Part I

BLE

CONFIDENTIAL

BLE Protocol

Introduction

BLE or Bluetooth Low Energy is the energy efficient replacement of Bluetooth Classic for short range RF connectivity. It is prominently used for those applications where power consumption is critical and small amount of data is transferred infrequently.It was introduced in 2010 and later in 2013, iPhone 4s became the first smartphone supporting BLE. It is also called Bluetooth Smart or BTLE. It has become a common protocol in IoT as well.

CONFIDENTIAL

Features of BLE

Some common features of BLE are:

- It works in the frequency spectrum of 2.4-2.4835 GHz. This frequency spectrum is divided into 40 "2 MHz" wide channels.
- The maximum rate of data transfer supported by it (Bluetooth version 5) is 2 Mbps.
- The range varies remarkably,depending on various factors and typically it is 10-30 meters.
- Power consumption also vary depending on the chipset used, application etc.
- Security is optional in BLE.
- It is primarily used for low bandwidth data transfer applications, as high bandwidth applications will eventually result in more power consumption.

Advantages of BLE

- Ultra-low power Consumption
- Low cost of modules and chipsets (nRF52 series)
- Highly resistant to interference at its 2.4 GHz license free band (Frequency Hopping)
- Its compatibility with most of the smartphones in the market.

Applications of BLE

The popularity of BLE is increasing exponentially and now almost every smartphone in the market incorporate BLE. The main applications include smart tags, fitness trackers (fitbit), Smart home, Wireless charging, Communicators etc.

labelsection:Applications of BLE

Architecture of BLE

It is divided into three categories: **??**

NTIAL

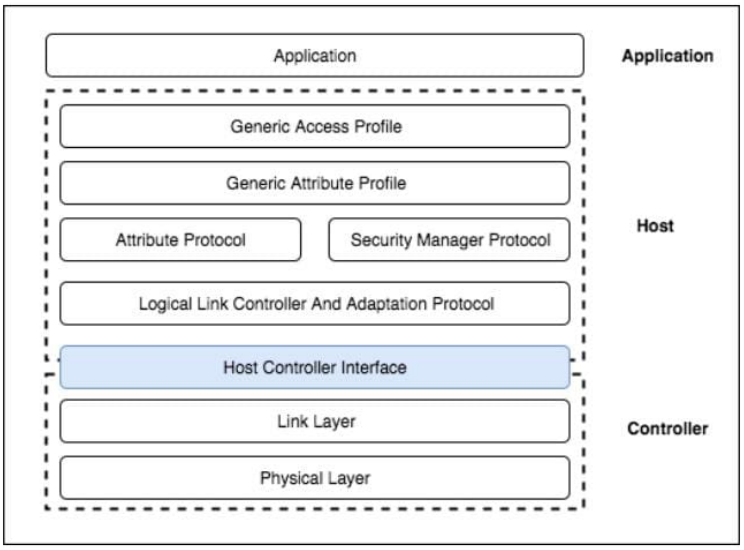

Figure 1: BLE Architecture

- **Application**
- **Host**
- **Controller**

Application

This layer contains the application logic,user interface, and the complete application architecture. It is use case dependent since it depends on the specific application.

Host

The host layer is further divided into various layers which are:

Generic Access Profile(GAP): It provides the framework about how the BLE devices will interact with each other. It incorporates about the roles & modes in which BLE devices would work, advertisement state, connection parameters and security. [1] [2] are the most commonly used terms for defining the role of BLE.

Attribute protocol(ATT): It defines how the data (attribute)is structured. Each attribute will have:

- **Attribute handle** This is a 16-bit value that the server assigns to each of its attributes — think of it as an address. This value is used by the client to reference a specific attribute and is guaranteed by the server to uniquely identify the attribute during the life of the connection between two devices. The range of handles is 0x0001-0xFFFF, where the value of 0x0000 is reserved.
- **Attribute Type or UUID** Universally Unique Identifier: It is a 16-bits or 128-bits unique number used to identify services, characteristics and descriptors.For example, the UUID for a SIG-adopted temperature measurement value is 0x2A1C SIG-adopted attribute types (UUIDs) share all but 16 bits of a special 128-bit base UUID:00000000-0000-1000-8000-00805F9B34FB. A custom UUID can be any 128-bit number that does not use the SIG-adopted base UUID. For example, a developer can define their own attribute type (UUID) for a temperature reading as: F5A1287E-227D-4C9E-AD2C-11D0FD6ED640.
- **Attribute Value** It refers to the variable length.
- **Attribute Permission** It determine whether an attribute can be read or written to, whether it can be notified or indicated, and what security levels are required for each of these operations. These permissions are not defined or discovered via the Attribute Protocol (ATT) but rather defined at a higher layer (GATT layer or Application layer).

Generic Attribute Profile (GATT): It defines how the attributes or data is packed,sent or formatted from one BLE device to another while interacting with each other. There are two roles that a device can adopt: Server and Client

[1] peripheral device is a device that announces its presence by sending out advertising packets and accepts a connection from another BLE device
[2] A Central is a device that discovers and listens to other BLE devices that are advertising. It is also capable of establishing a connection to multiple BLE peripherals

CONFIDENTIAL

Services It is a group of two or more attributes that satisfy a specific functionality on the server.Similar kind of data or characteristics are bundled together for a particular task.

Characteristics It is the lowest type of attribute. Encapsulated by a related service, these are actually state variables. Each characteristic store a single piece of data and relevant measurement.

Profiles BLE profile describes the behaviour of the Bluetooth Low Energy device and its purpose. A common example of Bluetooth Low Energy Profiles is Blood Pressure Profile.

??

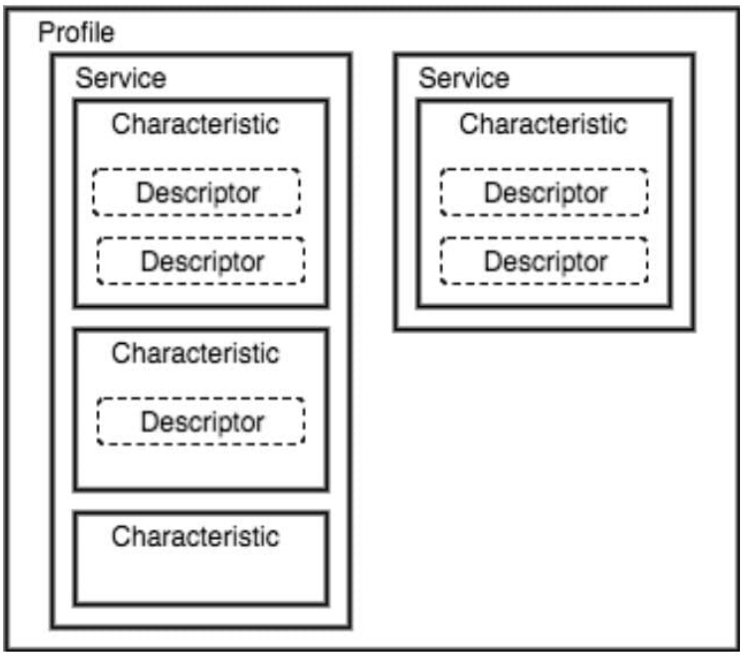

Figure 2: Profile, services and Characteristics

Controller

It is further divided into : Physical layer, Link layer, Direct Test Mode, Host Controller Interface (Controller side).

Physical Layer(PHY) It refers to the radio hardware that is used for communication and modulating/demodulating the data. BLE operates at 2.4GHz spectrum which is segmented into 40 "2 MHz" channels. Out of these, three channels are called Primary Advertising Channels and the remaining 37 are called Secondary Advertising Channels.

Link Layer This layer provides a way to interact with the radio (via HCL layer). It is responsible for managing the state of the radio as well as the timing requirements necessary for completing the BLE specification. It also manages operations like Random number generation, Encryption etc. The BLE device mainly operate in three main states:

- **Advertising state** In the advertising state, a device sends out packets containing useful data for others to receive and process. The packets are sent at a fixed interval defined as the advertising interval.

- **Scanning state** In order for a central to discover a peripheral, the central has to be tuned to the same channel on which the peripheral is advertising at that given point. This is achieved through the scanning state.

- **Connected state** A connection is considered established once the device receives a packet from its peer device. After a connection becomes established, the central becomes known as the master, and the peripheral becomes known as the slave.

Direct Test Mode (DTM) It is needed during manufacturing and certification tests.

Host Controller Interface (HCI) It allows the host layer to communicate with the controller layer. The main function of HCL layer is to relay commands from the host down to the controller and send the events back from controller to the host.

CONFIDENTIAL

GATT Design Guidelines

While GATT is a pretty flexible framework, there are a few general guidelines to follow when designing it and creating the services and characteristics within it. Following are some recommendations:

- It is mandatory to implement the following service and its characteristics:
  - Generic Access Profile (GAP) service.
  - Name and Appearance characteristics within the GAP service.
- Vendor SDKs usually do not require you to explicitly implement this service, but rather they provide APIs for setting the name and appearance. The SDK then handles creating the GAP service and setting the characteristics according to the user-provided values.

- Utilize the Bluetooth SIG-adopted profiles, services, and characteristics in your design whenever possible. They have the following advantages : Reduction in the size of data packets and minimising the development time as many bluetooth chipsets and module vendors usually provide implementations of these profiles, services, and characteristics in their SDKs.
- Avoid having services with too many characteristics. A good separation of services makes it faster to discover certain characteristics and leads to a better GATT design that's modular and user-friendly.

Step 1: Document the different user scenarios and data points In this step, you'll think about your system from a high-level in terms of data elements that need to be exposed on the Server device. In this step, you need to focus on the following:

- Defining the data elements that the server needs to expose to the Client.
- Defining whether each of these elements will be available for: *Read, Write, and Notifications of value changes back to the Client.*
- Eliminate any redundant data elements.
- Grouping the data elements into a meaningful number of groups based on related functionality.

Step 2: Define the services, characteristics, and access permissions Grouping of the characteristics into meaningful groups (services) based on their functionalities and define the access permissions for each of these characteristics.

Step 3: Re-use Bluetooth SIG-adopted services & characteristics Refer to the standard Services and Characteristics and see which ones match the data elements you came up within the design. Though it is not mandatory and you have the freedom to create custom Services and Characteristics.

Step 4: Assign UUIDs to Custom Services and Characteristics For any custom services and characteristics within the GATT, we can use an online tool to generate UUIDs such Online GUID Generator. A common practice is to choose a base UUID for the custom service and then increment the 3rd and 4th Most Significant Bytes (MSB) within the UUID of each included characteristic.

For example, we could choose the UUID:
00000001-1000-2000-3000-111122223333
for a specific service and then
0000000[N]-1000-2000-3000-111122223333, (where N > 1) for each of its characteristics.

CONFIDENTIAL

So, in this case, the service and its characteristics' UUIDs would look something like this:

Service A: 00000001-1000-2000-3000-111122223333
|—- Characteristic 1: 00000002-1000-2000-3000-111122223333
|—- Characteristic 2: 00000003-1000-2000-3000-111122223333
|....
|—- Characteristic n: 0000000[n-1]-1000-2000-3000-111122223333

The only restriction for choosing UUIDs for custom services and characteristics is that they must not collide with the Bluetooth SIG base UUID:

XXXXXXXX-0000-1000-8000-00805F9B34FB

You can find information about how to choose a single UUID here.

Step 5: Implement your GATT using the framework and APIs provided by the BLE solution vendor The solutions provided by the different vendors vary widely, so this step will be specific to the BLE module and development framework that you end up choosing.

GATT Design Example 1

The above concept would be more clear by this example. Consider a home automation system, which consist of multiple devices. The main user scenarios are:

1. Using the remote control to turn on/off the Bluetooth light bulb.
2. Monitoring the changes in the temperature and humidity of the environment sensor.
3. Notification of the battery level of the remote control, Bluetooth light bulb, and environment sensor.

??

CONFIDENTIAL

System Elements

1. **Gateway** It will act as BLE central when communicating with all other devices except the smartphone where it will act as peripheral. We have control over this device and will be designing its GATT. the command for controlling the light bulb will be routed from the remote control through gateway and to the Bluetooth Light bulb.
2. **Remote Control** It will act as peripheral only, and one that we will be designing GATT for.
3. **Environment Sensor** This is an off-the-shelf device over whose GATT we have no control.

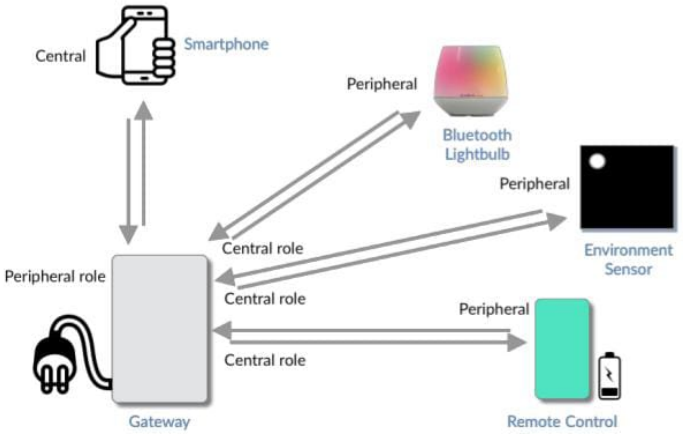

Figure 3: BLE home automation project example

4. **Bluetooth Light bulb** Another off-the-shelf device over whose GATT we have no control.

5. **Smartphone** One more existing device over whose GATT we have control.

GATT Design

The process is described below, step by step:

1. **Documenting the different user scenarios** Even though GATT focuses more on peripheral role, the central can still act as the server in most of the cases for specific data points it needs to expose.

   - **Gateway** The gateway device acts as both central and peripheral. In the peripheral role, the remote control notifies the gateway when specific buttons are pressed. Also some data points within the system need to be referred to cloud server via gateway. These include temperature & humidity reading, Bluetooth light bulb Status (on/off) and individual battery levels for the sensors. For the central role, the gateway needs to read some of the data exposed by devices within the system and get notified of data points exposed by this device.
   - **Remote control** Its main function is to turn on/off the Bluetooth light bulb. It acts in the peripheral role and exposes the following data points: (a) On button press (b) Off button press (c)Battery level

2. **Define the services, Characteristics and Access Permissions** In this step we will group the characteristics into meaningful groups

CONFIDENTIAL

(Services) and define their access permissions for each one of them.

- **Environment Sensor Service**
  - Environment sensor temperature reading characteristic: "Temperature". Access permissions: Read, notify.
  - Environment sensor humidity reading characteristic: "Humidity". Access permissions: Read, notify.
  - Battery level characteristic: "Battery Level". Access permissions: Read, notify.
- **Playbulb service**
  - Light Status Characteristic : "light status" Access Permission : Read, notify
  - Battery level characteristic: "Battery Level". Access permissions: Read, notify.
- **Remote Control Service:** Battery level characteristic: "Battery Level" Access Permission : Read, notify
- **GAP service:**
  - Name characteristic: the device name. Access: Read.
  - Appearance characteristic: a description of the device. Access: Read.

3. **Remote Control** We have one GATT server for the remote control. We can define the following services and characteristics:
   - **GAP service (mandatory):**
     - Device name characteristic: "Device Name". Access permissions: Read.
     - Appearance characteristic. Access permissions: Read.
   - **Battery service:** Battery level characteristic: "Battery level" Access permissions: Read, Notify
   - **Button service:**
     - On button characteristic: "On Button Press". Access permissions: Notify.
     - Off button characteristic: "Off Button Press". Access permissions: Notify
4. **Re-use Bluetooth SIG-Adopted Services and Characteristics**
   - **Gateway** The environment sensor,and remote control services are all custom services, since there are no SIG-adopted ones. We have three devices for which we need to expose the battery

CONFIDENTIAL

level, so we can reuse the SIG-adopted battery level characteristic. We will reuse it for each device within each device's service in the gateway GATT. We will also re-use the mandatory GAP service.

- **Remote Control** For this we can reuse both battery service and mandatory GAP service.

5. **Assign UUIDs to Custom Services and Characteristics** For any custom services and characteristics within the GATT, we can use an online tool to generate UUIDs such as the Online GUID Generator as described earlier.

Service	Characteristic	UUID	UUID Type
Environment Sensor		13BB0001-5884-4C5D-B75B-8768DE741149	Custom
	Temperature	13BB0002-5884-4C5D-B75B-8768DE741149	Custom
	Humidity	13BB0003-5884-4C5D-B75B-8768DE741150	Custom
	Battery level	0x2A19	SIG-adopted
Playbulb		19210001-D8A0-49CE-8038-2BE02F099430	Custom
	Light status	19210002-D8A0-49CE-8038-2BE02F099430	Custom
	Battery level	0x2A19	SIG-adopted
Remote Control		B49B0001-37C8-4E16-A8C4-49EA4536F44F	Custom
	Battery level	0x2A19	SIG-adopted
GAP		0x1800	SIG-adopted
	Device Name	0x2A00	SIG-adopted
	Appearance	0x2A01	SIG-adopted

Figure 4: Gateway GATT Design

Service	Characteristic	UUID	UUID Type
Button		E54B0001-67F5-479E-8711-B3B99198CE6C	Custom
	ON button press	E54B0002-67F5-479E-8711-B3B99198CE6C	Custom
	OFF button press	E54B0003-67F5-479E-8711-B3B99198CE6C	Custom
Battery		0x180F	SIG-adopted
	Battery level	0x2A19	SIG-adopted
GAP		0x1800	SIG-adopted
	Device Name	0x2A00	SIG-adopted
	Appearance	0x2A01	SIG-adopted

Figure 5: Remote control GATT Design

6. **Implement the Services and Characteristics Using the Vendor SDK APIs** Each platform, whether it's an embedded or mobile one, has its own APIs for implementing services and characteristics.It depends on the reader to implement for the specific platform they choose for their BLE device or application.

GATT Design Example 2

Let's look at an example of a GATT implementation. For this example, we'll look at an example GATT.xml file that's used by the Silicon Labs Bluetooth Low Energy development framework (BGLib).

```
<?xml version="1.0" encoding="UTF-8" ?>
<configuration>

    <service uuid="1800">
      <description>Generic Access Profile</description>

      <characteristic uuid="2a00">
        <properties read="true" const="true" />
        <value>Bluegiga CR Demo</value>
      </characteristic>

      <characteristic uuid="2a01">
        <properties read="true" const="true" />
        <value type="hex">4142</value>
      </characteristic>
    </service>

    <service uuid="0bd51666-e7cb-469b-8e4d-2742f1ba77cc" advertise="true">
        <description>Cable replacement service</description>

        <characteristic uuid="e7add780-b042-4876-aae1-112855353cc1" id="xgatt_data">
            <description>Data</description>
            <properties write="true" indicate="true" />
            <value variable_length="true" length="20" type="user" />
        </characteristic>
    </service>

</configuration>
```

Figure 6: GATT example 2

CONFIDENTIAL

- There are two services defined:
  - Generic Access Profile (GAP) service with UUID: 0x1800 (SIG-adopted service).
  - Cable Replacement service with UUID: 0bd51666-e7cb-469b-8e4d-2742f1ba77cc (a custom or vendor-specific service).
- Generic Access Profile service includes the following mandatory characteristics:
  - Name with UUID 0x2a00 and value: Bluegiga CR Demo.
  - Appearance with UUID 0x2a01 and value 0x4142.
- The Cable Replacement service has one characteristic named data:
  - The data characteristic has a UUID: e7add780-b042-4876-aae1-112855353cc1
  - It has both writes and indications enabled.

GATT example 3

Let's take a look at the definitions for services and characteristics within the MIDI BLE Specification. There is one service and one characteristic defined:

- GAP service (mandatory): Device name characteristic: "Device Name". Access permissions: Read. Appearance characteristic. Access permissions: Read.
- MIDI Service (UUID: 03B80E5A-EDE8-4B33-A751-6CE34EC4C700) MIDI Data I/O Characteristic (UUID: 7772E5DB-3868-4112-A1A9-F2669D106BF3). It has write, read and notify enabled.

GATT Design Example 4

General system description

The Smart Irrigation System is another application of BLE which is capable of improving the irrigation process by analyzing the moisture of soil and the climate condition. The system consist some off-shelf device, peripherals, gateway and smart phone. **??**

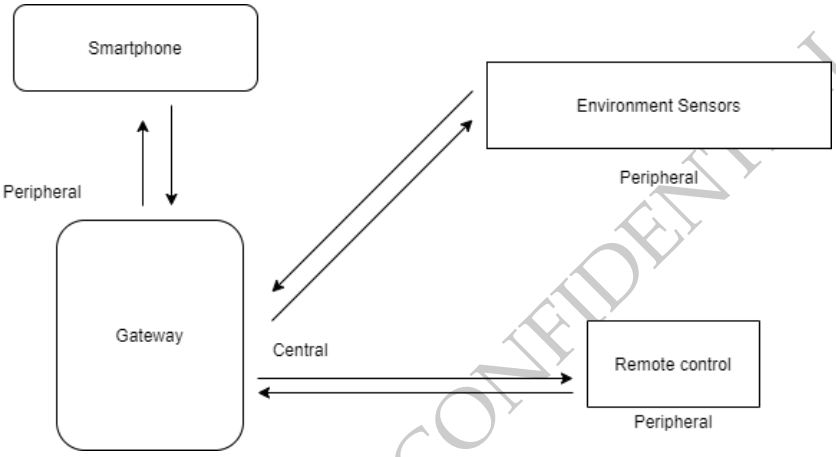

Figure 7: Block diagram of the Irrigation system

CONFIDENTIAL

The main user case scenarios are:

1. The user can use the remote control to turn on/off the power supply of the motor.
2. The user can monitor the changes in the temperature, humidity and moisture using the Environment sensor.
3. The user is notified of the battery levels of the remote control and Environment sensor.

System Elements

1. **Gateway** It will act as BLE central when communicating with all other devices except the smartphone where it will act as peripheral. We have control over this device and will be designing its

GATT. The command for controlling the power supply of the motor will be routed from the remote control through gateway and to the power supply of the motor.

2. **Remote Control** It will act as peripheral only, and one that we will be designing GATT for.

3. **Environment Sensor** This is an off-the-shelf device over whose GATT we have no control.

4. **Power supply of motor** Another off-the-shelf device over whose GATT we have no control.

5. **Smartphone** One more existing device over whose GATT we have control.

GATT Design

The process is described below, step by step:

1. **Documenting the different user scenarios** Even though GATT focuses more on peripheral role, the central can still act as the server in most of the cases for specific data points it needs to expose.

   - **Gateway** The gateway device acts as both central and peripheral. In the peripheral role, the remote control notifies the gateway when specific buttons are pressed. Also some data points within the system need to be referred to cloud server via gateway. These include temperature, moisture and humidity reading, Power supply of motor status (on/off) and individual battery levels for the sensors. For the central role, the gateway needs to read some of the data exposed by devices within the system and get notified of data points exposed by this device.
   - **Remote control** Its main function is to turn on/off the power supply of the motor. It acts in the peripheral role and exposes the following data points: (a) On button press (b) Off button press (c)Battery level

2. **Define the services, Characteristics and Access Permissions** In this step we will group the characteristics into meaningful groups (Services) and define their access permissions for each one of them.

   (a) **Gateway service** For the gateway to act as server to the smartphone, following services are defined:

CONFIDENTIAL

- **Environment Sensor Service**
  - Environment sensor temperature reading characteristic: "Temperature". Access permissions: Read, notify.
  - Environment sensor moisture reading characteristic: "soil moisture". Access permissions: Read, notify.
  - Environment sensor humidity reading characteristic: "Humidity". Access permissions: Read, notify.
  - Battery level characteristic: "Battery Level". Access permissions: Read, notify.
- **Remote Control Service:**
  - On button characteristic: "On Button Press". Access permissions: Notify.
  - Off button characteristic: "Off Button Press". Access permissions: Notify.
  - Battery level characteristic: "Battery Level" Access Permission : Read, notify
- **GAP service:**
  - Name characteristic: the device name. Access: Read.
  - Appearance characteristic: a description of the device. Access: Read.

(b) **Remote Control** We have one GATT server for the remote control. We can define the following services and characteristics:

- **GAP service (mandatory):**
  - Device name characteristic: "Device Name". Access permissions: Read.
  - Appearance characteristic. Access permissions: Read.
- **Battery service:** Battery level characteristic: "Battery level" Access permissions: Read, Notify
- **Button service:**
  - On button characteristic: "On Button Press". Access permissions: Notify.
  - Off button characteristic: "Off Button Press". Access permissions: Notify.

3. **Implement the Services and Characteristics Using the Vendor SDK APIs** Each platform, whether it's an embedded or mobile one, has its own APIs for implementing services and characteristics.It depends on the reader to implement for the specific platform they choose for their BLE device or application.

??
??

CONFIDENTIAL

Services	Characteristics	UUID	UUID type
Environment Sensor		00aec900-62bb-4ac6-aea5-32c305e4593d	Custom
	Temperature	00aec901-62bb-4ac6-aea5-32c305e4593d	Custom
	Humidity	00aec902-62bb-4ac6-aea5-32c305e4593d	Custom
	Moisture	00aec903-62bb-4ac6-aea5-32c305e4593d	Custom
	Battery Level	0x2A19	SIG-adopted
Remote Control		56dba5bc-725d-4172-8cee-1ea5da7c8303	Custom
	Buttton ON	56dba5bd-725d-4172-8cee-1ea5da7c8303	Custom
	Button OFF	56dba5be-725d-4172-8cee-1ea5da7c8303	Custom
	Battery level	0x2A19	SIG-adopted
GAP		0x1800	SIG-adopted
	Device name	0x2A00	SIG-adopted
	Appearance	0x2A01	SIG-adopted

Figure 8: Gateway GATT Design

CONFIDENTIAL

Services	Characteristics	UUID	UUID type
Remote Control		56dba5bc-725d-4172-8cee-1ea5da7c8303	Custom
	Buttton ON	56dba5bd-725d-4172-8cee-1ea5da7c8303	Custom
	Button OFF	56dba5be-725d-4172-8cee-1ea5da7c8303	Custom
	Battery level	0x2A19	SIG-adopted
GAP		0x1800	SIG-adopted
	Device name	0x2A00	SIG-adopted
	Appearance	0x2A01	SIG-adopted

Figure 9: Remote control GATT Design

Nordic BLE Example using Arduino API

Problem faced

- Configuring the UART port : There was no CMISS Configure file being generated. *Solution*:Alternative method - Double click on sdk_config.h–> press Ctrl + f to open search bar–> type UART and enable it
- Error in Interrupt code: too few arguments to function 'nrfx_gpiote_in_init' not showing in which line there is error and checked the arguments too
- sending img— You may like to have a look at our Code Example " External Memory OTA Bootloader " which uses the same concept of receiving an image over BLE and writing it to external memory.
- INFO : The documentation has used SDK version 15.0.0 but I have used an updated version. i have tried to use that version, but the files weren't getting extracted.

CONFIDENTIAL