BLE Advertising and Scanning with Angel Sensor

Learn how BLE scanner works with the Angel Sensor and run a simple example script in Python

Get the source code here.

In this post we are going to get into Bluetooth Low Energy advertising and scanning. Advertising is essential in allowing your device to interact with the outside world. We'll get in to a bit of general advertising properties and then go into detail about how the Angel Sensor advertises with a quick demo.

Bluetooth Basics

You may have wondered how devices are discovered on a Bluetooth network, advertising is the magic behind it all. All Bluetooth devices have a Generic Access Profile (GAP) which determines how the devices advertising and connections are controlled. Your device GAP most importantly defines if a device is a *peripheral* device (Angel Sensor) or a *central* device (phone/computer). A central device will listen for advertising data from peripheral devices and then can request more information afterwards in what is called a scan response.

With this information our devices are able to connect to each other and transfer data. We're going to skip the nuts and bolts of the packets and focus on what we are getting from the Angel Sensor. If you are interested, there are heaps of resources online that get into the technical details of BLE packets!

Angel Sensor and BLE Advertising

So what exactly is the Angel Sensor advertising and how can we see it ourselves? Let's start with the what. There are two fields that Angel Sensor advertises: flags and local name. The flags field determines the device discoverability. For this we are what we call general discoverable which means anything can see our advertisements. We're also going to see the local name field which in our case will say 'Angel Sensor <Device Serial Number>' with the serial number unique to your device.

Now let's move on to find your device specific Bluetooth data. If you've been following along with our experiment series (01, 02) then you should be all set up to run this next program. Make sure you have your BLED dongle ready and you have the 'bled112_scanner.py' file as well.

A Bit about the Code

The python we are using today comes from Bluegiga but we've modified it to only detect the Angel Sensor. If you go into the guts of the script, you'll find this snippet at line 459

```
if 'Angel Sensor' in get_local_name(ad_fields):
      mac = '%02X:%02X:%02X:%02X:%02X:%02X' % \
            (sender[5], sender[4], sender[3],
            sender[2], sender[1], sender[0])
      if mac not in known_macs:
            print mac
            known_macs.append(mac)
```

Can you tell what's happening here? We're looking at advertising fields with this first if statement, specifically field 9, the local name. This checks to see if 'Angel Sensor' is in field 9 and then returns the mac address of your device.

Running the Code

Now open a command prompt and navigate to your folder with ‘bled112_scanner.py.’ Now the program defaults to COM10 so you will need to know what COM port your BLED112 is on, then type this:

```
python bled112_scanner.py -p COM<#>
```

You’ll see a few things come up in the terminal and lastly you’ll see the MAC address of your device. Now that’s all fine and dandy but I got a bit more curious and wanted to see a little more information. If you open up ‘bled112_scanner.py’ in your favorite text editor, scroll down to line 458 and uncomment this:

```
print ‘ ‘.join(disp_list)
```

…then run the script again, this time you’ll see a bit more information. You can check the help file of the script with ‘-h’ and it will tell you what each of these numbers represent, I’m going to jump over to the payload data. Breaking that down we have our two fields, flags and local name, then the values for each.

- FIELD: 020106
  - 02: Field Length (including the length byte)
  - 01: Field Type (flags)
  - 06: Field Value (bits 1 and 2 set) – General Discoverable, BR/EDR supported
- FIELD: 1309416E67656C2053656E736F72203035383532
  - 13: Field Length (including the length byte)
  - 09: Field Type (local name)
  - 416E67…: ASCII Value which translates to “Angel Sensor 05852” which is my serial number

Hopefully you didn’t run into any hiccups here and everything works as expected! Play around with it and happy coding!