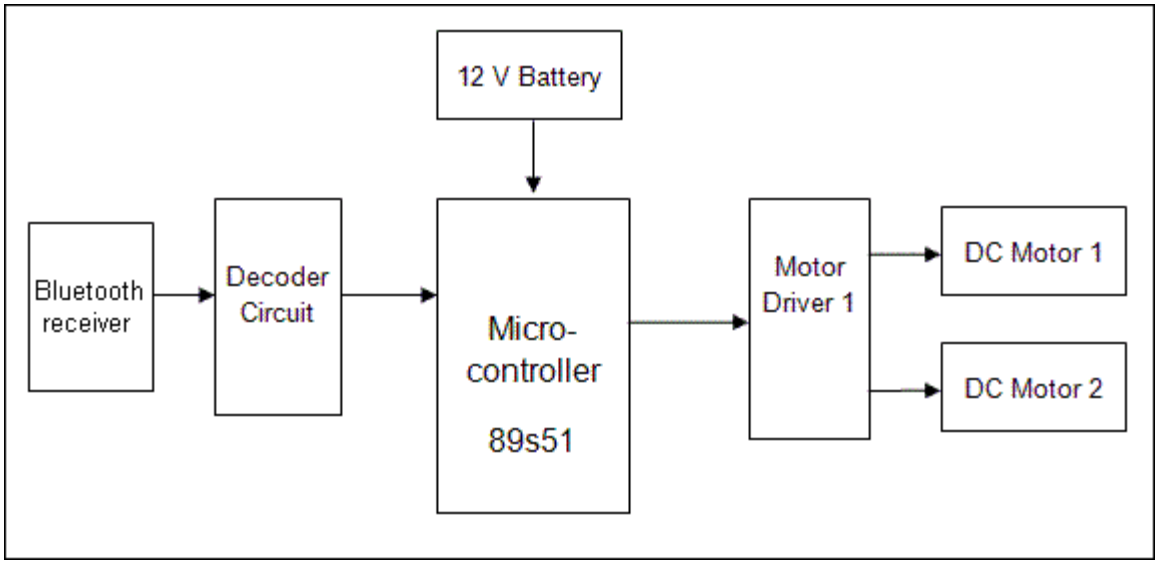

Figure 7.1: Discloses the Microcontroller and the Bluetooth.

Unlike any other wireless technology, ultra wideband is a quick, secure, and low power radio protocol used to pinpoint a position. Here is all the information you need to be aware of about the revolutionary ultra-wideband technology.

UWB is a radio technique that sends data packets in very brief, low-duty cycle pulses in the 3.1 to 10.6 GHz frequency range. Despite having been created initially for military use, it is today found in everything from computer peripherals to video games. It has the capacity to prevent interference from other wireless technologies like Bluetooth and WiFi while transferring massive volumes of data over short distances. The technology may be used inside, where it competes with conventional RFID systems on price and performance, or outside for high-speed communications between satellites.

UWB varies from standard narrowband communication technologies in that the bandwidth of the signal is substantially greater, and information is transferred not only by the precise phase and amplitude of the carrier wave but also by its perfect time. With only one UWB device, data rates may be quite high. The most popular implementations of UWB are based on extremely brief pulses, generally modulated by either the amplitude or frequency of the carrier wave. Better time synchronisation is made possible through frequency modulation, which also improves accuracy. Ultra-Wideband Technology (UWB) employs frequencies up to 10 GHz.

The FCC in the United States, ETSI in Europe, and ARIB in Japan have all defined UWB. These requirements are similar in that they permit usage in select bands that are shared with other applications or over the whole permitted UWB spectrum of 3.1 to 10.6 GHz (e.g., 2.3-2.7 GHz in the US). The 3.1 to 5.8 GHz spectrum is often shared with different service applications throughout the globe.

By shutting the gadget off while not in use, the battery life may be increased. A mobile device normally has to remain in standby mode after sending or receiving data for a minimum amount of time in order to save battery. The gadget may stay in standby mode for a shorter period of time if data is transferred more regularly.Due to its capacity to provide high bandwidth data connections over short distances, ultra-wideband technology (UWB) has attracted a lot of interest. With a range of little more than a metre, UWB seems to be able to provide data speeds greater than 20Mbps. In contrast, Bluetooth can only provide coverage of up to 10 metres and data speeds of roughly 721kbps.Bluetooth is a wireless technology standard for sharing data over short distances between fixed and mobile devices utilising short-wavelength radio transmissions in the ISM band from 2.4 to 2.485 GHz), building personal area networks (PANs) with high degrees of security. An estimated 2.5 billion gadgets are presently supported by the more established Bluetooth technology, which has been available for more than ten years. Nearly two million new devices are added every day.

UWB enables a broader variety of bandwidths (5 - 500 MHz), higher speeds (up to 480 Mbps), and various frequency ranges than Bluetooth. Like Bluetooth, UWB is mainly intended for point-to-point connections between two devices, such as direct connections between computer peripherals or communication with a cell phone or PDA that may subsequently transmit the information to other computers in the network.

UWB devices, on the other hand, can be programmed to operate in a "sleep mode" and then activated at will by an initiating device that transmits encoded signals that wake up the target device. This differs from Bluetooth, where both ends must be simultaneously activated by an initiating device, such as a laptop, before any communications can begin. This implies that it is feasible for several UWB devices to use a similar data channel and may minimise power consumption and latency (the time gap between requesting and receiving information) while boosting throughput (amount of information sent) (amount of information transferred).

What Benefits Does UWB Offer?

Supports considerably higher rates than Bluetooth 1.2 or 2.0 (up to 480 Mbps vs. 3 Mbps) (up to 480 Mbps vs. 3 Mbps operates at lower power levels (less than 100 mW) to extend the battery life

of gadgets may be utilised in places with significant levels of interference, when several wireless technology types interact with one another. UWB is designed to automatically steer clear of channels that are already being used by other devices enables all devices to "listen" for an open channel or time slot before switching back to standard data transmission, supporting numerous users sharing the same spectrum.

The whole spectrum of Bluetooth applications now available cannot be supported by a single radio chip. Utilizing a Bluetooth radio for "low energy, short-distance connections" and adding UWB when larger bandwidths are required for high-speed communications between stationary sites is the standard strategy.Another disadvantage of UWB is that, like Bluetooth, it requires line of sight and close proximity (within 30 feet) for communication between devices. This restricts their utility inside in big warehouses or office buildings.

UWB may be used for the following purposes:

1. monitoring and control of electronic equipment, such as security cameras, from a distance
2. medical supplies
3. sharing of health care information
4. Wireless products
5. between desktops and peripherals, wireless LANs
6. telecasting devices
7. Bluetooth microphones
8. Sonet/SDH wireless networks

When the computer platform is in "sleep" state, short-range communications may occur between PC peripherals.It may also be possible for employees to wear wrist-mounted computers that collect data from sensors positioned around workstations or in production facilities and transmit it back to a central computer for processing.

The benefits of 5G UWB over current wireless networks are many. First off, it offers more control over the hardware that connects to and interacts with the network. For instance, a system administrator may use in-built sensors like microphones, cameras, and GPS receivers to locate and disable certain devices. This might be helpful in places like data centres and power plants where security is a worry or at events like concerts where spectators shouldn't be able to record anything with their phones that may be streamed live online. Additionally, it permits quick transmission rates. The UWB link would increase capacity while minimising latency (the delay in data transmission) (the amount of data that can be transmitted in a specific amount of time).Additionally, UWB avoids the need for numerous radios by employing a single chipset that supports both wireless local area networks and short-range personal area networks (PANs) (WLANs). When it is essential to add or delete a device's functionality, it is feasible to connect several devices and then deactivate and reactivate them.

When comparing Ultra-Wideband with Bluetooth, you discover that each technology has advantages and disadvantages of its own.In other words, Bluetooth has greater connection, while UWB has a higher range. The benefit of a Bluetooth connection over a UWB connection is that devices may communicate without a direct line of sight. While UWB needs a direct connection

between two devices to send data, Bluetooth may leverage existing infrastructure, such as Ethernet cables, to offer Internet access at a quicker rate than Wi-Fi.UWB's speed and capacity to concurrently service many devices while avoiding other wireless traffic within the same spectrum are its primary features. Less power usage may compensate for the device implementation's possibly greater cost.

While UWB offers more bandwidth for high-speed data transfers between devices, Bluetooth is a superior option for low-bandwidth applications like speakers, keyboards, and mice. Due to its cheaper implementation cost, Bluetooth is still a preferable option for short-range wireless communications even if USB 3.0 now offers speeds that are equivalent to UWB.

While lacking Bluetooth's mobility, UWB offers the ability to provide better bandwidth for data transmission between fixed sites like desktop workstations. Its fast speed and low power consumption, however, might be a significant benefit in providing a single platform that could easily and adaptably manage both short-range and longer-range wireless networking demands.UWB does not, however, alone provide an efficient remedy for indoor positioning systems. However, fusing UWB with other technologies can create location-aware applications that could potentially enable personal navigation devices to pinpoint our exact location within a building without the need for satellite-based communications. The technology may be used to improve GPS services, making them faster and more accurate indoors.

Real-time location systems (RTLS) in use today are continually evolving. Operations leaders can experience choices overload. The Ultra-Wideband (UWB) sensor technology is one of such alternatives. The majority of people are acquainted with the phrase because to the popularity of the most recent Apple and Samsung mobile phones. But they may not be quite certain how UWB differentiates with Bluetooth Low Energy (BLE). Although businesses have long used BLE to monitor assets, UWB is an emerging technology that can link devices at close distances with unmatched precision. MarketsandMarkets projects that by 2025, "Ultra-Wideband will be valued 2.7 billion, with a CAGR of 19.6%." As a result, a common query is "Will UWB replace BLE"? We must first comprehend what Ultra-Wideband and BLE are and how they operate before we can respond to that question.

A constantly scanning radar called UWB uses a variety of low-power radio frequencies to function. This wireless, short-range system operates at a minimum 500 Mhz bandwidth. More particular, a band of radio frequencies between 3.1 and 10.6 GHz has been made available by UWB. While that range is fantastic, some people may be concerned that it could interfere with nearby devices. No, is the response. The fact that UWB uses so little power is the key to achieving "no interference." The graphic below shows that UWB utilises a maximum power of -41.3 dBm/MHz. UWB devices function best when they are near to one another and the line of sight is unobstructed due to the minimal quantity of power.UWB's most significant characteristic is communication and data transitioning from one device to another, even though it can accurately forecast the relative location and movements of assets within 10 to 30 cm. Due to the vast bandwidth, UWB has shown 110Mbit/sec transmission rates. That is sufficient to simultaneously exchange three video streams over a single connection

These two technologies have great promise for future developments that may bring about many of the things we've been longing for. Bluetooth will result in low-cost, quickly deployable systems that can handle everything from household appliances to industrial process controls. High bandwidth wireless connections will allow us to accomplish even more with UWB, but it's feasible that both technologies may be combined to provide a more comprehensive solution.Sunvera Software creates cutting-edge software programmes from scratch. iBeacon apps, security solutions, telehealth software, sales dashboards, custom mobile app development services, retail software development, supply-chain software, e-commerce, Shopify, web design, and unified access software are some of the areas in which we excel. We are a leading software and mobile app development company. We are happy to be partners with Google Cloud, Microsoft Azure, and Amazon AWS.Conversation or schedule a free 30-minute call with us to talk about your company.

CHAPTER 8

DIFFERENCE BETWEEN BLUETOOTH AND LIFI

Ryan Dias, Assistant Professor
Department of Electronics and Communication Engineering, Faculty of Engineering and Technology, JAIN (Deemed-to-be University), Ramanagara District, Karnataka - 562112, India
Email Id- ryan.dias@jainuniversity.ac.in

The late 1990s saw the development of Bluetooth technology, which uses wireless short-range transmission among both electronic devices like a MacBook Pro and a telephone or computers and televisions. Instead of the electromagnetic region utilised by conventional remote controllers, radio frequencies are employed by Bluetooth to communicate. As a consequence, Bluetooth does away with the necessity for both a wired connection and a constant line of sight between devices to communicate. In the same way that Bluetooth utilises radio waves to transmit high-speed short messages without either a cable connection, Wi-Fi does the same. Wi-Fi operates by fragmenting a signal and sending the parts across several radio bands.

By using this method, the signal may be sent at a lower power level per frequency and more devices can share a single Wi-Fi transmitter. Wi-Fi, which was first created in the 1990s, has undergone many standardization procedures that have been accepted by the International Electrotechnical Commission IEC Engineers (IEEE) to support higher data transmission capacity [1]–[3]. To transport data over the visible light spectrum, Li-Fi used LED lights. Let's take a step back and try to grasp why that's significant. Describe Wi-Fi. Via a Wi-Fi router, Wi-Fi, or Wireless Fidelity, transmits data using electromagnetic waves at radio frequencies. The distinction has several facets. Most obviously, Li-Fi utilises light, while Wi-Fi uses radio waves. This is genuine common light that you see every day.

Li-Fi also has impressive statistics:

It can reach 224 megabits, which is 100 times faster than the existing Wi-Fi average speed, according to certain tests. Applications for the web, music or even video may be included in the binary data. The Li-Fi bandwidth is likewise quite large. For instance, Li-Fi spans 10,000 fold the radio waves spectrum whereas Wi-Fi only operates at 2.4GHz, 4.9GHz, and 5GHz frequencies. Li-Fi is a blatant contender for game-changing technology when it comes to meeting the expanding need for data connectivity. Once implemented, Li-Fi maybe 10 times less expensive than Wi-Fi. Think about how straightforward and common LED lights are. Now think about all their Wi-Fi gateways, cables, and equipment. Blockers containing Li-Fi functionality for the façade. Walls are impenetrable to light [4]–[6]. There won't be any data transmission if anything fully stops the light source from reaching a gadget.

Additionally, the ability to utilise Li-Fi is limited since data transmission requires the light to be switched on. Additionally, its operational range is less than that of Wi-Fi in general. Li-inability Fi's to penetrate through barriers like walls or other obstructions is a very useful feature. By limiting data to a light stream, keeping data safe becomes immensely smart and straightforward. Bluetooth and Wi-Fi are both wireless types of communication, although they have different uses,

capacities, and other characteristics. Short-range data transmission between devices is possible using Bluetooth. For instance, it is often used in mobile phone headsets to enable hands-free phone usage. On the other side, Wi-Fi enables connections between devices and the Internet. Wi-Fi is accessible to more people and devices than Bluetooth, which has a maximum number of simultaneous connections. Additionally, Bluetooth tends to be easier to use and consumes less time than Wi-Fi since it only needs an attachment on each connected device. However, this is done at the sacrifice of range and data transmission speed, which Wi-Fi normally outperforms.

Version Comparison

When troubleshooting, it might be helpful to learn a little bit more about a device than simply its device address and the data present in the host and application levels. Version Information may be requested from devices by the host or the Link Layer autonomously and is solely relevant for debugging reasons. Version information does not need to be transmitted on each new connection since it cannot be utilised to alter the behaviour of the device. However, most devices would only ask for this data once per ten connections or more to allow a sniffer to get it and aid in connection troubleshooting.

The process for exchanging versions

Some individuals are troubled by the Link Layer's independent activity. Why would a Link Layer need to check for updates every 10 connections? Simply put, you may not be able to initiate a version information exchange from a host in either of the two devices when you have an issue that needs troubleshooting. Therefore, a version information exchange that happens seldom is far preferable to none at all to characterise these devices.

A gadget that is only connected with one other device such that no other device can connect and request this version information that initially displays unstable behaviour but does not show it for many months before showing it now is the classic example. This issue can be solved much more effectively now than it will be in a few months by having the Link Layer send these version swaps on its own after a few connections.

The following version details are provided:

1. Sub-version number; Version number; Company identification
2. The Bluetooth standard, with which this device is compatible, has a version number that is assigned to each published version of the specification.
3. The firm identification is a number given to the business that made this controller by the Bluetooth Special Interest Group (SIG).

The controller manufacturer will assign the sub-version number, which must be unique for each installation or revision of the controller. This number cannot be assigned in a specific manner; instead, the device's manufacturer should be contacted if debugging is crucial. At UnPlugFest gatherings, engineers test and confirm these controllers before distributing them in bulk, exchanging the contact information of the device producer. Additionally, the qualification database for all shipping items has this contact information.

If neither device has sent a version information indicator packet during the currently active connection, they both may do so to communicate version information. A device will react with a version information indication packet if it receives a version information indication from a peer device and has not yet delivered its message. It is not possible to request this operation once it has finished. Since this data is static, it won't change while a connection is made. Therefore, there is no purpose in requesting the same static information again. While the connection is open, the controller should save this data in case the host requests it again.

Feature Exchange

A peer device uses the feature information to ascertain what capabilities a device has. It reveals the range of extra functions that a gadget can support. The feature request packet (LL FEATURE REQ) may be used by the master to request this data once, if at all, during any connection. The feature response packet (LL FEATURE RSP) is provided by the slave in response to the master's request. Since all features must be enabled and all optional features provide a mechanism to refuse this request, this information does not need to be transmitted to utilise an optional feature. For instance, a refuse indication packet is provided, securely denying the encryption request, if encryption is not available or enabled.

The process of exchanging features

There is just one feature bit specified for Bluetooth low energy's first release: encryption capability. This indicates that a gadget supports encryption.

Disconnecting from Connections

The termination method is the only other Link Layer control operation that is possible. Naturally, this breaks the connection and puts the master and slave connections back in sleep mode.

The method of termination

A device sends a terminate indication packet (LL TERMINATE IND) to discontinue the connection. It then waits for this packet's acknowledgement at the Link Layer before disconnecting.

A device will simply time out this method and the connection will still be severed if it sends the terminate indication packet but does not get any acknowledgement of it. The supervisory timeout value is equivalent to this timeout. Once a device gets the terminate indication packet, it instantly disconnects after sending an empty packet in response to acknowledge the message. Two other circumstances may cause connections to be terminated:

1. Pause for supervision
2. Failure of MIC

The connection simply disconnects and the hosts on both sides are informed when one of these other causes occurs; no terminate indication packet is transmitted.

Robustness

Two very powerful algorithms are used by the Link Layer, which is resilient, to guarantee that data travels without interruption and that when it is transmitted from one site, it is sent to a peer host unaltered.

Adaptive Frequency Hopping

The effective transfer of data between devices communicating utilising the 2.4GHz ISM band depends on adaptive frequency hopping. Technology that prevents interference is crucial since there are so many different devices utilising this frequency. The introduction of adaptive frequency hopping into the Bluetooth classic standard in late 2003 marked a sea shift in performance improvements over pre-2003 devices.

Simply described, adaptive frequency hopping is a technique for hiding unwanted channels on a frequency-hopping radio by remapping them into desirable channels. Both devices contain a channel map for this purpose. Each bit of this 37-bit channel map corresponds exactly to a certain Link Layer channel. Link Layer channel 0 is tied to bit 0 in the channel map, Link Layer channel 1 is related to bit 1, and Link Layer channel 36 is related to bit 36 in the channel map. When a channel's bit is set to one, it indicates that the channel is in use; when it is set to zero, it indicates that the channel is not in use. This channel map must have at least 2 bits set as utilised.

Consider the channel map if all three of the primary Wi-Fi channels are in use. These are the 2412MHz, 2437MHz, and 2462MHz-focused channels. This indicates that just 9 Link Layer channels 9, 10, 21, 22, 23, 33, 34, 35, and 36 would be regarded as excellent. The channel map is defined by three values: Used, which is the set of useful channels from the channel map, and numUsed, which is the total number of useful channels. Therefore, these values for the example above would be as follows:

Channel Map = 00011110000000011000000002

Used = [9, 10, 21, 22, 23, 33, 34, 35, 36]

#Used = #

The adaptive frequency-hopping channels used for this connection would be determined by using the frequency-hopping equation, if a connection were to be established with this channel map, sent in the connect request packet from the master to the slave, and with the hop interval set to 7. Following the calculation of the starting frequency fn, these channels are adjusted.

Channel 7 should be the initial connection event on the link. This would be remapped into the list of good channels as it is not among the excellent channels. To achieve this, the number of good channels, num Used, is modulated with the number of good channels, used, which in this example is 9, and then the unmapped channel number is mapped into this. Link Layer channel 35, the eighth effective channel, is used as the connection event's channel. Data may always be transported via adaptive frequency hopping by remapping channels that are designated as unused to utilised channels that are known to be effective, even if there are several nearby interference sources.

Powerful CRCs

In comparison to Bluetooth classic and the majority of other wireless short-range devices, the size of the CRC value in Bluetooth low energy is 50% bigger. Unluckily, a 16-bit CRC is insufficient for the 2.4GHz ISM band by Bluetooth classic. To address this, a second 16-bit CRC was added to the Logical Link Control and Adaptation Protocol Layer (L2CAP), ensuring that the host could still provide bit error protection even if the controller's CRC had forwarded a packet containing unprotected bit errors up to L2CAP.

Sadly, this increases the size of each L2CAP packet, and for devices with limited resources, the implementation is further complicated by the challenging nature of performing a CRC calculation. Because Bluetooth low energy has been learned from Bluetooth classic, the CRC at the controller is substantially stronger. The packet is secured by this CRC, a 24-bit number. All single-bit, two-bit, three-bit, four-bit, five-bit, and other odd numbers of bit mistakes in the packet will be found by this CRC. This offers the same robustness as much bigger packets used in wired protocols like Internet Protocol (IP), which are substantially stronger than a 16-bit CRC. This is so because low-energy packets are substantially shorter, allowing a comparable degree of security to be provided with a shorter CRC.

Another issue with Bluetooth classic is that its packet headers are only adequately secured by an 8-bit CRC, which is a rather poor header error-check value. The main CRC should also fail if this incorrectly detected a packet. Again, our experience has shown that there is not much benefit to having a separate header error check and main packet CRC. The controller must still execute two different LFSRs, one for the payload and one for the header, even if the packet is valid. The likelihood of an error occurring in the header of a defective packet is rather low; the whole packet would likely need to be received nonetheless, negating the predicted power savings from having a separate header error check in the first place.

Bluetooth Low Energy uses a different approach to design. It employs a single robust CRC that defends against the header, length, and payload fields of the packet, as well as the message integrity check value of any encrypted packets. Except for the preamble and access address, both of which must be received bit-for-bit flawlessly for the packet to be initially received, the whole packet is subject to the full power of the strong CRC. This is as effective and more durable. Additionally, the method is easier to implement since it uses only one CRC as opposed to the three needed in Bluetooth classic. This demonstrates why creating every layer of the specification simultaneously is the best method to develop a comprehensive, high-quality specification that can satisfy market demands.

Low-Power Optimizations

The Bluetooth low energy standard's ultra-low power consumption optimization is covered in the parts that follow. These are some of the main techniques: Short packets, a high physical bit rate, little overhead, an improved acknowledgement system, single-channel connection events, subtracting connection events, and offline encryption are all used. There are many techniques to quantify power consumption, however in this case it will be assumed that two categories of power consumption are essential for a device to have low power consumption. First, a gadget must have

a low peak power consumption to be powered by inexpensive button-cell batteries. Too much peak power would cause the batteries to deplete too rapidly and shorten the device's lifespan. For a device to be operated for an extended period while providing a particular amount of application data, low power-per-application-bit is crucial.

Short Packets

The actual radio that is utilised is one of the wireless technology's most intricate components. The vast majority of these radios are made using bulk CMOS 2 technology. Designers are thus faced with a conundrum since to make the radio stable, they must raise the cost by adding circuitry to maintain the frequency. Because the packet length is so short, the heating impact is reduced, and Bluetooth low energy offers them a solution. It just takes a brief packet to produce this issue. Bluetooth Classic's 3-millisecond packets are excessively lengthy and might lead to issues. The attention to detail that the Bluetooth low energy designers showed in improving the Link Layer standard by taking into consideration the physical characteristics of the silicon fabrication techniques used is shown by this very straightforward design choice.

The radio or stability circuitry won't need to be calibrated if the packets are never longer than a few hundred microseconds. For the duration of this time, the frequency may wander without violating the specifications' specified limits on frequency drift. The maximum packet length in Bluetooth low energy is 376 microseconds; this is small enough that the silicon's heating won't cause the frequency of sent packets to stray too far from the permitted limits. As a result, by keeping packets brief, radio calibration is not required constantly. Minimising the amount and complexity of circuitry that must be powered during a packet transmission or reception, lowers peak power usage.

It should be noted that a pause of 150 microseconds is necessary after sending an extremely large packet. The silicon may cool down in between packets thanks to this interpacket interval. As a result, there is no requirement for frequency calibration between sending and receiving or receiving and sending packets, thus lowering power usage. This indicates that the maximum duty cycle for data transmission on an encrypted connection while travelling in just one direction is just: A wireless technology has an extremely low duty cycle of 58%. The duty cycle of Bluetooth Classic is 72%, whereas that of extremely high-speed wireless technologies will be in the high 90% range. Instead of being tuned for maximum data throughput, Bluetooth low energy is designed for sending short, discrete chunks of data.

High Bit Rate

Radio needs a high amount of current to relay data. The need to operate a 2.4GHz oscillator, on which the radio signal is modulated, uses the majority of this current. High currents are needed for CMOS to operate at extremely high frequencies. Because the majority of gates in a digital system do not change state constantly, CMOS is designed for such gates. Therefore, creating this 2.4GHz signal will utilise a comparable amount of current in any radio operating at 2.4GHz constructed using bulk CMOS technology.

The efficiency of the modulated signal becomes important since it is almost impossible to lower the power consumption below a particular level due to the oscillator. Your radio will be more effective if you can transfer a certain quantity of data more quickly.

Data is sent via Bluetooth Low Energy at a rate of one million bits per second. For instance, it would take four times longer to send a given sequence of bits if a device utilising a different technology could only transmit at a quarter of the pace of Bluetooth low energy.

Therefore, four times as much power would be required to transmit the same amount of data as a Bluetooth low-energy system. Therefore, large bit rates are advantageous.

Additionally, it should be remembered that there is a limit to how fast data can be sent. The power required to modulate and demodulate bits in a complicated modulation system that, for instance, can transmit 10 times as many bits per second as Bluetooth low energy is often quite high.

Therefore, even though it may take less time for these bits to transmit, due to the additional energy required for modulation, the power per bit is almost the same. Due to clock drift, the current consumed while getting nothing is substantially greater.

The most effective method of transferring data is therefore using a straightforward modulation technique with low peak-power consumption and high data speeds. The Bluetooth low-energy designers chose this as their sweet spot.

Low Overhead

The amount of overhead is crucial to take into account when choosing a radio technology since every bit matters. In Bluetooth low energy, anything that cannot contain application data is considered overhead. The preamble, access address, header, length, CRC fields, and optional MIC value are all included in this.

The efficiency for an unencrypted packet increases from 29 to 73 percent depending on the amount of the application data relative to the overall packet size needed to convey this data.

The radio is more efficient in terms of overhead the bigger the packet. The additional 4 octets of MIC that are added in each encrypted packet cause the efficiency to be poorer. When compared to the efficiency of comparable proprietary radios, this efficiency is quite excellent. Before any encryption overhead, Zigbee, for instance, has a packet overhead of somewhere between 15 and 31 octets. A brief four-octet application data in Zigbee may need up to 10 times more energy than a Bluetooth low-energy solution, given the four times slower physical bit rate for Zigbee. Figure 9 discloses the Arduino and the Bluetooth connection.

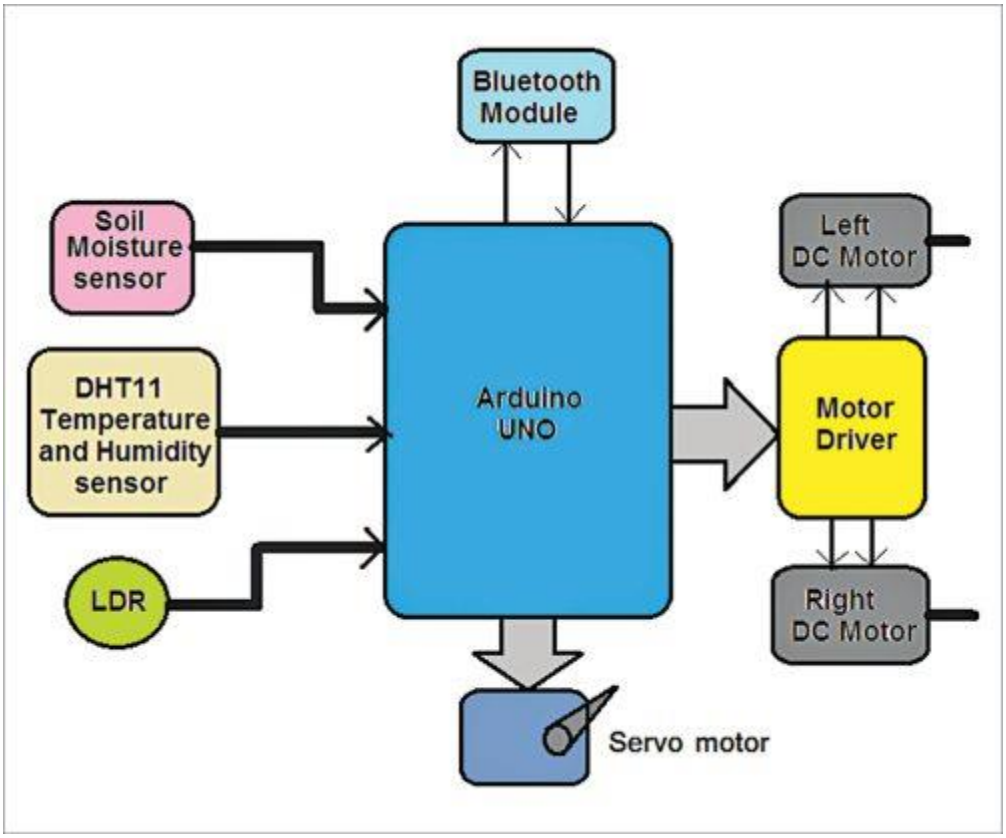

Figure 8.1: Discloses the Arduino and the Bluetooth connection.

The Application Data Overhead

Acknowledgement Plan

The Link Layer acknowledgement approach has an intriguing side consequence in that it does not need instantaneous performance or even delivery of packet acknowledgement. This is a significant distinction between Bluetooth Low Energy and Bluetooth Classic. The packet must be acknowledged by the receiver in Bluetooth classic at the next chance for transmission. The receiver must broadcast a negative acknowledgement in the next packet it transmits if the acknowledgement is not immediately received. When synchronous connections are active and the slave is trying to transfer data, this creates the greatest issues.

Every Bluetooth low-energy packet delivered can acknowledge the most recent packet sent, even if it was sent some time ago. As a result, devices are never required to broadcast right away to submit their acknowledgements. The device has the option to not acknowledge the last packet until it has data to send or until it must transmit for another reason, such as timeouts. This enables the very speedy and effective acknowledgements needed for sending massive amounts of data quickly.

Single-Channel Connection Events

A connection event is when all communication between a master and a slave takes place. A packet sent by the master initiates a connection event, which is then followed by a sequence of alternating

packets sent by the slave and master. Every packet broadcast in Bluetooth v1.1, whether it was a master or a slave, was sent on a distinct channel. The answer would have been sent on a separate channel if the master-to-slave packet had been sent on one. The issue with this is that even though the slave may have flawlessly received the transmission from the master, the channel used for the reply may have interference, making it unlikely that the acknowledgement for the master's data packet would be received.

Bluetooth v1.2 made every slave communication utilise the same channel as the master's broadcast when it included Adaptive Frequency Hopping. The assumption is that if a packet is good when data is transmitted from the master to the slave, it will be good when data is transmitted from the slave to the master. In the previous example, the slave would have replied on the same channel that the master used to transmit its data, so that packet would most likely get through. This is a sound supposition that has enhanced the real data rates in conditions of high interference.

This has now been brought to a logical conclusion by Bluetooth low energy. Why cease using a good channel if it is excellent? A decent channel needs to be used for as long as feasible. The most illogical course of action would be to just discontinue frequency hopping. After all, if the data is received, you should just continue on that frequency until it stops functioning, at which time you should switch to a different channel. Unfortunately, you cannot transmit any signalling packets to coordinate the frequency shift when you do need to do so. Resynchronizing the single frequency when the channel begins to break also consumes a lot of energy. This model soon fails due to the temporary nature of interference sources in a region, particularly with the bursty nature of most Internet traffic via a Wi-Fi network.

The number of co-located networks is further decreased by this one-channel model since each would eventually drift to a clear frequency. If there aren't many clean channels, there won't be many colocated networks either. A frequency-hopping method distributes network traffic in terms of both frequency and time, enabling several concurrent networks to operate in the same space at once. The alternative approach, which calls for maintaining a frequency until it breaks, is ineffective.

The low-energy strategy is to remain on one frequency for a single connection event, acting as if you are certain that it is functional and capable of sending data and acknowledgements and then switch to a different frequency at the subsequent connection event. This implies that the frequency to be used is entirely predictable at all times. The two devices will immediately cease utilising a specific channel if it is inoperable, resynchronize at the subsequent connection event on an alternative channel, and resume exchanging data. This implies that even in environments with high levels of unexpected interference, such as those found in many homes and companies, data may flow with the least amount of delay effect.

Subrating Connection Events

The master device often has access to far more resources than the slave device. The battery in a computer keyboard is significantly smaller than that in a computer, and although the computer may need to be recharged often, the keyboard battery is designed to endure for many years. But many gadgets just need a little bit of delay. The peer device must receive a keystroke from the user

as soon as feasible when they hit a key on a keyboard. User interaction requires low latency. Low latency and low power, however, must coexist together.

Low power calls for the slave device to only occasionally listen to the master device. The slave device must be able to transmit as often as feasible to achieve low latency. Low latency calls for the master to continually poll the slave to check if it has any data to transmit since the slave cannot communicate until the master does so in a connection event. However, it would not be low power if the slave had to hear every connection event.

The answer is to provide the slave with the freedom to disregard the majority of connection events coming from the master. The latency of the data to the connection event interval would remain the same as that of the master since the slave would still be able to synchronise and deliver data as soon as it is necessary. The term "slave latency" refers to the slave's ability to ignore a certain amount of connection events. The amount of power that the slave can save depends on the number of connection events that may be missed. The slave might have a lesser power the more connection events that can be missed. Figure 10 discloses the Ultrasonic Sensor and the LED Stripe.

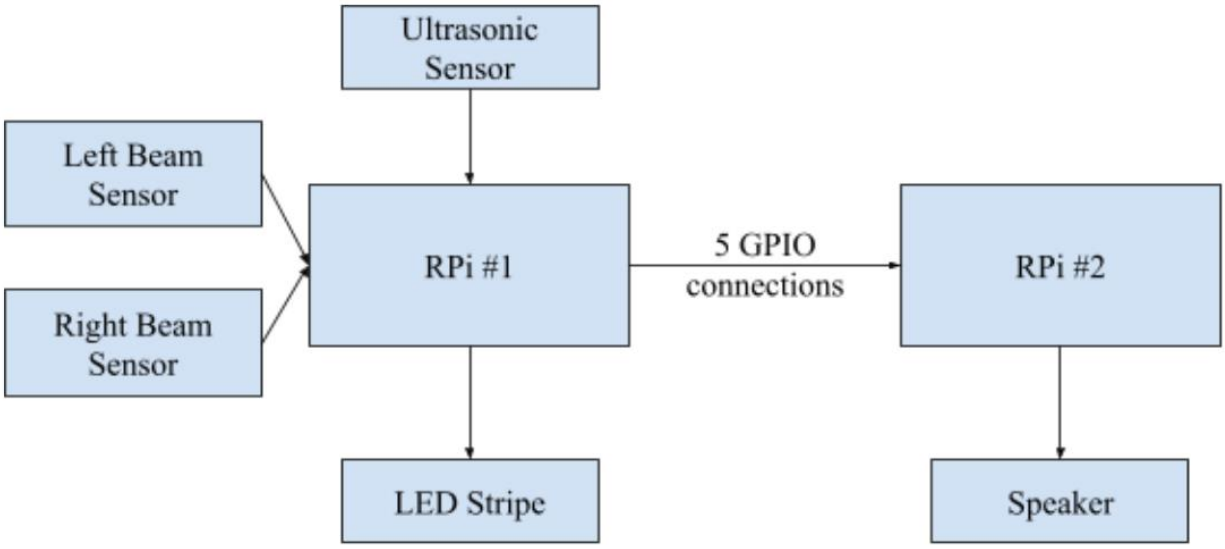

Figure 8.2: Discloses the Ultrasonic Sensor and the LED Stripe.

Subpar connection occurrences

The delay of the slave has a maximum. It is not feasible to have a slave latency that exceeds the connection's supervision timeout. Additionally, it is not advised to have a slave latency that provides the slave with less than six chances to resynchronize with the master. A slave delay of 4,650 milliseconds, for instance, would allow the slave six chances to resynchronize at the end of the slave latency if the supervision timeout is 5 seconds and the connection event interval is 50 milliseconds. The slave may nonetheless send data to the master at an average of only 25 milliseconds with a duty cycle of just 0.0069 percent and just once per 4.5 seconds of communication.

In-App Encryption

Data encryption is often regarded as a highly high-power operation. The authentication code itself, as well as the cypher blocks for the payload and message authentication codes, must be computed by the device. This calls for a total of seven cycles of the AES-128 encryption block for a maximum-sized packet. The peak current of the packet transmitter and receiver would be substantially larger if this task had to be completed in real-time. Data encryption and authentication codes may be generated in the background using Bluetooth low energy. The encryption of the data may be carried out before a packet is broadcast while the radio is still off. Data encryption is independent of the sequence number, the projected next sequence number, or the amount of additional data. Therefore, from the moment the data enters the Link Layer until shortly before being transferred, it may be encrypted at any point. Additionally, even if the next anticipated sequence number or another data bit changes, the encryption and authentication code will remain the same after any number of retransmissions of this data. In addition to eliminating the expense of retransmitting encrypted packets, this reduces peak power usage.

The only number that can be used to assess whether encrypted data was successfully received when it is received is the CRC value, which is generated in real-time. Until there is enough power to decode the packet and the radio activity ends, the encrypted data may be held in the Link Layer. Before the packet is sent to the host, this decryption of the packet may be performed whenever it is convenient. As a result, a receiver's peak power is decreased. Additionally, if the packet is retransmitted, it need not be again decrypted upon arrival since the sequence number is repeated. This eliminates the expense of re-receiving encrypted packets and reduces peak power usage.

Security Concepts Security is a complex subject that for many people is just a black box. If the technology is secure, that is enough for lots of people. However, many things about security need to be understood in the context of Bluetooth low energy. These include the following topics:

A. Authorization
B. Integrity
C. Confidentiality
D. Privacy

Authentication is defined as a way to prove that the device with which you are connecting is the device it claims to be and not a third-party attacker. Re-authentication using a previously shared secret For example, when a user opens a bank account, she must provide documentation to prove that she is who she says she is. This authenticates the user to the bank, typically because the bank trusts the issuers of these documents. Passports, identification cards, and other government-issued documents such as a driver’s license are typically used for this. The bank then gives the customer a plastic bank card and a Personal Identification Number (PIN) that she can use at a later time to re-authenticate herself with the bank. When the customer wants to remove money from this bank account she must authenticate herself to the machine by using the card and the shared secret PIN.

Any person who has that card and that PIN can authenticate himself as the account holder to the bank even if he isn’t in actuality. In Bluetooth low energy, authentication is performed in three different ways: During the initial pairing of devices, an authentication algorithm is used to

authenticate the connecting device. This might involve entering a passkey into one or both devices. This allows the link to be encrypted, and any shared secrets that will be required later can be distributed. When these shared secrets are stored, the devices are said to be "bonded." When reconnecting to a device with which you have previously bonded, one of these devices can send a signed command to the other device to authenticate that it knows the shared secret that was previously distributed. The signature is created by using the shared secrets exchanged at bonding, and as such, it cannot be falsified by a third party.

Part of this signed command must be a counter that is incremented for each message sent to prevent replay attacks. When reconnecting to a device with which you have previously bonded, either device can initiate encryption. Every data packet that is transmitted from then on will incorporate a message integrity check (MIC) value that authenticates the sender of that message to the other device by using the previously distributed shared secrets.

Authorization

Authorization is defined as the assignment of permission to do something. This is usually done in two ways:

1. Documentation that provides authorization
2. Authorization that is actioned directly

A concert normally has few, if any, authentication requirements, but authorization to enter the concert is typically enforced by using something called a ticket. These tickets are provided by the event organizer to the guest who presents the ticket at the concert as authorization to enter. There is no authentication that the person holding the ticket is the same person who authenticated the payment for these tickets; in other words, no photo identification is needed. Another example of authorization is the instant approval to allow somebody to do something. For example, if you quickly lent somebody your computer and ask him only to use a single program, you are authorizing him to use that program and no others.

This same authorization model can also be used wirelessly. When something connects with your device, you can authorize it to access certain parts of your device, but not everything within it. Integrity Integrity is defined as the internal consistency and lack of corruption of data. When any data is sent from one device to another, either by using a wired or wireless communications protocol, the data is subject to introduced errors. These errors are important to detect and guard against. From a security point of view, some errors could be introduced by a third-party attacker to attempt to change a valid message into a malicious message. For example, by replaying a captured message but changing one bit in a message that means "lock door" to "unlock the door," the security of a building could be severely compromised.

It should be noted that cyclic redundancy checks (CRC) are used to protect against bit changes, but these are typically too weak to be considered a security measure. It is too easy to change not only a few bits in the message but also a few bits in the CRC to match. To ensure integrity, a much stronger form of message authentication is required that also checks the integrity of the original message. Confidentiality is defined as the intent to keep something secret. The most common

representation of confidentiality is in films when you see the characters handling a company or government reports marked as “confidential.” Unfortunately, this is not a good use of the word because anybody who can see that report can read it. In Bluetooth low energy, confidentiality means that even if a third-party eavesdropper receives a message, she cannot decode it. The process of enciphering a message is called encryption. The enigma machine developed during the Second World War is a classic example of a device that could encrypt or decrypt a message.

Privacy Another security concern that should be considered is how private any communication is. Complete anonymity is difficult to provide. Take, for example, a famous person boarding an aircraft; there are plenty of people who will recognize him just by his face. It is therefore almost impossible to be granted complete anonymity in every location. Wireless communications should not make it easy to track somebody. If the devices that somebody carries are constantly allowing other parties to track his movement throughout a space, there could be some interesting and spooky consequences. For example, stores might give you special offers based on what you have bought in other stores. That’s not so bad, right? But a stalker could seed an area with devices and automatically track you through that area, which is a less attractive prospect than getting a few discounts.

Thus, privacy is the ability to prevent others from recognizing you by the devices that you are carrying, and not to allow them to track your movement throughout a space. Encryption Engine within Bluetooth low energy, there is a single cryptographic block that is used as a one-way function to generate keys and also to encrypt and provide integrity checks. This encryption engine is called the Advanced Encryption System (AES) as defined by the NIST publication FIPS-197. Bluetooth low energy uses the 128-bit version of this, known as AES-128. A generic way of looking at AES is to consider a single function, E, that takes both a key and some plain-text data, and results in a cipher-text data block. AES is therefore a block cipher. The key is 128 bits in length, the plain-text data is 128 bits in length, and the resultant cipher-text is 128 bits in length. It is interesting to note that if the key doesn’t change rapidly, the algorithm is very efficient. Each time a new key is used, lots of calculations need to be performed to set up the internal state of the encryption engine.

After the setup is performed, each new plain-text value that is input can be quickly converted into cipher text. The security algorithms in Bluetooth low energy make use of this property of the AES encryption engine. Shared Secrets virtually all security is based on shared secrets. Your bank card’s PIN is a secret that is shared between you and your bank. Your computer password is a secret that is shared between you and either your computer or your company’s information systems departmental computers. Your house key is a shared secret between the metal that you hold in your hand and the physical formation of the metal within the door lock. Within Bluetooth low energy, there are many shared secrets known as keys. A key is just a shorthand way of saying “shared secret.” There can be plenty of keys, just like you have a car key, a door key (or two or three), a bicycle lock key, or a key to access your work.

There are five main keys in Bluetooth low energy:

1. Temporary Key

2. Short-Term Key
3. Long-Term Key
4. Identity Resolving Key
5. Connection Signature Resolving Key

The Temporary Key Temporary Key (TK) is used during the pairing procedure. It is set to a value that is determined by the pairing algorithm and used to calculate the Short-Term Key. “Just Works” is a mode designed to make a connection to Bluetooth low energy devices possible when very limited user interfaces prevent user entry or verification of pass key values. The TK value when using “Just Works” is zero. This means that there is no authentication being performed and therefore this connection and any keys distributed over it would be vulnerable to man-in-the-middle attacks. “Passkey Entry” is a mode used when the user interfaces on both devices allow at least the display or entry of a number value.

The TK value when using the “Passkey Entry” algorithm would be set to the numeric value that is to be input on both devices. This numeric value is a value from 0 to 999999. This means that there is a significant probability that a man-in-the-middle attack will not guess the same value that is being used by the connection.

There is only a one-in-one-million chance that the right value is guessed. For authentication, this is a reasonable probability; therefore, a key generated using this algorithm is considered an authenticated key, protected from man-in-the-middle attacks. The last TK value is when the “Out Of Band” algorithm is used. This is when both devices have information that has been acquired by using another technology than Bluetooth. For example, if NearField Communication (NFC) was used to transfer a value between the two devices, this value can be used as the TK value for authentication.

A key generated by using out-of-band data is considered authenticated and protected from man-in-the-middle attacks because it is assumed that the out-of-band technology is also not subject to these types of attacks. Short-Term Key the Short-Term Key (STK) is used as the key for encrypting a connection the very first time two devices pair. The STK is generated by using three pieces of information: The Temporary Key is used as the key for the encryption engine, and two random numbers, rand and rand, are contributed by both the slave and master devices in the initial pairing request. Rand and rand are concatenated with the | symbol, as illustrated in Equation and then encrypted using the Temporary Key. The contribution of random numbers by both the slave and the master increases the security of the whole system because any attacker can only contribute 64 bits of the 128-bit random value. It is much harder for a man-in-the-middle attacker to guess which one of 2 64 possible values the peer device used. Figure 11:Discloses the Bluetooth Module and the Channels.

Long-Term Key the Long-Term Key (LTK) is distributed once the initial pairing procedure has encrypted the connection. The LTK can be a random number that is stored in a secure database. It is also possible that the LTK is generated on the slave device. Slaves are by design resource-constrained, so having a security database might be considered too much. To solve this problem,

the slave also distributes two other values: EDIV and Rand. These two values are stored on the master and sent upon reconnection to the slave.

The slave can then calculate the LTK that should be used, or more accurately, the LTK that it had previously given to the master. Upon reconnection to a previously paired and bonded device, the LTK is used to encrypt the link. This means that full pairing is not required every time a device connects.

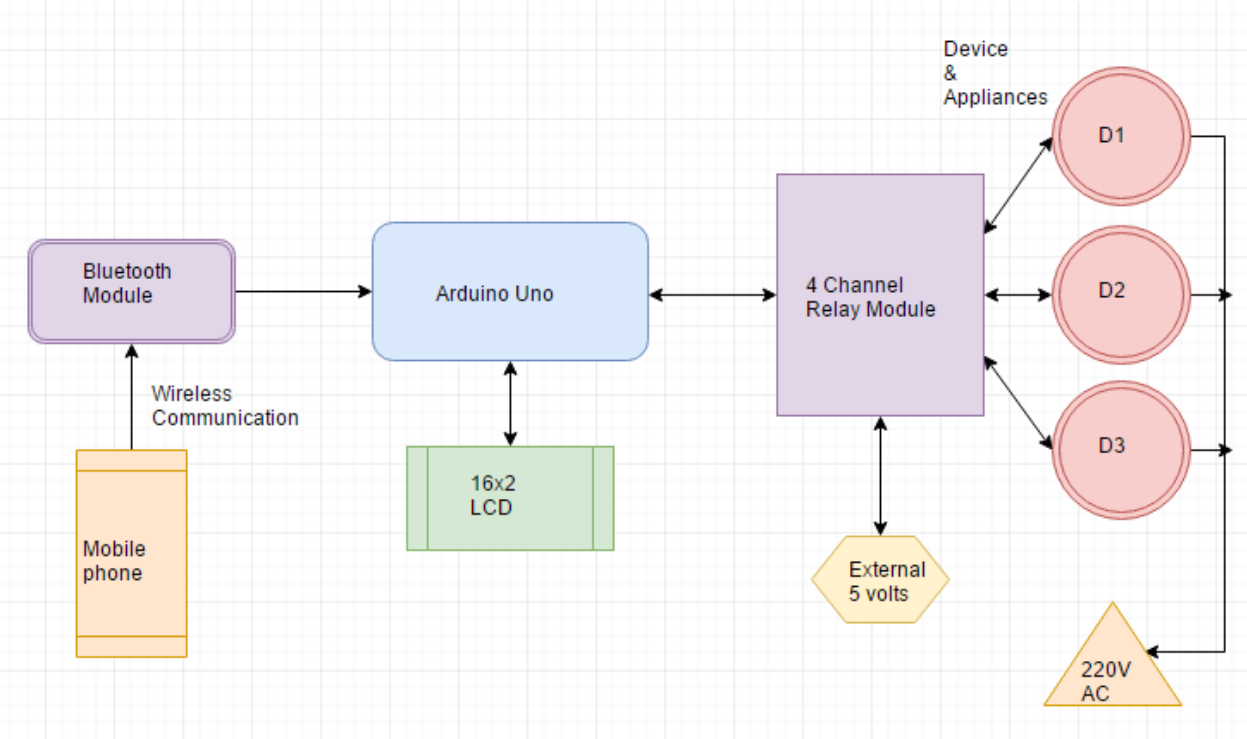

Figure 8.3: Discloses the Bluetooth Module and the Channels.

Identity Resolving Key The Identity Resolving Key (IRK) gives a device that knows a peer device's IRK the ability to resolve (work out) a peer device's identity. Privacy could be performed by always using a fully random address. These devices could change their random addresses at random times, and the device would not be trackable or even connectable by any trusted device. The problem is how to be both private due to using a random address, and also be identifiable by trusted devices. To solve this problem, the IRK is used when generating the random address.

By placing both the random number and the hash into the address field, a peer device that knows the IRK can be checked to see if they match. One way to think of this is to consider a private person who uses a different alias each time he talks to another person. To keep this simple, assume that instead of 2 24 possible name pairs, we have just three. The first time he could be known as "Floella Benjamin," the next time as "Bob Hope," and the next time as "Charlie Dimmock." If you see the names Floella with Benjamin, Bob with Hope, and so on, then you know it is that person. If the name were "Floella Hope" or "Bob Dimmock," it is not that person because "Floella" always

comes with “Benjamin.” Therefore, it’s possible for a device that has a list of IRKs for each separate bonded device to do an exhaustive match on each of these IRKs with the received private addresses.

A match likely means that the correct device has been found. It should be noted that there is no one-to-one correspondence of devices to random numbers and hash values. There are approximately 70,000 billion possible fixed device addresses and, therefore, devices, yet there are only 4 million random numbers that can be used. Therefore, there is a reasonably high probability that for two given devices with the same random number and different IRKs, it would be possible to have the same hash value. This is why privacy is typically combined with authentication to ensure that it is the correct device and not somebody who has the same combination of address parts. Connection Signature Resolving Key the Connection Signature Resolving Key (CSRK) gives a receiving device the ability to resolve a signature and therefore authenticate the sender of the message. The CSRK is distributed from the source of the message to the destination device for the message. Once distributed over an encrypted link, the link can be disconnected. Upon reconnection, because the message only needs to be authenticated and not sent in a confidential matter, the message can be signed.

To sign the data, the CMAC function defined by NIST Special Publication 800-38B 2 is used. This uses the CSRK as the key for this function, along with a Sign-Counter. The SignCounter is a 32-bit value that must be incremented for each message from the source device to the destination device for the duration of the bond. The SignCounter is set to zero immediately after bonding and incremented for each new packet sent, regardless of whether the devices disconnected in the meantime. In pairing and Bonding to enable most of the security features in Bluetooth low energy, two things must happen: The devices must pair with each other, and then once the connection is encrypted, they must distribute keys that can be used to encrypt, enable privacy, and authenticate messages. If these keys are saved for a future time, the devices are bonded. Therefore, to understand how security works, it is essential to understand how the pairing and key distribution system works. It is also important to understand that the initial connection between two devices is different from the subsequent connections between the same two devices.

Pairing two devices that initially have no security but wish to do something that requires security must first pair with each other. Pairing involves authenticating the identity of the two devices to be paired, encrypting the link, and then distributing keys to allow the security to be restarted on a reconnection much more quickly the second time around. The pairing has three distinct phases:

A. Exchange of pairing information
B. Authentication of the link
C. Key distribution.

Exchange of Pairing Information the first phase of pairing involves the exchange of pairing information that is used to determine both how to pair the two devices and what keys are distributed during the last phase. It is important to note that just because there can be very complex algorithms used during the pairing operation, how the devices pair is probably the biggest single opportunity to have the user reject the whole wireless technology.

If pairing is difficult and complex for the user, there is a risk that the user will fail to pair the device the first time around and take the product back to the store. Bluetooth low energy uses the same pairing process as that used for the Secure Simple Pairing feature in Bluetooth classic. Each device first determines its input and output capabilities, together with other paring information. The input and output capabilities are selected from a list of possible capabilities:

a) No Input No Output
b) Display Only
c) Display Yes/No
d) Keyboard Only
e) Keyboard Display

To determine which of these five possible values should be used, the device determines its input and output capabilities and feeds them into a matrix, which is shown in this paragraph. The input capabilities can be either "no input," the ability to select "yes/no," or the ability to input a number by using a "keyboard." The output capabilities can be either "no output" or "numeric output." Numeric output in this context means the ability to display a six-digit number. These five device input and output capabilities are communicated between the devices by using the Pairing Request message. The Pairing Request message is sent as the first security message from a device. This contains not only the capabilities but also other pairing information, including a bit stating if out-of-band data is available, and what the authentication requirements are if any.

It also includes the list of the keys that are being requested to be distributed at the end of the pairing procedure. The authentication requirements include whether bonding is enabled and whether man-in-the-middle protection is required. In response to this, a Pairing Response message or a Pairing Failed message can be sent. The Pairing Response message includes the same information as the Pairing Request. However, if the request indicated that out-of-band data was present but the responder doesn't have any out-of-band data, then the Pairing Failed message is sent instead. The Pairing Failed message can be sent at any time during the pairing procedure to allow devices to fail the current pairing; for example, when a parameter doesn't match what is supported or expected.

Once the Pairing Request and Pairing Response have been exchanged, the two devices can then move to the second phase of the pairing procedure. Authentication using the information from the Pairing Request and Pairing Response, the two devices can deterministically use the pairing algorithm. The two input and output capabilities are used in the following table to determine which algorithm is used: It should be noted that for some combinations, it is entirely possible to lead the users into doing the right thing. For example, if device A has a Display Yes/No capability while device B has a Keyboard Display capability, then device A can display a six-digit number, and the user of device B can see that number and type it into device A. That value can then be used for the TK value. Once the TK value has been determined, a really simple but hard-to-attack procedure is used to help stop a man-in-the-middle attack. A random number is generated by each device, and a confirmation value is also calculated based on that random number, the TK value, the known values of the pairing so far including the device addresses, and the parameters from the pairing request and response messages see Equation.

This value confirms that all the known parameters and addresses used so far are the same on both peer devices. This protects against man-in-the-middle attackers. Both devices exchange the random numbers and the confirmed values; therefore, they can check that the confirmation values match the random numbers and all the other shared information. The interesting twist is that the confirmation values are sent before the random numbers are exchanged. By doing this, an attacker would have to guess which one of the two 128 possible random numbers the peer device would use to calculate the confirmation value before it knows what that random number might be. If the confirmation values do not match the random numbers, then a Pairing Failed message would be sent to terminate the pairing because something was wrong.

If the confirmation values do match the random numbers, both devices have the same input parameters from the pairing request and response, the same address information, the same TK value, and the correct random numbers. Assuming that everything is confirmed correctly, the random numbers exchanged during the authentication are then used to calculate the STK value. This STK value is then used to encrypt the link by using the Link Layer encryption procedures, The Link Layer, Key Distribution Once the connection is encrypted by using the STK, it is then possible to distribute the required keys. These keys are distributed one at a time because, at 128 bits in length, they only just fit into a single packet. The following keys can be distributed:

a) LTK
b) IRK
c) CSRK

The LTK is distributed along with EDIV and RAND because the slave does not have a security database. The slave can use these bits of information to generate the LTK for that master directly. Both the master and slave can distribute all types of keys. This is because, for the current connection, the topology might be that device A is a master and device B is a slave; however, in subsequent connections, the topology might be reserved such that device A is the slave and device B is the master. It is therefore possible to distribute the LTK from the master to the slave so that if they reconnect in a different topology, they can still reconnect quickly. A slight issue that must be considered is that the addresses used during the pairing might not be the actual address of the device; the addresses used during pairing can be either random or private address.

It is therefore also useful to distribute the identity information of the two devices so that this information can be used as the unique value in the database. This allows future connections to be performed by using this identity information rather than a random address that could be out of date. Bonding is a Generic Access Profile discussion; however, it is useful to consider its operation at this point also. Bonding is nothing more than the saving of the keys and associated identity information in a secure database. If the device does not save these values, the devices will have paired but not bonded. If one device saves them but the other doesn't, upon reconnection, only one device will have the LTK, and thus the starting of encryption will fail.

To avoid this prospect, both devices exchange bonding information during the initial pairing so that they know whether the other device saves this bonding information. If the other device does not save the information, then once the attempt at starting encryption fails, the hosts will attempt

to pair again. The signing of Data When a device is connected but not encrypted, it is possible to send data that is authenticated without confidentiality. To do this, a CSRK is exchanged during the very first connection at which pairing occurred. After this point, as long as no data is exchanged that requires confidentiality, signing can be used. To sign the data, the CMAC algorithm is used. This algorithm takes the message to be authenticated, a Sign Counter, and the CSRK is used to authenticate the sender and then generates a signature value. The message that is authenticated is typically an attribute protocol signed Write Command and consists of the opcode, handle, and value that is being written. The SignCounter is a 32-bit value that must be incremented on each new data packet that is sent. This is used to stop replay attacks.

If the same SignCounter is transmitted in a subsequent data packet, it must be assumed that an attacker has received the previous message and is attempting to replay the message in the hope that it might do something interesting. Unfortunately for the attacker, the receiver just discards any message with a SignCounter that is less than the next expected value. It should be noted that the peer device needs to store this next expected SignCounter value for each device with which it is bonded if signed data is supported. The SignCounter value must be included in the message that is sent because the receiver doesn't know if the next message received has the next expected Sign-Counter. Consider the case when this is used to open and close the garage door. The message that commands the garage door to open or close is not confidential; anybody watching from the street will see the car turn up, followed by the garage door opening. If you go out one day far away from the garage and press the garage door remote control button several times, each time it is pressed a new message needs to be generated. By the time you return home and press the button to open the garage door, the Sign Counter value might be higher than what is expected.

The Generic Access Profile It is no coincidence that in no known language does the phrase "As pretty as an Airport" appear. Douglas Adams the final part of the core specification is the Generic Access Profile (GAP). This defines how devices can discover and connect and how they bond. It also describes how devices can be broadcasters and observers and, as such, transfer data without being in a connection. Finally, it defines how the different types of addresses can be used to allow private and resolvable addresses. Background one of the most important things to understand with Bluetooth low energy is how two devices first find one another, work out what they can do with one another, and how they can find and connect repeatedly.

This is really what GAP defines. To illustrate how Bluetooth low energy works, let's consider a typical user scenario: A user has just returned home from the store from which she has just purchased a new heart-rate belt. She already has a low-energy-enabled phone and wants to connect the heart-rate belt to the phone. She has been buying a lot of low energy devices recently; the list includes a low energy enabled television with remote control, a low energy lighting system, and, of course, low energy sensors in her computer, shoes, and watch. How does she con, the heart-rate belt to work with her phone? The user opens the heart-rate belt box and reads the instruction leaflet that tells her to remove the plastic tab from the heart-rate belt to turn the device on. It then instructs her to discover the heart-rate belt. On her phone, she accesses the Bluetooth menu, taps the "Add Devices" button, and then watches as several devices appear on the screen. At the top of the list of devices is the heart-rate belt she just purchased along with an icon of a heart-rate belt.

She selects the heart-rate belt, and the phone moves to its application store and displays a list of applications that work with this device. Some applications are free, some cost money, and some are from the same brand as the heart-rate belt. She selects an application and installs it on the phone. A few seconds later, the application is running and displaying heart-rate information on the screen. The user goes out for a run to test the application. The next day, the user puts on the heart-rate belt again and starts the application she downloaded yesterday. Again, the heart-rate information is displayed on the screen. It's almost like magic she simply uses the application, and the device works. A few days later, one of the user's friends recommends a different application that he downloaded for his heart-rate belt. She goes to the application store, searches for this application, downloads it, and then runs it.

The application again displays the heart-rate information, but now also includes additional information about how hard the user is working out. The application is using the same heart-rate information from the belt, but this time it's using the data differently. This example shows how a user can simply begin by using a Bluetooth low-energy device, and how the flexibility of Bluetooth low-energy services means that they are not tied to a single application to use that device. Initial Discovery to discover a device, you must scan for devices that advertise. Advertising devices transmit packets to any scanning devices by using a many-to-many topology. The problem with this is that every device that is connectable, but not necessarily discoverable, will be scanned.

To solve this, in addition to data, some flags are also broadcast that reveal whether the device is discoverable or connectable. There are two types of discoverability. The first type is used by devices that are "limited discoverable." This is used by devices that have just been made discoverable temporarily. For example, the first time a device is powered on, it would be limited-discoverable, or if the device had a button that allowed it to be discoverable temporarily. A device is not allowed to remain in the limited-discoverable mode for very long. This is because the limited-discoverable mode is intended to allow devices to stand out from the crowd of general-discoverable devices. If devices stayed limited and discoverable for a long time, they would not stand out. Therefore, a device can only be limited and discoverable for about 30 seconds.

The second type is used by devices that are "general-discoverable." This is used by devices that are discoverable but have not been recently engaged in interaction. For example, a discoverable computer, such that other devices can find and connect to it, but has not recently had this discoverability turned on, would be generally discoverable. A device that is searching for other devices would typically place generally discoverable devices lower down the list of found devices because these are probably not going to be as immediately important to the user as those limited discoverable devices are now. Determining device discoverability is the combination of scanning for all devices and filtering on the discoverable flags that each of these devices is broadcasting. It is also possible to use additional filters when presenting the list of devices to a user. For example, the proximity of the device can be used to present the devices that are closer to the search devices at the top of the list and those that are farther away at the bottom of the list.

To do this, the transmit power at which the advertising packets are transmitted is compared with the received signal strength to calculate the path loss of the communication. Devices with a smaller path loss will likely be closer than devices with a larger path loss. Another possible form of filtering

involves using the service information in advertising data to segregate based on what a device does. For example, if the user is looking for a heart-rate belt, then the fact that one device in the area supports this particular service probably indicates that this is the device with which the user wants to connect. There are two types of service information that a device can expose: a device can expose a list of the services that it has implemented, and a device can expose a list of services that it would like a peer device to support. Therefore, with no discoverable devices, limited-discoverable devices, general-discoverable devices, path loss range filtering, and service-based filtering, an intuitive interface can be made for the user. If the user interacts with a device, it appears at the top of the list; if he moves the device closer, it appears nearer the top of the list; if the device supports the services that are being searched for, they will appear closer to the top of the list.

A user just performs the search, looks at the item at the top of the list, and, with high probability, connects to this correct device. One little wrinkle in the device discovery procedure is name discovery. Users don't like to look at hexadecimal 48-bit numbers; instead, they prefer to look at user-readable and understandable names. To allow every device to have a name, the GAP defines a characteristic that exposes the device name. The device name can also be included in the advertising data or the scan response data for devices. To obtain the scan response data, active scanning needs to be used. Therefore, when discovering devices to display names on the screen, active scanning is typically required. This is because the device name is static data that would normally be included in the scan response data and not the advertising data. However, some devices have so much information they need to broadcast that they are unable to include the complete device name in the advertising or scan response data; instead, they include just part of the name or none of the name.

To obtain the complete device name in this situation, a connection must be made with this device and the device name read. Thankfully, the device name has the well-known characteristic type of Establishing the Initial Connection Once the list of devices has been found and a device has been selected, the next step is the initial creation of a connection to the device. This initial connection is performed by initiating a connection to the same device address that was found in the advertising packets. Once the devices are connected, the connecting device performs either an exhaustive enumeration for all the services and characteristics of that device, or it looks only for the service or services that it is interested in, and the characteristics of those services. For example, a phone or computer that has an application store would enumerate all the services that the device exposes.

This service information can then be sent to the application store, and any applications in that store that support those services would be presented to the user. So, instead of the user having to use the application that came with a device, an ecosystem of applications becomes available. Some would be free, some would cost money, some would be made by the manufacturer of the device, and some would be from independent software companies. The alternative method is that the device only performs service discovery for one or possibly a very limited list of services. For example, a television that is connected to a remote control would only search for the human interface device service and possibly the battery service, but nothing else. This means that even if the device

supported more services than these two, the television would never even perform service discovery on them.

This method would typically be used by devices that have a limited user interface or by devices that are connecting to peripherals that only use a very simple set of functionality. The result of service discovery is a list of those services that a device exposes. The client can then use these services. In the application store model, it is the application that takes the next step of characteristic discovery and configuration. Characteristic discovery is just like service discovery in that a device can either enumerate all the characteristics within a service or just use the well-known characteristics that it knows a service must expose. For example, the battery service must expose the battery level characteristic. So, if it doesn't need to discover any characteristics within the battery service, a client can read the battery level characteristic directly.

Service Characterization For a heart-rate belt, the characteristic discovery and configuration might be a little more elaborate. For example, the heart-rate service might expose just the heart rate, or an aggregation of the heart rate and the time intervals between heartbeats, or just the time interval between heartbeats. It can also expose other information that it has calculated from the heart-rate sensor. A client can pick and choose which characteristics to read. Some clients might only be interested in the heart-rate value, whereas others might want to read all the information. For the efficient transfer of data between devices, notifications or indications should be used. To con, the characteristics for which a client wants to receive notifications or indications, the client must write the client characteristic configuration descriptor.

This descriptor enables the required functionality. The device then starts sending these notifications or indications whenever necessary. For the heart-rate example, a client might con, the heart-rate value to be notified, and for that service, the value would be notified once each second, even if the heart rate doesn't change. For other services, the time interval can be more flexible. The battery service, for example, might only notify a value when the value changes. Thus, the application can con, the set of characteristics to be notified or indicated and then wait until the service sends this data through. This means that even if the user changes the heart-rate application, the new application will continue to receive the heart-rate notifications that were previously con,d.

Long-Term Relationships Most of the time, a peripheral is associated with a single central device. Your proximity tag is associated with your phone; your keyboard with your computer; your garage door opener with your garage door. When one device is associated with another, they are said to be bonded. Bonding is a two-step process by which two devices that barely know one another authenticate themselves and share secrets. For bonding to be successful, both devices also need to be con,d to be bondable. A device that is not bondable perhaps because it is already bonded with another device and can only manage a single bond at a time doesn't need to advertise that it is bondable. When both devices are bondable and one of these devices wants to bond, the first step of the bonding process has started.

After this, the input and output capabilities of the two devices are exchanged, an authentication algorithm is chosen based on these capabilities, and the devices authenticate one another. With the link now encrypted, the second step of the bonding process can be performed. This step involves

exchanging shared secrets that can be used when reconnecting to the same device. These shared secrets are typically keys that have various uses: a Long-Term Key (LTK) for encrypting subsequent connections, an Identity Resolving Key (IRK) for resolving private addresses, a Connection Signature Resolving Key (CSRK) for checking the signature of signed attribute protocol commands, as well as the distribution of some identification information sent by the slave to the master so that the slave doesn't have to store the information.

Reconnections Sometimes, devices will discontinue a connection. This might be because they've already sent everything they needed to send and don't want to waste energy maintaining a connection. A light switch will create a connection, send the "turn on the light" command, and then quickly disconnect. Sometimes, the connections might be maintained for significantly longer periods. For example, a keyboard might remain connected to a computer until the computer is turned off, at which point, it disconnects. When the computer is turned back on again, it needs to reconnect to the keyboard. Reconnections are both easy and hard. In Bluetooth low energy, all devices that want to be slaves in a reconnected connection need to be advertising by using connectable advertising events. It might not be discoverable, or it might be either limited or general-discoverable, or it might not allow connections from any device. Also, for a device to connect to an advertising device, the master must scan or initiate a connection to the device with the particular address that is advertising.

This means that the reconnecting device must be in the white list of the scanning or the initiating device if white lists are being used Private Addresses Some complications are introduced if the device that is advertising is using private addresses. A private address is a random address that changes periodically, for instance, once every 15 minutes. This means that even if you discovered a device that is advertising now, you will not be able to determine if that same device is around in 20 minutes, because it might be using a different address. This can at first appear to be an impossible problem to solve. The solution to this problem is a three-step process. The first step is to save an IRK during bonding; the second is to use this key to generate a resolvable private address; and finally, the master must scan for all devices and resolve these private addresses by using all the IRKs and only connect to devices that it believes it has identified.

A resolvable private address is a type of random address that comprises three parts. The first part is a set pattern of two bits to identify that this random address is a resolvable private address. This reduces the computational load on the scanning device so that it only attempts to resolve private addresses on resolvable private addresses. The second part is a 22-bit random number. The third part is a hash of the random number with the IRK that was shared during bonding. The combination of the random number and the IRK means that each private-addressed device effectively has four million possible addresses that fingerprint it.

A master that wants to reconnect to a slave device that is using private addresses must therefore take each resolvable private address that it scans, check with every IRK for each device that it knows could be using private addresses, and then compute the hash value that would have been used for each of these devices, given the random number. If this hash value matches the value used in the resolvable address, then there is a fairly high probability that this is the device identified by that key. This is not certain. Another device with a different IRK with the same random number

may generate the same hash value. However, a quick connection and encryption can quickly check that this is the correct device. The IRK and the LTK used for encryption or the CSRK used for signature authentication will be different for each of these devices, which can quickly confirm that this is either the right device or the low probability that a duplicated hash value has happened and it is the wrong device. Private addresses have some disadvantages.

The biggest one is that the host of the master must perform brute-force checking of every IRK for each resolvable private address that it receives. If that host knows many private devices, this could take some time. The HCI Encrypt command is very useful in this instance, especially if the host has few computation resources available. Another disadvantage is that white lists cannot be used to make connections easily. The only way to connect to private devices is to first scan for the resolvable private addresses, compute if this is a private address of one of the devices to be connected to, and then connect to it manually. This increases the power consumed by the host because the host must perform many computations to resolve the addresses. GAP Roles There are four GAP roles defined for a Bluetooth low-energy device:

1. Broadcaster
2. Observer
3. Peripheral
4. Central

A broadcaster is a device that sends advertising packets. Typically, this is used to broadcast some data from a service to other devices that happen to be in an observer role. A broadcast must have a transmitter but does not need a receiver. A broadcast-only device, therefore, only needs a transmitter. An observer is a device that scans for broadcasters and reports this information to an application. An observer must have a receiver; it can also optionally have a transmitter. A peripheral is a device that advertises by using connectable advertising packets. As such, it becomes a slave once connected. A peripheral needs to have both a transmitter and a receiver. A central is a device that initiates connections to peripherals. As such, it becomes a master when connected. Just like a peripheral, a centre needs to have both a transmitter and a receiver. A device can support multiple GAP roles at the same time. For example, a device can be a broadcaster and a peripheral at the same time. Modes and Procedures within GAP, two basic concepts are used to describe how a device behaves.

These are modes and procedures. When a device is con,d to behave in a certain manner for a long time, this is known as a mode. If a device is con,d to perform a single action that has a finite period over which this behaviour will occur, this is called a procedure. For example, when a device is broadcasting, this is called “broadcast mode.” Broadcasting typically lasts a long time; perhaps that is the single purpose of the device. When a device is looking for broadcasters, this is called the “observation procedure.” Observations typically occur for a very short period to build a user interface or find specific information that is needed.

Within GAP, the following modes are defined: Broadcast mode Terminate connection procedure Bonding procedure to understand GAP is therefore to understand how these modes and procedures interact with one another. For example, the broadcasting mode and observation procedure can be

logically combined to allow observations of broadcasters. Broadcast Mode and Observation Procedure When a device is in broadcast mode, it is using the Link Layer advertising channels and packets to transmit advertising data. This broadcast data can be observed by devices by using the observation procedure. It should be noted that some devices might have only a transmitter and are therefore broadcast-only devices. For these devices, it would not generally be possible to use private addresses or signed data in the broadcast data. Both of these require knowledge of keys that are distributed during the bonding procedure that requires a connection, which requires both transmitter and a receiver. Of course, if these keys could be distributed out of band, then they could still be used; however, there is no standard defined method for this to occur.

Discoverability in GAP, only devices operating in the peripheral role is discoverable. Devices that are trying to discover these devices will be in the central role. A peripheral device can be in one of three different modes used for discoverability: no discoverable mode, limited-discoverable mode, and general-discoverable mode. Discoverability, in the context of GAP, does not mean that a device is advertising, but that the device is discoverable by a user interface on a peer device. A device can transmit advertising packets without being discoverable, as understood by GAP. Therefore, it's necessary to include discoverability information as part of the advertising data for devices to be discoverable in the context of GAP.

This differentiates them from the devices that are advertising or broadcasting data that are not discoverable. To do this, the Flags AD information includes two bits that are used to determine if a device is in non-discoverable mode, limited-discoverable mode, or general-discoverable mode. No discoverable Mode A peripheral that is in nondiscoverable mode cannot set either the limited-discoverable mode or general-discoverable mode bits in the Flags AD information. If no other bits in the flags information are set, then the Flags AD does not have to be included in the advertising data. Nondiscoverable mode is the default discoverable mode; therefore, the host must command a change from this default to one of the other discoverable modes. Limited-Discoverable Mode A peripheral that is in limited-discoverable mode sets the limited-discoverable mode bit and clears the general-discoverable mode bit in the Flags AD.

Limited-discoverable mode is used by a device with which the user has recently interacted, for example, the user just turned the device on for the first time or pressed the connect button, or the user just made the device discoverable in some user interface. Limited-discoverable mode is only allowed to be used for approximately 30 seconds. This means that if another device finds a limited-discoverable device, it can be fairly certain that this is a peripheral with which the user has very recently interacted. Therefore, this device is most likely to be the one to which the user is trying to connect at this moment. Because limited-discoverable mode is primarily designed for devices that are trying to be discovered and enter a connection, it is highly recommended to also include the following useful information in the advertising data that will help build the user interface: Tx Power Level AD to allow sorting of devices found by path loss and therefore range. Local Name AD to allow the name of the device to be displayed. Service AD to allow filtering based on what the device supports Limited-discoverable devices should advertise at a reasonable advertising interval to both allow the user interfaces to be populated quickly and also to reduce the time

required by the master device for scanning for Bluetooth low-energy devices. The recommended interval should be somewhere between 250 milliseconds and 500 milliseconds.

A device that is discoverable and wants to connect should allow connections from any device, and, as a result, white lists would not be used. General-Discoverable Mode A peripheral that is in general-discoverable mode sets the general-discoverable-mode bit and clears the limited-discoverable mode bit in the Flags AD. A device uses general-discoverable mode when it wants to be discoverable. When compared with limited-discoverable mode, general-discoverable devices are very similar, except in the following ways: General-discoverable devices can be discoverable for an unlimited period; limited-discoverable mode has a 30-second maximum time limit. General-discoverable devices have a slower recommended advertising interval of between 1.28 seconds and 2.56 seconds; limited-discoverable devices are between 250 milliseconds and 500 milliseconds. It should note that general-discoverable devices will not be found very quickly.

This is by design; it is always useful to populate a user interface with those devices that are in limited-discoverable mode first and then place those devices that are generally discoverable lower down the list. The same transmitted power-level information should still be included in the advertising data, along with the other user interface-building information, to allow the priority sorting and filtering of the discoverable devices to be used by the connecting device. Discoverable Procedures A central device that wants to find discoverable peripherals either uses the limited-discovery procedure or the general-discovery procedure. These two procedures are essentially identical except for the filtering they perform based on the Flags AD information.

If there is no Flags AD information available in an advertising packet, or this information exists and neither the limited-discoverable mode bit nor the general-discoverable mode bit is set, then the advertiser is non-discoverable and would not be discovered. If the Flags AD information has the limited-discoverable mode bit set, the device is always discovered. This is true regardless of whether the limited-discovery procedure or the general discovery procedure is used. If the Flags AD information has the general-discoverable mode bit set, the peripheral is only discoverable when using the general-discovery procedure. These peripherals are not discovered by the limited-discovery procedure. As shown, the general-discovery procedure finds all discoverable peripherals regardless of whether they are limited or generally discoverable, whereas the limited-discovery procedure finds limited-discoverable peripherals.

Connect ability in GAP, only devices operating in the peripheral role use connectable modes. Devices that are trying to connect to these devices are in the central role and use connection establishment procedures. As with discoverability, connectable peripherals can be in one of three modes: no-connectable mode, directed-connectable mode, and undirected-connectable mode. However, connect ability is more complex for the central device because there are four different connection establishment procedures: auto, general, selective, and direct. From the GAP perspective, connect ability is much easier to manage than discoverability. This mode is the default mode, so the host must act to make a peripheral device connectable.

Directed-Connectable Mode A peripheral device that wants to connect very quickly to a central device uses the directed connectable mode. This mode requires the use of the ADV_DIRECT_IND

advertising packets and, as such, cannot be combined with the discoverable modes because these advertising packets have no host-generated advertising data. Because directed advertising packets are sent very quickly, this mode can only be used for a maximum of 1.28 seconds, after which the controller will automatically stop advertising. The host should not immediately start directed-connectable mode after it has timed out because this can severely restrict the ability for other devices to broadcast, be discoverable, or establish connections.

Therefore, it's recommended that if the directed-connectable mode times out, the undirected-connectable mode should be the fall-back mode. When using the ADV_DIRECT_IND advertising packets, both the current peripheral's device address and the central's device address are included in the packet. This means that the peripheral must have at least been connected with this device previously to know the device's address. Undirected-Connectable Mode A peripheral device that is connectable but does not need to establish a connection very quickly or wants to be connectable while saving as much energy as possible would use the undirected connectable mode. This mode requires the use of the ADV_IND advertising packets. Because ADV_IND advertising packets can include the Flags AD information, a device can be discoverable at the same time as being in undirected-connectable mode.

A peripheral stay in undirected-connectable mode until the host either moves to no connectable mode or a connection is established. Therefore, as soon as the new connection is terminated and the device wants to continue to be connectable, it will move back into the undirected-connectable mode again. A device that was connectable but is now in a connection cannot still be connectable because this would require that the Link Layer support two connections in the slave role at the same time, and this is not a supported state for the Link Layer.

Auto-Connection Establishment Procedure The auto-connection establishment procedure is used to initiate a connection to many devices at the same time. A typical user scenario for Bluetooth low energy is that a central's host is bonded with several peripherals and it wants to establish a connection with each of these devices as soon as they start to advertise. For example, a central device might want the ability to establish a connection to any sensor device that has a new reading when the device starts advertising. To allow a central device to make a connection to many devices at the same time, the host must first populate the white list with the set of devices that should be connected and then initiate a connection to the white list.

Typically, this would be the set of bonded devices about which the host is aware. Once one of these devices is found, because it is using directed-connectable mode or undirected-connectable mode, a connection is established. If other peripherals still need to be connected, the auto-connection establishment procedure would be restarted. There are two downsides to this procedure: the procedure only has one common set of connection parameters that can be used, and this procedure cannot connect with devices that are using private addresses. Because the connection parameters are determined by the center's host when it initiates a connection to the white list, it is not possible to have different connection parameters for different peripherals. This, therefore, works best when the types of peripherals are reasonably similar, at least from a connection parameter point of view. It is possible to change the connection parameters once the connection has been established. So, it might be necessary to initially use a fairly compromised set

of connection parameters when using this procedure. Private addresses also cause severe problems for this procedure. Because private addresses must be changed frequently and randomly to provide privacy, it is almost impossible to predict the private address that a device will be using at any given point in time.

As a result, it's not possible to use this procedure to connect to peripherals that are using private addresses. General-Connection Establishment Procedure In an attempt to solve the problems outlined in the preceding section, the general-connection establishment procedure does things slightly differently. Instead of using white lists, this procedure uses passive scanning to find all the devices that are advertising. For devices that are using resolvable private addresses, these addresses are checked against all known IRKs for the devices to which the central devices want to connect. If the address resolves, the host stops scanning and uses the direct-connection establishment procedure by using the resolvable private address that it received.

If a device address was received that was not a resolvable private address but is instead in the list of peripherals to which it wants to connect, it stops scanning and uses the same direct-connect establishment procedure by using the known device address. The downside of this procedure is that there is a time gap between the host discovered that a device is an advertising and the time when a direct connection establishment starts. This means that at a minimum, the connectable peripheral must send two connectable advertising packets before a connection can be established. This is a natural consequence of using private addresses. This general connection procedure also requires the processing of all advertising packets received by the controller in the host, even if the advertising packets are not connectable or are not from devices that are interesting to the central device.

This can require significantly more energy consumption in the host than the auto-connection establishment procedure. Selective-Connection Establishment Procedure The selective-connection establishment procedure is used to initiate a connection to many devices at the same time, but for which each device has different connection parameters. This solves the single connection parameter problem from the auto-connection establishment procedure. To perform the selective-connection establishment procedure, the host places the set of devices that are to be connected in the white list and then starts scanning by using this white list. This means that only peripheral devices that are on the white list and are advertising are passed up to the host; all other devices that are advertising in the area are filtered out immediately by the controller.

When the host receives the advertising information from the controller for a device in the white list, it can first check that this device was using a connectable advertising packet type. If it was, the host stops scanning and uses the direct-connection establishment procedure to initiate a connection to this specific device. Because each peripheral might want to have a different set of connection parameters, the host can also look up the desired connection parameters for the peripheral that can be used by the direct-connection establishment procedure. This has the same downside as the general-connection establishment procedure in that it takes a minimum of two advertising packets to establish a connection, but it solves the problem of having the same connection parameters as the auto-connection establishment procedure. Unfortunately, this does not solve the privacy issue of the auto-connection establishment procedure regarding the resolution

of private addresses. Direct-Connection Establishment Procedure Many of the previous procedures reference the direct-connection establishment procedure. This procedure is used to establish a connection to a single, specific device by using a set of connection parameters.

It does not use the white list; instead, it initiates a connection directly to a single device address. This is why the general and selective connection establishment procedures reuse this procedure to make the actual connection, once they have found the device address of the device that is advertising. Bonding Just like discoverability and connectivity, bonding defines modes and procedures, except that there are only two modes: nonbondable mode and bondable mode; and just one procedure: the bonding procedure. Nonbondable Mode The default mode for devices is the nonbondable mode. This means that a device will not accept bonding at this time. No keys will be exchanged or stored by the device.

Bondable Mode If a device wants to be bondable, then it is in bondable mode. When a device is in bondable mode, it will accept a bonding request from a peer device. To be in bondable mode, the bonding bit is set in the authentication requirements of the Pairing Request message during pairing. If possible, the device exchanges security keys and stores them if necessary. Bondable Procedure if a device wants to bond with a device it believes is bondable, it uses the bondable procedure. When using the bondable procedure, the device initiates pairing with the same bonding bit set in the authentication requirements of the Pairing Request message. Therefore, for bonding to work, the device that uses the bondable procedure initiates pairing with the bonding bit set. If the peer device is bondable, it will respond with the bonding bit set. If all this happens, the keys will be distributed after the link is encrypted, and the keys are then stored. Once the keys have been distributed and stored, the devices are bonded.

CHAPTER 9

DIFFERENCE BETWEEN BLUETOOTH AND ZIGBEE

Vinay Kumar S B, Assistant Professor
Department of Electronics and Communication Engineering, Faculty of Engineering and Technology, JAIN (Deemed-to-be University), Ramanagara District, Karnataka - 562112, India
Email Id- sb.vinaykumar@jainuniversity.ac.in

Advertising Data Whenever a device transmits advertising packets, the advertising or scan response data has a defined format. The format is just a sequence of advertising data structures. Each structure starts with a length field, which determines how many more bytes of data are part of this structure. Immediately after the length is the advertising data type. This is normally just 1 byte in length but could be 2, 3, or more bytes if necessary. A device that doesn't understand a given advertising data type can just ignore it and skip to the next structure. Any additional data bytes within the structure are defined by the data type. For example, the TX Power Level data type defines that the data, after the length and data type, is a single byte in length. This means that to output the TX Power Level in an advertising or scan response packet requires 3 bytes in total: one for the length, one for the data type, and one for the actual power level.

Some advertising data can be variable in length. The local name of a device could be anything from a few bytes to many bytes in length. The length field at the start of the data structure, however, binds the number of bytes thatare being output in the advertising data. Therefore, it is not necessary to include any terminating bytes. It is also possible to not include the complete value of a given value, especially if that would overflow the packet size or remove something else just as valuable from the packet. Most data that could be expected to overflow is typically represented by multiple data type values: one for the complete value if possible, and another for the shortened version. This way, the receiver can determine that the value is not complete, and thus, it can attempt to find the rest of this data through some other means.

For example, the local name might be longer than is possible to include in a short advertising packet, but it is also possible to read the complete name by using the GAP's Device Name characteristic. Flags The Flags AD is a sequence of bit fields that can be any length, from zero to many bytes long. Any bytes that are not included in the advertising data are assumed to have the value zero. This means that the flags field can be extended with additional flags bits as necessary. A device that receives a Flags AD that is longer than it expects can just truncate the data without a problem. The following flags are defined for Bluetooth low energy:

1. Limited-discoverable mode
2. General-discoverable mode
3. BR/EDR Not Supported
4. Simultaneous LE And BR/EDR
5. To Same Device Capable (Controller)
6. Simultaneous LE And BR/EDR

To Same Device Capable (Host) The BR/EDR Not Supported flag bit is used to notify a peer device before making a connection that it cannot make a connection by using Bluetooth classic; instead, it must use Bluetooth low energy. This is important because a dual-mode device cannot make a connection to another dual-mode device by using Bluetooth low energy. Therefore, a dual-mode device must check this bit to determine how it should initiate a connection to the device. The Simultaneous LE And BR/EDR to the Same Device Capable flag bits one for the controller and one for the host are used to determine if the peer device can initiate a connection over Bluetooth low energy if a Bluetooth classic connection already exists to that device. The controller and host might have different capabilities, and, therefore, both bits must be examined before two connections to the same device are attempted. Service there are multiple, varied types of service advertising data types. This advertising data type exposes a list of service UUIDs, one for each service. UUIDs come in two different sizes: 16-bit values or full 128-bit values. Some devices will not want to expose all the services that they support, or the complete list of services might be too long. So, both a full list of services and a partial list of services must be supported.

Therefore, it is necessary to expose four different service advertising data types:

1. Complete list of 16-bit service UUIDs
2. A partial list of 16-bit service UUIDs
3. Complete list of 128-bit service UUIDs
4. A partial list of 128-bit service UUIDs Local Name The local name advertising data type comes in two variants:
5. Complete Local Name
6. Shortened Local Name

If the local name is too long to fit into the advertising packet, the Shortened Local Name data type will be used. The local name is a UTF-8 string, possibly truncated. TX Power Level the transmit power level advertising data type is used to expose the transmit power level used to transmit this information in the advertising packet. It consists of a single byte of data delineated in dBm. Slave Connection Interval Range The slave connection interval range exposes the preferred connection interval range for any subsequent connection to this peripheral. The master is in complete control of the connection parameters, and these parameters are typically set before the directed connection establishment procedure is used. Thus, this information is useful to allow the central to at least determine a range of connection intervals that it should use when connecting to the peripheral. The interval range consists of two 16-bit values. The first value is the minimum connection interval; the second value is the maximum connection interval. Both values are in the same units as those used in the HCI command used to initiate a connection. Service Solicitation Sometimes a peripheral wishes to use a service on a central device.

Unfortunately, the central device might not want to advertise, or the peripheral might not be designed to spend energy scanning for all possible central devices. The solution to this is to include a list of services that the peripheral would like the central device to support. This solicitation of services gives a central device that is looking for peripherals the ability to connect, and it can use this information to determine which peripherals are more likely to include the client functionality for one of its services. The service solicitation advertising data consists of a partial list of either

16-bit service UUIDs or 128-bit service UUIDs. Service Data When a service is con,d to broadcast, it uses the service data advertising data type. The first two bytes of the data are the 16-bit UUID for the service that is broadcasting the data. Any additional bytes are the actual service data that is being broadcast.

Manufacturer-Specific Data The last advertising data type is manufacturer-specific data. The first two bytes of the data are a 16- bit company identifier for the ensuing data, followed by data that is specified by that company. This means that any company can define its own advertising data structures and expose them in an advertising packet. GAP Service GAP defines its own generic attribute profile service, the Generic Attribute Profile, which provides a device with a way to ascertain information about the device, including its name, what it looks like, and how to connect to it. The service exposes up to five characteristics:

1. Device Name
2. Appearance
3. Peripheral Privacy Flag
4. Reconnection Address
5. Peripheral Preferred Connection Parameters

The Device Name Characteristic the Device Name characteristic is a UTF-8 string that exposes the name of the device. Interestingly, a device can only have one Device Name characteristic, so it's possible to use the Read by UUID Request to quickly read the device's name without having to perform service discovery or characteristic discovery first. The Appearance Characteristic the Appearance characteristic is a 16-bit value that enumerates what the device looks like. Bluetooth Classic had a class of device fields that could be discovered during the inquiry procedure, but this was a mix of what the device did and what it looked like. This has proven to be a real problem because some manufacturers filtered on bits in the class of device, creating interoperability problems. For Bluetooth low energy, the chosen solution was to just concentrate on what was the most important information to build up a user interface rather than attempting to do filtering as early as possible.

Therefore, the appearance is only allowed to be used to drive the icon that is displayed on the user interface next to the device. The value in the characteristic is an enumerated value that is defined in the assigned numbers document, which is a living document. New values can be added to this enumeration when required by just requesting an assignment to be made. The Peripheral Privacy Flag Peripheral Privacy Flag is used to expose if the device is currently using privacy. It is both readable and writable; this is where the use of this characteristic becomes confusing. If this characteristic does not exist, this device does not support privacy and will always use a public Bluetooth address. If this characteristic does exist but is set to "disabled", again, this device is not using a private address and therefore will use the public Bluetooth address.

If this characteristic exists and it is set to "enabled", it will always advertise by using a private address. However, that doesn't mean that it will be connectable by using a private address. If the Reconnection Address characteristic exists, the address used when making a connection will be this reconnection address and not the advertising address. This means that only when the

reconnection address exists and the privacy flag is set to “enabled” is this device truly able to use private reconnections. A central device might not want to use privacy with a device, or it might want to use privacy with a device that currently has the Peripheral Privacy Flag set to “disabled”. In either case, the central device can attempt to write this characteristic.

However, this writer might not succeed. If the peripheral has more than one central device bonded with it, having the second or later central device able to enable privacy would mean that the first bonded central device would not be able to reconnect. Reconnecting would fail because it would be attempting to reconnect to the public address of the device and not the value it last read in the Reconnection Address. Disabling privacy can also cause similar problems. This direction is not as dysfunctional as the central device that still thinks that privacy is enabled on the peripheral because the central at least can detect that the peripheral is using its public address in advertising packets, allowing it to reconnect to this address. Upon reconnecting, it can check the value of the Peripheral Privacy Flag to confirm that privacy has been disabled. Therefore, the only safe way to use privacy is to have it always enabled from the start of the peripheral device being connected. It is not useful to constantly change the privacy state. Reconnection Address As explained in the preceding section, the Reconnection Address is used when privacy is enabled. This is the address that a peripheral device uses when it attempts to reconnect to a central device that it is bonded with, and the central device knows that the peripheral has privacy “enabled”. When used, the Reconnection Address will be a no-resolvable private address.

Therefore, the only devices that can make a mapping from this reconnection address to a peripheral are the devices that are bonded with the peripheral and have read the reconnection address. The reason that the reconnection address is part of the system is to allow for power optimization when privacy is used. If there were no reconnection address, all private peripherals would advertise by using a resolvable private address. The centre would then need to check every received advertising packet with a resolvable private address against every IRK that it knows. This is an expensive operation, typically because the host must get involved. By using a static no resolvable private address, the value is static and can therefore be placed into the white list to allow the controller to efficiently establish a connection to the peer device without using the power-hungry host. The interesting thing about the reconnection address is that for it to be considered private, it must be changed on each reconnection.

Therefore, each time that a central device reconnects to a peripheral using the reconnection address, it will write a new reconnection address into this characteristic before doing anything else. There is a possibility that the connection fails during this write request. In this case, the peripheral might have received the written request and not be able to send back the written response. Alternatively, the peripheral might not have received the written request. Therefore, the central must assume that the peripheral could be using the old reconnection address or the newly written reconnection address. It will therefore have to place both of these addresses on its white list. Peripheral Preferred Connection Parameters many peripherals have been designed for a single use case and have a preferred set of parameters with which they work optimally. Instead of having the central device guess what these parameters are, the peripheral exposes them in the Peripheral Preferred Connection Parameters characteristic.

The central can then read them on the initial connection and quickly change the connection parameters to something that the peripheral likes. Also, when reconnecting to a peripheral for which the central has remembered the parameters, it can place the appropriate values into the connection request, removing any need to change the connection parameters after the connection has been established, Central describes how an application can use Bluetooth low energy to interact with peripherals. Peripherals, explains how a peripheral can be designed to make the best use of the Bluetooth low energy technology, Testing and Qualification, explains how to qualify a design so that it can be sold or distributed Central one is and is not in the centre of the maelstrom of it all.

Harold Pinter The application of Bluetooth low energy can be broken into two separate disciplines: the design of applications that find and interact with peripherals, and the design of peripherals that can provide information to these applications running on a central device. Both disciplines require knowledge of how all the previous parts fit together to make a functioning whole. This chapter looks at the applications from a central point of view. The next chapter subsequently deals with how a peripheral can be designed. Background Central devices are vitally important for Bluetooth low energy to be useful. Typically, these devices are highly functional and have complex user interfaces.

This chapter will not go into how to write an application for a particular type of device; that is the job of that platform's developer program. Instead, important considerations will be given on how to optimize the user experience and the best ways to save power. Discovering Devices the first thing that any central device needs to do is to discover other devices. To do this, one can use either passive or active scanning. For passive scanning, a central device passively listens to any advertisement packets that peripherals are transmitting. Active scanning is when the central device, after hearing a peripheral, asks for more information. If the central device is only looking for what devices are around, and perhaps any information they may also contain, it should use passive scanning.

If the central device is also populating a user interface such as a screen or a window, active scanning should be used because the additional information can be useful to build the list of discoverable devices on a user interface. The information that can be found by scanning includes the name of the device and a unique number that identifies the device. This identity can be used if the central device later needs to connect to it. It is also possible to find some broadcast data within the scan responses so that information that is being broadcast by a service can be obtained. This could be useful information such as the battery level or the current time. Some information is not immediately available but can be obtained by making a quick connection to the device and reading it as needed. For example, the complete name of the device might be very long and might not fit into a single advertising or scan response packet.

Therefore, it would be necessary to make a connection with the device to read the rest of the device's name. When doing this, the application on the central device should be very careful about only reading characteristics within the correct service. Therefore first, the application should perform a service discovery of the required service, for example, to look for the GAP Service. It can then look within that service for the Device Name characteristic value. This can be read directly using the Read Characteristic Value by UUID procedure. The GAP Service specification

requires that only a single GAP service can exist on a device, and it will have only one device name characteristic value. If the device name is longer than what can be read in a single attribute protocol response, additional requests for the rest of the device name value can be made by using the handle that is returned in the first response.

This same procedure can be used for other useful information, such as the Appearance characteristic value in GAP. It should be noted that when passively or actively scanning, not only can the application obtain the contents of advertising packets but it can also receive the received signal strength (RSSI) of these packets in the controller. This RSSI value can be subtracted from the Tx Power that might be included within the advertising packets to give a very basic estimate of the path loss and therefore an estimate of the distance between this device and the central device, as illustrated in the following equation:

$$\text{path loss} = \text{TxPower} - \text{RSSI}$$

If the path loss is very small between 0 and 20 this indicates that the device is very close. If the path loss is very large, for example, above 70, the device is very far away. Because of the nature of wireless transmissions, these values should be averaged over several seconds so that any multipath interference is averaged out. Once this is done, the list of devices presented to the user should place the closest the smallest path loss at the top of the list. This way, the user will preferentially see those devices that are very close. When there are lots of devices, the user will need to scroll down to find those devices that are farther away.

Another consideration is that some devices might be advertising but not discoverable. Unless the user interface is displaying the broadcast service data, the devices that are not discoverable should be removed from the list of found devices. Connecting to Devices In the preceding section, the list of discoverable devices was created. This list can be presented to the user so that he can select a particular device (or devices) with which he wants to interact. The next task will be connecting to the device. Connecting to a device selected from a user interface should just be a case of initiating a connection to that device.

If, however, the chosen device is using a private address, care should be taken to recover gracefully if the device found has recently changed its private address. If that does happen, the initiation of the connection should be a timeout and the list of discoverable devices should be refreshed. The device might still be there, but it's using a different private address because it wants to protect its privacy. This is not a problem once the devices have bonded and exchanged Identify Resolving Keys (IRKs) because the central device can automatically refresh the list, resolve the private addresses to identify which one belongs to the desired device, and then connect to it. When initiating a connection, a set of connection parameters need to be chosen. The parameters used depend on what the two devices are intending to do.

Typically, peripherals have a Client Preferred Connection Parameters characteristic that gives a very strong hint to a central device about the types of connection parameters it prefers. When making the very first connection with a device, this information will not be available, and therefore a compromise between low power consumption and rapid characterization of a device needs to be struck. The best set of parameters is those that have a fairly rapid connection interval, allowing for

a rapid exchange of Attribute Protocol and Link Layer control messages during the initial connection. A reasonably large slave latency should also be offered so that the peripheral device can save power whenever possible.

For example, a connection interval within the range of 15 milliseconds and 30 milliseconds, with a slave latency of 150 milliseconds, allows for both rapid collections of data about the peripheral using up to 60Hz connection interval and a 6Hz idle frequency for the slave. The slave might request different parameters from those that the application on the central device chose. The central device should always try to honour these requests, especially after it has finished reading all the data that it wants at that moment. It is always possible for the central device to change the connection latency again, once it has more data to send or receive connecting to a peripheral device, the central device will want to know what the device does. To gather this information, it uses four procedures, in a specific order: primary services discovery, relationship discovery, characteristic discovery, and descriptor discovery. The first procedure is the primary services discovery.

These are the services that describe what the device does. For example, if the device has a battery, primary services would expose the Battery Service; if the device has a temperature sensor, it would expose the Temperature Service; if the device had a temperature sensor within the battery, this secondary temperature would not be exposed through a primary service because this might confuse the central device. Primary services only expose the primary functionality of a device. Next, for each primary service that the central device knows could include another service, these relationships need to be discovered. These relationships could be because of an extending, combining, or reusing relationship. The set of services that a device has does not necessarily determine the set of profiles that a peripheral device supports. There is no way to quickly determine the set of profile roles that a device supports.

Instead, a more complex algorithm has to be used, matching the profile roles that the central device supports with the set of services that are exposed on the peripheral and checking which roles are valid. This could be a nontrivial operation; however, the devices that would be doing these checks have plenty of resources, and this type of complexity is not considered a Bluetooth low-energy issue. The benefit of this approach is that future client profile roles that use a set of services on a peripheral don’t need to be designed into the peripheral when it is manufactured. This becomes a very flexible and extendable system ideally suited to the downloadable application models being deployed in many central devices. Once the services have been discovered, the set of characteristics and their descriptors can also be discovered.

There are no version numbers in Bluetooth low-energy services; therefore, the only way to know if a given optional feature exists is to check for the exposure of a given characteristic that is linked to this optional feature. Alternatively, characteristic properties and descriptors such as the notification and Client Characteristic Configuration descriptor can be used to differentiate optional behaviour. Generic Clients It is possible to build entirely generic clients. These are clients that can read and display characteristic values, possibly in a human-readable format. This gives central devices that have no understanding of the meaning of the individual services or characteristics on a peripheral device the ability to make them available to the user. There are two levels of generic

clients: those that use the available information on the peripheral device directly, and those that augment this information with characteristic information available via the Internet. The first level of generic clients finds all the characteristics within a peripheral and filters out any that cannot be read directly.

It then also filters out any that do not have a Characteristic Presentation Format descriptor. This descriptor includes most of the information needed to change the binary representation of the data into a human-readable value. It does this by using the format and exponent fields to determine how to convert the value from a fixed-point value into a more intuitive number. The unit field is a UUID that encodes the unit of this value. There are units for most physical quantities, which are taken from the BIPM1 list of units. All standard SI units and most common units are included. Finally, a namespace: description pair is included in the descriptor to allow for an even finer-grained client display.

This could allow not only for the weight of the item being measured to be converted from a 16-bit unsigned value into pounds weight but also noting that this weight is from a hanging weighing machine as opposed to a vehicle weight bridge or a set of bathroom scales. The second level of generic clients can display the most complex values in a characteristic. The Characteristic Presentation Format descriptor is limited in that it can only represent a very small subset of all possible data structures. Therefore, the second level of generic clients does not rely on that descriptor. Instead, it uses the knowledge that every character type has a unique number, the UUID that can be looked up on the developer website of the Bluetooth Special Interest Group (SIG) 2 to find an XML representation of the data format. For every characteristic type in any service specification, there must also be an XML file defined for that characteristic. These files are used to help validate devices when they are being tested, but they can also be used to determine the structure of the characteristic value.

The XML files can encode every possible data representation required, including enumerated types, bitfields, optional fields based on the value of an enumerated value or a bitfield, and binary and decimal exponential fixed point formats. These fields can also be concatenated together to make very complex data structures. A generic client with a connection to the Internet can therefore find a readable characteristic value and perform a simple query of a website to download the characteristic representation's XML file. This file can be used to display the value to the user. 13.6. Interacting with Services Once the central device has determined that it is necessary to interact with a service on a peripheral device, it makes a connection to that device and starts to read and write characteristic values and descriptors. The protocol used to do this, Attribute Protocol, is essentially stateless.

The protocol has no state when connected or when between one connection and the next. There is no "session protocol" either. To get around this limitation, all state is maintained at the application layers, where the applications can make intelligent decisions on how to save energy. The next sections describe how this is done. Readable Characteristics The most basic of services simply exposes a set of readable characteristics. For example, the Device Information Service contains one or more characteristics that provide additional information about a device. These basic services are easy to use. For each character in the service that the client understands, it reads the value,

either in a single request if the value is short, or by using multiple requests if the value is long. Writable Characteristics the next level of complexity is a service that has a characteristic that is both writable and readable.

The Link Loss Service is a good example of this type of service. It has a single writable characteristic, Alert Level that the client can write to con, the behaviour when the link between the two devices is lost. If the client writes “No Alert” into this characteristic, when the devices disconnect, the server does nothing. If the client writes “Mild Alert” into this characteristic, when the devices disconnect for any reason, the server will use a mild alert to the user to notify her of this occurrence. If the client writes “High Alert” into this characteristic, when the devices disconnect, the server will use as many bells, whistles, flashing lights, and other “alert” methods that it has at its disposal. Some people who have experience using connection-oriented protocols can be worried by this. The key element to understand here is that the state of the service is exposed in the service characteristics. The Alert Level characteristic in the Link Loss Service determines the device's behaviour. If this holds the value “No Alert”, the server will do nothing. You could consider this the “not-connected” state. If this holds the value “Mild Alert” or “High Alert”, the server will do something when the client disconnects.

You could consider that when this service’s characteristic holds one of these values, it is “connected”. Of course, this implies that if the client wants to gracefully disconnect from the server that has a mild or high alert level, it must change the alert state by writing “No Alert” to this characteristic before it disconnects. Many people will also ask about what this means if a server can have more than one connection at the same time, and one client writes the value “Mild Alert” to the link loss service’s alert level characteristic, but the other doesn’t and then disconnects. This is not a problem. The value for each characteristic of each service can be different for each client. If client A writes “Mild Alert”, this does not mean that the server will alert when client B disconnects because the value for client B is still “No Alert”.

An important difference between this type of service and those described in the following sections is that these services have characteristics that can be readable and writable. This means that a client can always check the current state of these services without having to remember what they had done previously. This is most useful when the client application unexpectedly terminates, perhaps during debugging of the client application software. When restarting, the client can just refresh their knowledge of the state by reading the appropriate characteristic values.

Control Points

Another type of service is one that holds no state, but the client can still write values to the service. This might appear strange at first, especially considering the services described in the preceding section were holding states. How can a service have a characteristic that is writable and not holds on to that state? The answer is easy: The service uses the value written immediately, and the server does not have any need to store that value after it has been consumed. This type of character is called a control point. The previously described Link Loss Service had an Alert Level characteristic that could be written that determined the behaviour when the two devices disconnected. It could write the appropriate value into this service and then disconnect. But this is

incredibly disruptive to any other applications that are also using other services on this device that might need the connection. Of course, there is, and it uses a control point characteristic.

The service is called the Immediate Alert Service and the characteristic is called Alert Level. Yes, it is the same alert level characteristic that was used in the Link Loss Service. But characteristics are simply a data format, in this case, an enumeration of three values, “No Alert”, “Mild Alert”, and “High Alert.” The behaviour is determined by the service, not the characteristic. In the Immediate Alert Service, the Alert Level characteristic is only writable, and it causes an alert immediately. Because the alert is immediate, this characteristic cannot have a state. Any value written is immediately consumed, used to make an alert, and not stored. Therefore, there is no point in making this characteristic readable. The characteristic has no state. There is another type of control point that is discussed in the Notifications and Indications subsection that follows. The advantage of this type of control point-based service is that it doesn’t matter which slave commanded the control point. Instead, the device will act upon the command written into the control point.

State Machines the next type of service expose a few writable control points along with one or more readable characteristics. These expose the state of a state machine. A state machine in this context is a “machine” that has an exposed state and a way of internally or externally changing states. Essentially, the only difference between a state machine and a control point described earlier is that the state is remembered in a state machine. This state can therefore be read and notified to the client if it changes. To help illustrate this, let’s consider a state machine for time synchronization. It would have the current state: the machine is doing nothing, and the machine is busy trying to find a more accurate value for the current time. Let’s label these states “Idle” and “Searching.” This is the exposed state of the state machine. Next, we need a way for the device to control the state machine.

This is done by using a control point. This is a control point as described just a moment ago, except that the control point is connected with the state machine that has an external state. For example, with the time synchronization service, this could be enumerated with two commands: Start Synchronization and Cancel Synchronization. This has several advantages. Any device can interrogate the current state of the state machine through the machine’s exposed state characteristic. This means that if three devices all want to synchronize the time at the same time, they can all check the state and only command it to start a new synchronization if the machine is in an idle state. More realistically, each of the devices can just command the state machine to start synchronization.

This should not be a problem with a well-defined state machine that can continue to operate even when given potentially invalid commands. For example, if the state machine stays searching when it was commanded to start synchronization, it would be safe to have multiple clients all ask it to start synchronizing any time they want. This is much preferred over the alternative of sending error responses to all clients when any sends a second start synchronization command and then having to instigate some form of random back-off procedure to start synchronizing again. The state machines must be sufficiently robust that all commands will have a defined state transition. For example, if the time synchronization service was idle and a client asked it to cancel

synchronization, the state machine could take this as an error, or it could accept that the client was unaware that it was already in the idle state and do nothing. This latter approach is typically required in state machines, so all commands should be safe for the client at any time. However, this scenario points up an interesting side effect: When one device asks for synchronization to start, it has no guarantee that synchronization will complete because it might be cancelled by another device. As an alternative, a service could be defined in which synchronization would always run to completion once started, unless cancelled by that client.

Due to restrictions on battery consumption, wirelessly protocol is not generally recommended for long-term applications. Therefore, a new wireless information protocol was required that could handle functioning with minimal power consumption for tiny devices that operate on batteries or restricted power sources. To meet these needs, a protocol called Zigbee was introduced in 1999, just as Bluetooth was beginning to gain popularity. Zigbee, however, gained popularity for business applications in 2005. Although described in the same IEEE standard of 802.15, it is interesting that these ISM band-using technologies, Bluetooth and Zigbee, are substantially different on several fronts. The distinctions between the Headset and Bluetooth low-energy protocols will be discussed in depth in this article.

a) Origin: German telecom Major Ericsson introduced Bluetooth in 1994, while Zigbee was developed in 1998. While Zigbee derives its name from the dancing moves of a beehive, Bluetooth takes its name after the famous European unification king Harald Blat.
b) IEEE Standards: Zigbee and Bluetooth were both developed under the IEEE 802.15.1 standard, even though Bluetooth is no longer an IEEE standard.
c) Management Alliances: The Zigbee Alliance oversees the management of the technology and certifies and tests Zigbee-based products. Under IEEE, all Zigbee-based protocols are standardised. However, the Bluetooth SIG is the only organisation in charge of managing Bluetooth's standards and devices Special Interest Group.
d) Updates to the protocol: Versions 2.0, 2.1, 3.0, and 4.0 of Bluetooth have been released, whereas versions 2004 through 2007 of Zigbee have. Currently, there are two varieties of Zigbee: Zigbee and Zigbee Pro, which have distinct addressing algorithms.
e) Wi-Fi operates at frequencies of 2.4 GHz and above, whereas Bluetooth operates at 2.4 GHz Frequency, 915 MHz, and 2.4 MHz.
f) Channel Bandwidth: Frequency and duration where the connections are communicated, wirelessly channels may use up to 1 MHz of bandwidth, while Zigbee-based networks can use up to 3 MHz of bandwidth.
g) Zigbee was created to interact inside a Personal Area Network, but Bluetooth-based communication works for Wide PAN (WPAN).
h) Max number of all cell nodes: Up to 8 cell terminals can be paired together using Bluetooth, while over 65,000 cell nodes may be linked using Zigbee.

Range: Wi-Fi-based networks have a range of 10 to 100 metres, compared to Bluetooth networks' maximum range of 10 metres. Self-Healing Networks: These networks ensure that networking continues even if a node sustains damage. In self-healing networks, the nodes next to the injured node will initiate a reversing loop, blocking the transmission of any data to the damaged or non-operational node. The network is maintained and made more stable as a result. While Bluetooth

has self-healing network technology described in it, there are still modifications and research being done for it, Zigbee based systems are intrinsically known to support this technology.

Bluetooth networks may be set up using the point-to-point master-slave technique, piconet networks with a master and up to seven slaves, and scatter networks. Zigbee devices are, on the other hand, maybe interconnected in star, mesh, and other standard topologies. A cluster of several Zigbee-based network topologies may be formed by connecting them. Three different sorts of nodes make up a Zigbee network: a coordinator, a router, and an endpoint. Data Transmission Rates the minimum data transfer rate for Bluetooth-based networks is 1Mbps, whereas the greatest data transfer rate for Zigbee is 250Kbps. The procedures by which a signal is spread throughout the frequency domain are referred to as "spread spectrum techniques." This is done to integrate more signals, improve signal security, lessen signal interference, etc. Most wireless connection designs have some kind of spread spectrum strategy stated in them.

Zigbee employs the Direct Spread Spectrum approach, while Bluetooth uses the Wideband Spectrum technique. Frequency hopping involves changing the carrier signal's amplitude while using a direct spread spectrum; at the transmitter to the receiver, the output waveform is scrambled and recovered from a pseudo-random code, respectively. Energy Consumption: For every network, the amount of energy used is influenced by the kind of data being transferred, the separation between transmission and reception, the amount of power that is wanted to be kept by the signal, and other elements. The Headset protocol is well recognised for transmitting practically any sort of data, including text and video. Contrarily, the Zigbee protocol is specifically for operating instructions and is known to communicate a limited range of data.

Servers, Clients, and Databases

A database is a collection of characteristics. A database may be extremely tiny and basic, with a minimum of six properties, very massive and sophisticated, or any size in between. However, it is how those attributes are utilised in services and profiles that make the attribute database complicated, not the attribute layer. An attribute server always houses the database, while an attribute client communicates with the server via the Attribute Protocol. Regardless of whether Bluetooth low energy or Bluetooth classic is used to connect to the other device, there is only ever one attribute server on each one. Each device only has one attribute server, which also means that each device only has one attribute database. The attribute database has a Generic Access Profile service that must be supported for a Bluetooth low-energy device. This implies that an attribute server and an attribute database are present on every Bluetooth low-energy device.

A good illustration of an attribute database

This indicates that the cost of disclosing a little quantity of data on a device, for example, only the battery level is very low.

Because every device already has an attribute database, introducing a service to expose this information merely adds three or four extra attributes to the overall cost. Each device comes with six qualities as a minimum, therefore it is rather simple to add three more attributes for the battery service.

Attribute Permissions

An attribute server may have information in certain attributes that may only be read or written. Every attribute in an attribute database has permission to help with these access limits. Access permissions, authentication permissions, and authorization permissions are the three primary forms of permissions. What kinds of queries may be made on a certain property depend on access rights. Recalling our previous examples, the state of a phone conversation may only be readable, the state of a light source might be readable and writable, and the light control point might be writable only. Similar to how a phone call requires permission to read its status and a light control point needs authentication to write it, the state of light may be accessible to everyone but can only be written by authorised devices.

Attribute permissions only apply to the attribute value, it should be indicated at this point. The attribute handle and attribute type are exempt from their application. The handles and types of every attribute that a device exposes are all accessible to other devices. This enables devices to check whether another device supports a feature before authenticating and requesting authorisation. For instance, it is easy to find out without authenticating if a device supports the light control point property. This protects the private and personal information exposed by the device in those services while making the first device and service discovery highly user-friendly. When reading an attribute, the access permissions are tested to see whether the attribute value may be read. An error message explaining that the client is unable to read this attribute value will be returned if it cannot be read. Similar to reading an attribute, writing an attribute involves checking the access rights. If the client is not allowed to write the attribute value, an error will be provided. There are two types of authentication permissions defined: those that need authentication and those that don't.

The authentication permissions are checked when an attribute is accessed, either for read or write, to determine if the attribute needs authentication. If authentication is necessary, this device authenticates the client that submitted the request. The value should be available if the attribute does not need authentication, subject to further permission restrictions. Only clients that have successfully authenticated will be granted access if the attribute does need it. An error message saying that there is insufficient authentication will be returned if a client tries to access an attribute that needs authentication but has not been authenticated with the device.

A client may both disregard the request and pass the error on to the application if it gets the inadequate authentication error, or it can try to authenticate the client using the SM and resubmit the request. The error code does not convey the necessary degree of authentication, it should be highlighted. To access the attribute value, the client could thus need to either ask for authentication or increase the authentication level. This behaviour has the intriguing side effect of giving the client total control over when and how authentication is carried out. Additionally, the status of the received request does not need to be kept on the server. Authentication in Bluetooth classic normally takes place when an L2CAP channel is created. The responder caches the channel request sent to an authentication-required channel, sends a pending answerback, starts the security processes, and then restarts the original request. This requires a lot of memory and is complicated. In Bluetooth low energy, the server just answers each request as best it can; the client is responsible

for the complexity of ensuring that authentication is carried out and if required, reissuing the initial request.

There are specified authorisation permissions for:

a) No permission
b) Approbation

Authentication and authorization vary somewhat from one another. Similar behaviour is triggered; if an attribute access is attempted and the client is not permitted, an error response is given with the error code insufficient authorisation. However, the client is unable to fix this problem. The server's ability to grant or deny permission for a client to access a set of characteristics is known as authorization. Therefore, client authorization is up to the server. What's more, the client lacks signals that would trigger the server to request the user's authorization. As a result, the server may ask the user for authorisation anytime a client tries to access a specific attribute that needs it. The request can be instantly rejected by the server. After then, the client would have to wait before making another attempt. After configuring the other devices to add the client to the list of allowed devices, the user of the client device will often initiate the retry.

One of the following five fundamental sorts of messages may be used to access any one of the attributes in an attribute database. Discover Requests, Read Requests, Write Requests, Write Commands, Notify Users, and Indicate Users. A client may locate attributes in an attribute database using Find Requests so that more effective handle-based requests can be utilised. To read the value of an attribute, a read request is issued. To choose which attribute value to read, these either employ one or more attribute handles, a range of attribute handles, or an attribute type. To write a value for an attribute, the Write Request is issued. These always use the value to write and an attribute handle. Additionally, it is possible to get ready several values for writing before carrying out these writes in a single atomic action. The attribute server always sends a single answer in response to each of these queries. The customer must submit a new request if further information is needed. For instance, if the attribute value is extremely large and does not fit within a single Read Response, the client may use another Read Blob Request to request the remaining portions of the property data.

Only one request may be sent at a time to reduce the server's complexity. Only when the first answer has been received may a new request be made. The Write Command may also be used to write an attribute value. There is never a reaction to this. This command may be delivered at any time since it has no response. As a result, it is advantageous to enter instructions into a state machine's control point.

There are two more message categories, both of which are started by the server and provide attribute data to the client without their request. The Attribute Handle and Current Attribute Value of the Notified Attribute are both included in the Notification, which may be issued at any time. The identical attribute handle and value are included in the indication, which consistently results in the return of an attribute confirmation. While Notifications may be issued at any time, these confirmations both acknowledge that the stated value has been received and that another Indication may be sent.

Atomic Operations and Transactions

A single transaction includes all Attribute Protocol messages delivered by a client to a server and vice versa. A single signal followed by a single confirmation, or a single request and answer, constitute a transaction. Transactions are crucial because they reduce the quantity of data that must be stored in between transactions. This implies that a device doesn't need to keep any information about a request it gets to execute the subsequent request. The fact that a new transaction cannot be begun until the previous one has finished is another crucial feature of the transaction model. A device cannot submit another request before it has gotten the answer to the previous one, for instance, if it sends a Read Request for an attribute. These transactions only apply to one specific device. A device that initiates a transaction is unable to start another one, although it is still able to respond to requests from peer devices. There are a few exceptions to this straightforward rule: Prepare/Execute writes, notifications, and commands.

Commands and Notifications

A device may communicate with another device via one of two Attribute Protocol messages, termed Commands or Notifications, without needing to wait for a response. When you need to transmit a certain Command or Notification but are present in the midst of another operation, they are helpful. As an example, let's say that after sending a Read Request to a certain device and waiting for a response, you now need to publish a value to the same peer device. You would utilise a Write Command to do that.

No confirmation or reaction is necessary for commands or notifications. As a result, the transmitting device is unable to determine if the message has been read and processed. This is unacceptable for certain applications, and a request/response or an indication/confirmation is necessary. However, this is completely appropriate for certain uses. There is no limit on how many of these messages a device may send, which is an intriguing byproduct of the absence of a response or confirmation. It is possible for a device to effectively bombard a peer device with commands or notifications. Any Command or Notification that a device gets may be dropped if it lacks the buffer space necessary to retain or process it as a safeguard against this. Therefore, it is necessary to regard these signals as being untrustworthy.

The Prepare Write Request and Execute Write Request messages constitute the second exemption to the aforementioned transaction rules. A device may prepare a lengthy series of writes using these messages, which can then be executed as a single transaction. Each Prepare Write Request and answer is a separate transaction from the perspective of transactions. Other requests may be interspersed throughout the whole process of preparation and execution. Long writes and write verification are two of this command's noteworthy side effects. Each Prepare Write Request additionally specifies the offset within the value of the attribute where this portion of the value will be written in addition to the handle of the attribute that will be written with the value. This implies that you may write a single, extremely big attribute for each component of the attribute value in a single execution by using a series of Prepare Write Requests.

Another intriguing side effect is that the attribute handle, offset, and part value that was sent in the Prepare Write Request is also included in the Prepare Write Response. Although it may first seem

like a waste of bandwidth, as the values in the answer will match those in the request, this safeguards against errors. A 24-bit cyclic redundancy check (CRC) that can identify up to 5-bit mistakes protects every bit in the payload of Bluetooth low energy, which may be a little bit protective of data at times. There is an extremely slim chance that the CRC may mistakenly accept a packet that contains 6-bit mistakes if it is received. The 32-bit message integrity check (MIC) value found in each encrypted packet serves as the following defence. There is virtually no assurance that this won't also fraudulently pass an invalid packet, but it should reject a packet that has wrongly passed the CRC value. Therefore, the possibility of receiving a packet that has wrongly passed the tests is quite remote.

Sometimes, the chance of any size is excessive. For instance, you don't want to write "close" on a Bluetooth low-energy device to control a city's sewage outflow valve only to find that the valve misinterpreted it as "completely open" and inundated the park and kids' play area with sewage. The prepared answer has the same information as the request because of this. The likelihood that the same data in the answer has been damaged in the same manner as the request is as near to zero as you can make it since a packet is delivered in two distinct ways, often by utilising two different radio channels and each using a different encryption packet counter. Of course, if the answer was incorrect, you may use the cancel code in the Execute Write Request to end the whole series of scheduled writes and begin preparing to write again.

Only a flat structure of attributes is defined by the Generic Attribute Protocol. Each attribute has a handle or address. Modern data organisation approaches, however, need a lot more organisation than this simple flat framework. The Attribute Profile makes this possible. The Attribute Profile specifies sets of attributes rather than simply a single set of characteristics. Let's examine how this may be accomplished to see why it is required. There is the potential for "pages" of characteristics. A certain set of values would be present on each page. For example, one page may be used to explain the device, another if it has a battery, and still, another to display the temperature. A page would be specified for each use case.

The gradual adoption of object-oriented paradigms has been the major advancement in software engineering over the last several decades. This simply groups the techniques you may use to modify the behaviour of the data with the data that characterises an object. Each item in an object-oriented design is self-contained, which is its fundamental advantage. I'd want to define a few terms right now. If someone mentions object-oriented programming, you may immediately think of objects, classes, and interfaces. A description of external behaviour is an interface. An implementation of that interface is a class. An instance of that class is what makes up an object. An instance of the automobile class that implements the driving interface, for instance, is a car.

Although not all automotive items have the same appearance or implementation, they always have the same fundamental driving interface, such as the steering wheel, accelerator, and brake pedals. Traffic congestion is evidence that while the driving interface is the same, the class that implements it might vary and be instantiated several times. Grouping is used in Bluetooth low energy for both services and attributes. A service declaration is used to group a service, while a characteristic declaration is used to group a characteristic. A characteristic is a collection of one or more attributes, while a service is a collection of one or more characteristics.

In software engineering, if you create and implement behaviour for a certain class, other components of the system may reuse an object based on that class as long as the interface to that class remains fixed. Additionally, if that class has a flaw, you only need to patch it once, and all other components of the system will instantly benefit from that change. You must create an abstract interface that is immutable to guarantee that classes are reusable. A strong term that signifies "unchanging across time" is "immutability." The sole factor that guarantees an interface's long-term sustainability is its immutability. If interfaces were changeable, the user of the object would have to spend more time figuring out what interface it has than actually using the object to perform its necessary functions.

A new class with a new interface inherits the functionality of an old class and then adds to or modifies this behaviour as necessary. This is how inheritance works in object-oriented systems to allow modifications to interfaces. It is possible to reuse interfaces effectively for many years by making sure they are immutable. The Generic Attribute Profile for Bluetooth Low Energy specifies two fundamental types of grouping: services and characteristics. An object with an immutable interface is analogous to a service.

A trait or qualities are often included in services, and additional services may be mentioned. A characteristic is a disclosed behaviour or an item of data. Because these traits are self-explanatory, they may be read and shown by clients of any kind. Service is thus only a set of attributes and some behaviour that these features reveal. The service's immutable interface is comprised of a collection of attributes and the behaviour that goes along with them. And this design gains a great deal of strength from this simple idea. One service may point to another service as part of a reference. The reference might indicate a variety of things, such as that this service is used to expand the functionality of the original service, that this service makes use of the other service, or that this service and the other service have been merged to form a larger set of services. Let's take a closer look at each of these references.

A service that has been in use for a while needs to be expanded. We cannot just add new behaviour to the old service since Service A is immutable. As a result, an extension is required without changing the initial service, we create a new service, Service AB, which has the necessary extra behaviour. However, we must additionally include an instance of that Service in each device that implements Service AB to ensure backward compatibility for the many millions or billions of current devices that only support Service. The two old Service and instances will still be found and used by an outdated device that only knows Service A; however, the outdated device will disregard the additional Service AB instances and will only be able to utilise the nonextended behaviour. Finding the Service AB instances and following the references to their Service An instances will be easy for a new device that is aware of both Service AB and Service A. Since Service AB defines the new behaviour, the new device will be able to utilise it.

When a new device and an old device are communicating, the new device will first try to locate Service AB, fail, and then locate Service A; as a result, it will be able to immediately revert to the interoperable behaviour that was specified in Service although it seems difficult, this is significantly easier than the alternative: Service A would be upgraded into a new version with feature bits to identify which features a certain service supports as well as potentially highly

complicated behaviour due to the necessity to test every potential feature combination. Each service is self-contained and immutable, and the interaction between services is openly revealed thanks to the extension technique. A gadget utilising the services has control over how it acts. The immutability of the original server also ensures legacy compatibility.

Referencing another service is another way to reuse it. This is the most straightforward allusion that can be made. A particular service, Service A, wishes to use the behaviour and state data of Service B. Service a just needs to make use of Service B to accomplish this. It is more like a generic reference to another instance of a class than reuse in the traditional object-oriented sense. The last reference style requires a separation of interface from implementation, making it more difficult than the previous two. Two separate service instances that are tied to one another and exhibit extra behaviour when merged are sometimes required. A third service that references the first two services must be established to do this utilising services.

Service C, which encapsulates the combined service activity, may expose the functionality that is necessary when interacting with both A:1 and A:2. To provide a service of a light that could only be turned on when there was no daylight, for instance, a light service and a daylight sensor service may be coupled. Additional states in the combined service's state machine come from combining the states of the two fundamental services it refers to. The autonomous A:1 and A:2 services continue to behave in an unchanging manner. As a result, Service C is required to discern between the behaviour linked to the combined services and the behaviour of the independent services very precisely.

There is one more thing about services that you should be aware of: they might have two distinct "flavours." Both main and secondary services are available. It is sometimes necessary to set up services that expose the device's external behaviour, and other times it is necessary to set up services that expose a functionality block that can be reused many times in many different ways but is never actually used or understood by the end user, as you can see from the description of the services and how they are designed in the preceding paragraph. A principal service is often one that makes available the actions that a device does. For instance, Service B would be launched as a main service if you have a device that supports it. Service D would be invoked as a backup service if you needed some more information for this device that is accessible in Service D but is unrelated to what the device performs. Consequently, a secondary service is an encapsulation of behaviour and traits that the user is not required to comprehend.

The Attribute Protocol may be used to rapidly and effectively locate primary services. They may either have a "parent" service or operate independently. Secondary services must always have a primary service that points to them since they can only be discovered through reference. With a main service at the top of each tree and a branch pointing to either a primary service or a secondary service, and a branch from each of these leading to yet another primary service or secondary service, it is implied that a tree of services may be built.

To reuse the functionality accessible in a secondary service, a main service may refer to that secondary service. Another secondary service or another major service may be referenced by a secondary service. Because secondary services are often leaf nodes in a service tree, pointing from

one secondary service to another is quite uncommon. Pointing from one major service to another is highly uncommon, although it is conceivable. One more benefit exists for primary services. A customer may seek main services very fast when they are looking for a certain service. This may be improved even further by enabling a particular principal service to run just once on a specific device. For instance, if a service is specified such that it can only have one instance of itself on a device, a client device would undoubtedly be able to quickly ascertain whether the service is there.

An important advantage of this optimization is: With the least amount of hassle, a straightforward customer who just needs one instance of a main service may accomplish that goal. Simple clients may utilise simple services without having to read the whole list of services on a device or, out of their relationships. Without this optimization, every simple client would have to traverse the whole service tree to decide how to make the greatest use of the services offered on a device. Both the electricity used for transmission and the memory used to hold all calculations and intermediate outcomes are being completely wasted in this action.

The ability to take a device's collection of service trees and look for apps that can use them is another intriguing feature of the service model. To do this, the generic client would first undertake a thorough service enumeration, starting with the principal services and moving on to the relationships to additional mentioned services. Once this tree has been created, it is feasible to give this service "forest" to an application store to get a list of programmes that are known to function with all or part of this forest.

Only one principal service could be supported by certain apps. Some applications could offer a main service that expands another primary service, perhaps as an addition to the primary service being supported by the application or possibly as a separate primary service. Multiple service trees could be supported by certain applications. Various apps could be able to either integrate this information on the client in novel ways or offer the information from these services in an engaging all-in-one application. A server that supports the services shown in,s 10-18 and 10-19, for instance, would be able to handle the main Service A, together with another primary Service C, the primary Service B, along with a secondary Service D, and the primary Service C by itself: A(C), B (D), and C. To know whether apps provide this set of services, the client may utilise this information. To decide whether apps may support this device, the list of services for each App is compared to this set.

Another strategy is to use App8, a universal application that can communicate with any service. Although they won't normally be able to communicate as effectively as a custom-written application, these client apps may be able to support devices for which the client does not yet have a custom-written application. By combining services with a pure client-server approach, this common client behaviour is expressly enabled. However, immutable services are the essential component that makes this feasible. Generic client apps could not be created that can access services that have a known immutable behaviour. By keeping each component's flexibility to a minimum, the whole system has been built for maximum adaptability. These distinct, unchangeable components work together to provide items on the market with the richness and utmost flexibility they need.

Service Statement

A service declaration is used to group services this is an attribute of the Primary Service or Secondary Service attribute type. All characteristics that appear after this service declaration and before the net service declaration are regarded as belonging to this service and are grouped with it. A major service encompasses all the device does, as previously mentioned. One that aids the main service in achieving its objective is referred to be a subsidiary service. The main service is the source of all references to secondary services. This is done for one very clear reason: to keep the client's simplicity. Simple clients are non-user interface devices that may nonetheless access services on a peer device. A simple gadget may simply search for essential services and discover the services it requires. It doesn't need to explore every service that a device could expose. In reality, by enabling basic clients to look for a certain principal service, the Attribute Protocol is optimised for them.

Some firms provide assistant services to aid in disclosing their actions or emotional condition. There is no need for each medical service to create its device information since, for instance, the majority of medical equipment will offer device information. The situations involving automation and batteries both call for the same device information. The device information service only has to be defined once and may then be utilised several times by including this information in a service. These straightforward consumers may also avoid worrying about such details by ignoring the secondary services in favour of their principal services as a result of this.

A Service UUID is the value of the service declaration. This is either a 128-bit or a 16-bit Bluetooth UUID. It is safe to disregard any services that a device does not comprehend. All the characteristics that are grouped with this service declaration may be disregarded, for instance, if a device contains a secondary service with a 16-bit or 128-bit UUID that this device does not recognise. The range of attribute handles for services may be identified thanks to the Attribute Protocol, which can aid with this. Secondary services need to be found independently. Each service may have zero or more Include characteristics to do this. Include declarations are always placed before any other service characteristics and shortly after the service declaration. Along with the Service UUID for the included service, the Include definitions also include the handle range for the referenced service. As a result, the referred services, their aggregated properties, and the service type may all be found extremely quickly. Because it is irrelevant, it is not stated whether the referred service is a main or secondary service.

A Service UUID that is a complete 128-bit UUID will not fit inside the typical response packets used to discover the included services since four octets are utilised for handles in the Include value. Therefore, the Service UUID is not a part of this declaration's value when the included service has a 128-bit UUID. This indicates that to determine the nature of the service is added, an extra Attribute Protocol read is necessary. A main service may refer to another primary service or a secondary service, and a secondary service can refer to another secondary service or a primary service if the nature of a referred service whether primary or secondary does not matter.

When a main service was first released and then expanded with another major service, the first service would need to be referred to. This initial main service cannot be converted to a secondary

service since doing so would prevent current customers from accessing the previous service. A consistent interface to a block of activity may be provided by combining characteristics into a single group inside a service. The design of Bluetooth low energy also enables the grouping of characteristics, enabling the exposure of a service's status and behaviour.

An attribute is only one value. It may be the outside temperature, the amount of cycling someone has done, or the status of the time synchronisation finite state machine. A feature, however, is much more than that. A characteristic must reveal the kind of data that a value represents, whether it can be read or written, how to set it up to be indicated, alerted, or broadcast, and what it implies. To do this, a characteristic is made up of three fundamental components: A characteristic begins with a declaration, which collects all of the other qualities of the characteristic. The actual value for this feature is included in the value property. Additional details or setup for this attribute are included in the descriptors.

At this stage, a common query is: Why is the value an attribute inside a characteristic rather than a stand-alone attribute? The solution is somewhat intricate. A characteristic is more than simply a value; it also contains rights, extra settings, and descriptive information that should be taken into account. It could have been conceivable to extend the Attribute Protocol's semantics to access this data, but doing so would have increased the protocol's complexity for the little chance that this would be required. Instead, it was decided to retain the device's structure and its features, as specified by the Generic Attribute Profile, distinct from the flat structure of attributes, as disclosed by the protocol. This implies that although it is considerably simpler to acquire the necessary information for the majority of features, it is more difficult to collect specific information about particular characteristics.

A Characteristic attribute is used to begin a characteristic, this has three fields: characteristic properties, the handle of the value attribute, and the type of the characteristic. If the characteristic value attribute may be read, written, informed, signalled, broadcast, commanded, or authenticated in a signed write, it depends on the characteristic characteristics. The related method may be used to get the value of the characteristic value if the bit in this field is set. The server characteristic configuration descriptor must also exist if the character has the broadcast bit set. The client characteristic configuration descriptor must exist if the characteristics notify or indicate bit is set. In this field, there is also an extended properties bit. This was added since the maximum size allowed by this Descriptors value had already been reached and there were more attributes to put in the 8-bit field. The characteristic expanded properties description contains these extra characteristics. There are just two more characteristics: writable auxiliary and dependable write support. The most intriguing section is the writable auxiliaries, which control whether the distinctive user description descriptor may be written.

The handle of the attribute containing the value for the characteristic is included in the characteristic value handle field. This is required so that a client that just provides characteristic declarations may do a very rapid search for the characteristic. This declaration makes the attribute that contains the value instantly accessible. If this field didn't exist, the client would have to do a separate search for attributes and infer which attribute's value was following the declaration. The value attribute now follows the characteristic declaration as the following attribute, however by

adding the handle for the value attribute in the declaration, this convention might change in the future.

The distinctive UUID is the last field. This is where the UUID is stored, which is used to specify the kind of characteristic value. The type of property that stores the characteristic value must match this UUID. This effectively indicates that there is a duplicate of data that may be found by making further calls to the server. However, doing so would result in power waste since additional over-the-air protocol messages would need to be delivered. Directly including the type of information in the declaration is more effective.

It has sometimes been questioned as to why the type of the characteristic value property is not a static UUID-like Value. This would lessen the issue that was previously noted; nevertheless, further optimizations may be made, suggesting that this would not be the best option. It would be much more effective to only ask for the battery state of a device rather than searching for the characteristic that includes the battery state UUID in one of its fields for a basic client that just needs to obtain the battery state of a device. The following examples show how these straightforward optimizations affected the declaration's structure:

The type of attribute in the characteristic value must match the characteristic UUID field in the characteristic declaration. It is an ordinary trait apart from that. The primary distinction is that the kinds of operations that may be carried out on this characteristic value attribute are listed in the characteristic declaration properties field and may also be in the characteristic extended properties descriptor.

There is a specification document that details the format of the characteristic for each one. Additionally, as characteristics do not by themselves contain behaviour, it is necessary to look at the service specification with which this characteristic is grouped to ascertain the behaviour that is revealed by this particular instance of the characteristic. There are many ways to describe quality. Although, as was just said, they may be necessary depending on the characteristic declaration, most descriptors are optional. A service definition could additionally call for certain descriptors. The following words may be used to describe a characteristic:

Extraordinary Characteristics

Characteristic User Description, Client and Server Characteristic Configurations, Presentation and Aggregation Formats, and Characteristic User Description. The description used to record the extra expanded attributes is this one. Currently, only two are described in detail: the capacity to write values in a trustworthy manner and the capacity to create Characteristic User Description descriptors.

The characteristic's user-described-descriptor

This descriptor allows a device to link a text string to a property. When used with equipment that consumers can con, themselves, this is most helpful. For instance, a thermostat may be set up by the user to specify the room in the building it is measuring. For maximum configurability, certain devices may have numerous temperature sensors, therefore having this setup at the characteristic level is crucial.

The Client Characteristic Configuration Descriptor

This descriptor must exist if a trait is notifiable or indicatable. It is a two-bit value with a notification bit and an indicator bit. Since Notification and Indication are complementary processes, these bits cannot be set simultaneously. The service standards, not the core specifications, specify how the value is informed or indicated.

The Server Characteristic Configuration Descriptor

Except for having one bit for broadcast, this descriptor is quite identical to the Client Characteristic Configuration descriptor. Setting this bit allows the device to broadcast certain data related to the service that this characteristic is grouped under. It is a single bit. Again, the provider chooses the time of this transmission. Oddly, you can't broadcast a single trait. The data that is broadcast when this bit is set is instead determined by the service for which it is grouped. It is up to the service to specify how an observer might ascertain which characteristics are broadcast by the service. Some services may state that several characteristics can be broadcast.

At first glance, it could seem a little odd that a character has a bit that can activate its broadcast without having the capacity to broadcast the characteristic directly. This is because characteristics do not by themselves have behaviour; hence, broadcast characteristic data without the context of service has no value. The information "Temperature: 20.5°C" on its own is not very useful. You may get the necessary context for a temperature by asking for "Room Temperature Service: 20.5°C" or "Car Engine Service: 65°C."

The Generic Attribute Profile was created to enable generic clients. A device that can read the values of a characteristic and present them to the user without comprehending what they signify is referred to as a generic client. Generic clients may establish a connection to a refrigerator and show the inside temperature without realizing that a reading over 10°C is probably not healthy. In contrast, a profile outlines how a client may interact with a refrigerator's temperature service and what to do if the temperature veers outside of an acceptable range.

For generic clients to function, they must first be able to identify features that may be presented to the user and then comprehend the values of those qualities well enough to do so. Finding all the characteristics in a device depends in part on the characteristic declaration having a recognised attribute type. Generally speaking, legible traits are also beneficial. The Characteristic Presentation Format descriptor is the most crucial element that indicates whether a characteristic may be utilised by a generic client. If this exists, the generic client could show its value, and reading this value is secure.

The following fields make up the presentation format, which is a multiple-field value: Format, Exponent, Unit, Namespace, and Description. The structure of the value is determined by the format, which is an enumeration of the common data types. Boolean and unsigned 2-bit and 4-bit formats are available. For both unsigned and signed integer numbers, there exist formats with bit widths ranging from 8 to 128. Standard IEEE-754 floating-point numbers, like those used in the majority of high-end computers, come in two sizes. There are two integer-based fixed-point numbers with different sizes that are mostly utilised by medical equipment. Two string

representations employing both UTF-8 and UTF-16 encodings are available as a final option. The opaque structure or an aggregate format, as described in the following description, should be used if the format of the characteristic doesn't fit into one of these buckets.

The exponent follows the format. This parameter defines a fixed exponent that may be added to the integer value before it is shown to the user, and it is only valid for integer values. Since it is a base 10 exponent, it is possible to position the decimal point using a procedure in the final output rather than by utilising intricate mathematics. The formula below may be used to describe the value the characteristic value represents: The unit field is the following field in the presentation format. The unit is a UUID listed in the paper on issued numbers. There are several specified units. For instance, if the unit is Temperature Celsius in the example above, the displayed result will be 6.2°C. Undoubtedly, it is expected that a generic client is aware of what each of these unit UUIDs stands for.

It is best to treat the last two fields as a single value. Additional information about the value is determined by the namespace and description variables. The namespace field, which only contains one byte, identifies the entity in charge of the description field. A 16-bit unsigned number makes up the description field. In reality, the description field is essentially a single "adjective" that may be attached to the characteristic to help the user identify which value corresponds to a certain device attribute. Take a thermometer with probes for measuring the temperature inside and outdoors as an example. This would reveal two temperature characteristics, with the sole difference being whether "inside" or "outside" is included in the descriptor field of the characteristic presentation format.

Some character traits have many values that are more complicated than simply one. Take the typical meaning of a place on the planet Earth, for instance. This is made up of two values that have been concatenated to create a single "value." A latitude value and a longitude value are combined to create the position value. The Characteristic Aggregate Format descriptor enables numerous presentation format descriptors to be referenced to support such complicated characteristic values and to explain the value's many fields.

Concerning the aforementioned example, the characters would have two Characteristic Presentation Format descriptors one for the latitude and one for the longitude and a Characteristic Aggregation Format descriptor that correctly referred to the two Characteristic Presentation Formation descriptors. The characteristic value's format may thus be accurately deconstructed by a generic client, who can then show the user the value. It should be noted that the descriptors from the characteristic aggregate format that are referred to in the characteristic presentation format do not have to be in the same characteristic. Even though they are in the same service or device, they could not be. The attribute that groups them has no significance for the aggregate format; they are only referred presentation formats.

CHAPTER 10

DIFFERENCE BETWEEN BLUETOOTH AND WIFI

Mamatha G N, Assistant Professor
Department of Electronics and Communication Engineering, Faculty of Engineering and Technology, JAIN (Deemed-to-be University), Ramanagara District, Karnataka - 562112, India
Email Id- gn.mamatha@jainuniversity.ac.in

There is a considerable probability that you can see the Wireless cons if you possess a phone, laptop, workstation, or any other contemporary electronic device. What all these two phrases signify, Here is all you have to know about Bluetooth and its differences and if Bluetooth can be used without WiFi.

Wi-Fi vs. Bluetooth

Both Headsets and WiFi are wireless technologies that may be used to connect your device, nevertheless, they function quite differently. Bluetooth solely connects your devices; WiFi primarily connects your gadgets to the internet. It's essential to describe Bluetooth and WiFi's operational principles in more depth to comprehend how the two vary from one another. To connect your gadgets to the internet, WiFi is a Bluetooth method that is widely utilised. To subscribe to a subnet (LAN) via a router using WiFi, you often need to input Login details on your device. In essence, WiFi enables online browsing without requiring a phone connection or cable plug for your device.

To do this, a WiFi router is required to convert the ethernet connection from your modem into a radio transmission. Once the radio signal is picked up, your WiFi-capable gadgets convert it directly into an internet signal. You may use your gadget to browse the internet wirelessly thanks to this. The majority of modern computers, laptops, smartphones, tablets, Chromecast, scanners, digital cameras, digital music players, and other smart home appliances all include WiFi receivers. A device can access the web electronically if it has the WiFi emblem, which can be found on most electronic devices. Additionally, if the emblem is shown at eateries, cafés, or airports, it implies that such locations provide wireless internet connectivity for your devices [1]–[3].

The quality of your router affects how strong your Wi-Fi connection is. But normally, you may subscribe to a WiFi connection up to 14 paces away. Check read this post on how to test your WiFi speed if you want to learn more about your WiFi signal. WiFi allows you to connect to the internet as well as to link your gadgets to another network. For instance, you may wirelessly transfer data between two computers connected to about the same area network.

Slow-computer

Data may be sent over short distances via Bluetooth between your smart gadgets. Bluetooth may be used, for instance, to link your device with a Stereo system or connect your workstation to a wireless keyboard. Bluetooth employs radio waves, much like WiFi, to transfer data between devices. Bluetooth utilises all of the same spectrum as WiFi, which may interfere with WiFi-

enabled devices. However, Bluetooth does not need a username to connect to a device, in contrast to WiFi.

Typically, you must press and hold a button on one Bluetooth device to link two devices. Next, choose that device from a different device. For instance, you may need to press a Bluetooth connecting toggle on your earphones before choosing them from your computer's list of Bluetooth devices that can be discovered. Bluetooth connections between devices are often less secure than WiFi connections since a password is not required to connect. Additionally slower and with a smaller range than WiFi, Bluetooth is usually up to 30 feet [4]–[6].

In practically every wireless gadget you can find today, Bluetooth is enabled. Bluetooth was first developed as a means to link devices without requiring wires. Most modern PCs, laptops, smartphones, tablets, keyboards, mouse, loudspeakers, gaming consoles, wristbands, wearable technology, cochlear implants, and many more peripherals now have Bluetooth technology. If a device has the Bluetooth logo, it can wirelessly connect to other Bluetooth-enabled devices with the official Bluetooth logo. Bluetooth may be used, for instance, to link your mobile to wireless speakers and headphones. Remote workstations may also be connected to your computer, laptop, or tablet via Bluetooth.

UART

Of all the possible transports, the UART interface is the most straightforward. It specifies a straightforward global asynchronous receiver/transmitter that is linked between the host and the controller in a null-modem setup. To connect transmit data (TXD) from the controller to receive data (RXD) on the host and vice versa, a null modem merely implies that the transmit and receive wires cross over at the opposing ends of the connection. Only TXD, RXD, and ground are available on three-wire UARTs. Additionally, Five-Wire UARTS contain flow control wires that link the controller's request to send (RTS) and the host's clear to send (CTS) signals.

The UART interface in Bluetooth low energy always employs 8-bit characters with no parity and one stop bit. Each byte ends with a stop bit, which is an additional bit of time. When utilised, the parity bit is used for error checking. The parity check is not necessary if the Bluetooth controller and host have a relatively short connection since mistakes seldom happen in these situations. Hardware flow control lines CTS and RTS are used. One of the following three packet-type codes must be appended to an HCI packet before it may be sent. So that's all. It is just that.

Command = 0, Data = 0, and Event: 0, respectively.

Unfortunately, the UART interface is not appropriate for very low-power devices since it is unable to provide any low-power signalling. To enable the interface to be shifted into a very low-power mode, some devices may employ extra hardware signalling lines. However, these extensions are often proprietary, and each device will implement them differently. Since synchronous data packets are not used by low energy, they will never be transferred to or from a low energy-only controller even if the HCI packet type 0x03 is defined for them in Bluetooth classic.

The 3-wire UART is a little more complicated than the UART transport that was previously discussed since it is made to function without any hardware flow control lines and even when there

are some bit mistakes. Using the 3-wire UART is strongly advised if your host and controller are more than a few millimetres apart and are in a noisy electrical environment. Channels are utilised in a similar way to the UART interface previously explained for the 3-wire UART to function. With 0x1 for instructions, 0x2 for data, and 0x4 for events, the channel numbers remain the same. The three-wire UART additionally specifies two extra channels: one for acknowledgements on channel 0x0 and one for connection setup on channel 0xF. The size of the channel numbers in a 3-wire UART is merely four bits.

Link creation and active state

Low-power condition

To verify if the peer device is awake and to set any settings, utilise the connection-establishing the channel. Automatic baud-rate detection may also be done via the connection-establishing channel. It does this by sending connection setup messages at various baud rates and seeing which one elicited a response from the peer device. To put the two devices into the usual state, the connection setup procedure is essentially a three-way handshake. This involves setting the reliable sliding window size, choosing whether to apply a cyclic redundancy check (CRC) and deciding whether to utilise out-of-frame software flow control.

Reliable packets may be transmitted while the system is operating. A sequence number and an acknowledgement number are included in every packet, which is also framed. Both of these integers are three bits long, allowing for the simultaneous transmission of several packets. For really quick UARTs, where it could take some time to process a packet, this is helpful. According to RFC 1055, SLIP is used for packet framing. For framing purposes, SLIP appends a 0xC0 byte to the beginning and end of each packet. Any instances of 0xC0 in the packet are then swapped out for a two-octet stream of 0xDB 0xDC. This also has to be transformed into a two-octet byte stream of 0xDB 0xDD because 0xDB is an escape sequence. Additionally, the XON and XOFF octets will be escaped to 0xDB 0xDE and 0xDB 0xDF if out-of-frame software flow control is used.

A framed packet's packet length and header checksum are also included in the packet header. The header checksum only establishes the reliability of the header information. The whole packet is discarded if it is invalid, and a retransmission system will automatically retransmit this packet. The payload is secured by a CRC value and has a maximum size of 4,095 bytes. The 16-bit CRC-CCITT is the CRC used in Bluetooth classic baseband packets and is also used in the three-wire UART. Again, if the CRC is unable to verify the packet, it is simply disregarded, and the peer device will retransmit the complete packet.

By sending a "sleep" message, the link may also be put into low-power mode. It may take some time for the UART hardware on the peer device to wake up once a packet is delivered, and the UART is normally off in this state, making it unlikely that any packet transmitted would be completely received. A very brief wake-up message may be delivered to help a peer device wake up quickly before delivering an HCI communication. After receiving a "woken" message in response, the device has an active connection and is ready to transmit any packets. The 3-wire

UART is the best option if a device only has a UART interface and the standard UART interface isn't proving to be strong against big mistakes.

USB

The USB interface is designed to work best with hardware that already has a USB host. It specifies how data, events, and instructions are sent between the host and the device. A common class code for Bluetooth devices is also established for enumeration. This has made it possible for several manufacturers to provide plug-and-play Bluetooth dongles. The "host to device class request, device as target" request type is used to send commands from the host to a device through the Control endpoint (0x00). The host polls the device for events using an Interrupt endpoint (0x81). To guarantee that events may occur with the least amount of delay, this endpoint should be polled every millisecond.

Two endpoints are used to send data: Bulk data out (0x02) is used to send data from the host to the device, and Bulk data in (0x82) is used to send data from the device to the host. A one-millisecond poll period is once again employed, however numerous Bulk data USB packets may be transmitted in a single frame, enabling extremely fast data throughput when required. The main issue with USB is that it does not need a lot of power. A significant amount of high-speed hardware and software control is needed to establish a USB interface. This is a pricey undertaking. Another issue is that every millisecond, the USB host must poll for data from the device. As long as the device is in operation, this usually prevents the host from entering the lowest-power processing modes. Another USB function called Link Power Management may help with this. It is advised to use this with Bluetooth-enabled devices.

SDIO

A host may connect with a controller via the SDIO Card Type-An interface using the high-speed SDIO protocol. SDIO is a packet-based bus that employs 4 to 8 lines to carry data rapidly and efficiently in both directions. For instructions, events, and data, the transport employs the same channel assignments as UART and 3-wire UART transports. The SDIO interface is highly helpful for devices that already have an SDIO interface accessible because of its very low error rates. You must be a member of the SD Association to access the complete SDIO transport standards. A version of these requirements, however, has been made accessible to assist businesses in assessing the technology. These condensed specs don't include all that's required to make a product, but they do provide enough details to explain how the system functions.

Logical Interface

The logical interface sits on top of the HCI physical transports. It is a logical interface since it is not required to implement this interface as a message-passing interface between components in the single-chip device. The HCI logical interface is represented as physical packets sent over the physical interface in a system where the controller and host are on different chips coupled by this physical interface.

Regarding the logical interface, it is important to comprehend three ideas the lower HCI interface of a controller establishes an HCI channel whenever it is connected to another device, and this

channel is identifiable by a connection handle. This connection handle is used to identify all data delivered from the host to the controller and all data that the controller receives from a particular peer device before sending it to the host. Every time a connection attempt is successful, the host receives the connection handle. The connection handle is still valid after it has been locally, via the Link Layer termination process, or as a result of a link supervision timeout.

Command Packets

Command packets are provided to the controller by the host to instruct it to take action. These command packets are often used to either con, the controller's status or request action from it. The HCI command packet includes an opcode that identifies the command being issued, a parameter length field, and the command's parameters, as shown in, 8-1. Every command has a different set of specific arguments. In Bluetooth low energy, there are three fundamental sorts of instructions that may be used: Con, the controller state; Request a particular action; and Control a connection. The controller may be seen as a large state machine with a variety of configurable settings. Using the LE Set Advertising Parameters, LE Set Advertising Data, LE Set Scan Response Data, and LE Set Advertise Enable commands, for instance, one might arrange the state of advertising to have the following state.

The state that is being utilised inside a state machine in Bluetooth low energy often cannot be modified while that state is being used. As a result, when advertising is enabled, the advertising specifications cannot be changed. Therefore, it is important to turn off advertising, modify the advertising specifications, and then turn it back on. Some instructions ask for a certain action to be performed without changing the device's or a connection's status. For instance, the LE Encrypt command asks the controller to create some encrypted data based on a key and some plain-text data.

Controlling a Connection

Once two devices have established a connection, orders may be issued to control that connection. The connection handle is always included in these commands. For instance, the current adaptive frequency-hopping channel map for a specific connection is retrieved using the LE Read Channel Map command.

Event bags

Event packets are used to communicate information from the controller to the host, usually in reaction to a prior order from the host demonstrating that the HCI event packet includes the event parameters, a parameter length field, and an event code that identifies which event is being transmitted. Each event has a special set of requirements. The HCI event packet format, The Bluetooth low energy standard has three fundamental event types: Common command success occurrences Generic command status events and completion events for individual commands

Generic Command Complete

A generic command complete event is delivered whenever the controller receives a command that can be finished right away. It's the Command Complete event right now. The fact that the

command opcode this event is completing, as well as the return arguments that are given by that command, is included in the event parameters makes this event generic. A status code that indicates whether a command was successful or unsuccessful is the first parameter of all command return parameters.

The status code and the required random number are the two return arguments for the LE Rand command, for instance, which instructs the controller to provide a random number. When the controller can execute a command without performing any over-the-air transactions, the generic command complete event is utilised. The LE Create Connection command, on the other hand, requires that at least one Link Layer CONNECT REQ packet be transmitted before the connection is established; as a result, the generic command complete event cannot be used. As an illustration, the LE Encrypt command does not request any Link Layer packets to be transmitted, so the generic command complete event is used.

Generic Command Status Events

A generic command status event is often followed by a command-specific completion event for commands that carry out over-the-air transactions, such as forming the connection discussed in the preceding section. The Command Status event is a general-purpose command status event.

Command-Specific Completion Events

Some commands take some time to finish. There are always command-specific completion events for these commands. There is just one command completion event for each of these commands. For instance, the LE Create Connection command will initially send a Command Status event, then if the connection is established or fails, an LE Connection Complete event will be issued. The command is not deemed complete until this command-specific completion event is received.

Data Packets

Application data is sent using data packets from the host to the controller for transmission to a peer device as well as from the controller to the host for hosting after being received from a peer device. Data packets are always identified by a connection handle. The host receives this 12-bit value in the LE Connection Complete event. The host can't communicate any data to a peer device until it has received the event. However, as soon as this event is received, the host may begin sending data to the peer device and the peer device can begin sending data to the host.

The Packet Boundary Flag and the Broadcast Flag are the two flags that make up an HCI data packet. Some of these flags are meaningless in low energy since these packets are recycled from Bluetooth classic. If this packet is the beginning or end of a higher-layer (Logical Link Control and Adaptation Protocol) message, it is indicated by the packet boundary flag. This is comparable to the Link Layer's LLID bits in a Data Channel PDU.

It's important to note that although this may have values 00 (start) and 01 (continuation) from the host to the controller, it can also contain values 10 (start) and 01 from the controller to the host (continuation). This is so because, unlike Bluetooth low energy, which doesn't have the idea of flushing, Bluetooth classic uses the value 10 from the host to the controller to indicate that the data

packet may be flushed if required. This value cannot be utilised since it just drops the connection if the data cannot get through. Because flushable data in Bluetooth classic is utilised for the start indication from the host to the controller, 00 is used in place of the LLID value 10 at the Link Layer.

Command flow control

Command flow control and data flow control are the two types of flow control available on the HCI interface. The controller may regulate how many HCI instructions it can handle at once using command flow control. The simplest way to understand how this works is to imagine the controller as having sufficient memory to buffer a few instructions. For the host to know how many orders it may send to the controller at once, it transmits the number of these buffers to the host. Event flow control is absent. The number of instructions that can be processed is thought to be a limiting factor in the number of events that can be transmitted. The host is also thought to have greater resources than the controller, allowing it to buffer and handle these events sequentially.

All Command Complete and Command Status events have a parameter named Num HCI Command Packets that enables command flow control. The number of command packets that the controller can buffer is indicated by this parameter. One of these slots is used each time a command is issued. The quantity of open slots is always included in any Command Complete or Command Status events that are transmitted to the host. Additionally, a new Num HCI Command Packets together with a Command Complete event with the opcode "No Operation" are sendable at any moment. This is especially helpful during the controller's first boot-up, when it may be desirable to provide the controller extra command buffer slots to facilitate the host's initial setup of the controller.

Data Flow Control

Similar methods are used to regulate data flow. Data flows from the host to the controller and from the controller to the host exist. Data flow control from the host to the controller must be employed, while it is acceptable to disregard flow control from the controller to the host. Flow control is not required since the majority of hosts should be able to handle the volume of data being transmitted from the controller to the host.

To control the data flow from the host to the controller, the controller is thought to contain several fixed-size buffers. One of these buffers is used each time a data packet is sent from the host to the controller. The buffer that housed the data packet is returned to the host to make room for another data packet each time a data packet is successfully delivered by the controller to the peer device.

This flow management technique has a minor drawback in that a dual-mode controller may have two distinct buffers, one for low-energy data and one for basic rate data. Therefore, the Read Buffer Size command and LE are the two HCI commands available to find the buffers on a controller. Command read buffer size the host may thus learn which buffers are accessible by using these two instructions. Only the LE Read Buffer Size command will produce non-zero length buffers for a low energy-only device.

The host utilises one of these buffers each time it transmits a data packet to the controller after learning how many buffers are available. When several Completed Packets event is dispatched from the controller to the host, the buffers are freed. This contains a list of connection handles as well as the number of packets transmitted. Because of this, the host is also aware of the kind of data that was transferred to the peer devices in addition to how many buffers have been released.

Setup of Controller

Before attempting to send or receive any packets from peer devices, the host might take a variety of actions. As part of this, the controller is reset to a known state, the device address is read, event masks are set, flow control buffers are read, the supported features of the local controller are read, a random number is generated, some data is encrypted, the random address is set, and white lists are con,d.

Reset the Controller to a Known State

Before doing anything further, it is usually beneficial to reset the controller to a known condition. The host transport may have connected just now while the controller was engaged in another activity. Resetting should thus put the controller in sleep mode and reset all programmable settings to their default values. The host sends the Reset command to the controller to reset it. The Command Complete event for the reset command is provided to the host once the controller has been reset. This command does not restart the physical transmission, it should be noted. When a physical transport has to be reset, the reset process for that transport should be followed. Furthermore, it should be noted that command flow control prevents the host from sending any further commands while the controller is being reset, even if the host is capable of sending numerous instructions. When the reset is issued, all instructions that were being processed will be ignored.

Reading the Device Address

A preprogrammed device address will be present in many low-energy devices. The host has to be able to read this address since it is necessary. The host sends a Read BD ADDR instruction to the controller, to read the device address. A Command Complete event is then returned along with the fixed device address as a result. The controller has a wide range of specified events that may be transmitted. There must be a means for the controller to know which events that host is capable of receiving and processing in the extremely probable scenario that new functionality is described in a future revision of the specification. Interoperability issues will arise if the controller just sends the host every event it is aware of, but the host is unable to comprehend these events. Allowing the host to con, the controller with the events it can receive is the only way to fix this. Then, just those events will be sent by the controller.

Two instructions are needed to set the event masks, as shown in the Bluetooth classic events are first set up using the standard Set Event Mask command. The "meta-event" for low energy is one of these occurrences. As a result, the host must use this command to activate this meta-event. Second, any necessary low-energy events are enabled using the LE Set Event Mask command. The controller has up to two sets of buffers it uses to buffer data that is transferred from the host

to the controller before it is delivered to a peer device, as described in the section on data flow control.

As shown, the host must send both the LE Read Buffer Size and the Read Buffer Size commands to read the buffers that are accessible in the controller. The host may, out how many buffers are available to deliver low-energy data packets to linked devices after the Command Complete events have been returned. Allowing the host to ascertain what features a controller supports before issuing any feature-specific instructions to the controller is another technique to guarantee that a controller and host are forward-compatible. The host sends the LE Read Supported Features command, to read the supported features that a controller offers. The Command Complete event, which contains the features that this controller offers, is returned by the controller in response. Before sending any instructions that are particular to a feature, this should be delivered.

Identifying the supported features on the controller

Read Supported States in the controller may be either simple or quite sophisticated. A command is supplied that can get this information so that a host may scale its activities to the controller's capabilities without attempting potentially incorrect state combinations and getting failures back. The host sends the LE Read Supported States instruction, to retrieve the list of supported state combinations. The whole list of potential support states is included in the Command Complete event that the controller sends back in response.

Application of the LE Read Supported States command

The following states are among those that may be supported: Directed advertising, scan-able advertising, connectable advertising, non-connectable advertising, and active and passive scanning Setting up a link to the master role or the slave role. Additionally, it includes bits that define which combinations of states, including the following, may be supported: At the same time as passive scanning and non-connectable advertising, a connection in the slave role may also be present. This data allows the host to ascertain the success or failure of a specific instruction to begin advertising, scanning, or starting.

Random Numbers

The source of random numbers used by the controller is quite reliable. These random numbers often originate from the device's physical properties, such as the phase noise between a free-running oscillator and an external crystal time source. The host should request as few random numbers as it can since the quantity of random numbers that can be created is restricted, as is the case with most random number generators that are based on fundamentally chaotic physical phenomena. It may then include these random numbers either occasionally or as seeds in a more traditional pseudo-random number generator.

The host issues the LE Rand instruction to instruct the controller to produce a random number. The controller answers with a random number in a Command Complete event. The AES-128 encryption engine utilised by low energy might be handy for the host to encrypt data on occasion. It seems sensible to provide the host access to this encryption engine given that the controller must have developed it shown, the host may encrypt some plain-text data using an encryption key by

delivering it to the controller as part of an LE Encrypt instruction. The data is then encrypted by the controller using the AES-128 encryption engine, and the encrypted data is returned in a Command Complete event.

There is no decrypt command, it should be noticed. For private addresses, the host was required to create encrypted data, but decrypting data at the host layer was not necessary. The host cannot extract the plain-text data from the encrypted data and an encryption key; it can only verify that the encrypted data is identical every time anything is encrypted. Certain controllers don't have fixed device addresses, or the host prefers to use a private address in place of the fixed address. The controller must have a random address programmed into it by the host to account for this. This address may then be used for advertising, active scanning, or opening connections.

A random number or a combination of a random number and an identity-resolving key is often used to create a random address. By utilising the LE Rand command, the host asks the controller for a decent random number. Then, using the LE Encrypt command, it feeds the plain-text data and associated identity-resolving key into the encryption engine. The LE Set Random Address command may then be used to set the random address using the resulting value. When the Command Complete event is received, the random address is made accessible for use by further commands.

Black Lists

Device addresses that can be whitelisted by the controller are kept on file. This is particularly helpful when looking for a few well-known items among the clutter of promotional materials in crowded areas. The first thing to establish is how big this list is since the white list has a predetermined size. Using commands, the list may then be controlled by clearing, adding, and removing devices. The list, controlled by the host, may be used by the controller to filter advertising packets.

The host sends the controller the LE Read White List Size command to read the size of the white. The controller responds by emitting a Command Complete event that has the most white list entries possible. Using the LE Clear White List command, the host can remove every entry from this list. The LE Add Device to White List command can also be used to add devices to the white list. It is also possible to remove a single entry in the white list by using the LE Remove Device from the White List command.

Reading a white list it should be noted that it is not possible to change the contents of the white list while it is being used. For example, if the device is an advertising and using the white list to filter the devices to and from which it will respond to scan requests or connection requests, the white list cannot be changed until advertising is disabled. The white list can then be changed and advertising re-enabled. The most primitive form of communication possible between two Bluetooth low-energy devices is broadcasting and observing the model. These use advertising and scanning to transmit and receive data.

The controller has two sets of data that can be transmitted by using advertising: advertising data and scan response data. It also has a set of parameters that are used for determining how and when

it should transmit advertising packets. The LE Set Advertising Parameters command allows the host to con, the advertising parameters. This includes the minimum and maximum interval that the host requires the controller to advertise, anywhere from 20 milliseconds to 10.24 seconds. The type of advertising is also specified here. Four types of advertising are available Connectable undirected advertising is used for general advertising of the advertising and scan response data. This allows any other device to connect to this device.

a) Connectable-directed advertising is used to request a particular peer device to connect.
b) This does not include any advertising data.
c) Schnabel undirected advertising is used to broadcast advertising data and scan response data to active scanners.
d) No connectable undirected advertising is used to just broadcast advertising data HCI advertising

Another parameter of the LE Set Advertising Parameters command is the address type to be used in the advertising data. This can either be the fixed device address or the random address just described. If the type of advertising used is directed advertising, the peer device address is also included. The final two parameters are the advertising channel map that is used to determine which of the three advertising channels should be used and the advertising filter policy. The filter policy determines if the white list is used to help filter out advertising packets that are received in response to an advertising packet. This can be set to one of the following states:

1. Allow a scan request or connect request from any device.

2. Allow a scan request only from devices in the white list, but allow connect requests from any device.
3. Allow a scan request from any device, but only allow connect requests from devices in the white list.
4. Allow only scan requests and connect requests from devices in the white list.

The next command that can be sent is the LE Read Advertising Channel Tx Power command. This returns the transmit power used when advertising. This is useful because it allows this value to be included in the advertising data or the scan response data to enable proximity pairing or sorting devices by path loss in a user interface. The host can then convert, the advertising data and scan response data by using the LE Set Advertising Data and the LE Set Scan Response Data commands.

Passive Scanning

To receive advertising data from peer devices, you can use passive scanning. The LE Set Scan Parameters command is used to con, the controller's scanning parameters. These parameters include the scanning filter policy as well as the following: The scanning interval and window are recommendations from the host to the controller that determines how often and for how long the controller should scan. The interval divided by the window determines the duty cycle for which the controller should scan. For example, if the interval is 100 milliseconds and the window is 10

milliseconds, then the controller should scan for 10 percent of the time. The slowest practical duty cycle that can still pick up a directed advertising packet directed at the controller is a 3.75-millisecond window every 1 second a 0.4 percent duty cycle.

It is possible to have the interval and window set to the same value. In this case, the scanning is continuous, with a change of scanning frequency once every interval. For example, setting the default parameters at a 10-millisecond interval and a 10-millisecond window results in a 100 percent duty cycle that changes the scanning channel every 10 milliseconds. The scanning filter policy determines if the white list is used or not. This can allow either all advertising packets to be received, or only advertising packets from devices in the white list to be processed. It should be noted that directed advertising packets that are not addressed to this device would also be discarded, even if sent from a device in the white list.

Once the scanning parameters have been con,d, scanning can be enabled by using the LE Set Scan Enable command. Once scanning, the controller sends events up for any advertising packets received that pass the scanning filter policy and other rules. This advertising data is sent to the host in an LE Advertising Report event. In addition to the device address of the advertiser, this event includes any advertising data that was in the packet and the received signal strength indication of this advertising packet. The signal strength can be combined with the transmit power data included in the advertising packet to determine the path-loss of the signal and hence give an approximation of range. The host can also end scanning by using the same LE Set Scan Enable command but with a parameter set to disable.

Active Scanning

The next complexity up from passive scanning is active scanning. This is used not only to obtain advertising data from peer devices but also their scan response data if possible. The commands used to con, and enable active scanning are identical to those for passive scanning. However, because the controller will need to send SCAN REQ packets to the peer device to obtain the scan response data, these packets need to include a device address. Therefore, you use an additional parameter in the LE Set Scan Parameters command to determine whether these Link Layer packets use the fixed device address or the programmed random address.

Future of Bluetooth Technology

Because it satisfies a fundamental requirement for connection nearby, Bluetooth has a bright future ahead of it. It was developed by nine major manufacturers in the communications and computing industries, including 3-COM, Ericsson, Lucent, IBM, Intel, Microsoft, Nokia, Toshiba, and others. Well over 1800 manufacturers from across the globe have joined the project since the original group's founding. Before the end of 2002, Bluetooth technology is anticipated to be included in more than 100 million gadgets, according to a market research assessment. Another forecast from the market research company Cahners In-Stat Group predicts that by 2005, there will be more than 670 million Bluetooth-enabled devices globally.

Bluetooth will have a significant impact on daily life as a consequence of the popularity of (WAP) Wireless Application Protocol, the adoption of smartphones, and portable gadgets. One of the

important technologies that may enable the roaming innovation ecosystem and obfuscate the lines between the homes, the workplace, the outside and the economy is Connectivity. Due to the difficulty in letting different devices communicate with one another, it was even impossible to imagine the variety of digital and creates understanding-tailored services that Bluetooth's seamless communication promises to provide. Already, a large number of Bluetooth pilot devices supported by major suppliers have entered the market, which is a very positive indicator of the adoption of the technology. Not only businesses creating Bluetooth-enabled items can support Bluetooth. Numerous other sectors might also be significantly impacted by Bluetooth applications [1]–[3].

As early as 1989, the service's creators had the idea of creating a short-link architecture for signal transmission. The 1.0 video was launched in 1999. Even though the first iterations of earphones were acknowledged to have flaws, they nonetheless served as a critical foundation for the advance of technology. The first version of the technology regulated transmission at a velocity of around 721 kbit/s. In the version that was introduced in 2004, the max power was increased to around 3 Mbit/s. The speed of later versions has increased. Because communication is established once apartments are connected, allowing one unit to observe the mechanism in the other, there are security risks. To determine which devices the telephone or computer is connected to, which is often as easy as searching up the division's name, is thus necessary [4]–[6].

Later, in 1999, the Special Interest Subgroup was created as a specialized organization to advance Bluetooth technology (SIG). This alliance brought together top companies in the transportation and telecommunications industries, including Nokia, Intel, Ericsson, Toshiba, and IBM, as well as later entrants Microsoft and Motorola. This technology initially performed best when used on cell phones since it provides a practical means of free data exchange. It is often utilised nowadays and may be found in many electrical items, including TVs, Wireless headphones and phones, GPS devices, and even the newest household appliances. Similar to the passive scanning settings, the scan interval and scan window specify how often the controller should check for advertising packets from devices.

To connect to any device on the white list, the initiator filter policy should be con,d to "use white list." Initial connection parameters specify the supervision timeout, the expected amount of data to be sent to or from the slave at each connection interval, the frequency with which the master transmits to the slave, the latency that the slave is permitted to use while ignoring the master, and more. It should be remembered that each device in the white list will have the same initial connection parameters. Therefore, this command is particularly helpful when connecting to many identical devices, such as automation sensors, but it is less helpful when connecting to numerous different kinds of devices. The LE Create Connection command's worst-case connection settings should be used by the host in certain situations.

The controller will issue the CONNECT REQ packet with all the necessary information and then produce the LE Connection Complete event if an advertising packet of a connectable advertising packet type is received from a device that is on the white list. Once the peer device has received the CONNECT REQ packet, it will also transmit the LE Connection Complete event up to its host. The connection handle used to identify data packets delivered from the host to the controller and from the controller to the host for this connection is included in the LE Connection Complete

event. This event also provides the controller's current position either master or slave. A device can advertise that it can establish a connection while also doing so; this makes it easier to identify which attempt was successful. The peer device address, as well as the connection settings for interval, latency, and supervision timeout, are also included in the event. The precision of the master's clock, which is necessary to calculate how many windows widening a slave device needs, is also included in the event. The host only receives this parameter for informative reasons.

Any advertising or initiating that was going on while the connection was being established immediately comes to an end once the controller sends the LE Connection Complete event to the host. Therefore, the host must send fresh orders to the controller to establish connections with other devices or carry on advertising. The same LE Create Connection command is used by the host to establish a connection with a single device. While other parameters are used to describe the device address of the peer device to which it is connected, the initiator filter policy is con,d to disregard the white list. The connection process is identical to starting a white list apart from these small variations; when the connection has been established, an LE Connection Complete event is produced on both devices. Sometimes a connection is established to a device that is not responding likely because it is far away from the initiating device and the host wishes to terminate the connection so that they may do another task. The host in this example uses the LE Create Connection command to try to connect to a particular device. This is also effective when connecting to a white list.

When the host wishes to stop this connection from being established, it sends the controller an LE Create Connection Cancel command. The controller then replies with an LE Connection Complete event for the first LE Create Connection command and a Command Complete event for the cancel command. It's crucial to finish all commands, and whether or not the connection was established, the LE Create Connection command is finished with the LE Connection Complete event. There is a race condition present in this, it should be mentioned. Before the LE Connection Complete event is provided to the host to inform it of the new connection, the host may send the LE Create Connection Cancel command to the controller at about the same time as the controller sends a CONNECT REQ packet to the peer device. In this scenario, the LE Create Connection Cancel command will be finished by utilising the Command Complete event; nevertheless, the LE Connection Complete event will be sent up to the host informing it of the newly made connection. Therefore, even when attempting to terminate the same connection, a legitimate connection may have been established. Connection Management Once a connection has been established, devices may begin to manage it to reduce power consumption, alter latency, begin encryption, or even end the connection altogether.

Update on Connection

The host of the master may modify the connection settings if the ones being used on a particular connection are no longer necessary. This might be because while the services are in use, a considerably longer connection interval is far more advantageous to preserve power than when the connection was first established, which had a very quick connection interval to aid the devices in initialising themselves. The LE Connection Update command allows the master to modify the connection settings. This contains the desired updated values for the supervision timeout, new

anticipated connection event length, and connection interval and latency parameters modifying the connection's settings Before delivering a Link Layer connection update request packet to the peer device, the controller replies with a Command Status event. The time of the moment at which these new connection settings will take effect is included in this packet. The LE Connection Update Complete event informs the host that the new connection parameters have been updated after this instant has passed and the new connection parameters are utilised.

The host can be aware of how the local channel is being used and want to let the controller know. The potential of the two radios directly interfering with one another would be reduced, for instance, if they were co-located with a Wi-Fi radio that is linked to an access point on a specific channel and communicates over the low-energy channels in the same portion of the band. Direct commanding the controller to submit a Link Layer channel map request to the peer is not possible). The LE Set Host Channel Classification instruction may be sent by the host to the controller. Each Link Layer data channel is identified by a bit field as either faulty or unknown. From the host's perspective, it is impossible to mark a data channel as acceptable since the controller may already be monitoring the packet error rates per channel and may have discovered that certain channels marked as unknown by the host are problematic.

Updating the channel map,

When the LE Set Host Channel Classification command is used, the host receives a Command Complete event right away. The channel map may be changed at any moment by the controller using the Link Layer control methods. Using the LE Read Channel Map command, the host may keep an eye on the channel map for a specific connection. This command indicates if every Link Layer data channel is currently in use or not.

Feature Exchange

The characteristics that are accessible on a connection may be found by the host. As an example, even though the peer controller supports encryption, it can only be utilised if the local controller does as well demonstrates how the LE Read Remote Used Features command may be used by the master's host to request the remote used features. This results in the return of a Command Status event and the exchange of a Link Layer feature request (LL FEATURE REQ) and response (LL FEATURE RSP). The feature response Link Layer message information is then sent by the controller to the host in the LE Read Remote Used Features Complete event.

Finding the Link layer version information might sometimes be helpful when troubleshooting devices, particularly ones that you do not directly manage. The manufacturer of this item may then be contacted using this information to assist you to resolve the issue. Either the master host or the slave host may ask for this information, enabling both components to troubleshoot the connection if required. The host sends the controller the LE Read Remote Version Information command to communicate the version information. The Link Layer version exchange then begins once the controller replies with a Command Status event.

After exchanging the version information, the controller delivers the LE Read Remote Version Information Complete event to the host along with the version information for the peer device. It

should be noted that the identical chain of events is returned if the LE Read Remote Version Information command is issued a second time, but no Link Layer operation will be executed since the version information is static and is cached in the controller. Additionally, this holds if the distant device and the local host have previously communicated version information. The completion event is sent right after the status event since the controller already knows the version information from the remote device.

As long as both parties have a shared secret, the host can activate the encryption of data packets while a connection is active. The Security Manager establishes this shared secret either at initial paring or via key distribution during bonding. It is necessary to take into account two sets of instructions and actions: one from the perspective of the master and the other from that of the slave. By delivering the LE Start Encryption command with the connection's encryption key included, the master's host may request that the Link Layer begin encryption. While encryption is being initiated, the controller returns a Command Status event. After then, the Link Layer begins the encryption process. Once encryption has begun, the controller notifies the master's host through the Encryption Change event if encryption is currently active or if there was a problem with encryption.

The series of actions and orders looks a little bit different from the host of the slave. The LE Long Term Key Request event, as shown in, 8-25, is the first one to alert the slave's host that encryption is being activated. The LE Long Term Key Request Reply command, which contains the key to be used for encrypting the connection, must be sent in response to this event by the host. This is a command, thus to finish it, a Command Complete event is utilised. The Encryption Change event is delivered to the host of the slave to inform it of the new encryption state once the connection has been encrypted.

Sometimes the host must resume encryption, either by using a new encryption key or simply by refreshing the initialization vector that is created when encryption first begins. From the perspective of the master's host, the orders and events that follow are the same as when encryption first begins, as shown in. To suspend and resume encryption, the Link Layer must transmit further packets. HCI master resuming encryption in events and orders are also the same from the host's perspective of the slave, as illustrated in.HCI Slave Restarting Encryption in

It should be noted that sending application or host data when encryption is off is not feasible in low energy. As a result, after encryption has been suspended, there is no need to send an Encryption Key Refresh Complete. This is due to an issue that was found in early versions of Bluetooth classic. Maximum security hosts would attempt to periodically renew keys to make them more challenging to decrypt. Data may still be delivered even when encryption had to be off to update keys.

The controller could have data in its buffers waiting to be sent, so the host couldn't get around this by simply halting its flow of data to the controller. As a result, hosts aiming to boost security in early iterations of Bluetooth classic decreased security and caused the controller to communicate unencrypted data. A subsequent version of Bluetooth Classic addressed this. There is often no need to stop encryption on a connection after a host has requested it. As a result, the security flaw in the

earlier Bluetooth standards is closed when encryption is suspended and data transmission is likewise paused.

Terminating a Connection

Any host may terminate a connection after both hosts have agreed that no more data needs to be exchanged over it or when continuing the connection would use more power than just cutting it off and rejoining later. As shown, the host sends a Disconnect instruction to the Link Layer to break the connection. The controller notifies the host of the Disconnect Complete event after terminating the connection. The Disconnect Complete event may also be sent to the host at any moment if the connection breaks because of a supervisory timeout or a failed encrypted message integrity check. When two devices have moved so far away from one another that they can no longer receive Link Layer packets from the peer device, supervision timeout often happens. Except in very unusual circumstances when a highly uncommon sequence of bit errors passes the cyclic redundancy check but fails the message integrity check, encrypted message integrity check failures should never take place.

The information and computer industries are anticipated to utilise Bluetooth technology widely. The market for smartphones is anticipated to expand with the use of airtime for wireless data indirectly increasing as a result of Bluetooth technology implementation. In the long run. The possibility to swap many networking capabilities with a sole Bluetooth module will allow manufacturers to save money on manufacturing costs. The International Organization for standardization is helping to expand the technology by defining advantageous implementation conditions, such as open, royalty-free access to the standard.

A protocol multiplexing layer called Logical Link Control and Adaptation Protocol (L2CAP) allows Bluetooth low energy to multiplex three distinct channels. In addition, it makes it possible to split and reassemble packets that are bigger than what the underlying radio can handle. The L2CAP layer also conducts several more, intricate tasks on a Bluetooth vintage radio. One of the fundamental ideas behind Bluetooth low energy is a fundamentally different connectionless paradigm, which states that a connection only has to be established when data needs to be sent and that the device may always detach at any moment. Only fixed channels are available since this requires the connectionless paradigm to be extended up to the L2CAP layer. There is no time lost waiting for the channel to be constructed since fixed channels have no setup settings to negotiate and are there as soon as the lower levels have made a connection.

L2CAP was not used when Bluetooth Low Energy was first developed. A Protocol Adaptation Layer (PAL) was formerly intended to function as a highly efficient but strictly constrained multiplexer between two protocols. The Attribute Protocol and a signalling layer seemed to be the PAL. Due to flexibility and old implementations, this was undesirable. Only one higher-layer protocol or the PAL's signalling layer could be supported by the PAL. There was no way to divide distinct protocols, no segmentation, and no reassembly. Layering protocols is one of the fundamental principles of protocol design; each protocol is self-contained. This implies that it is feasible to develop several system components together, such as the Security Manager. Each protocol is a distinct layer that may be tested separately at the time of implementation. Thus, the

PAL disregarded this straightforward regulation. However, it wasn't the design that made this strategy ineffective; rather, it was the lack of adaptability.

Segmentation and reassembly are tasks performed by most multiplexing layers. As a result, a big protocol packet from a higher layer may be divided into many smaller packets with the proper labels so that they can be communicated via a system with packet length limitations. An excellent illustration of this is an ATM network, where each packet is limited to a small amount of data, enabling quick switching between other streams. This makes it possible to simultaneously send bulk data and low-latency audio traffic. To facilitate segmentation and reassembly, the Host Controller Interface (HCI) makes use of the "start" and "continuation" bits on each data packet. The PAL, however, did not provide such a fundamental function. This meant that any application data in this layer could only be a maximum of 24 bytes in size. This stringent limitation ultimately led to the PAL's demise.

The firms that already had Bluetooth implementations and the companies that didn't broke off in the group building Bluetooth low energy when L2CAP was put up as a substitute. This would have required several months of contentious voting in certain standards organisations to try to create division; this is also sometimes related to disruptive political behaviour like attempting to cram the room with voting members to try to tilt the decision one way or the other. This is not the common method in Bluetooth. Instead, a report comparing the expenses of each strategy and demonstrating the cost of L2CAP addition was prepared. The battery life of a gadget that provided information once per second was lowered from 3.3 years to 3.2 years, which was the determining factor. L2CAP did shorten the device's battery life, although it did so only little compared to the 3 bytes of cyclic redundancy check (CRC) and the 7 bytes that came before each packet's payload, regardless of whether it included data. This is just another illustration of the careful consideration Bluetooth low-energy inventors gave to the system design implications of every choice.

L2CAP enables you to connect Bluetooth low energy to an already-in-use L2CAP implementation. Even though there are no protocols that can be used with Bluetooth low energy that enable this packet size, it also supports the complete segmentation and reassembly from Blue-tooth classic, thereby enabling packet sizes of up to 65,535 bytes in length. The channel model used by Bluetooth Classic is likewise preserved by L2CAP. Fixed and connection-oriented channels are the two types of channels available in Bluetooth classic. For the duration of the connection between the two devices, a fixed channel exists. These are mostly utilised as signalling channels, either for standard L2CAP signalling instructions or, starting with version 2.0, an Alternate MAC/PHY signalling channel. A few L2CAP signalling instructions may be sent to a peer device at any time to establish connection-oriented channels.

Using connection-oriented channels in Bluetooth classic, it is possible to treat the data from a specific pair of apps as being distinct from the data from other channels. For instance, connection-oriented channels could have a different flow specification or they might be streaming channels as opposed to best-effort channels, even if they can include more data integrity checks. When you have a complicated system with several, different sorts of data being delivered at once, connection-oriented channels are fantastic. Several different protocols, such as one stream for the high-quality audio from the phone to the car stereo, one stream for hands-free operation, another stream for the

phone book, and possibly another stream for an Internet connection, can all be used simultaneously by a phone and a car, for instance.

It may be difficult to open connection-oriented channels. The most recent standard calls for seven configuration settings for each L2CAP channel. This implies that before any data can be delivered, each of the configuration settings must be agreed upon in addition to the two messages that must be sent to request the establishment of a connection. This may take just a few more messages, or it could take a few hours to go through all the suggested values and counterproposals. Another difficulty that connection-oriented channels provide is the ability for a device to renegotiate various settings after they have all been set up and data has begun to flow. All of this reduces flexibility while raising the data connection's latency. This is a reasonable cost for the majority of Bluetooth old protocols and profiles since these connections are maintained over extended periods.

A channel is a straightforward idea in L2CAP. After all, L2CAP is a multiplexing layer, and as such, it contains several channels. An individual packet sequence from and to a single pair of services on a single device is a channel. Multiple channels may be active simultaneously between two devices. Bluetooth Low Energy only supports fixed channels. A fixed channel does not need to be con,d; it just exists as soon as the two devices are linked. There is still the possibility to introduce connection-oriented channels if it is deemed required in the future. The IDs for L2CAP channels. In Bluetooth, each channel identification is a 16-bit number. Use never should be made of the reserved channel identification 0x0000. The fixed channel for Bluetooth classic signalling is channel identification 0x0001. Although no profile presently utilises it, channel number 0x0002 is a fixed channel that is used for "connectionless data." When transferring data at a high rate is necessary, the Alternate MAC/PHY protocol is utilised with channel identification 0x0003. The Alternate MAC/PHY controllers utilise a test channel with channel identification 0x003F.

Three Bluetooth low-energy channels are available: For further details, see Chapter 10, Attributes. Channel identification 0x0004 is used for the Attribute Protocol, Channel identifier 0x0005 is used for Bluetooth low energy signalling, and Channel identifier 0x0006 is used for the Security Manager. Channel numbers from 0x0040 to 0xFFFF may be used for connection-oriented channels, whereas all other channel identities from 0x0007 to 0x003E are reserved. There is a 32-bit header and a payload in each L2CAP packet. Since segmentation and reassembly are presumed to be in use, the length of the packet must be given in the packet header to pinpoint its termination. The employed segmentation and reassembly strategy necessitates the labelling of packets across the HCI interface as well as inside each sent packet as a start or continuation packet for more information. The Host Controller Interface. There is no method to indicate that the end of the current packet is at a certain L2CAP packet segment. As a result, the only method to tell whether the current packet is finished is to send a new one, provided that one is available, or to include the packet length in the very first packet sent.

The header has a 2-byte length field and a 2-byte channel identification. It is followed by a payload of length bytes of data. Some extra headers and data may be included in the information payload in Bluetooth classic, however, with Bluetooth low energy, there are no further structures that are significant at the L2CAP layer. The information payload begins with a Maximum Transmission Unit (MTU) size of 23 bytes for all Bluetooth low-energy channels. MTU is the maximum size for

a specific L2CAP channel's information payload. This implies that all Bluetooth low-energy devices must be able to accommodate 27-byte packets sent over the air, including an L2CAP header of 4 bytes and a payload of 23 bytes of data. Host-level signalling takes place through the LE signalling channel. Each LE signalling channel packet has a single opcode, followed by any arguments. The LE signalling channel supports the following command opcodes:

1. Connection Parameter Update Request
2. Connection Parameter Update Response
3. Command Reject

An identification is always delivered together with the data payload of a signalling command. This 1-byte-long identifier is used to match answers with requests. For instance, if a request had the identification 0x35 and was submitted, the answer would be any other response with the same identity. With each request having a unique identity, this enables numerous requests to be active at once. Only after all other IDs have been utilised may an identifier be used again. Because of this, implementations utilise an increment operation to guarantee that this rule is followed. The only exception to this is: Never to use an identifier with the value 0x00. Duplicate instructions may be discreetly deleted as a result of the usage of identifiers. This would be advantageous if the command channels weren't trustworthy, but because they're always conveyed through a trustworthy carrier, this rule is seldom used. The rationale for identifiers is fairly straightforward in Bluetooth low energy since there is only one specified request and because this request may only be issued when no other requests are pending.

Any communication that was received by the device that is not supported might be rejected using the command reject. This command is the same as the reject command for Bluetooth classic. It has a reason code and could have some data in it. Either Command not understood or Signaling MTU exceeded are possible cause codes. When a command that the device does not support is issued to it, the Command not understood reason code is used. Even for currently undefined command codes, this should be delivered; doing so enables a device to be backwards-compatible with future revisions of the standards. When a command that is larger than 23 bytes is received, the Signaling MTU Exceeded reason code is used. The signalling channel's MTU is only set to 23 bytes by default, therefore if a command were to arrive that was 24 bytes or longer, the command refuse would be sent back in response.

Another reason code for Bluetooth classic is Invalid CID in Request, however, no instructions using a channel identification are provided for Bluetooth low energy, hence this code has never been utilised. Request and response for connection parameter updates. As shown in the connection parameter update request command enables the slave device to request that the Link Layer connection parameters be changed. These settings include the connection event interval, the frequency with which the slave wishes to be allowed to disregard the master, the slave latency, and the supervision timeout.

When the slave needs to change the connection settings for an active connection, it would use this command. For instance, if the connection event interval is too quick, too much power is wasted. If the slave latency were sufficiently high, this wouldn't be an issue, but if it weren't, the slave

would need to listen a lot. This may be helpful in some situations, such as when devices are initially bonding and exchanging a lot of messages to learn about one another's capabilities and services. For effective battery life, however, it is often essential to be able to reduce the number of connection events when the slave must listen.

The master may always start a Link Layer connection parameter update control operation at any time; this command is only usefully delivered from the slave to the master. The slave would react with a Command Reject command and the reason code Command is not understood if the command was delivered by the master since it would be seen as an error. This order may be issued at any moment by the slave. The master will reply with a Connection Parameter Update Response with a result code set to approve if it can alter the connection parameters after receiving the message. The Link Layer connection parameter update control mechanism will also be started by the master.

This is simply a request, after all, and the master may reject it if it doesn't like the parameters the slave requested by delivering a Connection Parameter Update Response with the result code set to denied. The slave is then given the choice to either acknowledge that the master requires or desires the connection settings it is presently utilising, or to cut off communication. At first, look, cutting off the connection would seem like a very extreme course of action, but if the slave's battery would drain in a week with the existing connection settings but would last for years with the required connection parameters, there may be no other rational course of action.

The slave might ask for a range of acceptable connection event intervals to lessen the likelihood that the master would reject the connection settings it sends. A well-thought-out slave would happily accept a variety of intervals. A master device's acceptance of a wide variety of connection intervals is severely constrained since it may simultaneously be performing other tasks, such as a low-latency conversational audio link or a high-quality audio connection. Depending on what it is doing at the moment, the range of intervals it can tolerate may fluctuate from what it was the previous time the two devices connected.

Having a reasonable-sized slave delay is another technique to boost the likelihood that the master will accept the connection specifications. The slave may then utilise a slave latency that offers it the optimum power usage by the master selecting the most appropriate connection event interval. For instance, the slave may ask for a connection interval range of between 100 milliseconds and 750 milliseconds with a slave latency of 5. If the slave wishes to synchronise every 600 milliseconds. The slave might synchronise every six connection events if the master selects a time interval of 100 milliseconds. The slave might disregard 2 of every 3 connection events if the master sets a synchronisation interval of 200 milliseconds, allowing it to reach its intended synchronisation interval of 600 milliseconds. Every other connection event might be disregarded by the slave if the master selects a 300 millisecond time limit. The slave might synchronise every 400 milliseconds if the master specifies that interval.

The Attribute Protocol Layer and the Generic Attribute Profile Layer are the two layers that will be discussed in this chapter. It is helpful to talk about both at the same time since they are connected in such a way. The Attribute Protocol's ideas were developed in a non-core working group before

being included in the core standard when Bluetooth low energy was developed in the Bluetooth Special Interest Group (SIG). However, an architectural choice was taken at the time of integration to divide the document into an abstract protocol and a generic profile. Although making this abstraction is helpful from the perspective of specifications, it is not helpful when trying to comprehend how attributes function. In theory, additional generic profiles might be positioned above the attribute protocol thanks to the abstraction of generic attribute profiles from the attribute protocol. Even while it's a possibility, it's not something that's being thought about right now.

Because any complexity would raise the protocol's cost and memory needs, it had to be very basic. Additionally, using the fewest amount of procedures was preferred. As a consequence, it was decided that the best first strategy would be to use a single protocol for everything. The three protocols used by Bluetooth low energy the Logical Link Control and Adaptation Protocol (L2CAP), the Security Manager Protocol (SM), and the Attribute Protocol mean that this objective was not achieved (AP). Every service above the Generic Attribute Profile (GATT), including the Generic Access Profile for name and appearance discovery, utilises the AP to keep the number of protocols to a minimum. This makes it possible to introduce new services that are built on top of the GATT for very little more money.

You may be wondering why protocols are considered to be so harmful. Protocols are fundamental to computers as a whole and, in some ways, to the rest of the world. For example, the Hypertext Transfer Protocol (HTTP) is used to download Web pages, the File Transfer Protocol (FTP) is used to transfer files, and the Secure-Shell protocol (SSH) is used to securely log in to another computer. Every protocol has been tailored for a specific use. Both utilising FTP to connect to computers and using HTTP to send a large number of files are inefficient.

Because Bluetooth's low energy does not attempt to communicate such a broad variety of data kinds, it stands apart from the multitude of Internet protocols. Given that streaming music or huge data transfers are not the goals of Bluetooth low energy, a single protocol may be created that just needs to handle the small number of data types that it supports. The cornerstone and main component of Bluetooth is a protocol known as the Attribute Protocol. Understanding Bluetooth low energy means understanding the Attribute Protocol.

It was evident when Bluetooth low energy was originally proposed that data is the foundation of any communications system. Many items contain data, and Bluetooth low energy provides a method for many other devices to access and use this data. Anything might be this data: the strength of your mobile phone's signal, the condition of your toys' batteries, your weight, how frequently you've opened the refrigerator today, how far you rode your bike this morning, the time, how much talk time is left on your headset, the most recent news story, who has just texted you, whether the chair is currently being sat on, who is in the meeting room, how long you've talked on your phone this month anything. A Bluetooth proximity device may display its transmit power level, which is an alarming level used to inform the user when the connection is lost. Additionally, it could have a device name so that the user can recognise it more quickly. Last but not least, the battery-powered gadget may provide its current battery level.

You must comprehend the crucial fact that some gadgets possess data while other devices want to employ it. This difference in Bluetooth low energy is crucial since it establishes which devices are regarded as servers and which as clients. A server is a device that stores data; a client is a device that uses data from a server and demonstrates this connection. There is one more crucial idea that you should comprehend. Data and state are quite different from one another. A value used to indicate a fact or measurement is known as data. Multiple devices may "know" data since it can be the room's temperature as read by the heating system or as measured by a thermometer, for example. A device's status or condition, including what it is doing and how it is running, is represented by its state, which is a value. Only one device has access to this state; it is claimed that only one device has this state information. The thermostat is supposed to reflect the current temperature in the room since it monitors the space's temperature.

In this book, the terms "state" and "data" refer to the information data that is stored on the server and, respectively, the information again, the data that is being transferred from the server to the client or is being retained on the client. Consequently, a server is an apparatus that stores a collection of state data. An object that receives or publishes this state data, maybe storing it locally as data, is referred to as a client. Since the server's status can have changed since the client's previous data transfer, the information on the client is not authoritative. Keep in mind that devices have a state and that this state will be stored on the server as you read the next sections.

External, internal, and abstract states are the three types of states used by Bluetooth low energy. The status of a physical sensor or comparable interface is represented by the current physical measurements. Let's use a bathroom scale as an example. Measurements for this device may include the room's current temperature, the battery level of the scales, or the weight of the most recent user. These are all referred to as "external states," states that, since they are being measured by an external sensor, may provide a different value each time they are read. The internal state is the next kind of state. State machines are used by certain devices to represent their internal states at any given time. They show how the gadget is presently operating rather than the exterior condition of a sensor. This may contain information about the status of the phone conversation, whether a GPS receiver is now being used to synchronise time, or even if the light is still changing brightness as a result of a previous dimming order.

An abstract state is the last category of state. This status information is only relevant at the current moment; it does not reflect the device's current internal or external condition. Examples of this kind of state include how to tell a light to be on or off, how to tell a device to alert right away, how to tell a device to control when time is synced and how to stop a synchronisation that is already in process. These are what are referred to as control points in the Attribute Protocol. These are often characteristics that can only be written or alerted and cannot be read. The Attribute Protocol permits exposing finite state machines, which is the most intriguing feature among the sorts of the state that may be disclosed. The internal state of the gadget is represented by a state machine. Additionally, a state machine may have one or more external inputs.

This abstract state, or control point, is decided by other state information or behaviour and these external inputs are brief instructions that shift the state machine from one state to another. It is feasible to completely reveal the operation and behaviour of a finite state machine on a device by

using the combination of internal state and control points. Two perspectives make this fascinating. The behaviour of a device may first be openly shown by disclosing finite state machines, their inputs, and their present state. Other devices may communicate with this device by exposing inputs. Second, a finite state machine's whole behaviour, including erroneous behaviour, may be defined. By doing this, the behaviour specified for that state machine will still describe what happens when any device sends an input on any control point into it. Think about a very basic state machine for a light for a minute. The light may be thought of as having a finite state machine with two states: on and off.

It could be able to read the light's present state and write this state to modify it. However, it is also important to take into account that there are three alternative inputs for this state machine: turn on, turn off, and toggle. A valid and logical future state may be mapped to the majority of state machine inputs. For instance, sending a turn to an off light will turn it on, while sending a turn to an on the light will turn it off. Even though this can be seen as improper behaviour, sending a turn-on to a light that is already on will keep it on. Similarly to this, sending a turn-off to an off light will keep it off. Additionally, sending toggle to light will turn it off if it was on. A light that was off will turn on when you send toggle to it.

The toggle input's intriguing feature is how much less data must be sent over the radio to alter the status of the light. A light switch would have to first read the current light state, toggle this information internally, and then write the new value to the light state if this abstract control point wasn't exposed. To do this, a minimum of three distinct messages must be sent: a request for the current light state, a reply that contains that light state, and a command to change the light state. The majority of these messages may be eliminated by simply instructing the abstract control point of the light state machine to accept the toggle command by adding the toggle command to the finite state machine of the light. The light may then be toggled on and off using the light switch. The light switch doesn't need to be aware of the previous or new state; the light performs the switching of the light state on the server.

As a result, exposing a state machine reduces the number of radio signals that must be delivered while simultaneously improving interoperability since it is difficult to direct a state machine into a state that has not been predetermined by its behaviour. Therefore, an interoperable and ideal protocol may be employed by describing all potential states and the behaviour of all inputs in all conceivable states. The service and profile architecture is the most intriguing architectural difference between Bluetooth classic and Bluetooth low energy. The majority of Bluetooth classic profiles also contain protocols, behaviour definitions, and compatibility standards. As a result, these traditional profiles are quite intricate and include several ideas. The fact that the profiles only include two different kinds of devices one for each end of the link is the largest issue.

Then, each device's behaviour is clearly stated. This could seem like a really helpful thing to do at first. It would be extremely helpful to specify what each device needs to do and how it interacts with the other device to allow a certain use case if you have a phone and a Bluetooth vehicle kit. Unfortunately, there are a few issues with this. The first issue with current profiles is that they do not precisely describe how a certain network device will behave on its own. The behaviour of the two devices is thus specified, but occasionally it is unclear exactly what the behaviour of each

particular device should be. This causes misunderstandings where one device thinks it is the other device's responsibility to do an action, and as a result, the action is never completed. In the Hands-Free Profile (HFP), for instance, it is stated that "either the HF or the AG should start the formation of an Audio Connection whenever required." So, the HF or the AG, which device begins the audio connection? What happens if they both try to accomplish this simultaneously? It's a nightmare for interoperability.

The simple answer to this is to explicitly state what each device should do by defining the behaviour of each device independently. The second issue with current profiles is that it is really difficult to utilise them in a manner that was not originally intended. It is challenging to make it function with a little different device since profiles specify how the two devices communicate with one another. It gets challenging to do this even inside profiles. For instance, the hands-free profile specifies a phone and a car kit, but the most common use case is a phone with a headset; the phone continuously sends user interface status updates to the other device. After all, it could be a car kit because it is unsure whether it is speaking with a headset or a car kit. A headset doesn't care if the signal strength drops from four bars to three bars, therefore this is a power waster. The simple answer to this is to specify each device's behaviour without requiring knowledge of its capability.

These issues have been resolved with Bluetooth low energy by using a completely new strategy. We have distinct papers that define the behaviour for a certain use case on the server and a client since we have a pure client-server architecture. The behaviour of the client is specified in a profile specification, while the behaviour of the server is specified in a service specification. This implies that the behaviour that is accessible via these attributes as well as the state that is exposed in the server by utilising an attribute database are both defined by the service design. A service may have certain properties that may be read, providing data that is either historical or current. Some properties may allow for writing, which enables orders to be transmitted to the service. When paired with control points, some features of a finite state machine disclose its state, resulting in completely exposable behaviour. The profile specifications specify how to utilise one or more services to allow a certain use case for instance, how to set up a service's exposed attributes in an attribute database on the server to request a specific action from the client.

The server has a known and defined behaviour, which is the split's major benefit. It doesn't care how the client uses it; it just performs what it is supposed to do according to the service definition. This indicates that the service is client-independent and that each unit can be tested. If a customer needs to utilise the service, they can. For instance, if there is a time service, one client may use it to get the current time, another client may use it to check the time frequently to see how its clock is drifting, and yet another client may use it to request that a GPS receiver be used to get the most precise time possible. The time service simply performs it, regardless of what the client is doing.

A light, for instance, may provide a client with access to a Light Service with two pieces of information: the present physical light state and the abstract light control point. Because the behaviour must be consistent across all instances of the light service, a light switch would implement the light profile that understands how to locate the light service, read the current light state, and control the light state. Using this method benefits the customer as well. The profiles are significantly easier to write since they do not specify server behaviour; instead, this is done by the

services. In essence, client profiles are a collection of guidelines for finding, connecting to, configuring, and utilising services.

They also provide common practices for carrying out different tasks that the customer requests. Any service combination may be used by the clients and their profiles to accomplish their objectives. To track temperature over time, for instance, a client might utilise both a time service and a temperature service without the requirement for a real-time clock within the client or for the data to be collected in real-time. Consider a home security system that detects an empty home but the owners want the property to seem inhabited by randomly turning on and off the lights. This might be achieved by creating a new profile that carried out a few very basic sets of instructions, as seen in the script below: An alternative profile that is represented by the following set of instructions may be used, for example, if you were in charge of an office and wanted to ensure that all the lights were turned down while no one was in the office:

One of the most effective ideas in Bluetooth low energy is the merging of several services in the client. Each service may be kept relatively basic; the system's complexity and richness come from the integration of services. Services must be atomic for this to be true. Atomic in this case refers to the fact that the services carry out a single set of operations. Services may be reused by several customers, each of whom is doing a separate task, by being made atomic. As a result, the first issue with classic profiles that it is almost difficult to create a classic profile with the explicit state can be resolved.

The services may be kept atomic and the amount of behaviour in each service can be kept to a minimum by describing them as independent specifications. Due to the minimal amount of behaviour, it is much simpler to explain exactly what a service performs, making it also simpler to verify this using accepted unit testing techniques. As a result, clients can rely on the explicit state that atomic services have.

This strategy also addresses the second issue with traditional profiles, which is that it is very hard to utilise a traditional profile in a manner that was not intended. It is feasible to design clients that may utilise these services by dividing the use cases into atomic services with known behaviour. These may be used by clients alone or in conjunction with other services. Combining services is less complicated since the services have specified behaviour. Additionally, because each service is atomic, behaviour does not "bleed" from one service to another. Each service is independent of the others, and neither does its behaviour rely on the condition of another service. This implies that services may be merged in any logical sequence.

This implies that customers may make inventive use of services on a smartphone. Profiles are just used to specify the common methods that devices may interact with for a certain use case; they are not necessary as services dictate how devices interact. A device may thus decide that it can integrate the temperature service, time service, and power production service from solar panels to identify the present weather.

The Bluetooth SIG is probably not going to establish a profile or use case for this, but a manufacturer could want to. The server's activity may be combined in innovative, creative, and practical ways in any client by breaking it down into services.

Bonding

Some gadgets need a relationship that lasts longer than just one connection. Other devices could want to communicate data to a certain device rather than any other device in the vicinity, or they might only want to transmit data to a device that has been verified to be the same as it was the previous time. This is accomplished by using a security paradigm that, at its core, produces a "bond." To create a secure connection with a device, a client must first connect to it, locate some services it can utilise, and then develop a long-term relationship with it. These secure connections confirm the identity of the device first. The connection is then encrypted to preserve secrecy. The devices then share a few pairing details. The important part is that the client forms a "bond" with the device if the pairing information is saved on the client. This is crucial because it prevents the client from having to reauthenticate and share pairing information every time it connects to this device. The devices just have an authorised and private data connection after it simply encrypts the link using the information it has saved as part of this connection.

The server will save the configuration information for this client when devices are bonded. With this technique, a device that reconnects may start receiving alerts right away without having to change the server configuration. The client would be informed if the battery level changes while it is connected, for instance, if the central device con,s a feature like a battery level to be alerted. However, the server won't be able to transmit the messages if the client loses connection with it. Bonding enables the notice to be issued instantly when the client reconnects to the server.

It's important to take into account this negative effect. The client could be required to check the current state after reconnecting, while the server might disregard any state changes for bonded devices. This may be wasteful for the client and server if the data changes rarely. This model also implies that the client doesn't regularly disconnect and rejoin since doing so would result in much higher reconnect costs because every item of state would need to be read. A different strategy used in Bluetooth low energy requires the server to keep track of both the fact that a character is set for notification and the fact that the value has changed when a client is offline. Each notifiable characteristic that would be set when the characteristic value changes must now have an extra bit associated with it. The characteristic value can only be informed if it changed when the client reconnects after checking these bits.

The client doesn't "connect" with a server as a result of the server storing the configuration data for a bonded connection. Instead, the client binds and sets up a peripheral to carry out a certain function, and then, as and when necessary, the client disconnects and reconnects to the peripheral. When the peripherals have data to deliver, they will be able to connect, and when necessary, the central device will reconnect to these bonded devices and promptly receive the data. This is true even if the peripheral is carrying out a request from the primary device. As an example, a central could request that a peripheral start syncing time before disconnecting it. Even if there is no assurance that the synchronisation was successful, the central device may reconnect to the peripheral at a later time and check on the status of the time synchronisation.

The client's ability to recall details about the handles of peripheral attribute handles is another significant benefit of this method. This implies that the client may remember the attribute handles

after the central device has identified the set of services on a peripheral. Without doing another service scan, the central device may read and write these attribute handles when it re-connects to the peripheral. This shortens the period needed to reconnect to a device before using it. The client of a central device may remember or cache the sets of services and characteristics between connections, as was mentioned in the section above. Some gadgets will be able to add or modify services. For instance, a peripheral's firmware may have been upgraded, or a computer or smartphone may have added an application that includes a service. All queries to that server will then fail, and the client won't be able to read any characteristics. This safeguards the client from reading the incorrect attributes.

The client will also get a notice from the GATT Service stating that the service changed in addition to these error messages. For the client to respond appropriately when it gets this message, it should have saved the handle of this attribute. The list of handles that have changed is included in the Service Changed notice. This implies that just the range of handles for a single service will be included in the notice if a device has only added or deleted that one service. The handle range may contain all of the device's handles if all of its services have been altered, such as when the operating system has been upgraded.

Service Changed is only important for bound devices, so keep that in mind. The client cannot cache attribute handles and will not be notified of any service changes if the central device does not bond with a peripheral device. As a result, whenever a client connects to a pair of non-bonded devices, the client must refresh the server's list of all services and attributes. The main consideration when designing a central device is what it supports. Typically, the client functions of one or more profiles will be implemented by a central device. The significance of this requires an explanation of profiles. A device's functionality for a specific use case is described in a profile. Roles are specified inside a profile. A device that may function as a component of an ecosystem of devices that support the use case of the profile is defined by a profile role.

The collection of services that a device must implement is specified by each profile role. Some profile positions don't need any services at all, while others merely need one service or a lot of services. Profiles often specify two responsibilities, such as a reporting role and a monitor role. The reporter job would be implemented in a peripheral device and would call for a single service. A central device would be used to implement a monitor function, which wouldn't need any services. The proximity profile, for instance, specifies two profile roles: the proximity reporter and the proximity monitor. The Immediate Alert Service, the Link Loss Service, and the Tx Power Service are all implemented through the proximity reporter. Each of these three services behaves as specified in its accompanying service specification when acting independently on the reporter. This profile model's flexibility and consequent future-proof ness relies on the independence of the services.

How a client makes use of these three services to provide the proximity use case is specified by the proximity monitor. For instance, the monitor may read the current Tx Power characteristic and compare it to the received signal intensity of packets from the reporter after the services and characteristics have been identified. The assumed distance between the two devices may then be verified by the proximity monitor as being appropriate. If the proximity sensor determines that the

distance is not acceptable, it might enter the alarm level characteristic in the Immediate Alert Service. The alarm level characteristic may also be recorded by the proximity monitor in the Link Loss Service. In this method, the proximity reporter may still warn in a controlled manner even if the signal strength declines so fast that the connection drops before the signal strength reduction is observed or the alert level characteristic in the Immediate Alert Service can be entered.

The proximity monitoring use case is made possible by this mix of services. However, by defining the alert level characteristic of the Immediate Alert Service, it is possible to utilise the Immediate Alert Service to enable various use cases, such as letting the user know which device is chosen on a user interface. A distinct profile would be used to characterise this client's behaviour. Finding the services that the peer device provides must be the first step for a profile. It may do this by first locating the main services. Locating all main services or finding primary services by service type may do this. This enables a profile that only needs to utilise one service to locate that service, omitting the need for any more services, on the device. The user interface alerting service, for instance, would only need to locate the one service it makes use of.

Finding multiple services could be necessary for a more complicated profile. Three services must be found to use the proximity monitor profile role. It could accomplish this either by locating all main services and then just saving the services it needs, or it could do this by doing the same single service search one at a time. A sophisticated client typically only locates all services once before caching them for subsequent use. The cached list of services is utilised when an application that implements a certain profile role requests all of the services on a device. This is feasible because notice is provided to the client whenever the list of services on a server changes, informing them that the services have changed.

The set of characteristics will be necessary after the set of services has been identified on the peer device. Similar to services, the profile may simply need one attribute or it may require several qualities. Some features could be optional, thus to see whether they exist in a service, you must search for them. The customer will typically find all of the previously identified features for each service. This is because this is a very quick and effective operation for the most basic service with only one attribute. Additionally, if the service implements multiple optional qualities, this procedure discovers every one of them. Once the traits have been identified, they may be put to use. The proximity monitor client, for instance, may enter the necessary alarm level into the Alert Level characteristic value for the Immediate Alert Service.

It should be noted that for certain services with just one readable characteristic, the process of discovering the characteristic and reading its value may be integrated into a single action. For instance, using the Read Characteristic by Type GATT technique, the Tx Power characteristic may be found and read after the Tx Power Service has been identified. This implies that an additional step to identify the traits mentioned before is not necessary. Certain traits may facilitate being informed or suggested. A client must first locate the Client Characteristic Configuration Descriptor to access these server capabilities. The client does this by locating any more characteristics inside this characteristic group following the characteristic value. This will uncover every extra descriptor for the attribute. Once it has been identified, the Client Characteristic Configuration Descriptor may be written with the appropriate value to allow alerts or indications.

Security for Profile

The client must connect to the server if it wishes to swiftly detach and rejoin in the future or if certain conditions call for an encrypted or authenticated relationship. Any characteristic value in a service may be attempted by the client to read or write. If the proper permissions are in place to read or write this characteristic, the service merely has to reply with the value of the characters or with the answer indicating the value was written. The server will reply with the proper error code if the permissions are incorrect. The client might try to pair or bond with the server to enable the proper degree of security if the error code indicated that the client had inadequate security. The pairing would be adequate if the client was simply viewing the information once. Bonding would also be necessary if the client would repeatedly reconnect. The connection will be encrypted once the devices have the necessary security, at which point the client may repeat the unsuccessful request.

Broadcast only

Some devices will desire to exchange a single quantum of data with several other devices. The broadcasting paradigm is the most effective and energy-efficient method to do this. For instance, a gadget that knows the time would broadcast that information to all other devices in the vicinity so that they could all receive the current time. An accessory that solely does transmitting might use very little power. It is not required to be connectable or discoverable. It is not necessary for it to accept any connections from central devices or to have a large, discoverable attribute database. Just relevant information is broadcast continuously. The device's broadcast interval or frequency is one thing to take into account. It is possible, for instance, to broadcast the time 10 times per second, or every 100 milliseconds.

However, it is not necessary to broadcast the time again within one second since the current time is often only accurate to the nearest second. As a result, the frequency of certain data broadcasting may depend on how long a user is prepared to wait for the data to become accessible. Since those who are interested in this information only sometimes listen for it, the current time is presumably something that can be broadcast at rather modest intervals. Let's say once each day or maybe once per week. Therefore, once every 10 seconds would be a suitable broadcast interval. It is noteworthy that a wall clock that receives time synchronisation from a broadcaster once per day only has to be precise to the nearest second for that day. By syncing with an accurate time transmitter once each day, a wall clock that would otherwise be off by 30 seconds per month would become more accurate. All peripherals will initially be discoverable, except those that exclusively transmit. When a peripheral is discoverable, it is promoting itself to any nearby scanning central devices searching for peripherals. Limited and universal discoverability are the two categories of discoverability. After engaging with the user, a peripheral is only temporarily discoverable. The peripheral would typically be discoverable at all other times when it is required.

For instance, a peripheral's batteries could only be partially discoverable when they are first installed by the user. This enables the user interface of a central device, often carried by the same user, to show this new device at the top of a list. Additionally, peripherals would be limited-discoverable as soon as a device's connect button is pushed. Again, the user would anticipate that

after pressing the connect button, the peripheral will appear at the top of the list of devices. There is no need that peripherals always be discoverable. Devices that have linked with a central device often stop being discoverable or only start to become discoverable again after pressing the connect button. This offers two benefits. First, information relating to discoverability, such as the device name or the current Transmit Power, is not required to be included in the advertising packets of peripherals that are not discoverable.

Second, central devices that are searching for peripherals to find will only wish to present peripherals that the user is interested in, not all of the nearby ones. The list provided to the users of central devices will be considerably more manageable and valuable if peripherals are not always discoverable. Connect ability and discoverability are extremely similar in that they both employ advertising packages but have distinct objectives. While connectable devices are normally bonded with one or more other devices and would only accept connections from those devices, discoverable devices are often not linked with any other devices. When a device is connectable, it accepts requests for connections from initiating central devices. A peripheral that is connectable essentially accepts connections from any device that offers it a packet of connection requests. This promiscuous conduct is not ideal in terms of energy use. It may take a long time to communicate with the peripheral if a central device can connect with it, consuming power from the peripheral and probably more importantly preventing the peripheral's linked centrals from connecting to it.

This issue may be fixed by configuring a peripheral device's controller to only accept connections from a select group of central devices. Any connection request from a device that is not on the white list of central devices is rejected. The list of central devices is kept in a white list. By doing this, the peripheral only has to utilise power to communicate with its bonded devices since it only needs to accept connections from the central devices that are bonded and on the white list. Only when the device has been bonded can it be used in this model, which only accepts connections from bonded devices in a white list. The device has to be promiscuous and accept connections from all devices before this may be enabled. This usually only occurs when the device is first connectable and has not yet been coupled with any other devices.

The peripheral normally exposes one or more services once connected. Each of these services captures the atomic operation of a peripheral component. Through the use of an Attribute Protocol server, these services are made available through attributes. Typically, an attribute database refers to the collection of attributes in a server. The Generic Attribute Profile determines how these characteristics are arranged (GAP). It is organized as follows: A service declaration that identifies the type of service comes first for each service in the attribute database. This statement is followed by the characteristics of the service. The service declaration for the subsequent service is shown next. There may be one or more other services that are included inside each service.

Included services enable the addition of more complex behaviour inside a service without expanding its primary functionality. As a result, it is encouraged to define services as those that only implement a tiny portion of functionality. The Battery Service, for instance, merely displays the battery level. Two batteries may be included in a device like a camera, one for the flash and the other for the main body of the camera. These parts are capable of having independent services, such as the Camera Flash Service and the Camera Service. Therefore, the battery service for the

primary camera as well as the flash service would be included in the camera service. A separate Battery Service for the flash would be part of the Camera Flash Service.

Each service may also have one or more qualities. Fundamentally, a characteristic is only one accessible value. However, each feature also includes a characteristic declaration that outlines the nature of the data, and how it may be accessed, and may also provide more details on its configuration or formatting. These individual attributes inside the attribute database comprise each of the service declarations, service includes, characteristic declarations, characteristic values, and characteristic descriptors. Therefore, depending on the functionality it is exposing, the peripheral just exposes the characteristics it needs to disclose and waits for a client to interact with them. The essential building elements of services are characteristics. Simply said, a characteristic is a value that a customer may engage with. The characteristic type and, in the end, an XML file that provides the specification for the characteristic, decide the format of a characteristic's value. This XML file does not describe a characteristic's behaviour; rather, a service definition does. As a result, numerous features of the same kind but distinct behaviours might exist inside a single peripheral.

For instance, the Immediate Alert Service and the Link Loss Service both utilise the Alert Level feature, but the behaviour is different. The Alarm Level characteristic in the Instant Alert Service is solely writeable and instructs the peripheral to send an immediate alert depending on the value provided. This value is a control point characteristic, which means that it cannot be read since its value is instantly consumed. The Alert Level characteristic in the Link Loss Service is both reading and written and only generates an alert when the central and peripheral become disconnected, depending on the value of the characteristic at the point of disconnect. The characteristic of this service has a state and may be read and written. The value is saved when it is written so that it may be read at a later time.

Questions for Practices

1. What is the small power factor in Bluetooth?
2. What is an IP license?
3. What is ism band?
4. Why do speeds almost always increase in Bluetooth signal?
5. What is the architecture for protocols?
6. What are the Bluetooth specifications documents?
7. What are the applications for Bluetooth?
8. What is the security of Bluetooth?
9. What are the Bluetooth capabilities?
10. What is the Bluetooth protocol?

Bibliography

[1] M. Cäsar, T. Pawelke, J. Steffan, and G. Terhorst, "A survey on Bluetooth Low Energy security and privacy," *Computer Networks*. 2022. doi: 10.1016/j.comnet.2021.108712.

[2] A. Aza, D. Melendi, R. García, X. G. Pañeda, L. Pozueco, and V. Corcoba, "Bluetooth 5 performance analysis for inter-vehicular communications," *Wirel. Networks*, vol. 28, no. 1, pp. 137–159, Jan. 2022, doi: 10.1007/s11276-021-02830-9.

[3] R. Admiraal, J. Millen, A. Patel, and T. Chambers, "A Case Study of Bluetooth Technology as a Supplemental Tool in Contact Tracing," *J. Healthc. Informatics Res.*, 2022, doi: 10.1007/s41666-021-00112-9.

[4] A. Comuniello, A. De Angelis, A. Moschitta, and P. Carbone, "Using Bluetooth low energy technology to perform ToF-based positioning," *Electron.*, 2022, doi: 10.3390/electronics11010111.

[5] A. K. Sultania and J. Famaey, "Batteryless Bluetooth Low Energy Prototype With Energy-Aware Bidirectional Communication Powered by Ambient Light," *IEEE Sens. J.*, 2022, doi: 10.1109/JSEN.2022.3153097.

[6] H. A. de S. Mourão and H. A. B. F. de Oliveira, "Indoor Localization System Using Fingerprinting and Novelty Detection for Evaluation of Confidence," *Futur. Internet*, 2022, doi: 10.3390/fi14020051.

[7] M. Ghavidel, N. Khademi, E. Bahrami Samani, and L.-M. Kieu, "A Random Effects Model for Travel-Time Variability Analysis Using Wi-Fi and Bluetooth Data," *J. Transp. Eng. Part A Syst.*, 2022, doi: 10.1061/jtepbs.0000624.

[8] W. T. Sung, I. V. Devi, S. J. Hsiao, and F. N. Fadillah, "Smart Garbage Bin Based on AIoT," *Intell. Autom. Soft Comput.*, 2022, doi: 10.32604/IASC.2022.022828.

[9] S. Xi, J. Huang, B. Chen, and G. Liu, "A compact dual-band multi-input multi-output antenna for 5G/WLAN/Bluetooth applications," *Microw. Opt. Technol. Lett.*, 2022, doi: 10.1002/mop.33089.

[10] Y. Liu *et al.*, "Electronic skin as wireless human-machine interfaces for robotic VR," *Sci. Adv.*, 2022, doi: 10.1126/sciadv.abl6700.

[11] M. del C. Rey-Merchán, J. M. Gómez-de-Gabriel, J. A. Fernández-Madrigal, and A. López-Arquillos, "Improving the prevention of fall from height on construction sites through the combination of technologies," *Int. J. Occup. Saf. Ergon.*, 2022, doi: 10.1080/10803548.2020.1815393.

[12] H. B. Amor, C. Bernier, and Z. Prikryl, "A RISC-V ISA Extension for Ultra-Low Power IoT Wireless Signal Processing," *IEEE Trans. Comput.*, 2022, doi: 10.1109/TC.2021.3063027.

[13] H. Pirayesh and H. Zeng, "Jamming Attacks and Anti-Jamming Strategies in Wireless Networks: A Comprehensive Survey," *IEEE Communications Surveys and Tutorials*. 2022. doi: 10.1109/COMST.2022.3159185.

[14] S. Sarkar and J. Hasan, “Contact tracing using Firebase and Bluetooth Low Energy,” *SSRN Electron. J.*, 2022, doi: 10.2139/ssrn.4009881.

[15] S. Sutar, K. Malladi, P. Mekala, and S. Goel, “Vulnerability Analysis of Bluetooth Technology for Defensive Application,” 2022. doi: 10.1007/978-981-16-5655-2_71.

[16] T. P. Barbosa, L. A. R. da Silva, F. J. P. Pujatti, and J. C. H. Gutiérrez, “Hydraulic hybrid passenger vehicle: Fuel savings possibilities,” *Mech. Based Des. Struct. Mach.*, 2022, doi: 10.1080/15397734.2020.1714447.

[17] S. Ahmed, Y. Xiao, T. T. Chung, C. Fung, M. Yung, and D. D. Yao, “Privacy Guarantees of Bluetooth Low Energy Contact Tracing: A Case Study on COVIDWISE,” *Computer (Long. Beach. Calif).*, 2022, doi: 10.1109/MC.2021.3125611.

[18] Z. He, W. Li, H. Salehi, H. Zhang, H. Zhou, and P. Jiao, “Integrated structural health monitoring in bridge engineering,” *Automation in Construction*. 2022. doi: 10.1016/j.autcon.2022.104168.

[19] Y. Wang *et al.*, “Tea Analyzer: A low-cost and portable tool for quality quantification of postharvest fresh tea leaves,” *LWT*, 2022, doi: 10.1016/j.lwt.2022.113248.

[20] W. Wu, L. Shen, Z. Zhao, M. Li, and G. Q. Huang, “Industrial IoT and Long Short-Term Memory Network-Enabled Genetic Indoor-Tracking for Factory Logistics,” *IEEE Trans. Ind. Informatics*, 2022, doi: 10.1109/TII.2022.3146598.

[21] M. Caux *et al.*, “PassStat, a simple but fast, precise and versatile open source potentiostat,” *HardwareX*, 2022, doi: 10.1016/j.ohx.2022.e00290.

[22] A. Dudnik, I. Bakhov, O. Makhovych, Y. Ryabokin, and O. Usachenko, “Models and Methods for Improving Performance of Wireless Computer Networks Based on the Decomposition of Lower Layers of the OSI Reference Model,” *Int. J. Emerg. Technol. Adv. Eng.*, 2022, doi: 10.46338/IJETAE0122_15.

[23] R. Brito-Pereira, C. Ribeiro, N. Pereira, S. Lanceros-Mendez, and P. Martins, “Printed multifunctional magnetically activated energy harvester with sensing capabilities,” *Nano Energy*, 2022, doi: 10.1016/j.nanoen.2021.106885.

[24] X. Li *et al.*, “Optimization strategy of wind energy harvesting via triboelectric-electromagnetic flexible cooperation,” *Appl. Energy*, 2022, doi: 10.1016/j.apenergy.2021.118311.

[25] T. M. Sharrukh, V. R. Rajapriya, P. Paviethra, and S. Sadhishprabhu, “Design of Rover with Live Video Transmission,” 2022. doi: 10.1007/978-981-16-4625-6_6.

[26] A. Mohanty and S. Sahu, “4-port UWB MIMO antenna with bluetooth-LTE-WiMax band-rejection and vias-MCP loaded reflector with improved performance,” *AEU - Int. J. Electron. Commun.*, 2022, doi: 10.1016/j.aeue.2021.154065.

[27] M. Barzgari, A. Ghafari, M. Meghdadi, and A. Medi, “A Current Re-Use Quadrature RF Receiver Front-End for Low Power Applications: Blixator Circuit,” *IEEE J. Solid-State Circuits*, 2022, doi: 10.1109/JSSC.2022.3147101.

[28] J. Gao, T. Fan, Z. Qin, and W. Sun, “A study on the cabin indoor localization algorithm based on adaptive K-values,” *Int. J. Distrib. Sens. Networks*, 2022, doi: 10.1177/15501477211073044.

[29] P. C. Ng, J. She, and R. Ran, “Compressive RF Fingerprint Acquisition and Broadcasting for Dense BLE Networks,” *IEEE Trans. Mob. Comput.*, 2022, doi: 10.1109/TMC.2020.3024842.

[30] A. L. Popescu, N. Popescu, C. Dobre, E. S. Apostol, and D. Popescu, “‘IoT and AI-Based Application for Automatic Interpretation of the Affective State of Children Diagnosed with Autism,” *Sensors*, 2022, doi: 10.3390/s22072528.

[31] Z. Wang, J. Chang, H. Zhi, C. Li, and L. Feng, “A PDA functionalized CNT/PANI self-powered sensing system for meat spoilage biomarker NH3 monitoring,” *Sensors Actuators B Chem.*, 2022, doi: 10.1016/j.snb.2021.131292.

[32] O. Salem, K. Alsubhi, A. Shaafi, M. Gheryani, A. Mehaoua, and R. Boutaba, “Man-in-the-Middle Attack Mitigation in Internet of Medical Things,” *IEEE Trans. Ind. Informatics*, 2022, doi: 10.1109/TII.2021.3089462.

[33] V. J. Schneider, N. Bush, D. Vitus, R. W. Carpenter, M. Robinson, and J. Boissoneault, “A virtual reality platform for the measurement of drinking topography,” *Drug Alcohol Depend.*, 2022, doi: 10.1016/j.drugalcdep.2021.109246.

Reference Books for Further Reading

1. Bluetooth Essentials for Programmers by Albert S Huang, Larry Rudolph
2. Make: Bluetooth: Bluetooth Le Projects with Arduino, Raspberry Pi, and Smartphones by Allan, Don Coleman, Sandeep Mistry
3. Bluetooth 1.1: Connect Without Cables by Jennifer Bray, Charles F Sturman
4. Bluetooth Low Energy: The Developer's Handbook by Robin Heydon
5. Inside Bluetooth Low Energy by Naresh Kumar Gupta
6. Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking by Kevin Townsend, Akiba
7. Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking by Kevin Townsend, Akiba