

Understanding the Architecture of the Bluetooth Low Energy Stack

Erick John Reyes, Senior Firmware Engineer
 Mary Grace Legaspi, Firmware Engineer, and
 Eric Peña, Embedded Systems Architect

Abstract

This article provides a deeper understanding of the Bluetooth® Low Energy (BLE) stack architecture and how to use existing BLE applications to maximize the full potential of low power, wireless communications. This knowledge is essential to efficiently and reliably design, troubleshoot, and optimize applications.

Introduction

Bluetooth Low Energy (BLE) is a key technology for the Internet of Things (IoT) ecosystem. While BLE was initially developed as a wireless protocol to replace cables in consumer products such as wireless keyboards, mice, and headsets, it has evolved into much more than just a cable replacement. It now plays a significant role in various industries, including medical, retail, and automotive, as well as industrial applications such as location tags and instrumentation control.

According to the 2023 Bluetooth Market Update, there is a 9% compound annual growth rate (CAGR) for Bluetooth wireless technology-enabled device shipments from 2023 through 2027! With this growth, BLE device shipments are expected to

more than double, with 97% of all Bluetooth technology-enabled devices incorporating BLE by 2027.¹

BLE is a wireless technology that was introduced with the Bluetooth 4.0 specification in July 2010. Previously known as Bluetooth Smart, BLE is specifically designed for ultra low power devices.

While the traditional Bluetooth technology we are familiar with is used for tasks like pairing our smartphones with headsets and transferring large amounts of data such as music and photos, BLE serves a different purpose. Bluetooth is capable of handling larger data transfers but consumes more battery power as a result. In contrast, BLE is optimized for applications that do not require extensive data transfers, making it ideal for a wide range of power-sensitive applications. Unlike Bluetooth, which remains active and consumes power even when not in use, BLE spends most of its time in sleep mode. It only wakes up when a connection is initiated, and the connection time typically lasts for just a few milliseconds. This efficient power management, combined with data rates up to 1 Mbps or even 2 Mbps in BLE 5.0, allows BLE devices to operate with minimal power consumption.

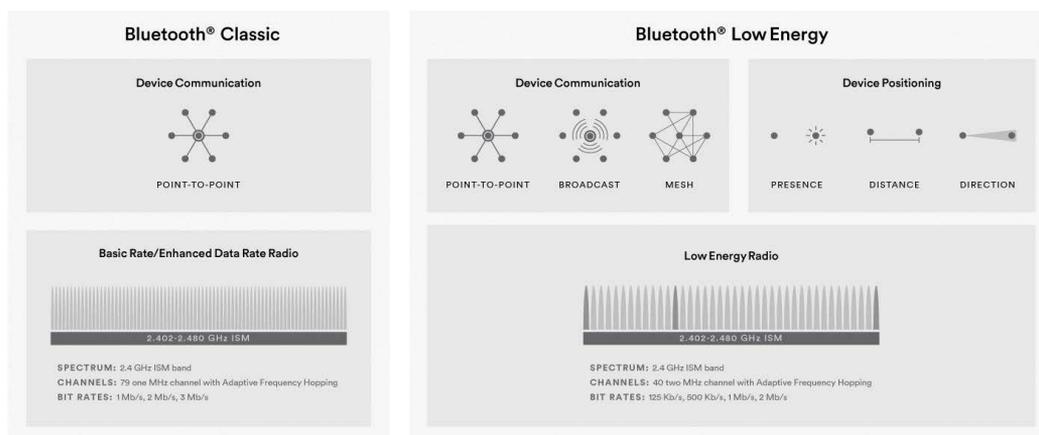


Figure 1. (a) Bluetooth Classic and (b) Bluetooth LE brief specifications.²

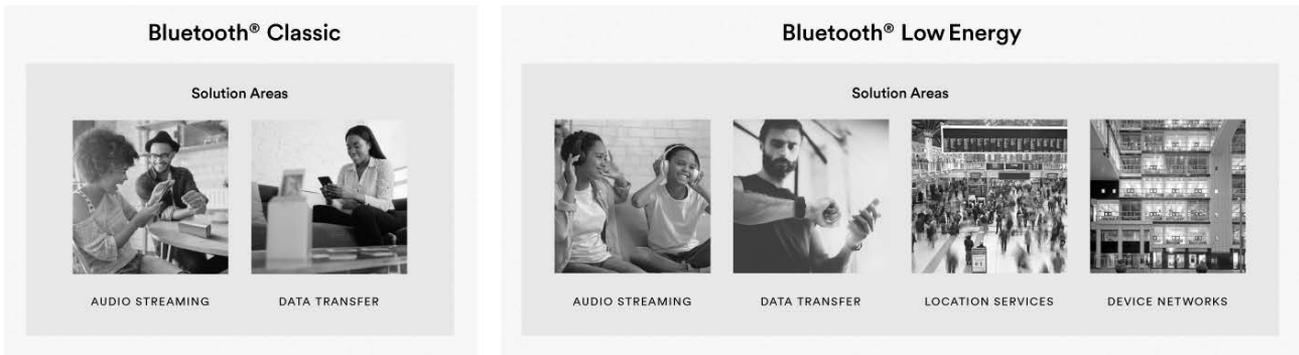


Figure 2. (a) Bluetooth Classic and (b) Bluetooth LE applications.²

Bluetooth Specifications

As shown in figures 1 and 2:

- ▶ *Bluetooth Classic*: represents the earliest versions of Bluetooth with higher data rate capabilities targeted for streaming, high bandwidth file transfers, and headsets. It has 79 RF channels and can be discovered on 32 channels.
- ▶ *Bluetooth Low Energy*: targets low power application with infrequent data transfers such as sensors and other low bandwidth transfers. It has 40 RF channels and can be discovered on three channels.

BLE Applications Overview

A typical BLE application consists of two devices: a peripheral and a central. Before establishing a connection, a peripheral broadcasts its presence via a process called BLE advertising. The central scans for available peripherals. Once the central finds the desired peripheral, a connection is established between the two. The applications in each device can then communicate with each other by propagating the data through the different layers of the BLE stack. See Figure 3.

A smartphone, for example, can act as the central device, while a fitness tracker can serve as the peripheral device. The fitness tracker, acting as a server, collects data such as heart rate, blood pressure, ECG, steps, and even sleep patterns. It advertises its presence to nearby devices, including the smartphone that acts as the client. The smartphone retrieves this data from the fitness tracker and displays it on an app that is easily understood by the user. This is just one example of the many applications that can be implemented using BLE, which will be discussed in the latter part of the article.

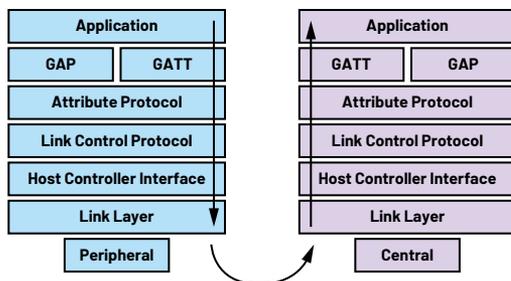


Figure 3. Layers of peripheral and central devices.

BLE Stack Architecture

The BLE stack architecture, as shown in Figure 4, is the structured software framework that enables communication between BLE devices. It defines the layers and protocols necessary for establishing, maintaining, and terminating Bluetooth connections, as well as facilitating data exchange between devices.

The BLE stack architecture is typically divided into three main layers: the application layer, the host layer, and the controller layer. The application layer is the topmost layer of the stack. It is where the actual data is utilized and processed by the applications running on the BLE devices. The host layer sits between the application and controller layers in the stack. It implements all the higher level protocols and profiles required for BLE communication. Additionally, it provides a high level application programming interface (API) that allows applications to interact with the lower layers of the stack. The controller layer is the hardware component of the BLE stack. It is responsible for the transmission and reception of Bluetooth signals. The controller layer handles tasks such as frequency hopping, modulation, and demodulation of signals. Together, these layers work in tandem to enable efficient and reliable communication between BLE devices.

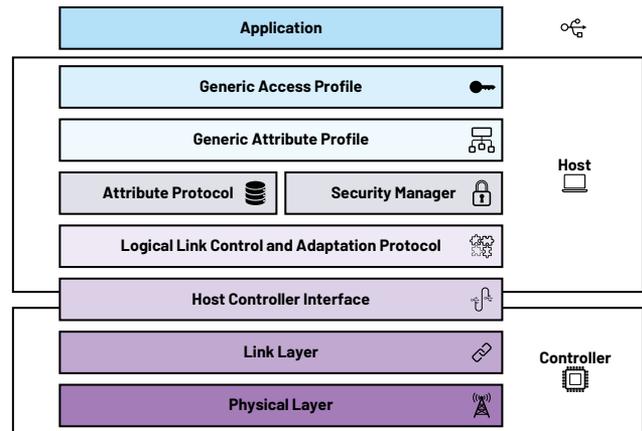


Figure 4. A BLE stack architecture.

Application

The application layer is where the specific functionality of BLE-enabled devices is implemented. It interacts with the lower layers of the stack through the generic attribute profile (GATT), which allows the definition of services, characteristics, and the corresponding data.

In the application layer, the features and behaviors of the devices are designed. This includes defining services and characteristics, specifying how data is exchanged, and implementing the logic for handling events such as connections, disconnections, and data updates. The application layer essentially customizes the BLE stack to meet the specific requirements of the device and its intended use cases.

Host

The host layer, which contains the remaining upper layers of the BLE stack, includes Logical Link Control and Adaptation Protocol (L2CAP), Security Manager Protocol (SMP), Attribute Protocol (ATT), GATT, and Generic Access Profile (GAP). L2CAP acts as an interface between the higher layer protocols and the lower layers. It is responsible for fragmentation and defragmentation of application data and uses the ACL links to transfer packets. L2CAP uses channel identifiers (CIDs) and channel multiplexing to properly locate the endpoints on the device. See Figure 5.

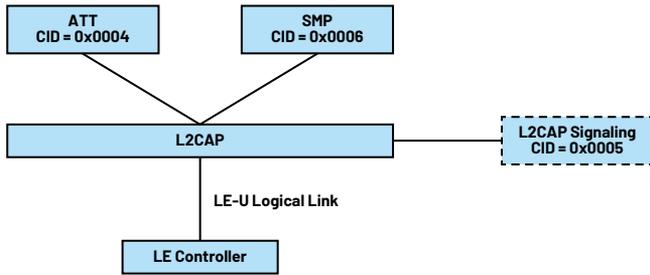


Figure 5. L2CAP packets.

L2CAP Signaling

In the BLE stack, commands are exchanged between devices in the form of requests and responses. Here are some key points regarding commands:

- ▶ Commands are sent in the form of requests and responses.
- ▶ One command can be sent per protocol data unit (PDU).
- ▶ PDUs that contain L2CAP signaling messages are known as C-frames (control frames), while data frames are categorized into B-frames (basic information frames) and LE-frames (low energy information frames). See Figure 6.

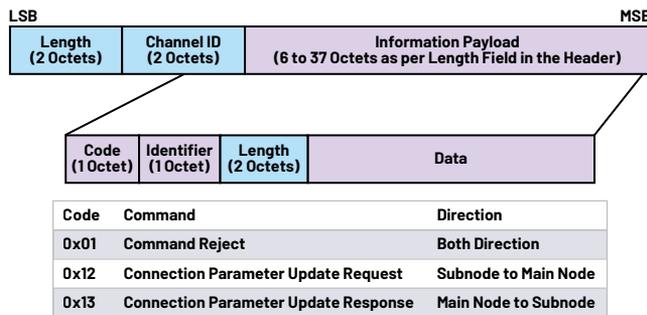


Figure 6. An L2CAP frame format.

- ▶ Command Reject
 - Response sent when the command code was not identified or the length of the command was incorrect
 - Possible reasons
 - Command not understood
 - Signaling maximum transmission unit (MTU) exceeded
 - Invalid CID in request

▶ Connection Parameter Update Request

- Sent by the LE node to the LE main to request a set of new connection parameters
- Connection parameters
 - Interval min
 - Interval max
 - Node latency
 - Timeout multiplier

▶ Connection Parameter Update Response

- Sent by the LE main to the LE node in response to the connection parameter update request

SMP defines the procedures for pairing, authentication, and encryption between BLE devices. SMP commands use the L2CAP services to carry out these procedures. The SMP command packet consists of a code field and data field. The code field identifies the type of the command, while the data field's length and format depend on the command type. All SMP procedures implement a 30-second timeout, which is used to tell if a procedure has failed. See Table 1.

Table 1. SMP Command Codes

Code	Description	Phase
0x00	Reserved	—
0x01	Pairing request	Phase 1
0x02	Pairing response	Phase 1
0x03	Pairing confirm	Phase 2
0x04	Pairing random	Phase 2
0x05	Pairing failed	Phase 2
0x06	Encryption information	Phase 3
0x07	Master identification	Phase 3
0x08	Identify information	Phase 3
0x09	Identity address information	Phase 3
0x0A	Signing information	Phase 3
0x0B	Security request	Phase 1
0x0C to 0x0FF	Reserved	—

ATT defines the rules for accessing attributes or data on a device. It enables the discovery, reading, and writing of attributes on a remote device. ATT follows a client-server model. The server exposes a set of attributes while the client can discover, read, and write on those attributes. An attribute structure in ATT consists of a handle, type, value, and permissions. An attribute handle is a unique nonzero value assigned to each attribute on the server. The attribute type specifies what the attribute represents and is identified by a Universally Unique Identifier (UUID). The UUID can be 16 bits as assigned by the Bluetooth Special Interest Group (SIG) or a custom 128-bit UUID. The attribute value is the actual data value and the attribute permissions determine the level of access that is permitted for the attribute. See Figure 7.

Handle	UUID	Value	Permissions
--------	------	-------	-------------

Figure 7. An attribute structure.

ATT defines six PDU types: request, response, command, confirmation, notification, and indication (Figure 8). A request PDU is sent by the client to the server, requesting a reply. A response PDU is the reply of the server to the client when a reply is requested. A command PDU is sent by the client to the server requiring no reply. An indication PDU is sent by the server to the client, requiring a reply. A confirmation PDU is sent by the client to the server as a reply to an indication. A notification PDU is sent by the server to the client, requiring no reply. These PDU types enable the exchange of information and control between the client and server in the ATT layer of the BLE stack.

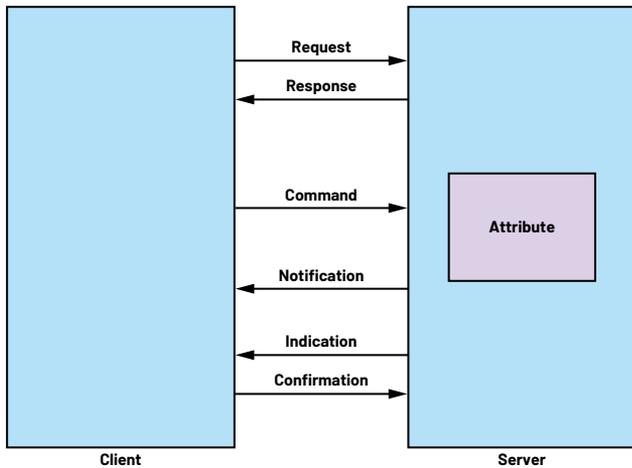


Figure 8. Different attribute PDU types.

An ATT PDU packet is structured with an opcode, attribute parameters, and authentication signature (Figure 9). The opcode field identifies the method/type of PDU, such as a request or response. It also includes a command flag to indicate if the PDU is a command and an authentication signature flag that indicates the usage of an authentication signature in the packet.

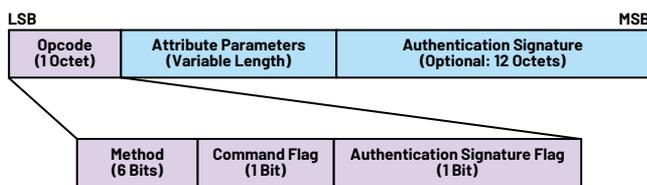


Figure 9. An ATT PDU packet format.

LL	Preamble (1 to 2 Bytes)	Access Address (4 Bytes)	Payload (2 to 257 Bytes)	CRC (3 Bytes)
L2CAP	L2CAP Header		ATT Data (0 to 247 Bytes)	
	Length (2 Bytes)	CID (2 Bytes)		
ATT	ATT Header		ATT Payload (Up to 244 Bytes)	
	Opcode (1 Byte)	Attribute Handle (2 Bytes)		
Data Ch PDU	Header (2 Bytes)	Payload (0 to 251 Bytes)	MIC (Optional) (4 Bytes)	

Figure 10. BLE layers packet format.

Figure 10 summarizes the packet format of the layers in the BLE stack architecture and provides a basic understanding of how data is structured.

Moving up the host layer, the next higher layer is the GATT. GATT defines how data or attributes are formatted, packaged, and exchanged between connected devices. GATT procedures consist of attribute discovery, read, write, notification, and indication. It provides a standard framework for managing data in a BLE device.

Multiple GATT profiles may exist in a BLE device (Figure 11). Standard profiles are defined in the Bluetooth specifications to ensure interoperability between BLE devices from different manufacturers. However, custom profiles can also be implemented based on the specific application requirements. With that, understanding the structure of a GATT profile is vital.

A GATT profile consists of services. A service is a grouping of related attributes defined in the ATT protocol. In GATT, the term characteristic is often used to refer to an attribute but a characteristic may contain a descriptor that is an attribute itself. Characteristics are containers for user data, while descriptors provide descriptions or additional information about the user data.

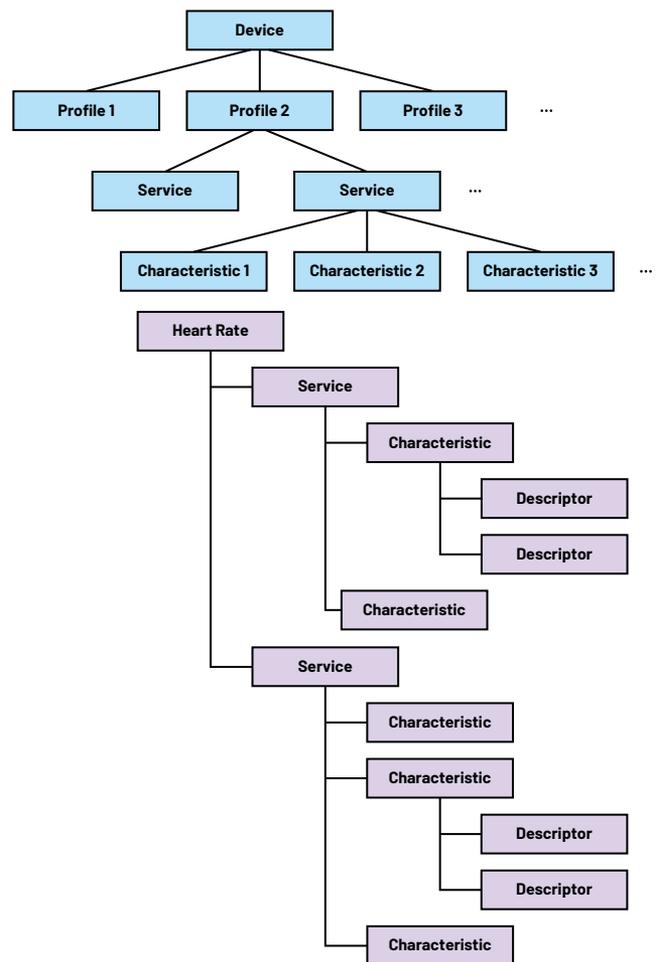


Figure 11. A GATT profile structure.

In the GATT, similar to the ATT, there are two roles: a GATT client and a GATT server. A GATT client is a device that accesses the data available on a remote

GATT server. A GATT server is a device that provides data access to a remote GATT client. The role of a device in GATT is determined by the direction of data access.

The top layer of the host is the GAP. GAP defines how BLE devices access and communicate with each other. It encompasses modes of operation, generic procedures for device discovery, connection establishment, and security. GAP is required to be implemented for all devices that support Bluetooth technology since it provides the standard framework for controlling a BLE device.

GAP provides a different role for a BLE device based on its activity. When no connection is required, a BLE device can act as a broadcaster or an observer. A broadcaster is a device that advertises itself within the vicinity. It mainly utilizes the advertiser role of the link layer to send its advertisements. An observer is the opposite of a broadcaster. It scans the area to receive advertisements from nearby devices utilizing the scanner role of the link layer. Real-life examples of broadcasters are BLE beacons, while examples of observers are BLE hubs that collect data. See Figure 12.

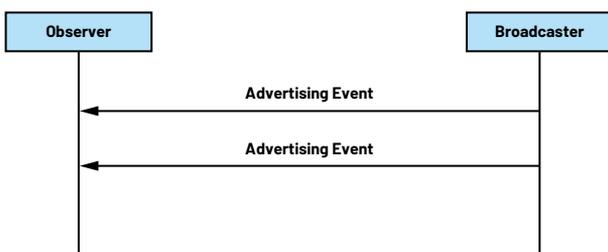


Figure 12. A broadcaster sends advertising packets to an observer.

When a connection can be established, GAP provides two additional roles for a BLE device: peripheral and central. The peripheral device, similar to the broadcaster, advertises itself and awaits a connection request from a remote central device. The central device, on the other hand, acts as an observer, scanning for peripherals and initiating connection requests to the desired peripheral. As mentioned before, examples of a peripheral device are smartwatches, fitness trackers, and home automation sensors, while examples of central devices are smartphones, tablets, and laptops. See Figure 13.

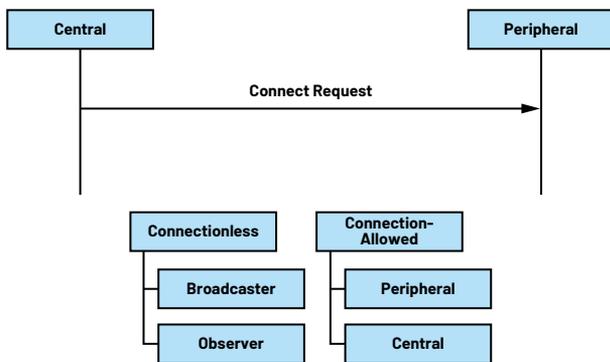


Figure 13. Central/peripheral vs. broadcaster/observer.

As a profile, GAP includes a service that is hosted by the GATT server, as mandated by the Bluetooth specification. The GAP service encompasses various characteristics that provide essential information about the device. These characteristics typically include the device name, device appearance, peripheral preferred connection parameters, central address resolution, and resolvable private address only.

Controller

The controller contains two layers: the link layer and the physical layer. The physical layer is positioned at the bottom of the BLE stack and is responsible for the actual transmission and reception of the signal over the air. Operating in the 2.4 GHz ISM band, the physical layer uses a Gaussian frequency shift keying (GFSK) modulation. This modulation scheme allows for efficient data transmission by shifting the frequency of the carrier signal.

The physical layer consists of 40 channels, each spaced 2 MHz apart from one another:

- ▶ Three advertising channels are used for broadcasting short packets of data to announce the presence of the advertiser and its available services or information.
- ▶ 37 data channels are used once a connection is established between a central and a peripheral device.

Prior to the Bluetooth 5 Core Specification, only three channels were dedicated to BLE advertisements. However, with the introduction of extended advertisements, the remaining 37 channels are used as auxiliary advertising channels. This extension unlocks new features in Bluetooth 5, including the ability to employ different coding schemes of the physical channels or PHYs. See Figure 14.

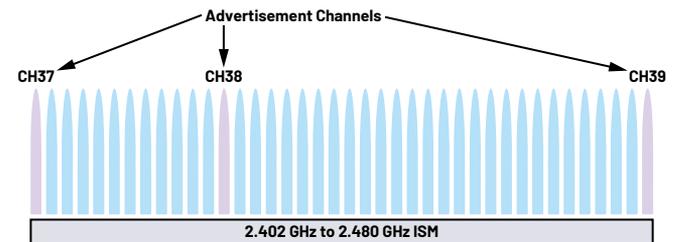


Figure 14. BLE channels.

With the introduction in Bluetooth 5, BLE supports different PHYs that offer three modulation schemes and four data rates (Table 2). The default PHY, known as LE 1M, operates at a modulation scheme of 1 Msymbols per second (Msyms) and achieves a data rate of 1 megabits per second (Mbps). This provides up to 100 meters of wireless range. Another PHY option is LE 2M, which supports twice as much data rate of 2 Mbps using a 2 Msymbols modulation scheme. The third PHY, called LE Coded, offers two data rates: 125 kilobits per second (kbps) and 500 kbps. LE Coded employs a 1 Msymbols modulation scheme, similar to LE 1M. However, the main difference is the addition of coding schemes. To reach a data rate of 125 kbps, a single bit is encoded using eight symbols, while for 500 kbps, a single bit is represented by two symbols. This coding scheme allows the LE Coded PHY to be used in long-range applications, reaching distances of up to 1000 meters in free space.

Table 2. Different PHYs in BLE

PHY	Modulation Scheme	Coding Scheme		Data Rate
		Access Header	Payload	
LE 1M	1 Msymbols modulation	Uncoded	Uncoded	1 Mbps
LE 2M	Pairing request	Uncoded	Uncoded	2 Mbps
LE Coded	Pairing response	S = 8	S = 8 S = 2	125 kbps 500 kbps

Sitting above the physical layer is the link layer, which is responsible for scanning, advertising, creating, and maintaining links or connections between devices. The link layer also manages frequency selection for data transmission, utilizing frequency-hopping spread spectrum to mitigate interference. The link layer has different states: standby, advertising, scanning, initiating, and connected (see Figure 15).

- ▶ The link layer in idle is in a standby state where there are no packets received or transmitted.
- ▶ In the advertising state, the link layer (acting as an advertiser) transmits advertising packets while listening to devices that request additional information.
- ▶ In the scanning state, the link layer (acting as a scanner) listens for advertisers and may request additional information from them.
- ▶ In initiating, the link layer (acting as an initiator) listens to the packets from the advertiser and responds to those packets by initiating a connection.
- ▶ The connected state is when the link layer is connected to the link layer of another BLE device.

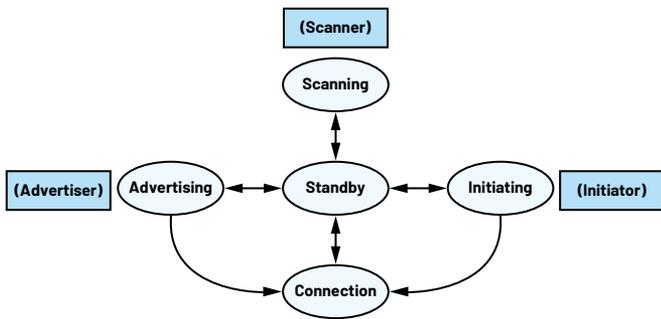


Figure 15. Link layer state transitions.

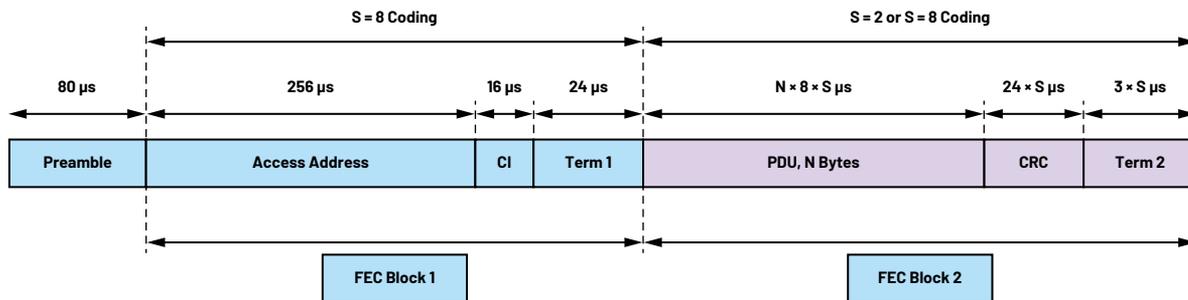


Figure 17. A BLE packet for a coded PHY.

In addition to different states, the link layer also defines events: advertising events and connection events. Advertising events involve the transmission of packets using the advertising channels, while connection events involve the transmission of packets during the connected state through the data channels.

The link layer also defines the packet format of the BLE packet that will be transmitted by the physical layer. The packet format may be categorized into two types: those used for coded or those used for uncoded PHY.



Figure 16. A BLE packet for an uncoded PHY.

A BLE packet for uncoded PHY as shown in Figure 16 starts with a preamble followed by an access address, a PDU, and a cyclic redundancy check (CRC).

The BLE packet for a coded PHY shown in Figure 17 consists of a preamble, forward error correction (FEC) block 1, and FEC block 2.

- ▶ The *preamble* is an alternating sequence of 1s and 0s used for frequency synchronization. It has a length of 1 byte for LE 1M or 2 bytes for LE 2M.
- ▶ The *access address* is used as a correlation code by devices tuned to the physical channel and has a length of 4 bytes. For physical advertising channels, the access address has a fixed value of 0x8E89BED6.
- ▶ The *PDU* contains the payload from the upper layers of the BLE stack. It can be an advertising PDU or a data PDU, which may contain important information from sensors and other devices necessary for communication. More information will be discussed in the next section.
- ▶ *CRC* is used for error checking.

- ▶ The *constant tone extension* (CTE) consists of a constantly modulated series of unwhitened 1s and is generally optional. CTE is important in the BLE feature called direction finding.
- ▶ *FEC block 1* contains the access address, coding indicator (CI), and block terminator (TERM1). The CI indicates the coding scheme used in FEC block 2, while TERM1 is a 3-bit block terminator.
- ▶ *FEC block 2* contains the PDU, CRC, and TERM2, which are coded as indicated in the CI field in FEC block 1.

The BLE packet PDU used in both uncoded and coded PHYs is categorized into two types: advertising channel PDU and data channel PDU. See figures 18 through 21.



Figure 18. An advertising physical channel PDU.



Figure 19. An advertising physical channel PDU header.

The advertising channel PDU shown in Figure 19 is used for advertising events. It is composed of a 2-byte header and a payload of up to 255 bytes.

The header contains fields for PDU type, RFU (reserved for future use) bit, ChSel, TxAdd, RxAdd, and length. The values of the ChSel, TxAdd, and RxAdd bits depend on the PDU type, while length describes the length in bytes of the payload.



Figure 20. A data physical channel PDU.



Figure 21. A data physical channel PDU header.

The data channel PDU shown in the figure above is used by the connection events. It consists of a 2- or 3-byte header, a payload, and a message integrity check (MIC) that is used in encrypted links.

A data channel PDU header contains different fields: LLID, NESN, SN, MD, CP, length, and CTEInfo.

- ▶ LLID describes the type of the link layer data PDU
- ▶ NESN (next expected sequence) identifies which packet from the peer device is expected next
- ▶ SN (sequence number) identifies the current packet
- ▶ CP (CTEInfo Present) denotes the existence of an additional CTEInfo field
- ▶ Length describes the payload length in bytes
- ▶ CTEInfo describes the type and length of CTE

Host Controller Interface (HCI)

The HCI serves as an intermediary between the host and the controller. It provides a standardized set of commands and events that enable communication between these two layers.

HCI supports multiple types of transport layers, including UART, USB, Secure Digital (SD), and three-wire UART. Each transport layer has its own specifications and requirements. For the purpose of this overview, the focus is on the UART transport layer.

As per the Bluetooth 5.2 specifications, the UART transport layer supports five types of packets: command, event, asynchronous connection-less (ACL) data, synchronous (SCO) data, and isochronous (ISO) data.

- ▶ The command packet is used by the host to send commands to the controller. These commands instruct the controller to perform specific actions or configurations.
- ▶ The event packet is used by the controller to notify the host about events that have occurred. Events can include connection status changes, data reception, or other relevant information.
- ▶ The ACL data packet is used to exchange data between the host and the controller. It allows for the transmission of asynchronous data, such as information from sensors or user input.
- ▶ The SCO data packet is used to exchange synchronous data between the host and the controller. However, it is important to note that SCO data packets are not supported in BLE and are primarily used in Bluetooth Classic for voice or audio transmission.
- ▶ The ISO data packet is the newly added type of packet that allows the use of BLE to transfer time-bounded data between devices. Isochronous data packets are designed for applications that require precise timing, such as audio streaming or real-time control.

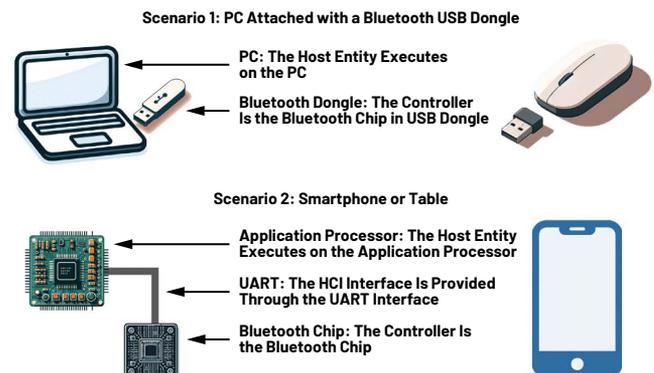


Figure 22. Bluetooth applications.

Why Is It important?

Understanding BLE and its importance is beneficial due to its widespread use in various applications, ranging from consumer products to industrial settings. BLE is a constantly evolving and upgrading protocol, offering limitless possibilities for application development.

Use Cases

BLE is extensively used in numerous industries and impacts daily life in ways that may go unnoticed (Figure 22). Being familiar with BLE allows individuals to understand and leverage its capabilities in various domains. Here are some of the areas in which BLE contributes greatly and revolutionizes the processes as we know it:

BLE in Medicine

BLE plays a crucial role in the field of medicine. It enables the use of devices such as glucose meters, pressure monitors, and even implantables like pacemakers

that require very low power consumption. These devices can collect data and transmit real-time reports to patients and healthcare providers. BLE is also used for patient tracking, locating room or floor numbers, and transmitting information to healthcare responders.

BLE in Location Tracking

BLE is utilized in advanced trackers or smart tags that can be attached to items such as bags, keys, and even pets to track their location. These tags are designed to be very small and energy-efficient, making the low energy capability of BLE essential. Industries also make use of this technology from warehouses that monitor storage locations, grocery stores, and indoor navigation.

BLE in Wearables

Wearables require a small form factor and long battery life, which makes BLE an ideal technology. Devices such as smart watches, fitness bands, and smart eyewear rely on BLE for wireless connectivity and efficient power consumption.

BLE in Audio Streaming

BLE plays a significant role in audio streaming applications. With the introduction of LE Audio, BLE enables low latency audio streaming with improved audio quality perception. LE Audio incorporates the Low Complexity Communications Codec (LC3) that can support low data rates without compromising audio quality. This opens up new possibilities for wireless audio consumption.

BLE in Home Automation

In the realm of home automation, BLE is a fundamental technology for creating smart homes. The IoT is heavily utilized in smart homes, and BLE enables seamless connectivity between various smart devices. There is a wide range of BLE-enabled smart devices available in the market for home automation including key fobs, home beacons, switches, and much more. BLE enables users to control and monitor various aspects of their homes, such as smart lighting, home energy management for efficient usage, smart door locks, wireless speaker systems, domestic robots, and security systems.

Conclusion

In the BLE stack, application data traverses through various layers before reaching the remote application on another device, which also has its own BLE stack. Here is the process:

1. Application layer: the application selects the appropriate attribute that will hold the data to be transmitted.
2. ATT layer: a packet is generated from the ATT layer, containing the information corresponding to the selected attribute on the remote device.
3. L2CAP layer: the packet from the ATT layer passes through the L2CAP layer. L2CAP handles data fragmentation and defragmentation if necessary. It adds an L2CAP header to every L2CAP packet.

4. Link layer: the L2CAP packet is then passed to the link layer, which relays it to the physical layer for over-the-air transmission. The link layer adds a link layer header to the packet, collectively known as a PDU.

5. Physical layer: before transmitting, the physical layer adds the necessary preamble, access address, and CRC to the PDU. The packet is then transmitted over the air.

On the receiving end, the remote BLE device receives the packet and performs the reverse process to extract the data.

When implementing BLE in different applications, it is crucial to choose the right hardware for optimal efficiency and optimization. Analog Devices offers a variety of BLE-enabled microcontrollers suited for different requirements and applications.

The [MAX32665/MAX32666/MAX32667/MAX32668](#) DARWIN family of low power microcontrollers is designed for a wide range of real-world applications. These MCUs support Bluetooth 5 Low Energy radio connectivity, allowing wireless interfacing with multiple devices for IoT applications while still maintaining the lowest active consumption and retention power as possible.³ DARWIN MCUs also have the largest embedded memories in their class, enabling support for larger applications and more stacks. This flexibility and capability open up endless possibilities for designing and preparing for any challenges in the IoT space, supporting the foundation of modern IoT solutions and paving the way for evolution.

References

¹[2023 Bluetooth Market Update](#). Bluetooth, 2023.

²[Bluetooth Technology Overview](#). Bluetooth.

³["Meet DARWIN: A New Breed of Low Power IoT MCUs"](#). Analog Devices, Inc., October 2022.

Bhargava, Madhur. *IoT Projects with Bluetooth Low Energy*. Packt Publishing, Limited, 2017.

["Bluetooth Core Specification Version 5.2 Feature Overview"](#). Bluetooth.

[Core Specification 5.4](#). Bluetooth.

Gupta, Naresh. *Inside Bluetooth Low Energy*. Artech House, 2013.

[Stack Architecture](#). Zephyr Project.

The Bluetooth® word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by Analog Devices, Inc., is under license. Other trademarks and trade names are those of their respective owners.

About the Authors

Erick Reyes is a senior firmware engineer working under the Software and Security Group and supports the Automotive and Energy, Communications, and Aerospace BU. His career in Analog Devices started in 2019 as a product applications engineer working with precision converters and RF wireless transceivers. He finished college with a bachelor's degree in electronics engineering at Mapua University Manila.

Mary Grace Legaspi is a firmware engineer under the Software and Security Group and supporting the Consumer Business Unit. She joined ADI in Cavite, Philippines in September 2018. She graduated with a bachelor's degree in electronics engineering from Tarlac State University and a Master of Management from the University of the Philippines.

Eric Peña is an embedded software architect under the Software and Security Group supporting the Industrial Automation Business Unit at Analog Devices. He joined ADI in Cavite, Philippines in April 2019. He graduated from Adamson University in Manila with a bachelor's degree in computer engineering. Eric previously worked at Technology Enabler Designer as a firmware engineer and also as a systems engineer at Fujitsu Ten Solutions.

Engage with the ADI technology experts in our online support community. Ask your tough design questions, browse FAQs, or join a conversation.



Visit ez.analog.com