Wireless & The IoT

Lab 5: Hardware! Packets! Beginning to play around with real BLE

Introduction

The purpose of today's lab is to introduce you to the nrf52840 Dongle, the Nordic Bluetooth ecosystem, the *littlest* bit of embedded programming, and generally just to play around with some real-world wireless.

The Assignment

I'll give an overview of the major points, but don't plan to talk for more than 60-120 seconds. Mostly, it's go through the lab and start figuring stuff out. Share some of the interesting stuff you found. Not a super formal write-up, just a 'show that you did this,' and hopefully, found something fun/interesting/etc.

What to submit?

Please use this document as a template, add your responses directly, and export it as a PDF to Gradescope. Folks are encouraged to collaborate as much as you like with others. If you work with others, please put everyone's name who worked together below. I believe I have also configured Gradescope to allow "group submission," so please submit to Gradescope as a group.

(your name**(S)** here)

Part 1: Setup nRF Connect

Nordic has a suite of *really* nice software tools[1] that help support experimentation with their hardware platforms. Not everything in the nRF Connect panel is supported by the nrf52840 dongle (and something that look like they wouldn't be, are; e.g. the "RSSI Viewer" works fine, despite saying it's for the nRF52832; sadly no direct test mode on the dongle).

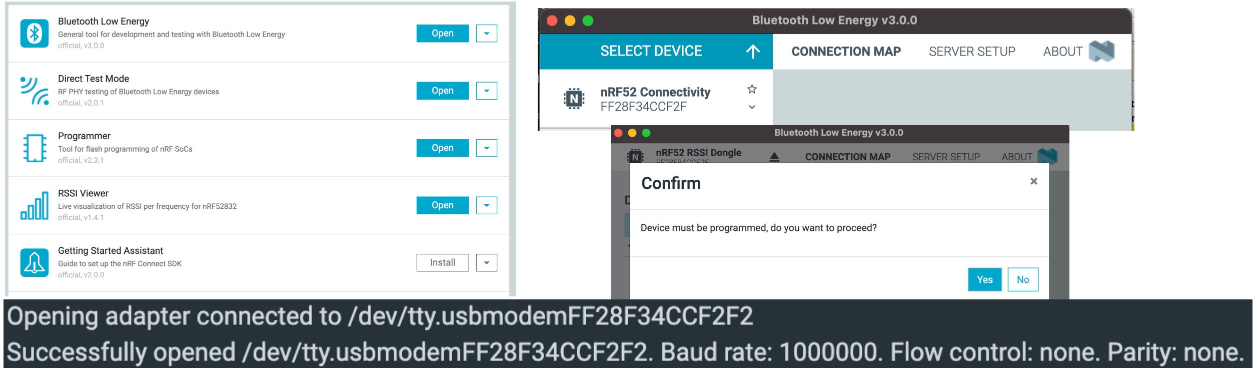
Download and install the nRF Connect for Desktop tools:
https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-desktop

While that's installing, go ahead and install the nRF Connect app on your phone too [it's just called 'nrf Connect for Mobile' probably easier to search, but here are links nonetheless]:

- https://apps.apple.com/us/app/nrf-connect-for-mobile/id1054362403
- https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp

By default, the app is just an empty shell that can install sub-apps. Go ahead and install the *Bluetooth Low Energy* app, the *Programmer*, and the *RSSI Viewer*. You can also install the *Direct Test Mode* app if you want, I'll bring a few boards that support that for folks to play with.

Once things are installed, plug in your dongle and fire up the *Bluetooth Low Energy* app. Not much happens until you select a device. It'll ask to program your device, which you should say yes to. Under the hood, a custom firmware image for this app is being installed. This firmware opens a serial link between the *Bluetooth Low Energy* app and the dongle, which the app uses to control the dongle and collect data. This basic concept of 'serial to app' is how all of the Nordic tools work. Notably, this circumvents the entirety of OS Bluetooth stacks. At no point does the dongle appear to be a Bluetooth peripheral to your OS [well, you *can* program it with firmware to do that; just not today], rather it's just a USB serial port.



Bluetooth Low Energy
General tool for development and testing with Bluetooth Low Energy
official, v3.0.0

Direct Test Mode
RF PHY testing of Bluetooth Low Energy devices
official, v2.0.1

Programmer
Tool for flash programming of nRF SoCs
official, v2.3.1

RSSI Viewer
Live visualization of RSSI per frequency for nRF52832
official, v1.4.1

Getting Started Assistant
Guide to set up the nRF Connect SDK
official, v2.0.0

Bluetooth Low Energy v3.0.0

nRF52 Connectivity
FF28F34CCF2F

Bluetooth Low Energy v3.0.0

nRF52 RSSI Dongle

Confirm

Device must be programmed, do you want to proceed?

```
Opening adapter connected to /dev/tty.usbmodemFF28F34CCF2F2
Successfully opened /dev/tty.usbmodemFF28F34CCF2F2. Baud rate: 1000000. Flow control: none. Parity: none.
```

Figures: nRF Connect app; drop-down to select device; Yes to programming; serial conn in log

Play around a little bit with *Bluetooth Low Energy* app. What devices do you see when you scan? Can you make your device advertise? Can you customize what your advertisement says? Can you see your dongle's advertisements on your phone? Can you the devices of other folks in class? Can you manipulate the RSSI of your device? What does the RSSI Viewer show?

[1] Insert "back in my day" grumbling here (but kinda really).

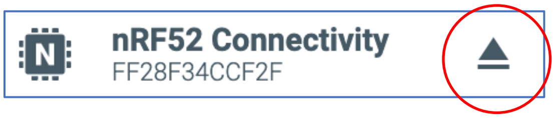
Part 1: What to Submit

Drop some screenshots / text of anything you found cool or interesting while playing around. Make sure to note a few device addresses of interest, they'll help with filtering in the next step.

Part 2: Sniffing BLE

Now, we're going to tie back to Lab 1 (!), and link our dongle to Wireshark. You should already/still have Wireshark installed, but if it's not on this machine, install Wireshark first.

Step 0: Disconnect from the apps you were using



First, grab a copy of the nRF Sniffer app:
https://www.nordicsemi.com/Products/Development-tools/nRF-Sniffer-for-Bluetooth-LE/Download

> ***Head's Up!*** This next step will erase the DFU off of your dongle. That means you'll no longer be able to program the device without an *external programmer*. (Unfortunately, I don't think Nordic has a firmware image of DFU + sniffer; though as you'll see in the Programmer, there's plenty of space…) I'll bring a few programmers to the lab for folks to reflash their devices to stock firmware if you want. This post gives more details about what's going on under the hood: https://devzone.nordicsemi.com/guides/short-range-guides/b/getting-started/posts/nrf52840-dongle-programming-tutorial

Open the *Programmer* app, and drag the `/hex/sniffer_nrf52840dongle_nrf52840_4.1.0.hex` precompiled firmware over for programming.

The sniffer receiver is written in Python. You'll need Python3 and `pyserial >= 3.5`. If you don't have Python3, follow the python install guide. For pyserial, you can run `python3 -m pip install pyserial` once Python is installed.

First, a quick sanity check that things are working:

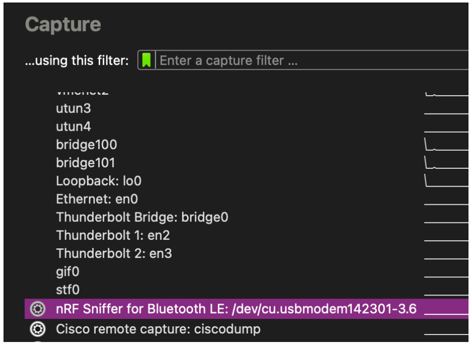
```
  $ cd extcap/
  $ cp ../doc/example.py .
  $ python3 example.py
Could not find device  ← this may print at first :shrug:
Sniffer Device List: [Bluetooth LE device """" ([67, 45, 61, 114, 35, 213, 1]),
Bluetooth LE device """" ([2, 80, 215, 201, 50, 109, 1]), Bluetooth LE device """"
([77, 62, 140, 133, 107, 159, 1]), Bluetooth LE device """" ([111, 48, 65, 15, 124,
253, 1]), Bluetooth LE device """" ([253, 36, 251, 90, 19, 135, 1]), Bluetooth LE
device """" ([105, 27, 16, 56, 55, 211, 1]), Bluetooth LE device """" ([85, 144, 143,
180, 73, 32, 1]), Bluetooth LE device """" ([226, 45, 78, 106, 96, 29, 1])]
inConnection False
currentConnectRequest None
packetsInLastConnection None
nPackets 4129
```

Next, we need to install the external capture device 'extcap' to wireshark. For this, you need to know where Wireshark was installed on your machine. Inside the wireshark install is an `extcap` folder that we'll need to add this new sniffer to:

```
  $ cd /Applications/Wireshark.app/Contents/MacOS/extcap
  $ cp -r ~/Downloads/nrf_sniffer_for_bluetooth_le_4.1.0/extcap/* .
```

Then, you can fire up Wireshark (or, if you already have, Capture Menu → Refresh Interfaces).

If everything went well, you now have a new capture interface!

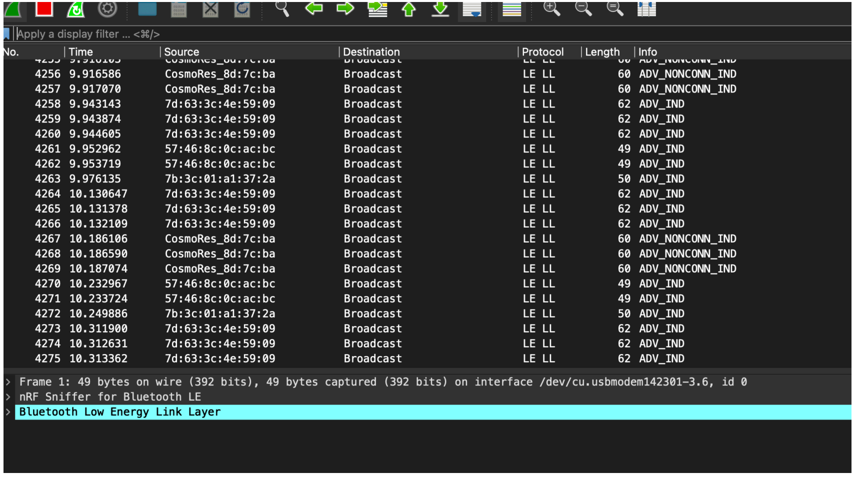


Capture

...using this filter: Enter a capture filter ...

```
utun3
utun4
bridge100
bridge101
Loopback: lo0
Ethernet: en0
Thunderbolt Bridge: bridge0
Thunderbolt 1: en2
Thunderbolt 2: en3
gif0
stf0
nRF Sniffer for Bluetooth LE: /dev/cu.usbmodem142301-3.6
Cisco remote capture: ciscodump
```

Double click, and start capturing!

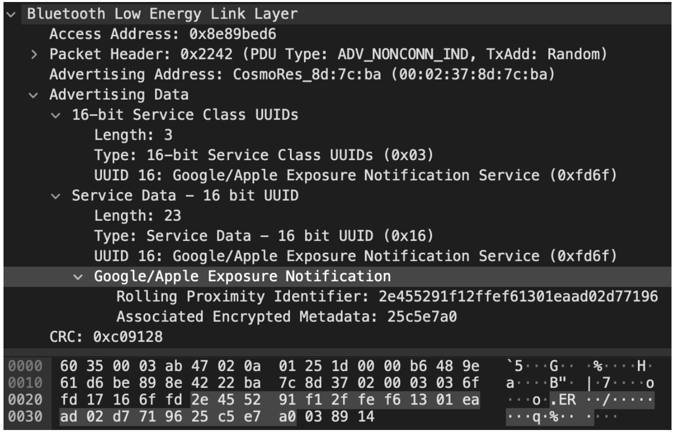


Apply a display filter ... <⌘/>

```
No.   Time       Source             Destination  Protocol  Length  Info
 4256 9.916586   CosmoRes_8d:7c:ba  Broadcast    LE LL         60  ADV_NONCONN_IND
 4257 9.917070   CosmoRes_8d:7c:ba  Broadcast    LE LL         60  ADV_NONCONN_IND
 4258 9.943143   7d:63:3c:4e:59:09  Broadcast    LE LL         62  ADV_IND
 4259 9.943874   7d:63:3c:4e:59:09  Broadcast    LE LL         62  ADV_IND
 4260 9.944605   7d:63:3c:4e:59:09  Broadcast    LE LL         62  ADV_IND
 4261 9.952962   57:46:8c:0c:ac:bc  Broadcast    LE LL         49  ADV_IND
 4262 9.953719   57:46:8c:0c:ac:bc  Broadcast    LE LL         49  ADV_IND
 4263 9.976135   7b:3c:01:a1:37:2a  Broadcast    LE LL         50  ADV_IND
 4264 10.130647  7d:63:3c:4e:59:09  Broadcast    LE LL         62  ADV_IND
 4265 10.131378  7d:63:3c:4e:59:09  Broadcast    LE LL         62  ADV_IND
 4266 10.132109  7d:63:3c:4e:59:09  Broadcast    LE LL         62  ADV_IND
 4267 10.186106  CosmoRes_8d:7c:ba  Broadcast    LE LL         60  ADV_NONCONN_IND
 4268 10.186590  CosmoRes_8d:7c:ba  Broadcast    LE LL         60  ADV_NONCONN_IND
 4269 10.187074  CosmoRes_8d:7c:ba  Broadcast    LE LL         60  ADV_NONCONN_IND
 4270 10.232967  57:46:8c:0c:ac:bc  Broadcast    LE LL         49  ADV_IND
 4271 10.233724  57:46:8c:0c:ac:bc  Broadcast    LE LL         49  ADV_IND
 4272 10.249886  7b:3c:01:a1:37:2a  Broadcast    LE LL         50  ADV_IND
 4273 10.311900  7d:63:3c:4e:59:09  Broadcast    LE LL         62  ADV_IND
 4274 10.312631  7d:63:3c:4e:59:09  Broadcast    LE LL         62  ADV_IND
 4275 10.313362  7d:63:3c:4e:59:09  Broadcast    LE LL         62  ADV_IND

> Frame 1: 49 bytes on wire (392 bits), 49 bytes captured (392 bits) on interface /dev/cu.usbmodem142301-3.6, id 0
> nRF Sniffer for Bluetooth LE
> Bluetooth Low Energy Link Layer
```

Anything timely around?



```
v Bluetooth Low Energy Link Layer
    Access Address: 0x8e89bed6
  > Packet Header: 0x2242 (PDU Type: ADV_NONCONN_IND, TxAdd: Random)
    Advertising Address: CosmoRes_8d:7c:ba (00:02:37:8d:7c:ba)
  v Advertising Data
    v 16-bit Service Class UUIDs
        Length: 3
        Type: 16-bit Service Class UUIDs (0x03)
        UUID 16: Google/Apple Exposure Notification Service (0xfd6f)
    v Service Data - 16 bit UUID
        Length: 23
        Type: Service Data - 16 bit UUID (0x16)
        UUID 16: Google/Apple Exposure Notification Service (0xfd6f)
      v Google/Apple Exposure Notification
          Rolling Proximity Identifier: 2e455291f12ffef61301eaad02d77196
          Associated Encrypted Metadata: 25c5e7a0
    CRC: 0xc09128

0000  60 35 00 03 ab 47 02 0a  01 25 1d 00 00 b6 48 9e   `5···G·· ·%····H·
0010  61 d6 be 89 8e 42 22 ba  7c 8d 37 02 00 03 03 6f   a····B"· |·7····o
0020  fd 17 16 6f fd 2e 45 52  91 f1 2f fe f6 13 01 ea   ···o·.ER ··/·····
0030  ad 02 d7 71 96 25 c5 e7  a0 03 89 14               ···q·%·· ····
```

Play around a little bit with the Wireshark captures. Can you identify any packets that are being sent as your devices? Can you identify packets from other folks in class? Look at the protocol breakdown for some advertisements, can you see the major fields we talked about in lecture? Any advertisements you can get meaningful data from (maybe ones others send)?

Part 2: What to submit

Drop some screenshots / text of anything you found cool or interesting while playing around. Were you surprised at the volume of traffic? Any interesting devices you were able to ID?