Developer Study Guide

Bluetooth® Internet Gateways

First Steps

Release	:	2.0.1
Document Version:		2.0.1
Last updated	:	18th October 2022

Contents

1. Revision History

Version	Date	Author	Changes
1.0.0	18th November 2020	Martin Woolley Bluetooth SIG	Initial version
1.0.1	6th January 2021	Martin Woolley Bluetooth SIG	Added *enabled* configuration property to the Bluetooth firewall.
2.0.0	24th June 2021	Martin Woolley Bluetooth SIG	**Release** Added new module covering gateways for Bluetooth mesh networks. Modularised the study guide to accommodate the two main cases of LE Peripherals and mesh networks. Updated to be based on Python 3 rather than Python 2. **Document** This document is new in this release
2.0.1	18th October 2022	Martin Woolley Bluetooth SIG	Added reference to the Bluetooth Low Energy Primer resource.

2. Introduction

In this module we'll focus on getting orientated within the world of Bluetooth internet gateways, consider questions such as *Buy vs Build* and set about considering different types of gateway, the kinds of requirement we might have and what a logical architecture might look like.

It is assumed that you already have a basic appreciation of Bluetooth Low Energy (LE) terms like *GAP, GATT and ATT*. If any of these terms are new to you then you are advised to read about these topics in the *Bluetooth Low Energy Primer*, which you can download from https://www.bluetooth.com/bluetooth-resources/the-bluetooth-low-energy-primer/ .

It is also assumed that you know what a Bluetooth mesh network is although a more detailed knowledge is not necessary at this stage. The Bluetooth SIG paper *Bluetooth Mesh Networking - An Introduction for Developers* is recommended reading for those who are completely new to the subject. See https://www.bluetooth.com/bluetooth-resources/bluetooth-mesh-networking-an-introduction-for-developers/ .

3. About Gateways

Definitions of the term "gateway" vary, largely according to the context in which the term is used. In general, though, a gateway is a system which allows two otherwise incompatible communications systems to talk to each other via a network or networks.

Gateways are often categorised as *middleware* because they sit in the middle, in between two other systems, providing a service which allows the two systems to work together.

There are many types of gateway. For example, an SMS gateway allows applications to communicate with a mobile network's *short message service centre (SMSC)* to send and receive text messages (i.e. SMS messages) usually using a simple protocol from the world of applications rather than one of the less commonly known telecommunications protocols supported by SMSCs.

A Bluetooth internet gateway allows applications to interact with Bluetooth devices indirectly using a protocol which runs over TCP/IP.

4. Buy vs Build

If your interest in Bluetooth internet gateways stems from needing one for a project, you have a decision to make as to whether to buy a commercial gateway product or to build one yourself. Which of these two paths you take, will depend on your requirements, skills, constraints (e.g. budget) and other factors and it is therefore, a decision which only you can make.

The "Find a Product" page at the Bluetooth SIG web site may assist you in identifying candidate products, should you decide to buy rather than build:

https://launchstudio.bluetooth.com/Listings/Search

5. Standards

Gateways do not fall within the charter of the Bluetooth SIG and therefore, no formal standards relating to gateways have been produced. Some white papers, which discuss this subject and which offer suggestions on a RESTful gateway API which uses HTTP were produced by the Bluetooth SIG Internet Working Group in 2014, however. Their content has informed this study guide to an extent.

https://www.bluetooth.com/bluetooth-resources/internet-gateways/

https://www.bluetooth.com/bluetooth-resources/gatt-rest-api/

https://www.bluetooth.com/bluetooth-resources/gap-rest-api-white-paper/

Note that these papers cover the subject only as it relates to Bluetooth LE devices other than those used in a Blueooth mesh network. There are no formal papers on the subject of Bluetooth Internet Gateways and Bluetooth mesh.

6. About Bluetooth® Internet Gateways

Gateways use one or more approaches to support the interfacing of Bluetooth devices, and applications connected to a TCP/IP network. They each have their own merits, capabilities and constraints and your requirements and priorities will help you appraise each.

6.1 Bluetooth Device Types

When considering gateways, we identify several different types of Bluetooth device, whose technical capabilities require a different approach to be taken in order that they can be supported. They are not mutually exclusive, and a single gateway solution could support all of them with a suitable architecture. The key types stem from the differing Bluetooth capabilities which devices might have.

Table 1 lists some of the possibilities.

Table 1 - Device Capabilities

	Capability	Comment
1	GAP Peripheral and GATT Server	The greater majority of Bluetooth LE devices, other than computers, smartphones and tablets, fall into this category. They act as GAP peripherals, advertising and accepting connections over which attribute protocol (ATT) PDUs from the remote client drive procedures involving GATT services, characteristics and descriptors.
2	GAP Central and GATT Client	Some devices can act as GATT clients. Typically, they also act as GAP centrals, scanning for advertising from GAP peripherals and then connecting to them. Once connected, the GATT client will interact with the attribute table on the remote GATT server using ATT PDUs. Computers, tablets and smartphones usually act as GAP central devices and GATT clients although many can operate in any combination of GAP central/peripheral and GATT client/server.

3	GAP Peripheral and GATT Client	Devices can act as GAP peripherals, advertising, accepting connections from GAP central devices. Having accepted a connection, in contrast to capability (1), these devices then act as a GATT client rather than server, sending ATT PDUs to access the attribute table in the remote GAP central device, which in this scenario is now acting as a GATT server.
4	GAP Broadcaster	Some devices advertise data but cannot accept connections. These are called GAP broadcasters. Bluetooth beacons are often GAP broadcasters.
5	IPv6 Over Bluetooth LE	A relatively small number of devices support Ipv6 over Bluetooth LE. See https://datatracker.ietf.org/doc/rfc7668/ for further details.
6	Bluetooth mesh node	Devices which are members of a Bluetooth mesh network are called nodes. Communicating with mesh nodes involves protocols which are specific to Bluetooth mesh technology but which are built upon the lower layers of the Bluetooth LE stack.

Note that GAP is the Generic Access Profile, the part of the Bluetooth LE stack which is responsible for advertising and accepting connections. GATT is the Generic Attribute Profile, which defines a number of procedures relating to the data attributes held in a device's *attribute table*. Any permutation of GAP peripheral / central and GATT client / server is technically possible, but the most common combinations are the GAP peripheral which acts as GATT server when connected to, and the GAP central which acts as GATT client after connecting to a remote peripheral.

6.2. Gateway Architecture and Bluetooth Device types

For a gateway to provide access to the various types of device listed in Table 1, the gateway architecture must accommodate the particular capabilities of those devices somehow. In most cases, the devices themselves do not need to have any special capabilities which were designed in anticipation of them being accessed via a gateway. This is not always the case, however.

Let's consider ways in which a gateway could provide access to the various device types featured in Table 1.

6.2.1 GAP Peripheral and GATT Server Devices

The gateway must be able to scan for connectable advertising packets and establish connections to discovered devices, perhaps maintain those connections with automatic reconnections and be able to disconnect when required. The gateway acts as a GAP central and GATT client.

6.2.2 GAP Central and GATT Client Devices

The gateway must act as a GAP peripheral, advertising and accepting connections from remote devices. It must also act as a GATT server.

The question then arises as to the GATT service(s) the gateway should support in its capacity as a GATT server and from this, precisely how the GATT client will communicate with the gateway and the IP network on the other side of the gateway.

In this scenario, the gateway may support a custom-selected set of GATT services which compatible client devices may then work with. An alternative approach though is for the gateway to implement the GATT *HTTP proxy service* and for the device to act as a GATT client which uses that service. The HTTP proxy service allows HTTP operations to be formulated by the client by writing to various GATT characteristics within the HTTP proxy service on the GATT server (the gateway in this case) and then to trigger execution of the operation by writing the HTTP control point characteristic.

The specification for the HTTP Proxy Service is available from
https://www.bluetooth.com/specifications/gatt/

6.2.3 GAP Peripheral and GATT Client Devices

The same comments covered in 8.2.2 apply to this case except that given the device is acting as a GAP peripheral, the gateway must act as GAP central, discover and connect to the device. Once connected though, the device acts as GATT client and the gateway as GATT server, either using a custom-selected set of GATT services on the gateway or the HTTP proxy service. It is not particular common for devices to act as GAP peripherals and GATT clients in that combination, but it is entirely possible.

6.2.4 GAP Broadcaster Devices

For the gateway to be able to "collect" data contained within advertising packets from GAP broadcasters, it must act as a GAP observer. The gateway must understand which advertising packet fields should be conveyed over IP connections to connected IP clients.

6.2.5 IPv6 Over Bluetooth LE Devices

The gateway must support IPv6 over Bluetooth LE so that connections can be formed with IPv6 over Bluetooth LE devices and data exchanged.

6.2.6 Bluetooth Mesh Nodes

The gateway must include a Bluetooth mesh stack and have been *provisioned* as a node in the mesh network it is required to make available to internet clients.

6.3. TCP/IP Application Protocols

TCP/IP is in widespread use[1]. In terms of the OSI seven layer model, TCP sits at layer 4 (transport) and IP at layer 3 (network). So, in implementing the TCP/IP part of a Bluetooth internet gateway, we must also choose a suitable protocol to run over TCP/IP, probably at layer 7 (application).

There are a great many possibilities. HTTP is very commonly used and has vast support in both server software and client software i.e. the web browser which computers and mobile devices are almost guaranteed to have. But there's also an argument for using a message-oriented protocol such as MQTT. HTTP is a request/response, transaction-oriented protocol whereas as stated, MQTT is message-oriented. One may fit your gateway requirements better than the other.

[1] The art of understatement 😊

7. Project - Building a Bluetooth® Internet Gateway

In this section, we'll spend some time going through a process which mirrors a typical development project, with the goal that we will ultimately have created a working Bluetooth internet gateway.

You'll be given the opportunity at each project stage, to do some work. Sometimes the work will require you to reflect on a set of issues and document your thoughts, and nothing else. Sometimes you'll do some hands-on technical work. The idea is that the work you do gets you close to the issues and helps instill a good appreciation of those issues in your mind.

7.1 Requirements

The first step in our project will be to determine the requirements for a gateway whose architecture we will next determine and which you will ultimately create.

7.1.1 Task

Think about the requirements which you believe to be important for a Bluetooth internet gateway. Jot them down in a tabular format like the one shown in Table 2 here. There's no need to print anything, just note down your ideas in a text editor or if necessary, on a piece of paper.

Table 2 - Your gateway requirements

Ref	Short Name	Requirement

When you've exhausted your ideas, turn to the next page to see the list we shall be working from in the next project stages.

7.1.2 Selected Requirements

For our project work, we shall focus on supporting both the most common type of Bluetooth LE device, specifically those that act as GAP peripherals and GATT servers and Bluetooth mesh networks. Table 3 contains the list of requirements we shall be using to inform our subsequent work.

Ref	Short Name	Requirement
1	Network interfaces	The gateway shall provide remote access via the internet or other TCP/IP network, to Bluetooth LE devices which are in Bluetooth radio range.
2	Bluetooth device types	The gateway shall support Bluetooth devices which support the GAP peripheral and GATT server roles. The gateway shall suport Bluetooth mesh networks.
3	Protocol and API choice	The gateway shall allow applications to use an API over a simple and commonly supported protocol.
4	API - device discovery	The gateway API shall allow the discovery of in-range Bluetooth LE GAP peripheral devices to be initiated and shall provide the results of the discovery process to the initiating TCP/IP client.
5	API - connect to device	The gateway API shall allow a TCP/IP client to instruct the gateway to establish a Bluetooth connection with a specified Bluetooth LE device.
6	API - disconnect from device	The gateway API shall allow a TCP/IP client to instruct the gateway to disconnect from a specified Bluetooth LE device.
7	API - service discovery	The gateway API shall allow a TCP/IP client to instruct the gateway to discover the GATT services, characteristics and descriptors available in the GATT server of a connected Bluetooth LE device and to return the results in a suitable format.
8	API - read characteristic	The gateway API shall allow a TCP/IP client to instruct the gateway to read and return the value of a specified GATT characteristic within a connected device.
9	API - write characteristic	The gateway API shall allow a TCP/IP client to instruct the gateway to write a new value to a specified GATT characteristic within a connected device.
10	API - enable/disable notifications and indications	The gateway API shall allow a TCP/IP client to instruct the gateway to enable notifications or indications relating to a specified GATT characteristic within a connected device.
11	API - deliver notifications and indications	The gateway API shall deliver characteristic notifications and indications to TCP/IP clients that have enabled them.

12	Concurrency	The gateway shall support access to Bluetooth devices via multiple, concurrent users.
13	Device sharing	It shall be possible for multiple gateway users to access the same Bluetooth device concurrently
14	Security	The gateway shall be secure
15	Scalability	The gateway shall be scalable
16	Webcam	Optional - support webcam streaming so that selected devices can be viewed remotely whilst controlling them via the gateway
16	API - messages from the TCP/IP client to the mesh network	The gateway API shall allow a TCP/IP client to request that the gateway send a mesh message of a specified type and wityh specified parameter values into the associated Bluetooth mesh network.
17	API - messages from the mesh network to the TCP/IP client	The gateway API shall allow a TCP/IP client to subscribe to specified mesh addresses and receive messages published within the mesh network to those addresses.

7.1.3 Appraisal

Review Table 3 and think about the stated requirements. We could have presented a much longer or more granular set of requirements, but this is good enough to allow us to proceed, remembering that this is an educational resource not a formal project document.

Requirement 13, *Device Sharing* may be controversial to some and it has implications for the technical architecture. Perhaps it would make more sense to restrict access to each a Bluetooth device to one gateway user at a time. Or perhaps multiple users could access a single device for read-only operations but only one user at a time is granted write access to it? There are various ways of approaching this topic and requirements will vary.

Requirements 14 and 15 are rather vague. What do we mean and what exactly are our security and scalability requirements? We'll revisit these questions in sections 11 and 12, respectively.

Requirements 16 and 17 are specifically about Bluetooth mesh. Bluetooth LE with GAP and GATT is technically very different to Bluetooth mesh. Is it possible to create a single gateway that supports both technologies? If it is technically possible, is it a good idea or is there an argument for dedicating distinct gateways to servicing each of the two Bluetooth technologies?

Additional requirements may materialize as we develop the architecture of the gateway and start to better understand capabilities, limitations and implications.

7.2. Architecture

We will now turn our attention to the question of how we might meet the requirements that have been identified, in terms of the overall solution architecture. We'll make the distinction between *logical architecture* and *physical architecture*. Logical architectures identify the overall system structure, the primary components it contains and their relationship with other system components, but without being specific about the technologies involved. A physical architecture maps specific technologies and implementation options onto that logical architecture and at this point, should illustrate quite clearly how our gateway will work and what we need to develop, assemble and integrate to build it.

7.2.1 Logical Architecture

7.2.1.1 Task

Review the requirements in Table 3 again, but this time think about the overall architecture of the solution. Think in terms of a *logical architecture* first, focusing on identifying the required structure and building blocks, without being specific about the particular technologies which might meet the need. Don't worry about requirements 14 and 15, covering security and scalability at this stage. These requirements are too nebulous as currently stated.

Draw a diagram which illustrates your ideas for the logical architecture and when satisfied, turn the page to continue.

7.2.1.2 Selected Logical Architecture

The diagram below represents the logical architecture we shall base our physical architecture on and which we shall eventually implement. Study it and the explanatory text.

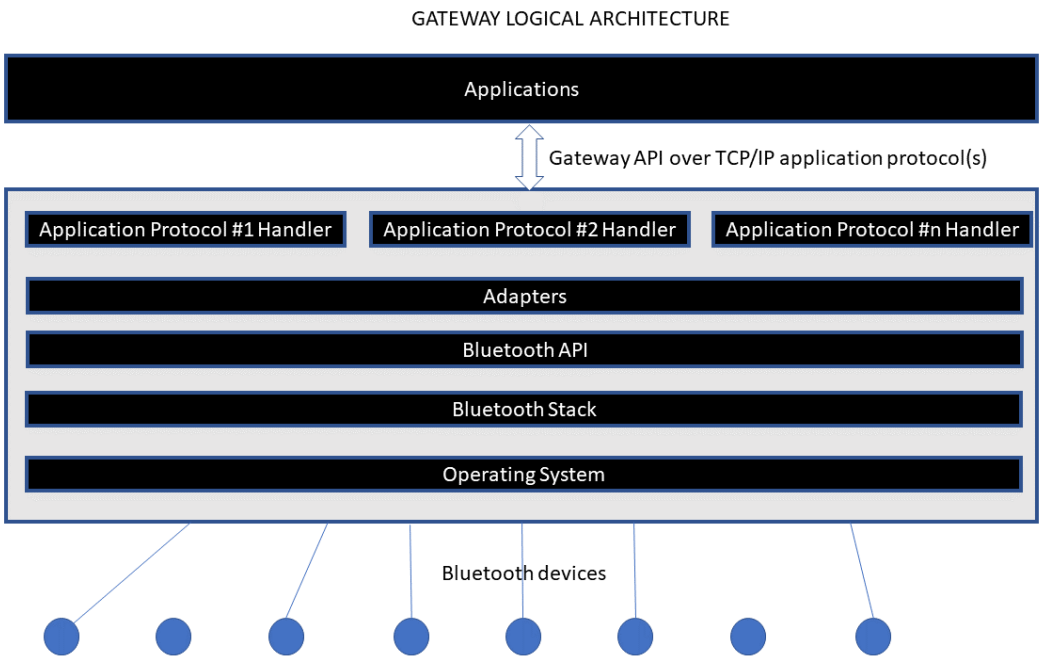


Figure 1 - Gateway Logical Architecture

The components of the logical architecture in Figure 1 are explained as follows:

Applications

Applications are not part of the gateway. They are users of the gateway. Examples include mobile applications and web sites.

Gateway API

The Gateway API provides a means by which applications can send requests to the gateway and receive responses or ad hoc messages from it.

Application Protocol(s)

The gateway API must be conveyed over a protocol which itself runs over TCP/IP. In fact, we may find that our requirements cannot be met by a single protocol and the logical architecture anticipates and allows for this possibility. Or we may simply want to offer a choice of protocols because some application types are better served by one protocol rather than another.

Application Protocol Handlers

The gateway must include a component which handles each supported TCP/IP application protocol. Protocol handlers might be servers, in a client-server protocol or message brokers or relays in a message-oriented protocol. This will become clear when we select the protocol(s) to be used, in the physical architecture stage.

Adapters

This set of components are responsible for converting requests made using the TCP/IP application protocol into Bluetooth communication made via the Bluetooth API, and for converting data received from the Bluetooth API layer into gateway API responses sent to applications over TCP/IP using the application protocol. They sit in between those two worlds and adapt one to the other.

Bluetooth API

Whatever platform we run our gateway software on, must include a suitable Bluetooth stack. To use it, our gateway code will require an API through which to invoke Bluetooth procedures.

Bluetooth Stack

A key component will be a Bluetooth stack, consisting of both software and hardware. This is somewhat close to being a physical architecture component, but we'll allow ourselves to include it in the logical architecture since it adds clarity as to how we intend the system to work.

Operating System

Unless we choose to implement on a microcontroller which does not have a true operating system (O/S), we'll need one and in the physical architecture stage, we'll need to select our gateway O/S.

Bluetooth Devices

These are devices which the gateway will exchange data with. Given our requirements, these are devices which support the GAP peripheral and GATT server roles or that are nodes in a Bluetooth mesh network.

7.2.2 Physical Architecture

Now we'll make some decisions about the specific technologies and components to be involved and how they will work together to provide the required gateway functionality.

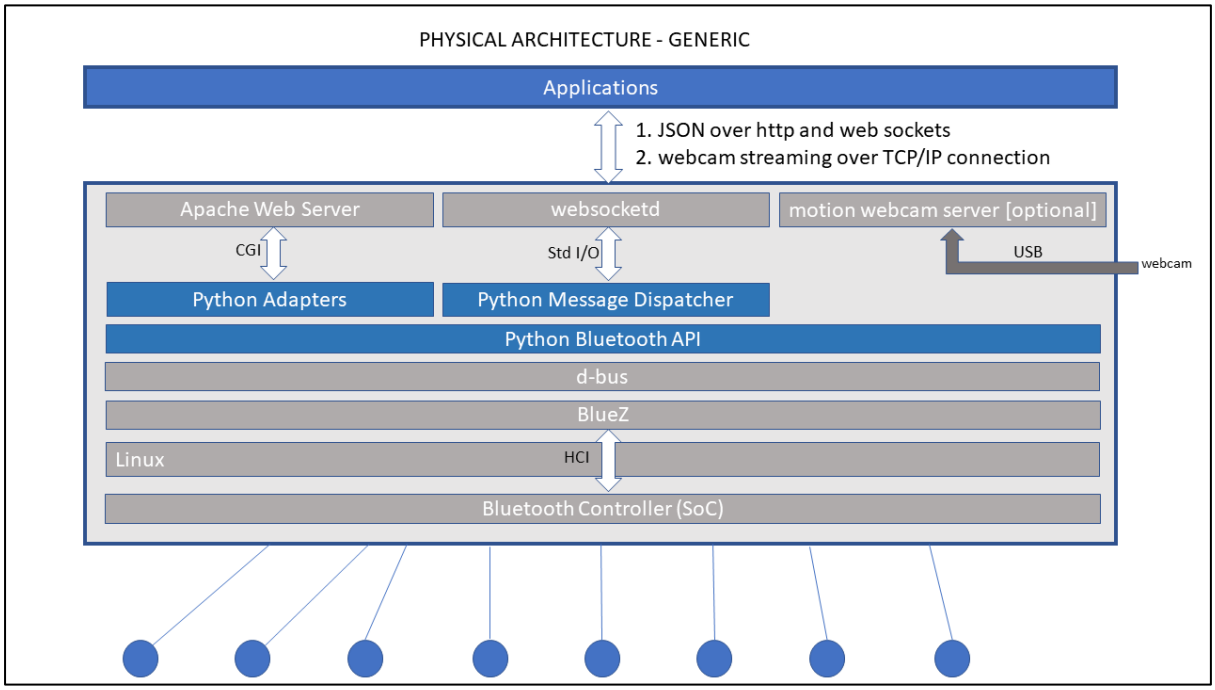
7.2.2.1 Task

How would you proceed? What protocol(s) would you make available to applications? What would your API look like? What platform would the gateway run on? What would you develop yourself and what readily available 3rd party components would you use? What programming language would you use for those areas you decide you want to develop yourself?

Try to answer these questions and more. Sketch a physical architecture that offers specific names for each of the components in our logical architecture. Then turn the page.

7.2.2.2 Selected Physical Architecture(s)

The diagram below represents the physical architecture which we shall implement in some form. Study it and the explanatory text.



Blue rectangles are components that will be developed from scratch or completed, starting from a partial implementation. Others are ready-made components, we shall install, configure and integrate. By selecting largely off the shelf components, we've saved ourselves a lot of work.

Applications

Applications fall outside of scope of our gateway architecture. They can be written in any language, provided the gateway API can be used and the gateway's TCP/IP application protocols are supported.

Gateway API and Supporting Protocols

The gateway shall support HTTP and web sockets for API communication.

The API shall use the HTTP GET for making requests which do not change the state of remote objects (e.g. characteristic values or device state), with an HTTP query string used to pass associated parameters.

The API shall use the HTTP PUT for making requests which do change the state of remote objects (e.g. characteristic values or device state), with JSON objects in the HTTP request body used to encode and transport associated parameters.

Responses and ad hoc messages will be delivered as JSON objects in all cases.

In the case of Bluetooth notifications and indications and Bluetooth mesh messages, web sockets shall be used since this use case requires bi-directional and server-initiated communication.

Enabling and disabling notifications and indications shall be achieved by writing an API JSON control object to the web socket and characteristic value notifications and indications will be delivered to the application using the same web socket connection, also expressed as a JSON object.

Subscribing to a Bluetooth mesh group address will be performed by sending a JSON object from client to server and messages from the mesh network which the gateway receives and which have a destination address to which one or more clients have subscribed will be delivered to those clients over a web socket.

Streaming webcam video data over a TCP/IP connection may be supported.

Application Protocol Handlers

This layer of the architecture consists of two mandatory components and one optional component. Given HTTP and web sockets have been chosen as the TCP/IP interfaces to our gateway, we need one handler for each.

The Apache web server is an open source, free of charge, mature, reliable and easy to use HTTP server daemon. It has an extensive collection of plugins called modules, with which its functionality can be extended, and custom code to be executed in response to HTTP requests can be written in a multitude of languages. For these reasons, it has been selected as the component that will act as an HTTP handler for our gateway.

Choosing the Apache HTTP server has implications for other aspects of the physical architecture, including the platform the gateway will run on. Apache runs on many platforms, another of its advantages. But if you had envisaged implementing your gateway on a constrained device like a microcontroller, it will not be suitable. It's perfect for this education resource, however.

Web sockets require a server to service connections and so we'll be using a web socket server implementation called websocketd. It's simple, lightweight and perfect for our needs.

We'll support plugging a webcam into our gateway so we can point it at a device or too, just because we can, which is often a good enough reason to do something! To that end, our architecture will include the open source webcam server software, *motion*.

Adapters and the Message Dispatcher

Adapters shall be written in the popular programming language Python. The Apache web server will invoke a particular python script, according to the URL in HTTP requests and by writing to standard output, the python scripts will be able to return a JSON results object back over the connection to the requesting HTTP client.

There are many ways to integrate custom code with web servers. The simplest, supported by most web servers, is called the Common Gateway Interface (CGI). It's not designed for high performance or efficiency. There are alternative approaches which fare better on those respects. But neither of these issues is a priority for us. The expectation is that a small number of concurrent users will make a small number of requests at a time.

A special adapter shall be responsible for sending ad hoc Bluetooth data over a websocket connection to a connected client and is called a Message Dispatcher in the architecture diagram. A dispatcher will handle Bluetooth notifications and indications from connected Bluetooth LE devices and messages from nodes in the mesh network.

Bluetooth API

A Python Bluetooth API which the adapter scripts can use has been written specifically for this Bluetooth SIG developer study guide. It provides support for all the required Bluetooth GAP and GATT procedures that you'll need for the project in module 03, *LE Peripherals*. For Bluetooth mesh, it

is incomplete however. If you opt to follow module 04 *Mesh* Networks and implement a gateway for mesh networks, you'll complete this code in that hands-on project.

Bluetooth Stack

We'll be using Linux and therefore will use its standard BlueZ stack which is included in Linux distributions. BlueZ supports both Bluetooth LE with GAP and GATT and Bluetooth mesh.

The hardware platform must have a Bluetooth LE adapter for BlueZ to work.

Operating System

Our gateway will run on the Linux operating system.

Hardware Platform

You can run the gateway on any computer which supports Linux and has a Bluetooth adapter. We used a Raspberry Pi 4 mostly but preferred a Raspberry Pi 400 for better performance when developing the mesh gateway.

7.3. Gateway Development

You've probably heard the expression "the devil's in the detail". It applies here. BlueZ can be used for Bluetooth LE with GAP and GATT **or** with Bluetooth mesh but not both at the same time. So to support both of these Bluetooth technologies, you will need to physically separate gateways and the physical architecture of each will vary slightly. The code you write to enable communication with the two types of device will be significantly different too. So the study guide project splits at this point. If you'd like to continue with building a gateway for Bluetooth LE GAP and GATT Peripherals, you should continue with module 03. If Bluetooth mesh is your topic of interest, proceed now to study module 04. Of course you can study both if you want to.