Chapter 4: Bluetooth Low Energy (BLE)

Time 3 Hours

At the end of this chapter you will understand the basics of BLE and how to create BLE projects on WICED devices.

**Most importantly, you will be able to use WICED to connect your IoT device to a Wi-Fi Network.**

4.1 BLE Introduction 2

4.2 GAP 2

4.3 GATT 2

4.4 Profiles, Services, and Characteristics 2

4.5 Security 2

4.6 CySmart 2

4.7 Advanced Topics 3

4.8 Using BLE in WICED Studio 3

4.9 Documentation 4

4.10 Exercise(s) 5

Exercise - 4.1 Create a BLE Advertiser 5

Exercise - 4.2 Connect using BLE 5

Exercise - 4.3 Save BLE Bonding Information 5

Exercise - 4.4 Add a Pairing Key 5

Exercise - 4.5 (Advanced) Join a BLE Mesh Network ??? 5

4.11 Related Example “Apps” 6

4.12 Recommended Reading 6

# BLE Introduction

With the addition of BLE in 2010, it has become very popular particularly in IoT devices such as smart watches, health monitors, beacons, etc. What these applications typically have in common is small batteries that are often not charged frequently. Therefore, low power is more critical than data transfer speed. Moreover, these types of devices don’t require a constant connection. Rather, they can connect somewhat infrequently to send a burst of data.

The scenario described above is ideal for BLE. In fact, the way low power is achieved in BLE is not by lowering the power of the radio (i.e. the range), but rather by having the radio turned off most of the time. That is, BLE connections can stay active while only turning on the radio for a small percentage of each connection interval (e.g. a few hundred microseconds). The connection interval can be varied depending on the application from 7.5 ms to 4 seconds to trade off power and performance.

The MCU is also put in sleep mode a large portion of the time to further reduce power.

BLE is also sometimes referred to as “Bluetooth Smart”. The two terms are interchangeable. Devices that support both Classic Bluetooth and BLE (e.g. smartphones) are sometimes called “Smart Ready”.

# Stack

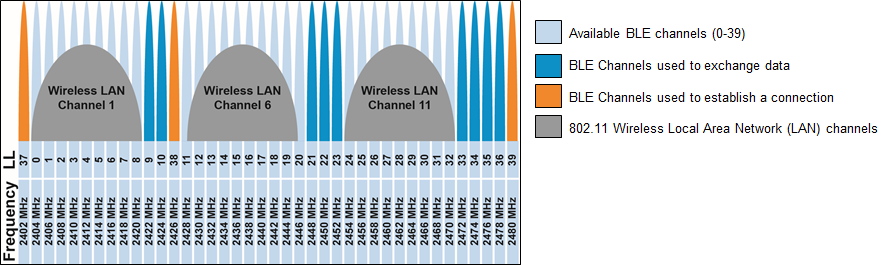
As with most complex systems, the BLE stack is broken into layers as shown below.



# Physical Layer (PHY)

BLE operates in the 2.4 GHz ISM band (2.400 – 2.480 GHz) using 40 channels with 2 MHz spacing between channels. 3 channels are used for advertising (i.e. establishing a connection) and 37 channels are used for data. Gaussian Frequency Shift Keying (GFSK) modulation is used.

In order to work in the crowded 2.4 GHz ISM band, the 3 advertising channels (37, 38, and 39) are spread across the spectrum. For example, a region with 3 Wi-Fi access points operating on 3 different channels may look like this when superimposed on the BLE channels:



BLE uses adaptive frequency hopping (AFH) to avoid channels with poor signal strength or high error rates. In the example above, channels 0-8, 11-20, and 24-32 will likely be identified as channels that should be excluded from frequency hopping due to the interference from the Wi-Fi signals.

The max raw data transfer rate in BLE is 1 Mbps. In Bluetooth v5, the data rate can be doubled to 2 Mbps at the expense of range. Including overhead, the actual data transfer rate is ~300 Kbps in Bluetooth v4.1 and is ~800 Kbps in Bluetooth v4.2 and beyond due to the data length extension which allows larger payloads in each packet (27 bytes vs. 251 bytes). The max payload size can be different between transmit and receive to optimize application throughput.

# Link Layer (LL)

The link layer provides the methods for devices to find each other and connect. It also handles maintaining a reliable link once it has been established.

A device that is available will *Advertise* so that it can be discovered by nearby devices. The advertisement packet includes device information such as services supported and what type of connections, if any, the device will allow.

Devices that want to gather information or form connections will *Scan* for nearby devices that are advertising. Once devices know about each other, the one that initiates the connection (i.e. the one that was scanning for devices) will be the *LL Master*, while the one that accepts the connection will be the *LL Slave*.

Once a connection is established, the link layer uses AES-128 encryption and 24-bit cyclic redundancy check (CRC) to guarantee a private and reliable connection. The link layer also implements AFH as described previously.

# Logical Link Control Adaptation Protocol (L2CAP)

The L2CAP layer is responsible for taking large packets of data from the upper layers and segmenting them into smaller packets for the link layer, and vice versa. The largest possible size for data packets being transmitted in BLE is called the Maximum Transmission Unit (MTU). It can be set in the range of 23 to 512 bytes.

# Generic Access Profile (GAP)

The GAP defines how devices discover each other, how they establish a connection, and how they interact with each other based on their roles. There are four GAP roles. The first two involve a connection, while the last two involve an exchange of data without a connection (i.e. advertise/scan only). They are:

|  |  |
| --- | --- |
| **GAP Role** | **Description** |
| Peripheral | A device that connects to a Central. Typically, this is an IoT device like a fitness monitor. |
| Central | A device that connects to a Peripheral. Most often, this is a smart phone or tablet. |
| Broadcaster | A device that only advertises. It may transmit useful data within the advertising packets. This may be an IoT device such as a beacon or a GPS tag. |
| Observer | A device that scans for devices and may use data from their advertising packets. |

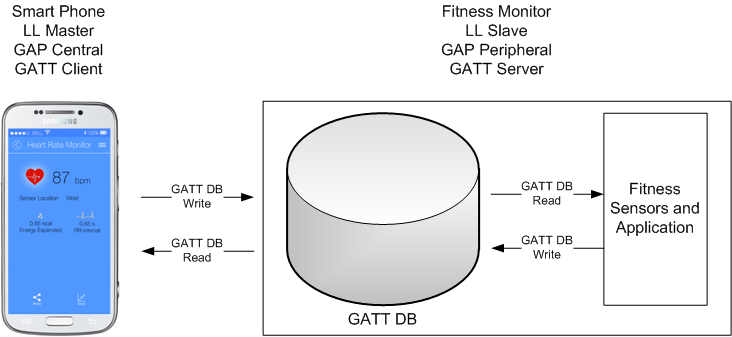
# Generic Attribute Profile (GATT)

Once a connection is made, the GATT determines how data is exchanged. There are 2 GATT roles:

|  |  |
| --- | --- |
| **GATT Role** | **Description** |
| Server | A device that contains data to be shared. Typically, this is an IoT device like a fitness monitor. |
| Client | A device that requests data from the server. Most often, this is a smart phone or tablet. |

Note: based on the above roles, the GATT server is typically a GAP peripheral while the GATT client is typically a GAP central.

Servers use a GATT Database to store data in a format defined by the Bluetooth spec. The database responds to read/write requests from server itself (e.g. when new data is available from a sensor) and from a connected client. Allowed transactions are defined when the database is setup in the server (e.g. which values the client can write/read vs. read). For example, a client may be allowed to write/read configuration settings on the server but may only be allowed to read the values of sensors.



# Profiles, Services, Characteristics, and Attributes

## Profiles

Once a GATT DB is available, how does each device know what data is stored and how it is represented? The answer is Profiles. A profile allows devices to understand what data is stored in the database without having to do a complicated exchange of information each time a device connects. When a connection happens, devices only need to tell each other which profiles and services (more on services in a minute) they support along with some basic configuration information about each service, and then they are ready to exchange data. In fact, advertisement packets often contain information about supported profiles and services for a device. (Is this more a description of services than it is profiles?)

The Bluetooth SIG defines a set of Standard (a.k.a. adopted) GATT Profiles. If two devices implement the same standard profile, they are guaranteed to be interoperable. Each standard profile as a 128-bit Universally Unique Identifier (UUID). Is this true or is it just the services/characteristics that have UUIDs? The CTW says that profiles have UUIDs but I only see service and characteristic UUIDs.

Non-standard (a.k.a. Custom) GATT Profiles are also supported by BLE and are often provided for proprietary technologies. For example, Cypress uses a custom profile for CapSense. In this case, a custom UUID is used. This UUID must be recognized a custom app on the client so that it knows how to interact with the GATT DB for the custom service.

## Services

A profile is a collection of one or more services. A service is something that provides some related set of information. For example, the Blood Pressure Profile requires a Blood Pressure Service and may also have optional services such as Device Information. All profiles require a Generic Access Service and a Generic Attribute Service.

Each service has a UUID – either one assigned by the SIG, or a custom one for custom services.

## Characteristics

A service is a collection of characteristics. The characteristics are different items that are all related to the service. For example, the Blood Pressure Service contains three characteristics: Blood Pressure Measurement, Intermediate Cuff Pressure, and Blood Pressure Feature. Each of these is related to blood pressure measurement but will contain different information.

Like profiles and services, characteristics also have UUIDs.

## Attributes

A characteristic is collection of attributes. An attribute specifies the format of the data and contains the data itself as a series of fields. For example, the Blood Pressure Measurement Characteristic contains an attribute structure with the fields Flags, Measurement Compound Value, etc. The exact fields included and the units used to represent the data in each field are specified by the Flags field. In that way, by reading the flags, both devices know what data is in the GATT DB and how it is represented.

# Attribute Protocol (ATT)

The ATT defines the rules for BLE communication. It enables GATT clients to find and access attributes on GATT servers using six operations: Requests, Responses, Commands, Notifications, Indications, and Confirmations. Examples of each operation:

GATT Client reads data:

1. Client sends a Request for the data.
2. Server sends a Response with the data.

GATT Client writes a value (such as registering for notifications or indications):

1. Client sends a Command to the server.
2. Server receives the Command.

GATT Server sends a notification of new data (assumes client previously asked for notifications):

1. Server sends a Notification.
2. Client receives the Notification.

GATT Server sends an indication of new data (assumes client previously asked for indications):

1. Server sends an Indication.
2. Client receives the Indication and responds with a Confirmation.
3. Server receives the Confirmation.

# Security Manager (SM), Pairing and Bonding

BLE has two security modes, and several levels in each mode. They are:

|  |  |  |  |
| --- | --- | --- | --- |
| **Security** | **Level 1** | **Level 2** | **Level 3** |
| Mode 1 | No security | Unauthenticated  Encrypted | Authenticated  Encrypted |
| Mode 2 | Unauthenticated  Data Signed | Authenticated  Data Signed | N/A |

Authentication can be done in one of several ways depending on the capabilities of the devices. The possible capabilities are:

1. No Input, No Output
2. Display Only
3. Display
4. Display: Yes/No
5. Keyboard Only

Need to understand these better and explain the possible options depending on the capabilities. What is display only vs. display yes/no vs display?

Need to understand/explain what data signed means. How is this different from encryption?

Need details on authentication and encryption schemes.

Once two BLE devices have established a connection (including authentication and key exchange if necessary), they are considered Paired. If the authentication information and keys are stored in memory, then the devices are Bonded. Devices that are bonded can connect in the future without going through the pairing process again.

The whole process looks like this:



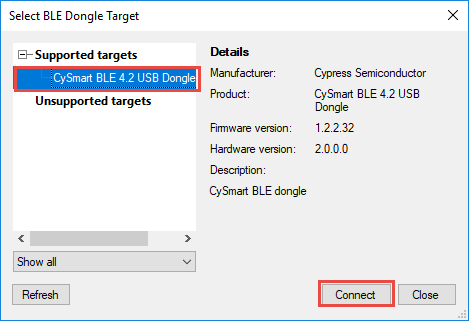
In Bluetooth v4.2, privacy 1.2 was introduced. This involves using a 48-bit resolvable private address (RPA) that can be changed frequently (every 1 second) to prevent tracking. Only peer devices that have the 128-bit identity resolving key (IRK) of a BLE device can connect to it.

# CySmart

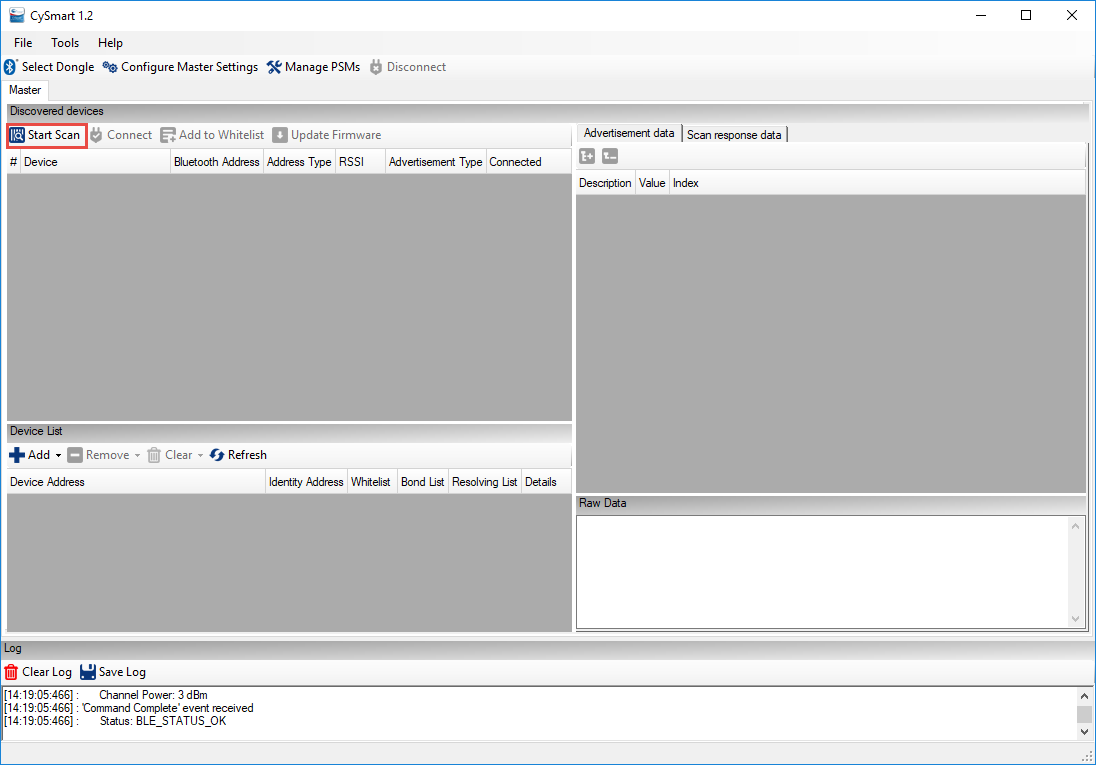
Cypress provides a PC and mobile device application (Android and iOS) called CySmart which can be used to scan, connect, and interact with services, characteristics, and attributes of BLE devices.

## CySmart PC Application

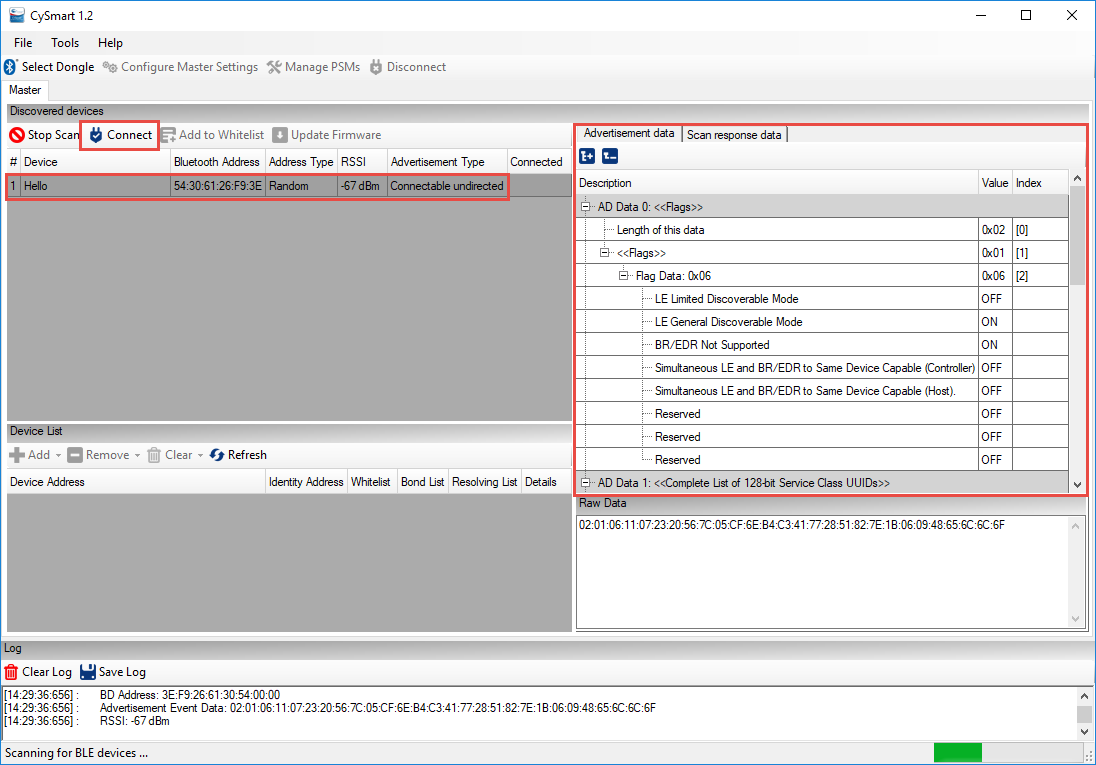
To use the CySmart PC Application, a CY5670 CySmart USB Dongle is required. When CySmart is started, it will search for supported targets and will display the results. Select the dongle that you want to use and click on “Connect”.



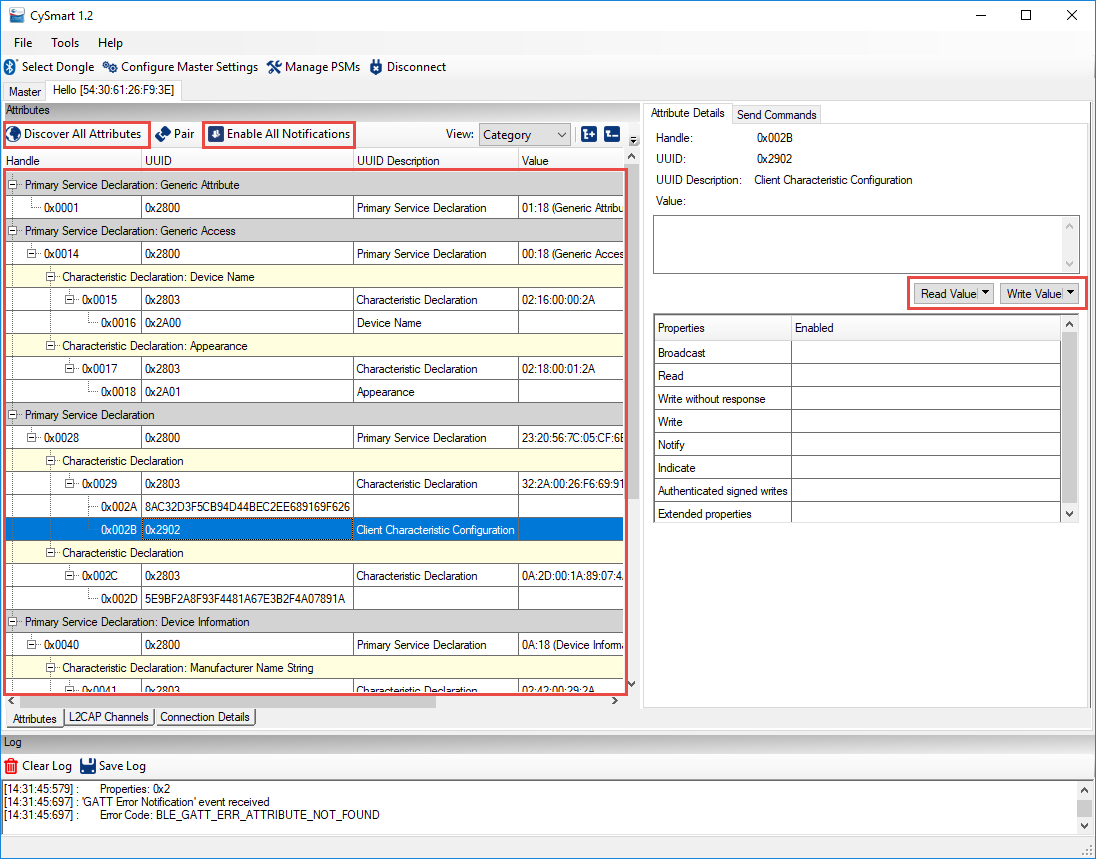
Once a dongle is selected, the main window will open as shown below. Click on “Start Scan” to search for advertising BLE devices.



Once the device that you want to connect to appears, click on it. You can then see its Advertisement data and Scan response data in the right-hand window. Click “Connect” to connect to the device.

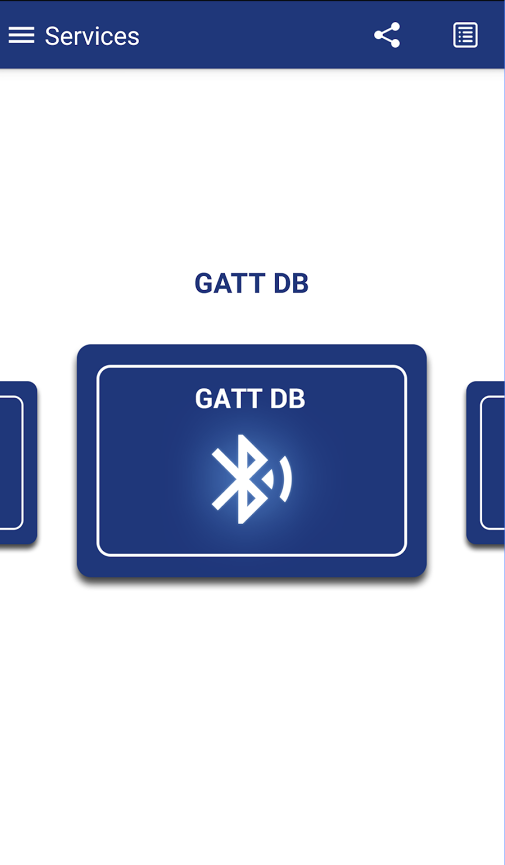
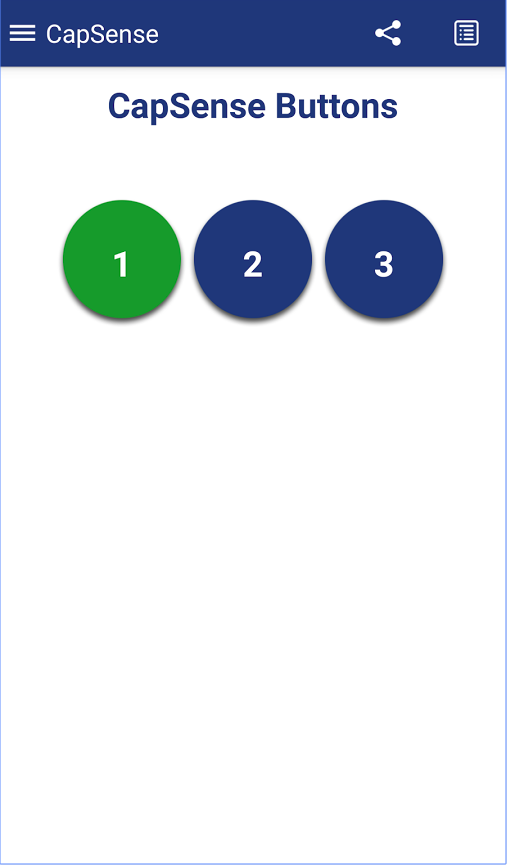
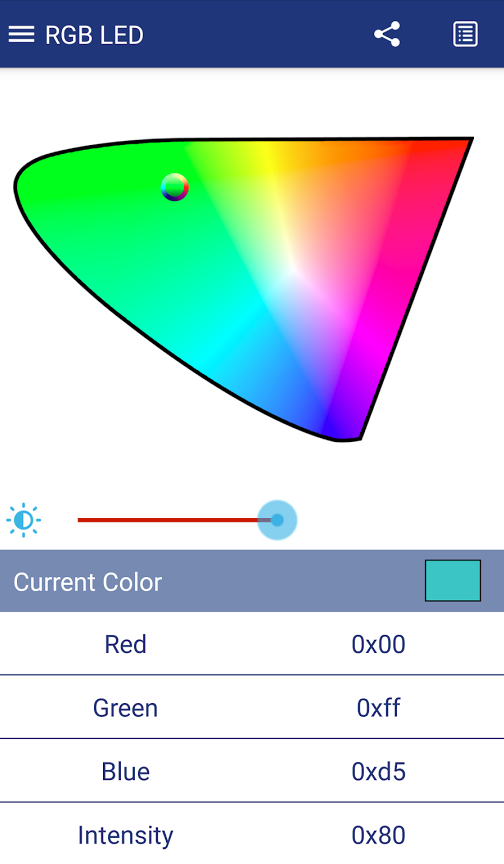
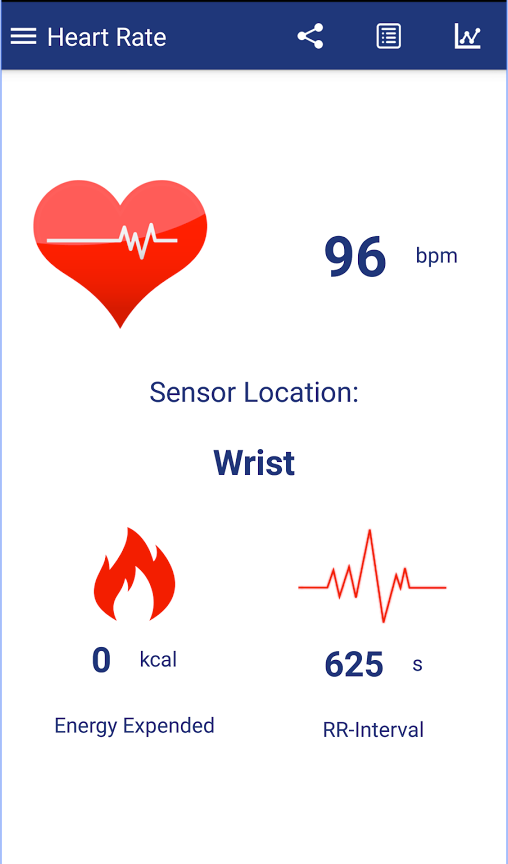


Once the device is connected, click on “Discover All Attributes”. Once that is complete, you will see a representation of all services, characteristics, and attributes from the GATT database. You can read and write values by clicking on an attribute and using the buttons in the right-hand window. Click “Enable All Notifications” if you want to see real-time value updates in the left-hand window for characteristics that have notification capability.



## CySmart Mobile Application

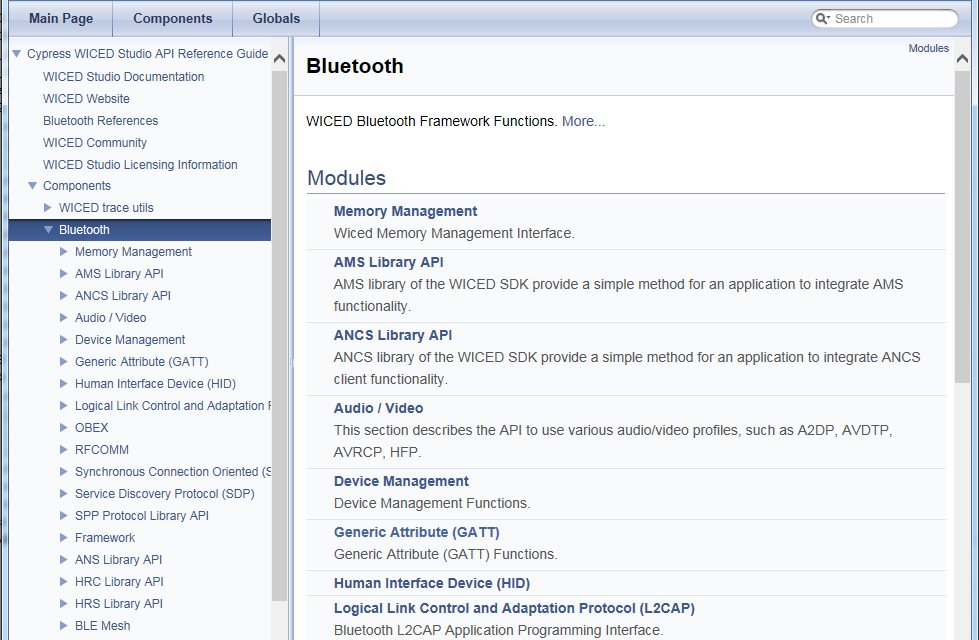
The CySmart mobile application is available on the Google Play store and the Apple App store. The app can connect and interact with any connectable BLE device. It supports specialized screens for many of the BLE adopted services and a few Cypress custom services such as CapSense and RGB LED control. In addition, there is a GATT database browser that can be used to read and write attributes for all services even if they are not supported with specialized screens.

# Using BLE in WICED Studio

## Documentation

The relevant documentation for Bluetooth functions are in the WICED SDK documentation under Components🡪Bluetooth.



## Bluetooth Stack Initialization and Callback

Same as Classic Bluetooth? – move or copy to here or put in a separate document?

## GATT Database Functions

Need to validate writes are allowed before writing the database. Need to send notifications/indications and look for indication confirmation – if not do we need to re-send (timeout?)

## Example Applications

The WICED Studio SDK contains a wealth of example snip and demo applications to illustrate how to use various BLE features. It is always recommended to start a new application based on one of the provided examples or by using *WICED Bluetooth Designer* (discussed in the next section) to simplify and speed up development.

Some of the example applications provided in the WICED Studio SDK are described below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Application Name** | **Connection/Pairing**  **Method** | **Services** | **Description** |
| snip.ble.ams |  |  |  |
| snip.ble.anc |  |  |  |
| snip.ble.ancs |  |  |  |
| snip.ble.ans |  |  |  |
| snip.ble.bac |  |  |  |
| snip.ble.bas |  |  |  |
| snip.ble.eddystone |  |  |  |
| snip.ble.env\_sensing\_temp |  |  |  |
| snip.ble.gatt\_db |  |  |  |
| snip.ble.hrc |  |  |  |
| snip.ble.hrs |  |  |  |
| snip.ble.ibeacon |  | ` |  |
| snip.ble.multi\_beacon |  |  |  |
| snip.ble.mybeacon |  |  |  |
| demo.hello\_cleint |  |  |  |
| demo.hello\_sensor |  |  |  |

## WICED Bluetooth Designer

Same (or at least similar to) Classic Bluetooth – move or copy to here or put in a separate document?

# Advanced Topics

DTM – Direct Test Mode

HCI - Host Control Interface

OTA updates

Multi-role devices

Mesh

# Exercise(s)

* 1. Create a BLE Advertiser

??? Use Bluetooth designer or do from a template?

* 1. Connect using BLE

??? Use Bluetooth designer or do from a template?

* 1. Save BLE Bonding Information

??? Use Bluetooth designer or do from a template?

* 1. Add a Pairing Key

??? Use Bluetooth designer or do from a template?

* 1. Implement a BLE Central Device ???

??? Use Bluetooth designer or do from a template?

* 1. (Advanced) Join a BLE Mesh Network ???

??? Use Bluetooth designer or do from a template?

# Recommended Reading