HOME   PRODUCTS   SOLUTIONS

ARGENOX   LIBRARY   BLOG   ABOUT US

CONTACT US

A BLE Advertising Primer
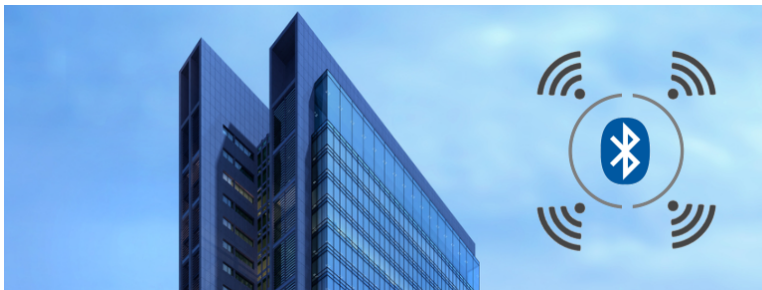
BLE Advertising is one of the most important aspects of Bluetooth Low Energy. Understanding how to properly use advertisements can help you lower your power consumption, speed up your connections, and improve reliability. We're going to go over how they work and how to use them.

Bluetooth Smart has two ways of communicating. The first one is using advertisements, where a BLE peripheral device broadcasts packets to every device around it. The receiving device can then act on this information or connect to receive more information. The second way to communicate is to receive packets using a connection, where both the peripheral and central send packets. We will focus on advertisement for several reasons:

Sign up to get updates and more

Enter E-m

Sign Up

G+

Related Articles

A Guide to

HOME   PRODUCTS   SOLUTIONS

ARGENOX

LIBRARY   BLOG   ABOUT US

CONTACT US

consumption.

- Users want responsive products, and the advertising interval is critical in quick connections.

Advertising is by design unidirectional. A **Central** device can't send any data to the Peripheral device without a connection. But a single peripheral can advertise to multiple masters in the area.

BLE Physical Layer

Before we get into how advertisement packets are sent, we want to talk a little about the BLE physical layer. The physical layer is in charge of actually sending the signals over the air. This includes the actual RF radio.

Bluetooth Low Energy shares some similarities with Classic Bluetooth. Both use the 2.4GHz spectrum. Basic Rate (BR) and BLE both use GFSK modulation at 1Mbps, but their modulation index is different. Enhanced Data Rate (EDR) uses a completely different modulation than GFSK. Classic Bluetooth has 79 channels compared to LE's 40 channels. The channels are also spaced differently. Both of these differences make LE and Classic different and incompatible, so they can't communicate. Dual Mode Radios, like the CC256x, support LE and Classic by switching their modulation parameters and the channels on which they're running.

Bluetooth Low Energy
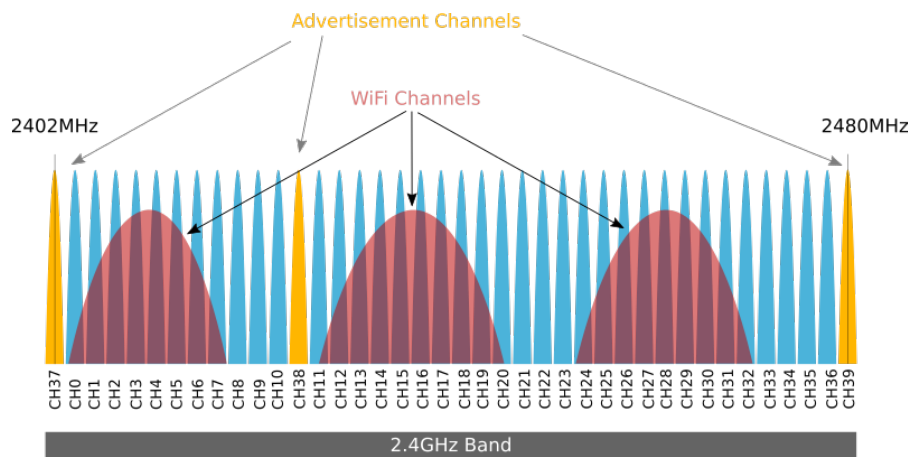
Starting Bluetooth with Arduino

Related Blog Posts

Does your iPhone suppor Blueto v4.2?
March 2016

Whoop iOS 9 crashe Blueto Low Energy Apps
Septem 22, 201

	BLE	Classic	
	BLE	BR	EDR

The 2.4GHz spectrum for Bluetooth extends from 2402MHz to 2480MHz. LE uses 40 1MHz wide channels, numbered 0 to 39. Each is separated by 2MHz.

Channels 37, 38, and 39 are used only for sending advertisement packets. The rest are used for data exchange during a connection. We're interested in what's happening in these 3 channels, and that's what we'll cover here.

During BLE advertisement, a BLE Peripheral device transmits packets on the 3 advertising channels one after the other. A Central device scanning for devices or beacons will listen to those channels for the advertising packets, which helps it discover devices nearby.

Channels 37, 38 and 39 are spread across the 2.4GHz spectrum on purpose. 37 and 39 are the first and last channels in the band, while 38 is in the middle. If any single advertising channel is blocked, the other channels are likely to be free since they're separated by quite a few MHz of bandwidth.

This is especially true since most other devices that interfer with BLE are narrow band. Channel 38 in particular was placed

When a BLE peripheral device is in advertising mode, advertising packets are sent periodically on each advertising channel. The time interval between packets has both a fixed interval and a random delay.
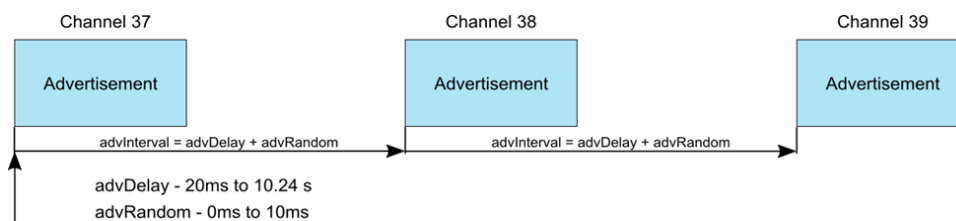
You can set the fixed interval from 20ms to 10.24 seconds, in steps of 0.625ms. The random delay is a pseudo-random value from 0ms to 10ms that is automatically added. This randomness helps reduce the possibility of collisions between advertisements of different devices. We mentioned that finding advertisements is critical, so avoiding collisions at all costs is extremely important. This is just another way Bluetooth Smart uses to improve robustness.

You may be thinking that you would want to advertise on just one or two channels, not all three to save power. Most companies frown on this approach because of the effect from interference. If the channels you selected are blocked, you device won't work. Apple, for example, recommends advertising on all 3 channels, as do other manufacturers.

The advertising interval is separate from the connection interval. So, just because your device is slower to form a connection doesn't prevent you from sending data quickly once the connection is established.
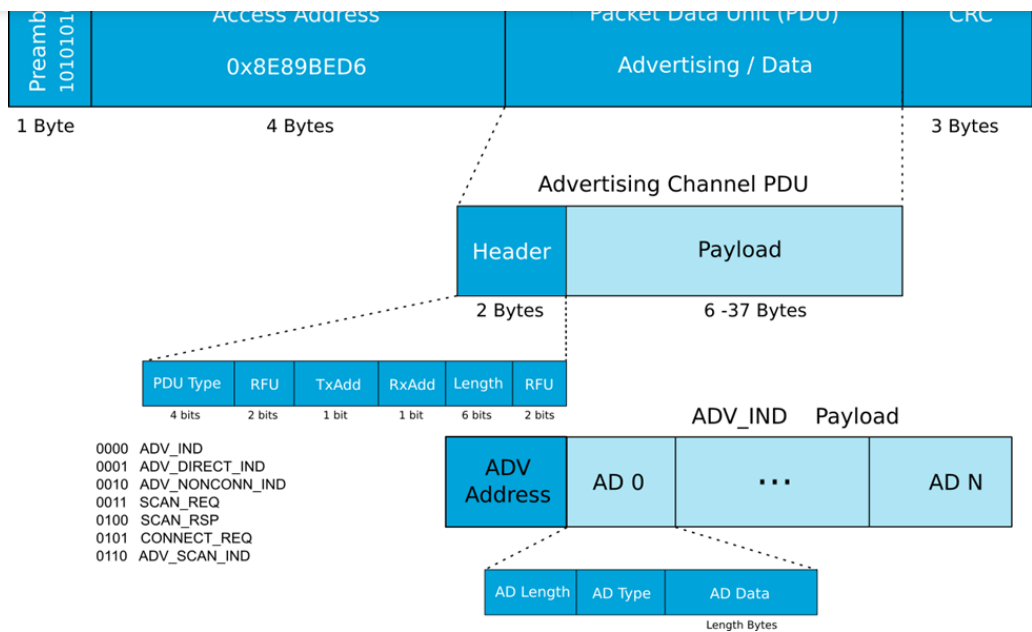
The Packet data unit for the advertising channel (called the Advertising Channel PDU) includes a 2-byte header and a variable payload from 6 to 37 bytes. The actual length of the payload is defined by the 6-bit Length field in the header of the **Advertising Channel PDU**.

It's important to note that there are several PDU types for the advertisements, but here we will focus primarily on **ADV_IND** and **ADV_NONCONN_IND**. ADV_IND is a generic advertisement and usually the most common. It's generic in that it is not directed and it is connectable, meaning that a central device can connect to the peripheral that is advertising, and it is not directed towards a particular Central device.

When a peripheral device sends an ADV_IND advertisements, it is helping Central devices such as Smartphones find it. Once found, a Central device can begin the connection process.

ADV_NONCONN_IND is the advertisement type used when the peripheral does not want to accept connections, which is typical in Beacons.

advertisement data structures.

After the advertisement address is taken into account (this is typically referred to as the Bluetooth MAC Address although it may change at will), we are left with 37 – 6 = 31 bytes for actual advertisement data structures. This has to fit a length, type and data itself.

We went through a few of the layers, and you can see that BLE encapsulates a lot of data the deeper you go. This gives you a lot of flexibility in supporting different behaviors that best suit your product.

At the lowest level, the advertisement has 31 bytes that can advertise any number of different things. You can see the full list of the Advertisement Data types on the Bluetooth Sig Website , each data type specifies a different standard of data in the payload.

Some of the most commonly used **Advertising Data Types** are:

- 0x06 **Incomplete List of 128-bit Services**: UUIDs of the services provided by the peripheral can be advertised in 128-bit format.
- 0x02 **Incomplete List of 16-bit Service Class UUID**
- 0x08 **Shortened Local Name** and 0x09 **Complete Local Nam**e

Since the number of advertising data structures is variable, you can combine them as needed.

The most important part of building the advertisement is getting the right information to the Central device (Smartphone or standalone Central device), and this usually depends on what's important for your product. If your product provides unique services, you can advertise those services so that a Smartphone can distinguish your product from others nearby. For example, Beacons have custom data like unique UUIDs, power levels and other characteristics that are important for finding and using beacons.

The Bluetooth SIG also has the 0xFF data type which is manufacturer specific, so you have the flexibility of defining your own custom payload. Apple did this for

Although every application is different, advertising the most important or unique services provided by the peripheral is the easiest way to connect to it, and it makes sense in a lot of products. When an iPhone or Android is looking for devices, it can use the custom service UUID to find the exact devices it wants to talk to and filter out other devices. Searching for a specific address can be impossible, but finding devices that have unique IDs is easier.

For example, let's say your product is a small light sensor. You can create a custom service with a unique 128-bit UUID. Then by including the UUID in the advertisement packet, an iPhone can ignore all other devices in the vicinity except your product. This makes finding devices faster since you don't have to connect to each device to discover its capabilities.

This is also good when it comes to power savings. New generations of Smartphones are making more and more of the decisions and filtering on the low level. They do this because it's more energy efficient to discard an advertisement packet early on if it's not used than it is to inform the OS and the user of it and discarding it later. But the phone needs to have information about what to filter which the application has to provide, and it depends on unique UUIDs for services or device addresses.

A Quick Look into UUIDs

Once you start working with BLE devices, you'll quickly realize that UUIDs are critical. Services,Characteristics and other items use UUID to uniquely identify them.

UUIDs are nothing more than unique 128-bit (16 byte) numbers:

75BEB663-74FC-4871-9737-AD184157450E

It's typical to arrange the UUID in the format above 4-2-2-2-6. Each pair of characters actually indicate a hexadecimal number. So 75 above is actually 0x75.

the short form 16-bit UUID for the Heart Rate Service is:

0x180D

In reality this represents a 128-bit UUID:

0000180D-0000-1000-8000-00805F9B34FB

If you're using existing services or profiles that were specified by the Bluetooth SIG, you can avoid using the full 128-bit UUID. But, custom services need a fully defined 128-bit UUID.

Creating UUIDs

The only important thing about UUIDs is that they'd be unique.

You can randomly generate them in various ways. One website that can generate them is Online UUID Generator. In Mac OS X you can use the uuidgen utility from the command line for the same thing.

Because of the number of bits, it's unlikely that you'll ever generate a similar UUID as anyone else. It's most important that you avoid the Bluetooth SIG base in custom UUIDs.

Optimizing BLE Advertisements for Power and Latency

One of the most critical things to realize is the big tradeoff between power consumption and latency. Every advertisement consumes power. The BLE radio has to power up and transmit. The less advertisements, the longer the system runs from a set of batteries. So is setting the advertisement interval to 10 seconds a good idea?

Well, let's assume that a user wants to connect to the peripheral. Can you wait 10 seconds? A long interval can be very frustrating to a user, especially if the environment has interference and packets get lost. So, making the interval too large is bad for experience. Somewhere in the middle, around 500ms to 1 second is a sweet spot

BLE Advertisements and the Smartphone

Bluetooth Smart was designed to allow peripherals to be extremely low power. It does thi partially by placing a lot of the burden on the smartphone, with the assumption that a smartphone has a larger battery and is easily recharged. But in real products you want to avoid causing significant battery drain on the smartphone. Enabling Bluetooth drains the battery faster, and some users will end up disabling Bluetooth in frustration. This is a prok for most products that want to provide the user with a fast experience.

Much of the power used by smartphones comes from scanning for advertisements. Beca of this, Android and iOS limit significantly the scanning, especially background scanning.

While your App is in the foreground, you basically have complete control of the BLE and y have a high priority. It makes sense because Apple and Google know you want to show da quickly to the user, so there's no point in delaying it.

Once your application is in the background, looking for nearby devices (if your application supports it), the OS usually downgrades the priority. This is done in a few ways:

- The scanning interval increases, so it takes longer to discover a peripheral device that i advertising. This is especially true if, for example, there is no BLE app in the foreground scanning.
- The OS will generate less advertisement discovery events, such as multiple discoveries the same device.

Android and iOS each handle this separately, so it's important to understand what effect running in the background has on your device and connection.

Scan Response

responds with a scan response.

The Scan Response packet has the same packet format as the advertisement, with the ex
the type on the higher layer indicating it's a scan response instead of an advertisement. S
scan response can provide the device name or other services you didn't mention in the ac
packet.
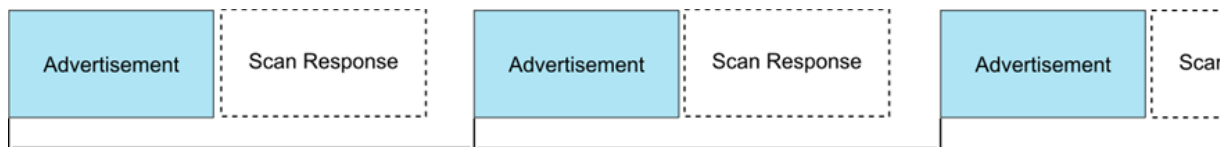
The scan response comes with a catch. If scan responses are enabled in a peripheral devi
device has to keep the radio in RX mode after it sent the advertisement to be able to recei
scan request packet. This has to be done even if there is no device out there that will actu
one (because the peripheral doesn't know who is really out there). This means more energ
consumed. For very low power applications, consider disabling the scan response if not al
needed.

BLE Beacons and iBeacons

No discussion of BLE Advertisement would be complete without discussing Beacons and i
particular Apple's iBeacon standard.
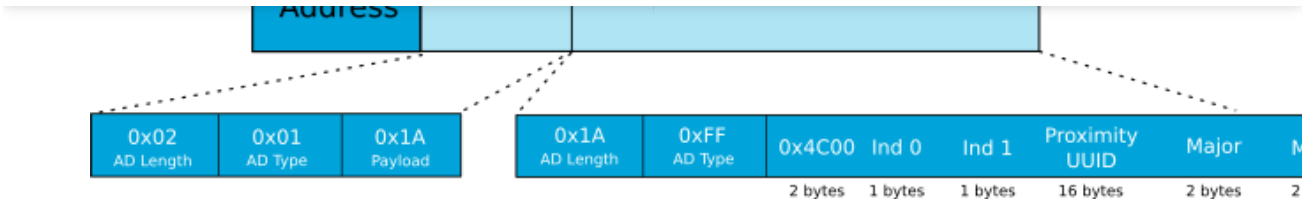
You can see the format for the iBeacon packets. These packets use the basic BLE format, specific fields. Let's go through them one by one.

The advertisement packet contains the Bluetooth MAC address and the payload. The payl composed of two AD Structures, the first one gives generic information using the Flags Da the second is the Apple specific iBeacon information.

Flags Advertising Data Type

This packet has data type 0x01 indicating various flags. The length is 2 because there are data type and the actual flag value. The flag value has several bits indicating the capabilitic iBeacon:

Bit0 – Indicates LE Limited Discoverable Mode

Bit1 – Indicates LE General Discoverable Mode

Bit 2 – Indicates whether BR/EDR is supported. This is used if your iBeacon is Dual Mode d

Bit3 – Indicates whether LE and BR/EDR Controller operates simultaneously

Bit4 – Indicates whether LE and BR/EDR Host operates simultaneously

Most iBeacons are single mode devices BR/EDR is not used. For iBeacons, General discove
is used.

iBeacon Data Type

The most important advertisement data type is the second one. The first byte inc　　　　th

HOME   PRODUCTS   SOLUTIONS

LIBRARY   BLOG   ABOUT US

CONTACT US

and minor fields.

Each iBeacon has to have a unique UUID so that an iPhone app can know exactly where it
to one or more iBeacons.

Finally, there is also a 2's complement of the calibrated TX power that can be used to impr
accuracy knowing the power level of the beacon.

There's nothing stopping you from creating your own beacons with a different manufactu
problem is that Apple specifically detects iBeacons with the particular format, so there wo
interoperability.

Conclusion

We've covered some of the most popular BLE and Bluetooth devices, their specifications a
key aspects to keep in mind when making a decision.

As always, there are many details and concerns that come into play when creating a BLE p
committed to help you get it done right, so feel free to get in touch with us to discuss.

References

[1] Bluetooth Special Interest Group. Bluetooth v4.2 Specification

[2] Apple Inc. iBeacon for Developers.

HOME   PRODUCTS   SOLUTIONS

LIBRARY   BLOG   ABOUT US

CONTACT US

f   🐦   g+

▶   in

Argenox Technologies LLC © 2018