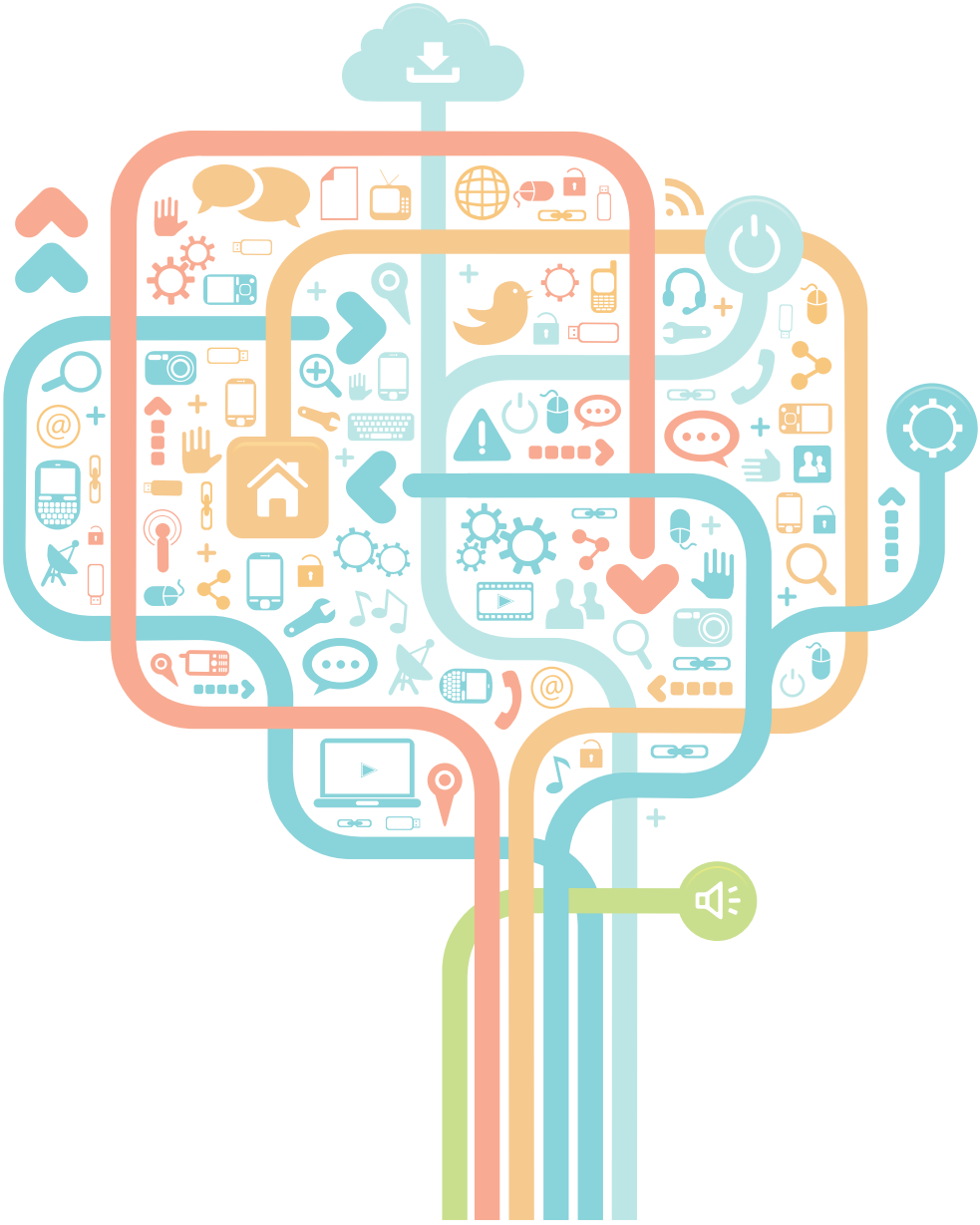
wimoto



WIMOTO MOTES
DEVELOPER GUIDE V1.0.4

Version	Release Date	Author	Notes
1.03	20-Nov-2014	Marc Nicholas	First public release
1.04	1-Dec-2014	Marc Nicholas	Minor fixes

1 Introduction

1.1 About Wimoto Sensors

Wimoto sensors are small, battery-powered wireless sensors that monitor various environmental conditions. They communicate using the Bluetooth Low Energy (Bluetooth Smart) standard to modern smartphones, tablets, computer and our own gateway (Mesh).

There are currently five sensors in the Wimoto family. Their names and capabilities are listed below:

	Climate	Grow	Sentry	Thermo	Water
Ambient light					
Ambient temperature					
Ambient humidity					
Soil moisture					
Passive infrared motion sensor					
Accelerometer					
Thermopile/infrared thermometer					
Probe thermometer					
Water detector					

The sensors include a lithium coin cell battery (CR2450) and will run for over a year under normal usage conditions. The battery is replaceable by removing the screws on the back of the Wimoto with a Philips PH00 screwdriver.

Things that affect battery life are:

- Being continuously in a 'connected state' with the sensor (an unusual use-case as it would require a constantly running smartphone or tablet to be in close proximity to the sensor)
- Aggressive use of the data logger downloading function

Note: even greater power efficiency is achieved when the sensor is running purely in Sensor (Broadcast) mode with no active Central connections (read on for a description of what a Central is).

The range of the sensors is highly dependent on physical obstructions with metal and concrete being among the least desirable materials. However, a range of 35m/100ft

in open air is achievable but is also partly dependent on the device making the connection.

1.2 Bluetooth Low Energy

Bluetooth Low Energy was introduced in the Bluetooth Core Specification version 4.0. It should be noted that Low Energy is an optional component of the Core Specification and not all Bluetooth 4.0 devices support Low Energy. To further confuse matters, Bluetooth Low Energy is marketed as “Bluetooth Smart”.

For a Bluetooth 4.0 host controller to work with Wimoto sensors, the host controller should be either “Dual Mode” capable (meaning it supports both classic mode Bluetooth and Bluetooth Low Energy) or a “Single Mode” Bluetooth Low Energy device (meaning it only supports the Low Energy protocol).

From this point on, we will refer to Bluetooth Low Energy by the acronym “BLE”.

1.2.1 Roles

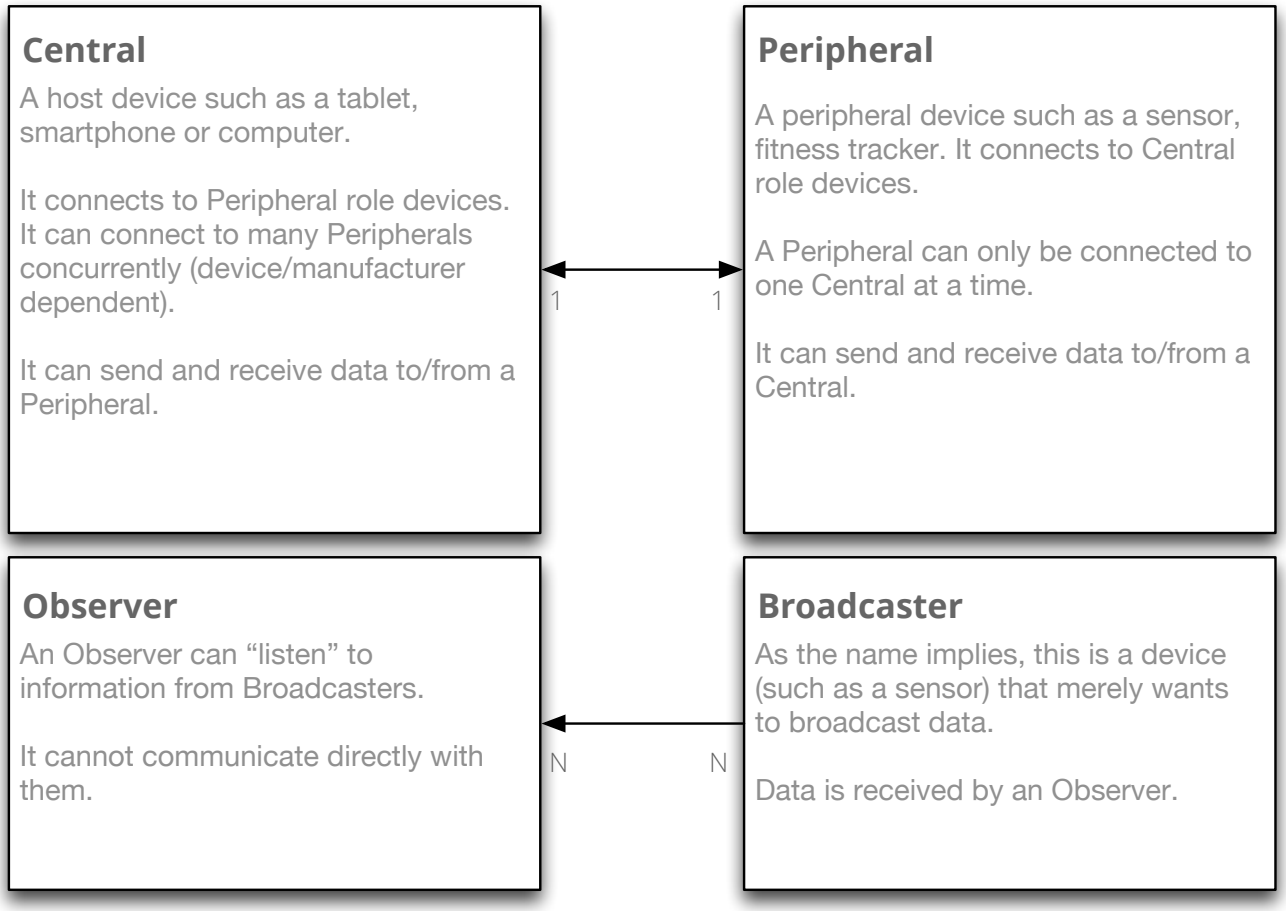
Bluetooth Low Energy has a master-slave relationship at both the communications protocol level and the data access level. For now, we will focus on the communications protocol layer, which is called the Generic Access Protocol (GAP).

GAP supports a number of “roles” and a Bluetooth Low Energy device is allowed to implement a minimum of one (1) role and may in fact implement them all.

Central and Peripheral roles have a 1:1 relationship: a Peripheral can only accept one connection at once from a Central. And a Peripheral can only talk to one Central at once. However, most Central can talk to multiple Peripherals at once.

Observer and Broadcaster roles have no restrictions, as they are passive rather than active connectivity.

The currently defined roles are in the chart below.



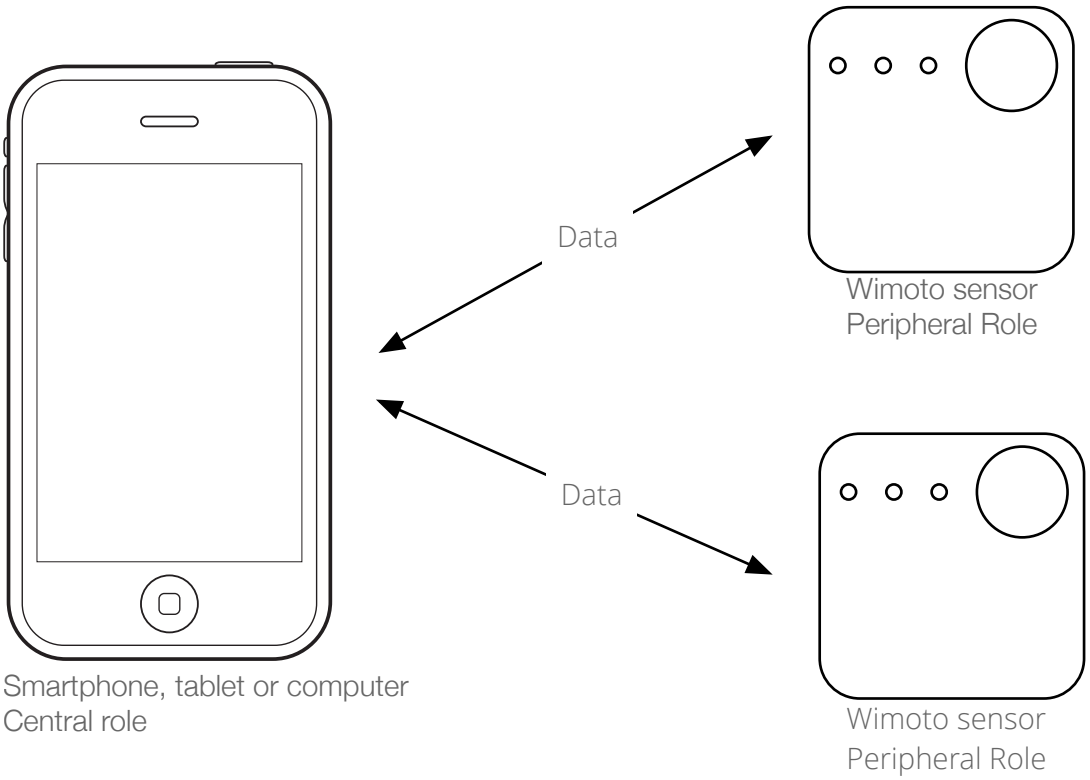


Figure 1: Central-Peripheral role

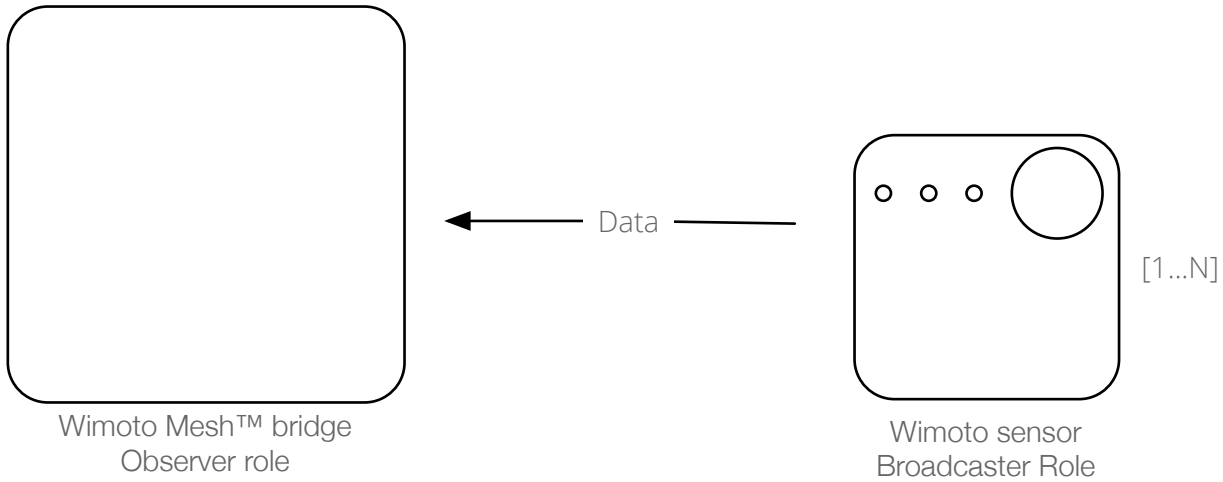


Figure 2: Observer-Broadcaster role

1.2.2 Profiles, Services and Characteristics

Bluetooth Low Energy implements access to data in an object-oriented fashion. A BLE device (a Peripheral or Broadcaster) implements one or more “Services” that contain one or more data characteristics (Characteristics). A collection of Services and Characteristics is referred to as a Profile. A blood glucose meter, for example, might have a profile that includes a current glucose reading Service and Characteristic, a Battery Level Service, and a Device Information Service with manufacturer information such as serial number and model.

The object-oriented nature of BLE means that a Service can also include other Services as sub-services, which in turn have their own data Characteristics. This is referred to as a “Secondary” rather than “Primary” Service.

Characteristics can be self-describing by having descriptors attached to them. The Bluetooth Special Interest Group (Bluetooth SIG) manages these descriptors.

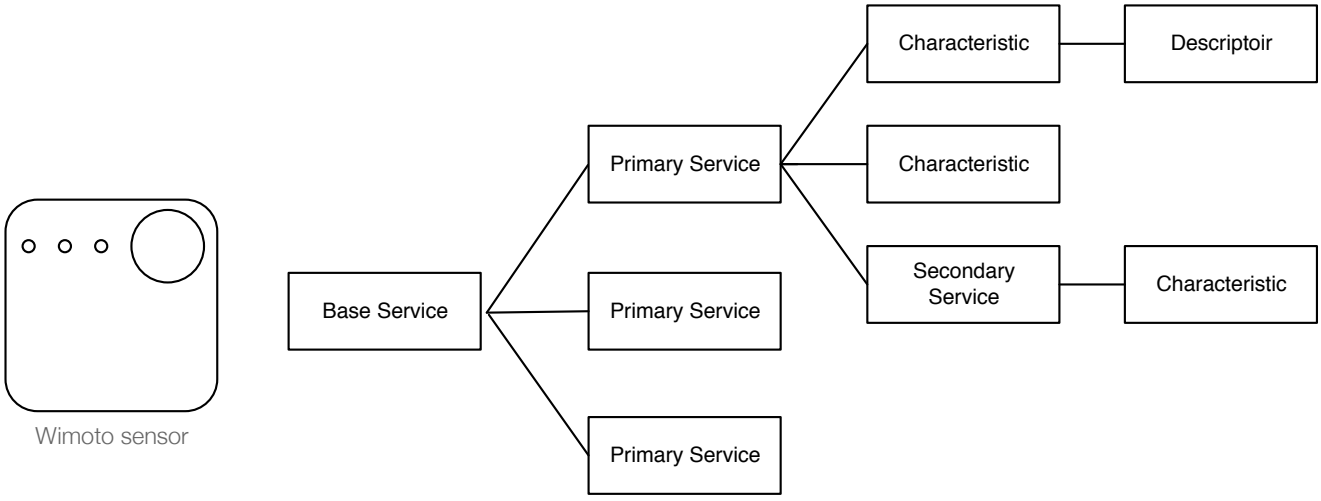


Figure 3: Relationship between Services and Characteristics

Services and Characteristics have unique hexadecimal identifiers (UUIDs). The Bluetooth SIG has defined some common Profiles and they carry shorter 16-bit (four hexadecimal characters) UUIDs. Custom Services and Characteristics carry long 128-bit UUIDs which are made up of a identifier unique to an organization plus 16-bits specific to the Service or Characteristic it’s applied to. A manufacturer may additionally have a Company Identifier – Wimoto’s is 0x0107 -- whoch

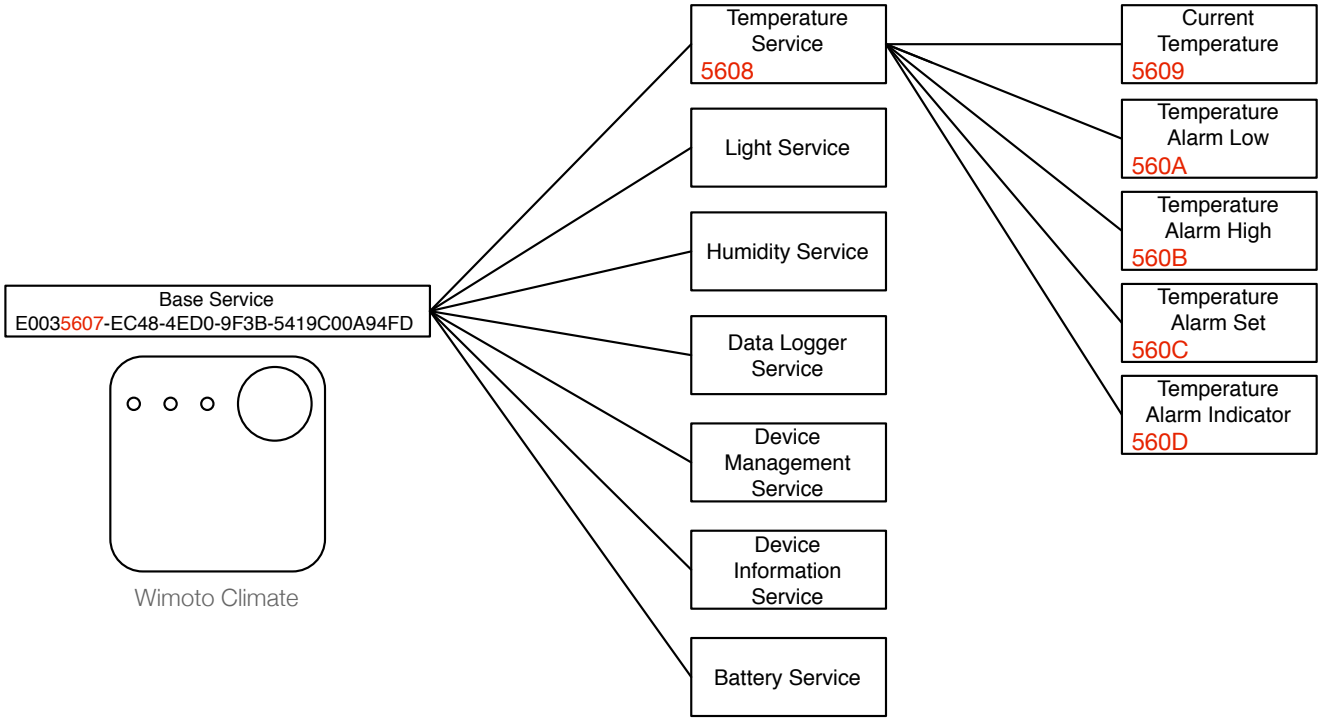


Figure 4: Partial map of Wimoto Climate profile

1.2.3 Notifications and Indications

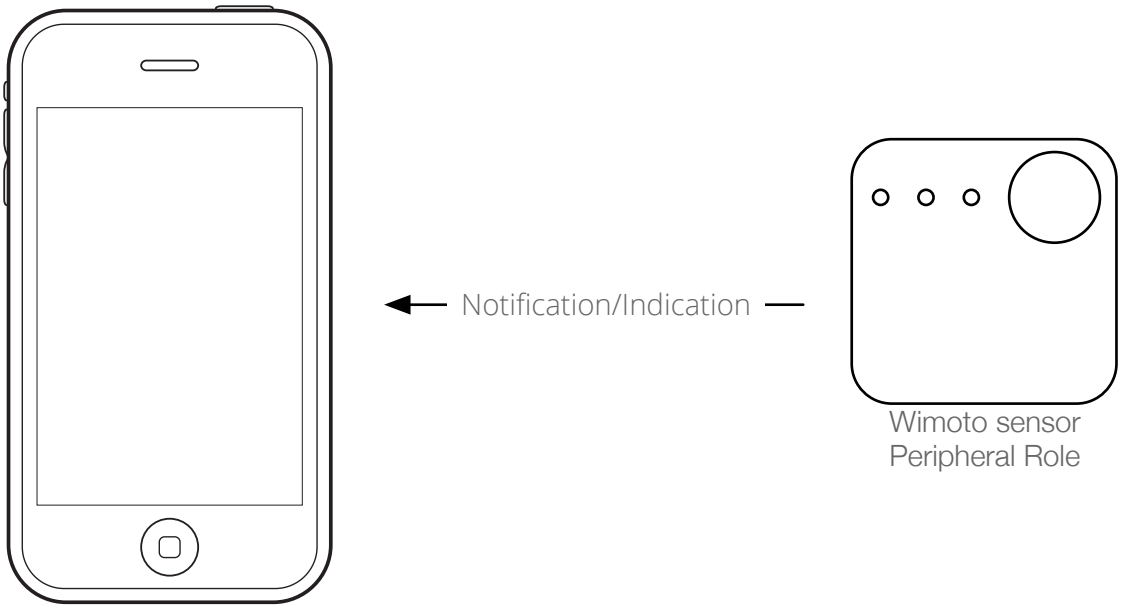


Figure 5: Central's 'subscribe' to Notifications and these are 'pushed' from a Peripheral asynchronously.

A BLE Central can "subscribe" to Notifications and Indications from a Peripheral (this is not available in the Observer-Broadcaster relationship) and have updates 'pushed' to the Central rather than continuously polling for changes. This is an extremely powerful feature and Wimoto takes advantage of this for things such as data updates, alarms, and transferring large amounts of data for the data-logger.

For example, a Central may subscribe to the current temperature characteristic on a Climate and then get a notification every time that temperature value changes. This happens completely asynchronously.

An Indication is similar to a Notification except the Central has to specifically respond to an Indication. Wimoto uses this for alarms.

2 Working With Wimoto Sensors

2.1 Development Requirements

To communicate with Wimoto Sensors, you will need:

- Hardware that supports Bluetooth Low Energy (referred to as "Bluetooth Smart Ready"). Many current generation smartphones and tablets support BLE – all Apple iPhones after the 4s, all iPads after 3rd generation, many Samsung, HTC and Google Nexus devices, most newer Macs.
- Access to the "Bluetooth stack" via the programming language of your choice.

iOS and OSX: Apple's SDK/APIs feature something called "CoreBluetooth" which is available in iOS > 6 and OSX > Lion.

Android: Android 4.3/Developer API 18 introduced Google-provided BLE APIs and a stack. Previously, manufacturers such as Samsung implemented their own, which caused app development challenges.

Linux: Linux kernel > 3.6 features the BlueZ Bluetooth Stack which has good support for Bluetooth Low Energy. Version > 4.99 of the BlueZ utilities is required to make best use of this environment.

Embedded: consult your silicon vendor as to the availability of a suitable BLE stack or BLE radio modules that extrapolate the BLE stack.

2.2 Sensor Specifics

We're going to use the Climate as a use-case, but this applies equally to all of our sensors.

Figure 6. shows the full map of the Climate's Profile. Our private UUIDs have been shortened from 128-bits to 16-bits for clarity. Please refer to the Excel spreadsheet "Wimoto Profiles" for full UUIDs.

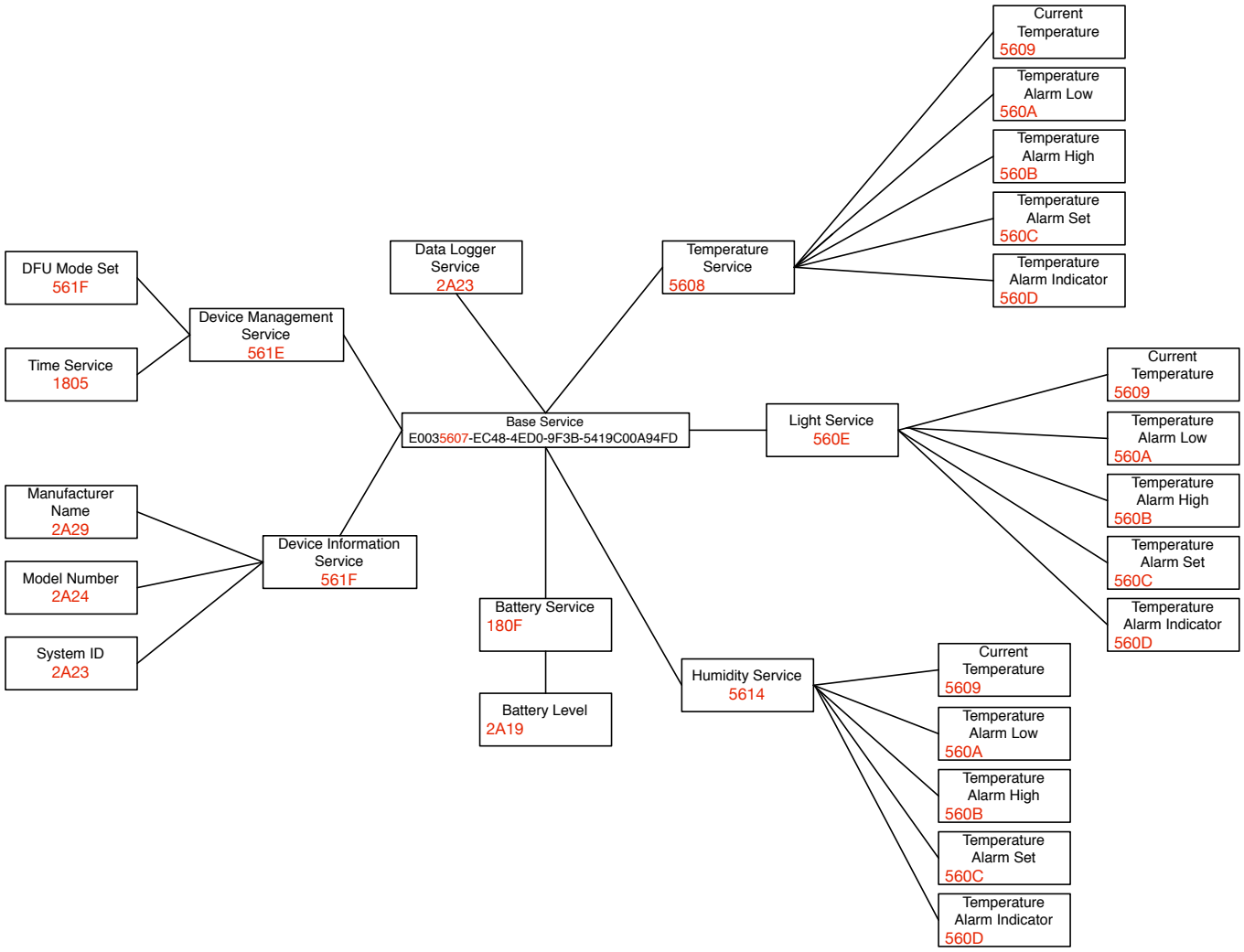


Figure 6: Wimoto Climate Services and Characteristics diagram

2.2.1 Temperature Primary Service

The Temperature Service has the following Primary Service UUID:

`E0035608 - EC48 - 4ED0 - 9F3B - 5419C00A94FD`

Within that Service, there are five Characteristics.

2.2.1.1 Current Temperature

`E0035609-EC48-4ED0-9F3B-5419C00A94FD - Current Temperature`

The current temperature is two bytes long and is stored in a proprietary format. The format is big endian and is converted to Celsius with the following pseudo-code:

```
temperatureC= -46.85 + (175.72/65536 ) *(float)u16sT;
/* T= -46.85 + 175.72 * ST/2^16 */
```

This characteristic can be subscribed to and when the temperature local to the Wimoto changes, a Notification callback will be triggered.

2.2.1.2 Temperature Alarm Low Value

`E003560A-EC48-4ED0-9F3B-5419C00A94FD - Temperature low value (for alarm)`

The temperature low alarm is two bytes long and in the same temperature format as current temperature. When this characteristic is set and Temperature Alarm Set is enabled, subscribing to Notifications of the Temperature Alarm Indicator will yield a callback when the current temperature crosses below this threshold.

2.2.1.3 Temperature Alarm High Value

`E003560B-EC48-4ED0-9F3B-5419C00A94FD - Temperature high value (for alarm)`

The temperature high alarm is two bytes long and in the same temperature format as current temperature. When this characteristic is set and Temperature Alarm Set is enabled, subscribing to Notifications of the Temperature Alarm Indicator will yield a callback when the current temperature crosses above this threshold.

2.2.2 Temperature Alarm Set

`E003560C-EC48-4ED0-9F3B-5419C00A94FD - Temperature Alarm set`

This is a single byte. Writing 0x01 will enable the alarm feature and writing 0x00 will disable the alarm feature.

2.2.3 Temperature Alarm Indicator

```
E003560D-EC48-4ED0-9F3B-5419C00A94FD - Temperature Alarm
```

This is a single byte. Subscribing to Notifications on this characteristic will trigger a call back with either a value of 0x01 for a Low Alarm or 0x02 for a High Alarm. 0x00 is a no alarm.

2.2.4 Light Level and Humidity Primary Services

The light level and humidity Services work in predominantly the same manner as temperature.

Light is delivered as a 16-bit decimal in lux where 0 is 0 lux and 65535 is 65535 lux.

Humidity is also stored in a proprietary format and can be decoded with the following pseudo-code:

```
humidityRH = -6.0 + (125.0/65536) * (float)u16sRH;
/* RH= -6 + 125 * SRH/2^16 */
```

2.2.5 Data Logger

All Wimoto sensors feature a data logger feature that will record sensor data at 15-minute intervals for 30-days in a cyclic buffer (i.e. on Day 31, Day 1's data is overwritten but Day 2-30 remain intact).

To enable the data logger on the Climate, write 0x01 to the the following characteristic:

```
UUID E003561B-EC48-4ED0-9F3B-5419C00A94FD - Enable data logger service
```

Writing 0x00 to the above Characteristic will disable the data logger.

To read out the data logger, write 0x01 to the following characteristic:

```
UUID E003561D -EC48-4ED0-9F3B-5419C00A94FD -   Trigger data logger output
```

And then subscribe to Notifications on the following characteristic:

```
UUID E003561C-EC48-4ED0-9F3B-5419C00A94FD – Data logger output
```

The data field is an unsigned integer array of size 16. The data format is as given below.

The 1st and 2nd bytes contain the year
The 3rd byte contains the month
The 4th byte contains the day
The 5th and 6th byte contain the hour
The 7th byte contains minutes
The 8th byte contains seconds
The 9th and 10th bytes contain the temperature
The 11th and 12th bytes contain light level
The last 4 bytes contain the humidity level

For example:

```
07DD0B15000B32000101020200000055
```

Year	Month	Day	Hour	Minute	Seconds	Temperature	Light	Humidity
07DD	0B	15	000B	32	00	0101	0202	00000055
2013	11	21	11	50	00			

2.2.6 Device Management Service

The Device Management Service consists of three Characteristics. We do not recommend that integrators or developers try to implement the Device Firmware Update routines themselves. Wimoto can supply code to integrate into iOS and Android apps to do this that has been tested and validated not to corrupt the device Flash memory.

Therefore, the only Characteristic that is of interest is the Time Characteristic.

```
UUID 1805 - Time Stamp
```

A counter on this characteristic gets started when the device powers up. In that state, it acts as an uptime counter.

An integrator or developer can also set the values for current date and time so that it functions as a real-time clock. The data format of time stamp is as flows:

yy-yy - mm – dd – hr – min - sec

2.2.7 Device Information Service

The following read-only Characteristics are included in the Device Information Service and are defined by the Bluetooth SIG:

```
UUID 2A29- Manufacturer Name ('Wimoto')
UUID 2A24- Model Number ('Wimoto_Climate')
UUID 2A23- System ID - unique identifier
```

2.2.8 Battery Service

Battery Service is also a Bluetooth SIG defined service. The Service UUID is 180F and the Characteristic 2A19 contains the battery level. Converting this to decimal yields an approximate percentage of battery capacity (0-100%).

```
UUID 2A19- Battery Level
```

2.2.9 Broadcast ("Sensor") Mode

All of the pervious information pertains to use the Wimoto in 'Peripheral' mode from a Central role device.

For some applications, it may be more desirable to use the Wimoto in a more classic "sensor" style function. We call this "Sensor Mode" and it utilizes Bluetooth Low Energy's Broadcaster role.

In Sensor Mode, the sensor broadcasts Manufacturer Specific Data that includes a well-known service (Battery Service) plus Wimoto proprietary data.

The data format for Broadcast data is, using the Climate as an example:

Company Identifier	Temperature	Light	Humidity
01-17	TT-TT	LL-LL	HH-HH

For a total of 8-bytes of custom Broadcast data.

3 Further Reading

The full list of our Profiles and Characteristics is available here:

https://github.com/Wimoto/Developers

Robin Heydon's book "Bluetooth Low Energy: The Developer's Handbook" is an excellent guide to BLE:

http://www.amazon.com/Bluetooth-Low-Energy-Developers-Handbook/dp/013288836X/ref=sr_1_1?ie=UTF8&qid=1416494608&sr=8-1&keywords=bluetooth+low+energy

Sandeep Mistry has created an excelled node.js module for Wimoto sensors:

https://github.com/sandeepmistry/node-wimoto