Lab 2: BLE Scanning

Introduction

The purpose of today's lab is to enable you to scan and explore BLE communication. We'll take our first steps loading code onto the nRF52840DK and thinking about the Nordic software ecosystem as well.

Goals

- Set up nRF Connection on your system
- Enable BLE Scanning with the nRF52840DK and Wireshark
- Explore BLE advertisements in the nearby environment

Equipment

- Computer
- nRF52840DK + USB cable
- Smartphone (optional)

Partners

- This lab should be done individually

Submission

- Write your answers up and submit a PDF to Gradescope.

  Remember: I'm not looking for a formal lab report. Just your answers in any format that makes sense. The goal is to prove that you did the lab and spent some time thinking about it.

1. Install nRF Connect for Desktop

Nordic has a suite of *really* nice software tools that help support experimentation with their hardware platforms.The app will work on Windows, MacOS, and Linux. and it uses your nRF52840DK hardware to actually interact with devices.

Not everything in the nRF Connect panel is supported by the nrf52840DK (and some things that look like they wouldn't be supported, are; e.g. the "RSSI Viewer" works fine, despite saying it's for the nRF52832).

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

While that's installing, you can optionally 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*.

The *Bluetooth Low Energy* app functions very similarly to the nRF Connect app on your phone. Both allow you to scan for nearby devices, connect to them, and investigate services they provide. Play around for a bit and see what's nearby. You might be surprised by what you find.

When you finish using an app, be sure to disconnect from it:

TASK: None. Continue to the next section.

2. Integrate BLE Scanning into Wireshark

Next, we're going to add an external capture source to Wireshark that allows it to sniff BLE communication by using the nRF52840DK. The full guide that we're following is here: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fug_sniffer_ble%2FUG%2Fsniffer_ble%2Finstalling_sniffer.html

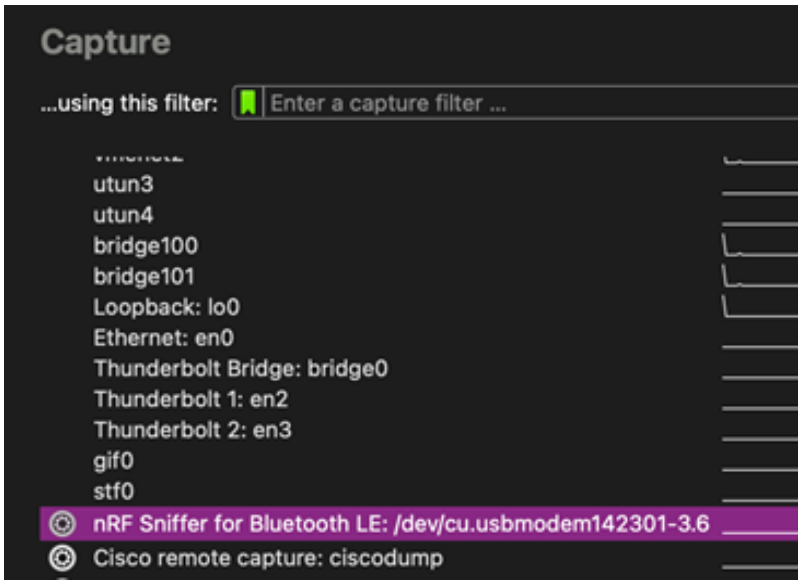
1. Get a copy for the sniffer ZIP: https://www.nordicsemi.com/Products/Development-tools/nrf-sniffer-for-bluetooth-le/download

2. Open the *Programmer* app, and drag the `/hex/sniffer_nrf52840dongle_nrf52840_4.1.0.hex` precompiled firmware over for programming. Then write that firmware to your nRF52840DK.

3. 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. This *does* work on Windows with a little bit of effort.

4. We need to copy over the "extcap" stuff to the correct folder so Wireshark can find it. I can't write better instructions than Nordic already did: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fug_sniffer_ble%2FUG%2Fsniffer_ble%2Finstalling_sniffer.html (Note: we've already handled the python requirements from step 1 by installing `pyserial`)

5. Finally, make sure you have your nRF52840DK reprogrammed and connected over USB, then either restart Wireshark or go to "Capture Menu -> Refresh Interfaces". You should now see a new capture interface: "nRF Sniffer for Bluetooth LE".

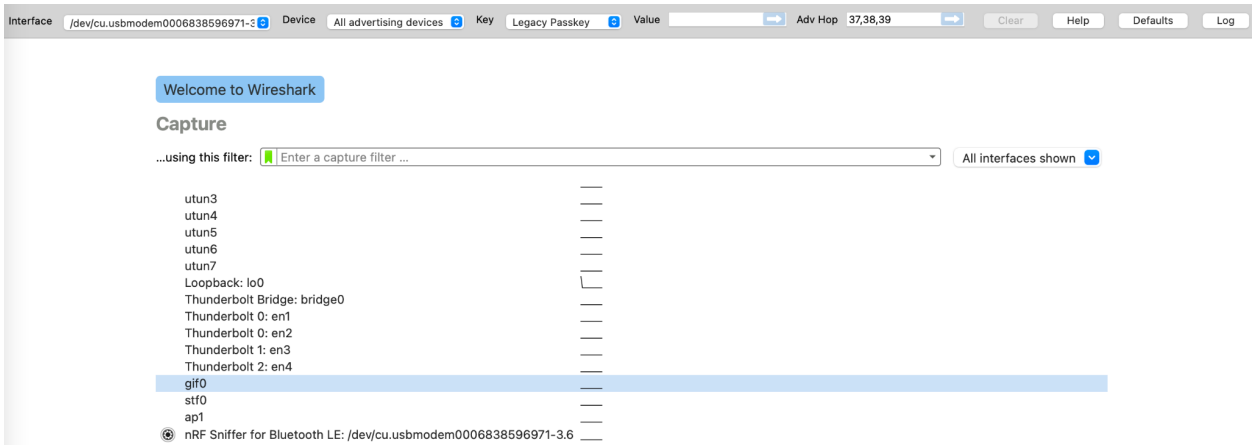


Double-click it to start capturing!

Lots of things can go wrong here! Be sure that you're following all the steps and didn't skip anything. Also check the troubleshooting steps here: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fug_sniffer_ble%2FUG%2Fsniffer_ble%2Finstalling_sniffer.html

If you're still having problems, definitely reach out and I'm happy to help!

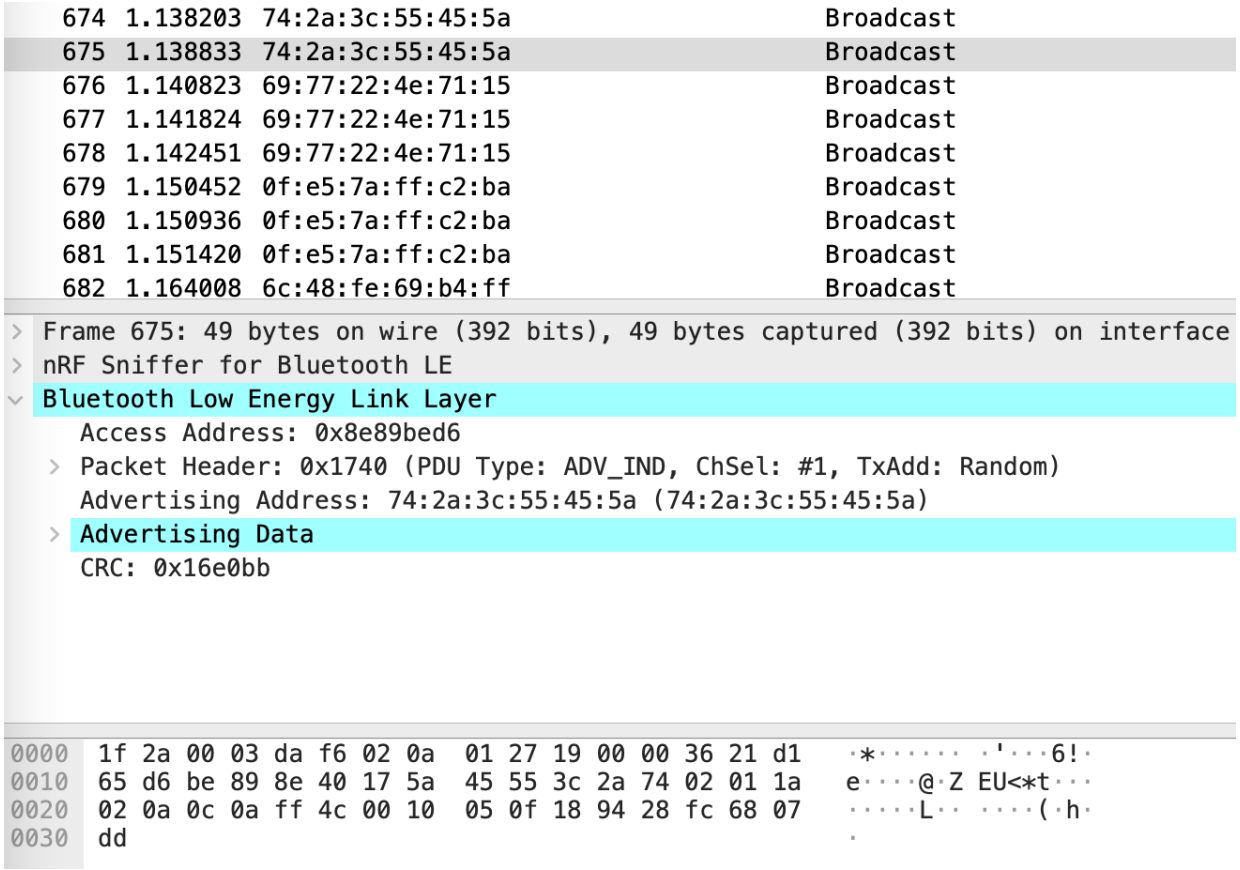
TASK: None. Continue to the next section.
Successfully connected screenshot.



3. Investigating BLE Advertisements

Now that you've (hopefully) got the Wireshark external capture working, let's investigate some BLE packets! Run wireshark and collect packets for a few seconds. Then take a look at the packets you received and answer a few questions. Include screenshots as makes sense.

1. **TASK:** How many transmissions do you see in one second?
   I saw more than 2 thousand transmissions.
2. **TASK:** Pick a received packet and explain the meaning of all of the bytes of it.



Here is a transmission of an advising channel. The access address is a part of the link layer package and the device that receives the message should match this address before listening to it. The packet header is the two bytes header of the BLE packet as explained in class as figure 1 shows. The advertising address is a unique identifier that helps recipients of the advertisement to know the identity of the device. The advertising data the CRC shows in figure 1 contains the information that wants to be advertised to other devices.

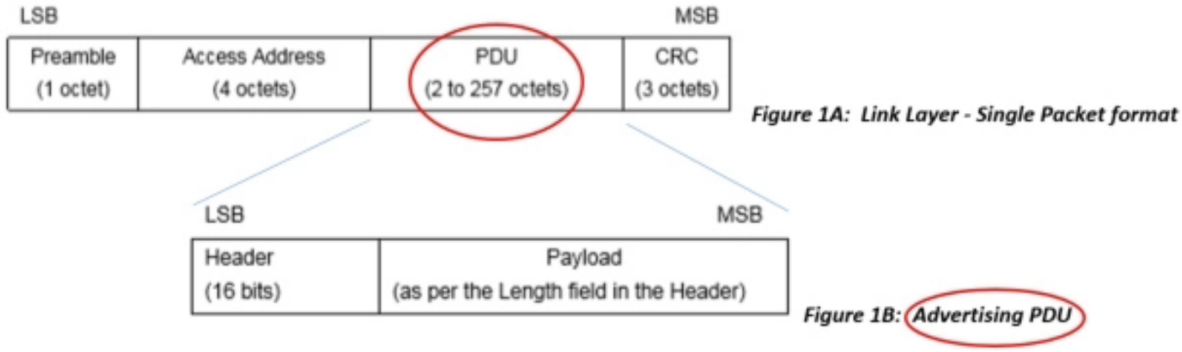


Figure 1

3. **TASK:** Identify five different AD Types across various advertisements. Here is the PDU type to name the form I found. (https://www.novelbits.io/bluetooth-5-advertisements/)

PDU Type	PDU Name	Channel	Permitted PHYs		
			LE 1M	LE 2M	LE Coded
0000b	ADV_IND	Primary Advertising	•		
0001b	ADV_DIRECT_IND	Primary Advertising	•		
0010b	ADV_NONCONN_IND	Primary Advertising	•		
0011b	SCAN_REQ	Primary Advertising	•		
	AUX_SCAN_REQ	Secondary Advertising	•	•	•
0100b	SCAN_RSP	Primary Advertising	•		
0101b	CONNECT_IND	Primary Advertising	•		
	AUX_CONNECT_REQ	Secondary Advertising	•	•	•
0110b	ADV_SCAN_IND	Primary Advertising	•		

PDU Type	PDU Name	Channel	Permitted PHYs		
			LE 1M	LE 2M	LE Coded
0111b	ADV_EXT_IND	Primary Advertising	•		•
	AUX_ADV_IND	Secondary Advertising	•	•	•
	AUX_SCAN_RSP	Secondary Advertising	•	•	•
	AUX_SYNC_IND	Secondary Advertising	•	•	•
	AUX_CHAIN_IND	Secondary Advertising	•	•	•
1000b	AUX_CONNECT_RSP	Secondary Advertising	•	•	•
All other values	Reserved for Future Use				

Figure 2

Figure 3: ADV_IND type (0000 with primary advising)

Figure 3: ADV_NONCONN_IND type (0010 with primary advising)

Figure 4: SCAN_REQ type (0011 with primary advising)

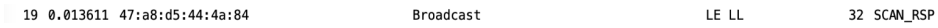
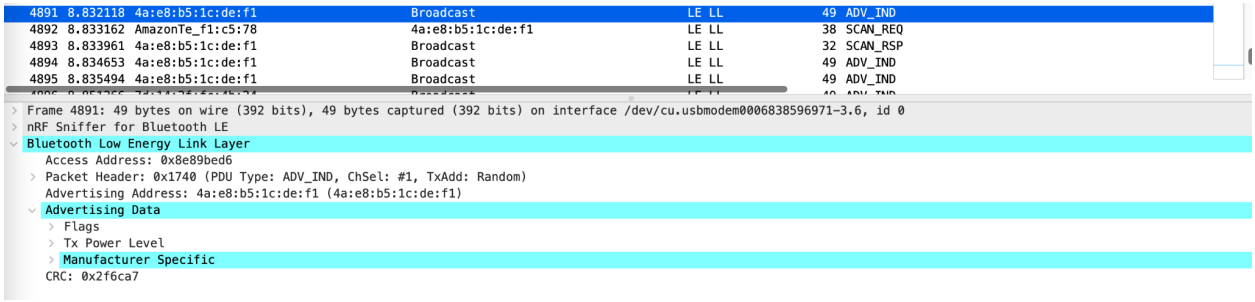


Figure 5: SCAN_RSP type (0100 with primary advising)

Figure 6: AUX_CONNECT_RSP type (1000 with secondary advising)

4. **TASK:** Identify fields in an advertisement and corresponding scan response for one device.



This is what I capture where the destination of the SCAN_REQ is the same as the ADV_IND and SCAN_RSP source. Both with address 4a:e8:b5:1c:de:f1.