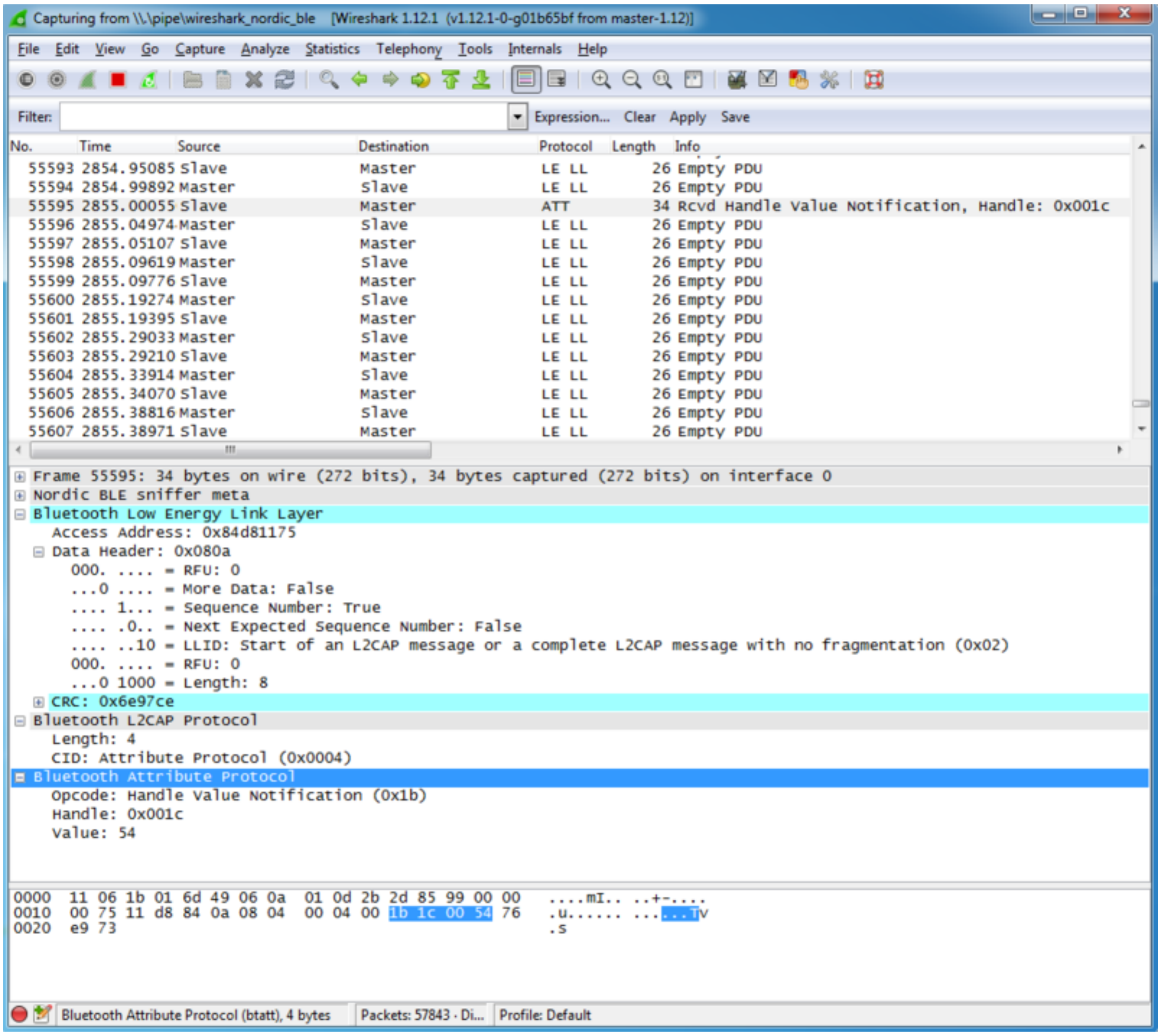
Introducing the Adafruit Bluefruit LE Sniffer

Created by Kevin Townsend



https://learn.adafruit.com/introducing-the-adafruit-bluefruit-le-sniffer

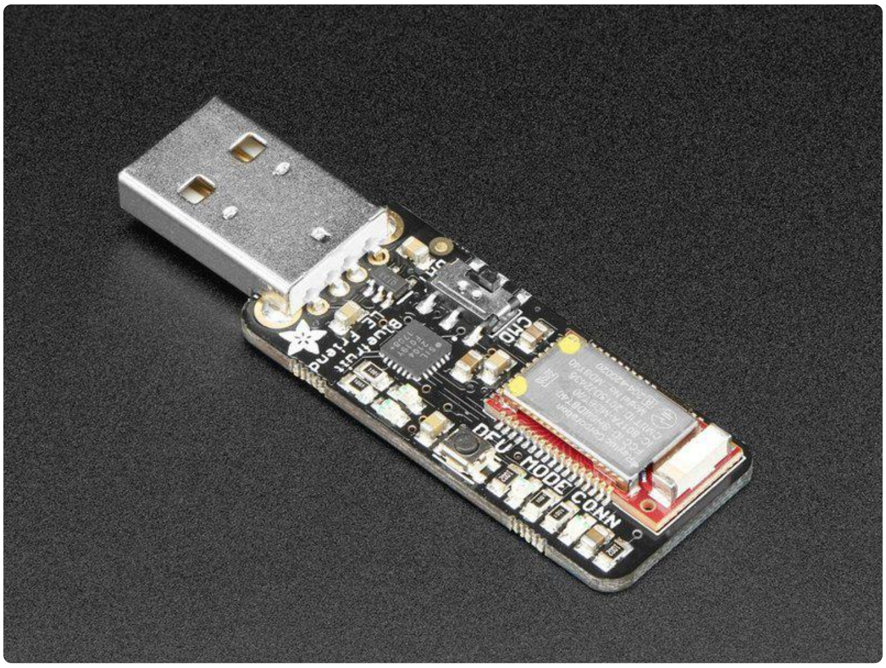
Last updated on 2023-08-29 02:42:34 PM EDT

Table of Contents

Introduction

Using a special firmware image provided by Nordic Semiconductors and the open source network analysis tool Wireshark, the Bluefruit LE Sniffer () can be used as a low cost Bluetooth Low Energy sniffer.

NOTE: This product can only be used to sniff Bluetooth Low Energy devices. It will not work with classic Bluetooth devices or transactions.

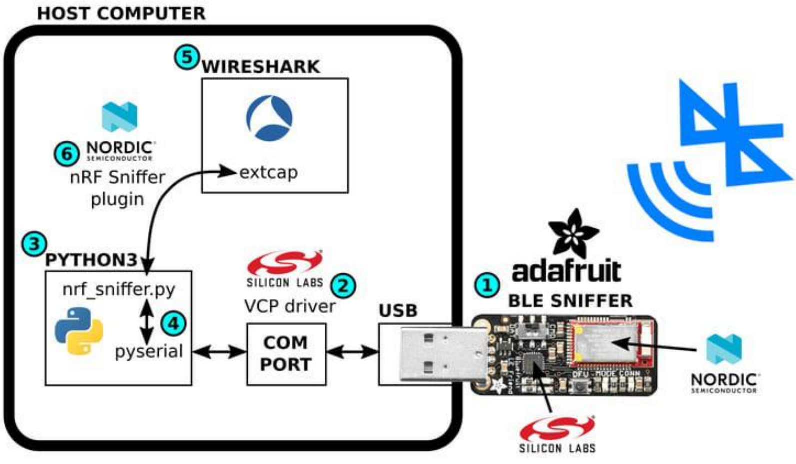


Since nRF-Sniffer is a passive solution that is simply scanning packets over the air, there is the possibility of missing packets using this tool (or any other passive sniffing solution). In order to capture as many packets as possible, be sure to run the sniffer on a USB bus that isn't busy and avoid running it in a virtual machine since this can introduce significant latency over USB.

Using with Sniffer V2 and Python3

Once things are all setup, usage is fairly easy. However, there are numerous separate items that need to be installed and configured. So the initial setup can be a bit cumbersome. We'll go through each step, but it can also help to have a general understanding of the overall setup.

Here's a simplified diagram of the setup:



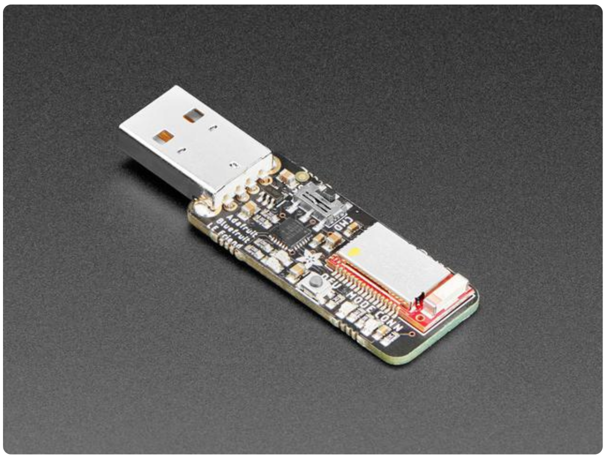
Here's a summary of all the parts needed:

1. The actual BLE sniffing hardware. This guide uses the Adafruit Bluefruit LE Sniffer with V2 firmware ().
2. The BLE Sniffer uses a Silicon Labs CP2104 to provide USB to serial conversion. In order for this to show up as a COM port, the Silicon Labs Virtual COM Port driver () is needed.
3. The BLE sniffing plugin uses Python ().
4. To talk to the virtual com port from Python, the pyserial module () needs to be installed.
5. Wireshark () is the main software front end used to facilitate BLE sniffing and decoding.
6. To talk to the BLE sniffer from Wireshark, the Nordic Semiconductor nRF Sniffer for BLE () plugin in is used.

These parts come from numerous different sources - at least 5 different vendors are shown in the diagram above. So this will be quite the journey. Here we go...

BLE Sniffer Hardware

You'll need one of these:



[Bluefruit LE Sniffer - Bluetooth Low Energy (BLE 4.0) - nRF51822](https://www.adafruit.com/product/2269)

Interested in learning how Bluetooth Low Energy works down to the packet level? Debugging your own BLE hardware, and trying to spot where something is going wrong? Or maybe you're...

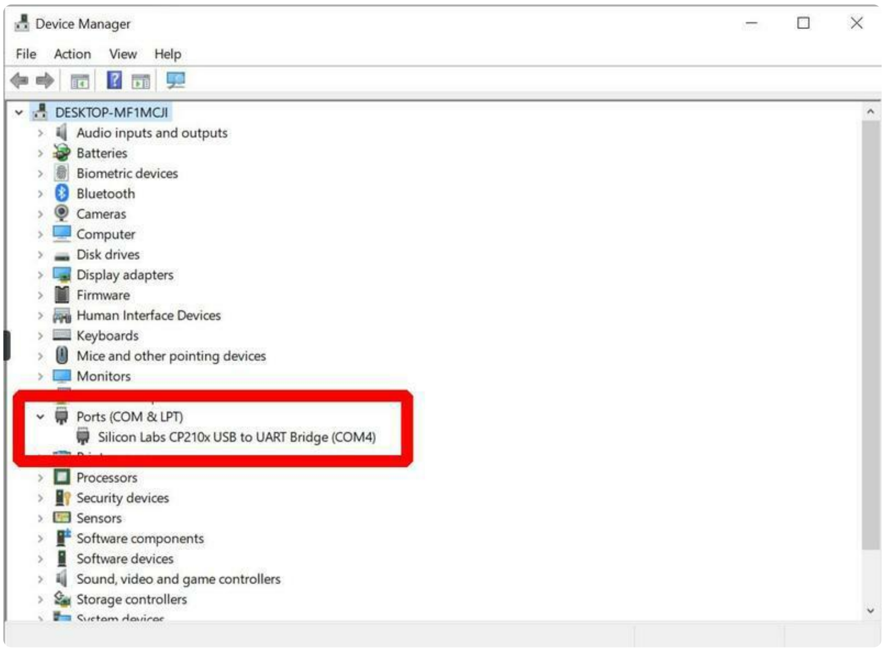
https://www.adafruit.com/product/2269

Silicon Labs VCP Driver

This driver allows the CP2104 chip on the Adafruit BLE Sniffer to show up as a COM port on your PC.

Silicon Labs VCP Driver

Once installed, a COM port should show up on your PC when the Adafruit BLE Sniffer is plugged into a USB port. It should have CP210x in the name.



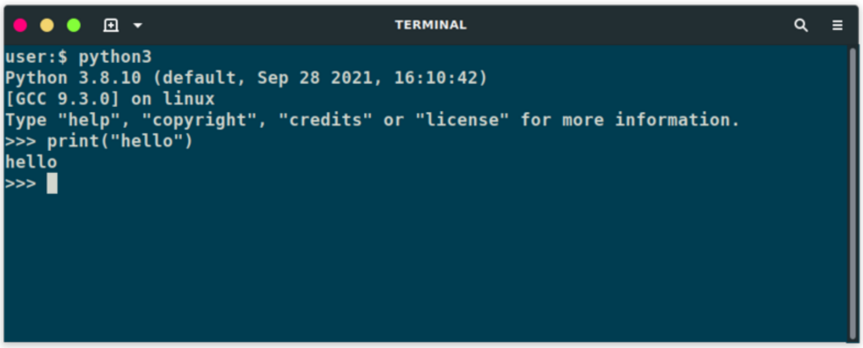
This check does not require any of the other software components we install later. So if a COM port is not showing up at this point, do not proceed further until determining why.

Python 3

If Python 3 is not already installed on your system, go to the Python main page to learn how to download and install it for your specific system:

Python

It should now be possible to launch Python and run some simple commands:



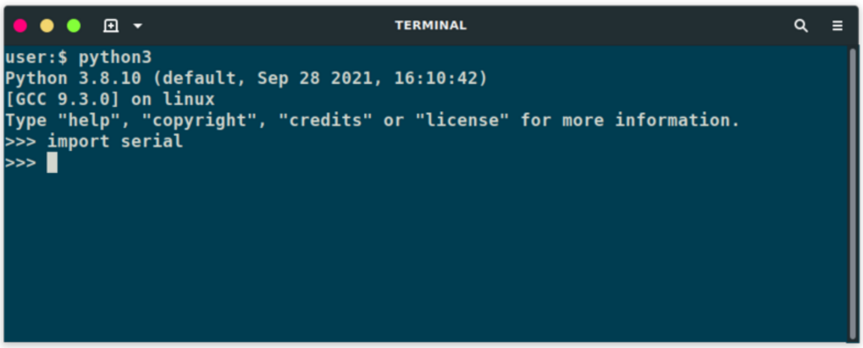
On Windows, try using `py` to launch Python.

Python Serial Support

To provide access to the COM port, install the pyserial package.

pyserial

It should now be possible to launch Python and import the pyserial package:



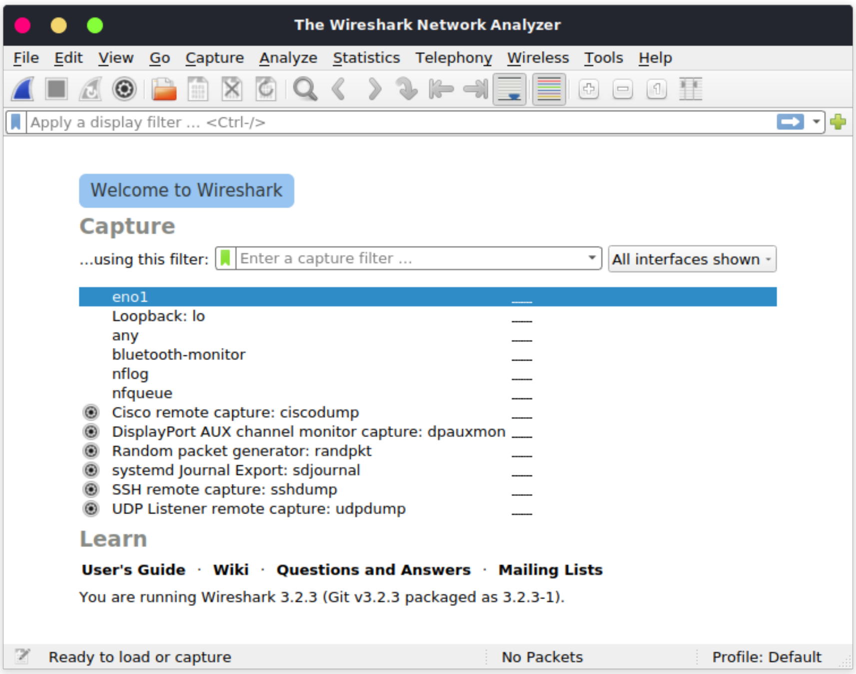
NOTE: the import is actually `serial`, not pyserial.

Install Wireshark

Go to the Wireshark main page to learn how to download and install Wireshark for your specific system:

Wireshark

Once complete, it should be possible to run Wireshark and at least get the start screen:



Install BLE Sniffer Plugin

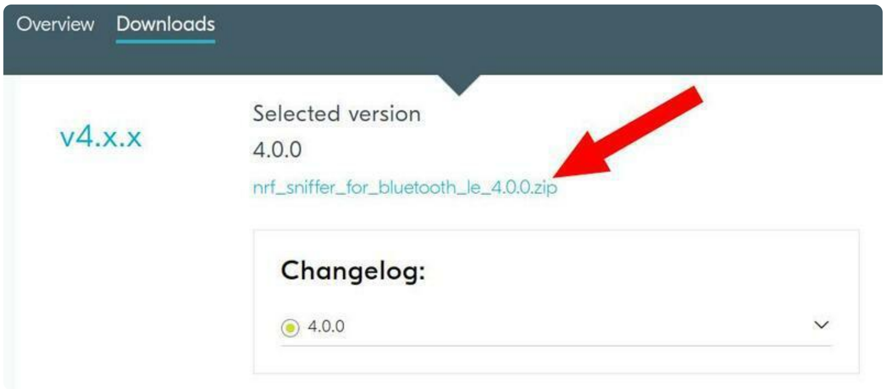
OK, finally, the thing we actually care about. The thing that will let us talk to the Adafruit BLE Sniffer and do some actual BLE sniffing. Let's download and install that BLE sniffing plugin!

Download Plugin from Nordic

Start by downloading the nRF Sniffer for BLE package from Nordic Semiconductor:

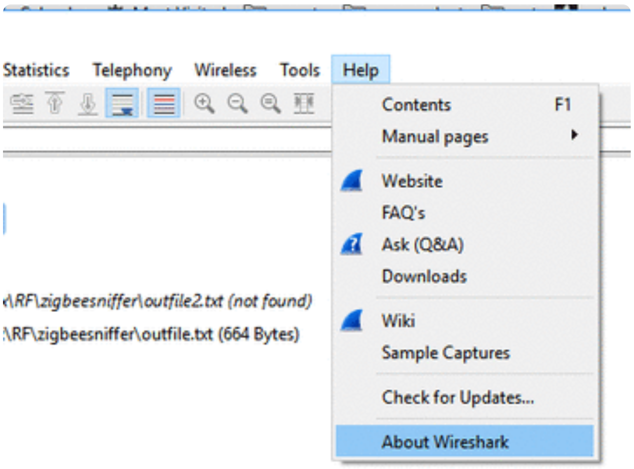
nRF Sniffer for Bluetooth LE

This will be a ZIP file. At the time of this guide, the version is 4.0.

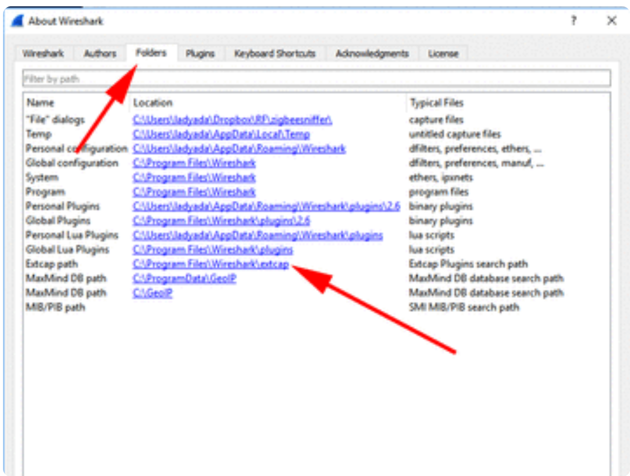


Determine Wireshark Plugin Folder Location (extcap)

We need to install items from the ZIP file downloaded from Nordic into a specific Wireshark folder location. This location is different on different systems. To determine it for your system, do this:



Open Wireshark, in the Help menu select About wireshark



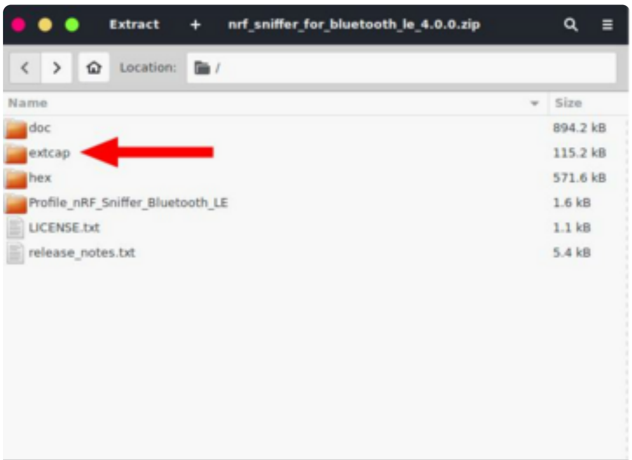
In the Folders tab, find the extcap path

We'll refer to this folder location as the Wireshark extcap folder.

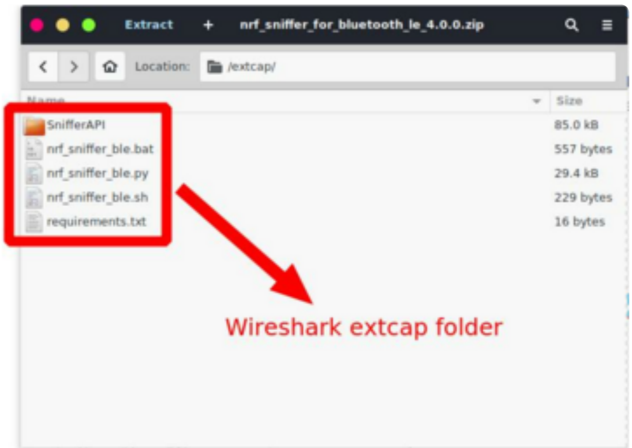
Install BLE Sniffer Plugin into Wireshark

To install the plugin, simply copy the files shown below from the ZIP downloaded from Nordic into the Wireshark extcap folder location determined above.

Open the ZIP file downloaded from Nordic:



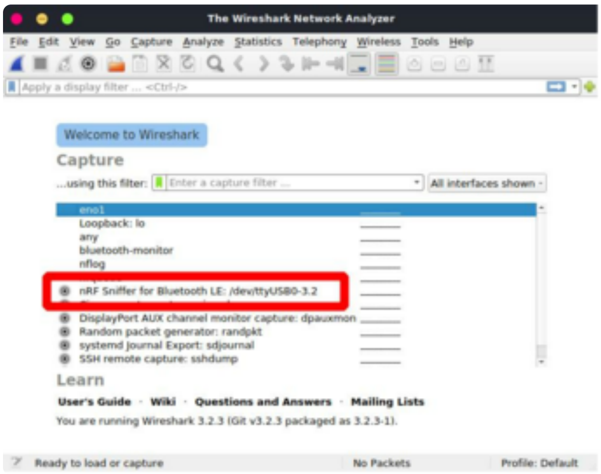
We only need the contents of the extcap folder from the ZIP file.



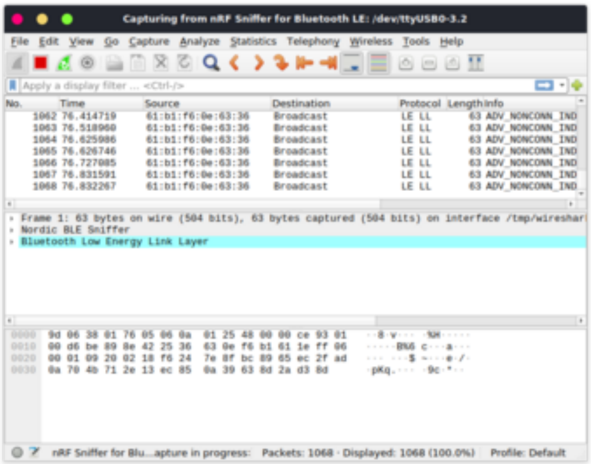
Extract and copy all of the contents of the extcap folder to the Wireshark extcap folder location.

Final Check and Test Capture

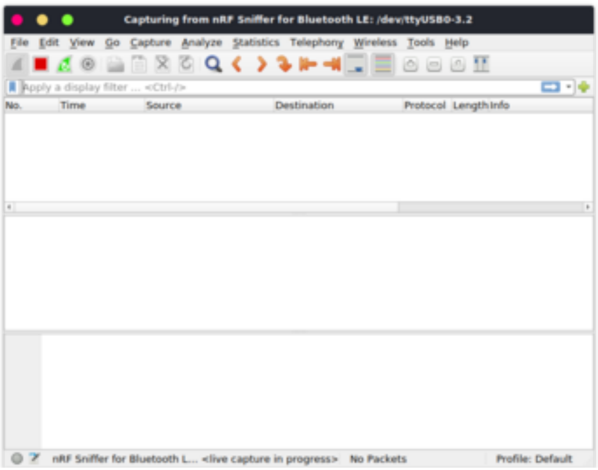
OK, now we can test things out with some real actual BLE sniffing! woot!



Plug in the Adafruit BLE Sniffer.
Launch Wireshark.
The sniffer should show up under the available capture devices.



Double click on the sniffer capture device.
This will open the device and start capturing.
If there is BLE traffic, it will be seen right away.



If there is no BLE traffic, it will look like this.
Note the device has opened properly and is sniffing, there's just nothing to be seen.

MacOS device names do not parse correctly with the 4.1.1 extcap/nrf_sniffer_ble.py script. This currently requires a small change of interface.split to interface.rsplit to parse correctly. See forum post for additional details.

https://forums.adafruit.com/viewtopic.php?t=202787

```
interface, extcap_version = interface.split('-')
```

```
interface, extcap_version = interface.rsplit('-', 1)
```

Next Steps

Once everything is working as shown above, you are ready to move on to working with these BLE packets.

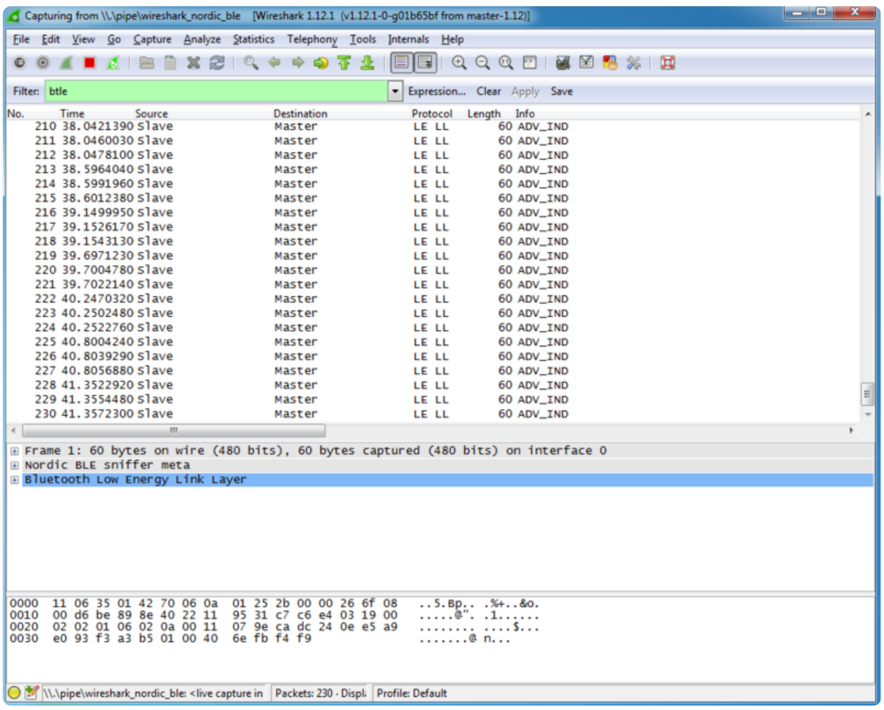
Now go here to learn how to look at BLE packets with Wireshark

Working with Wireshark

This page will work with both V1 and V2 sniffer firmware, once you've got the software installed

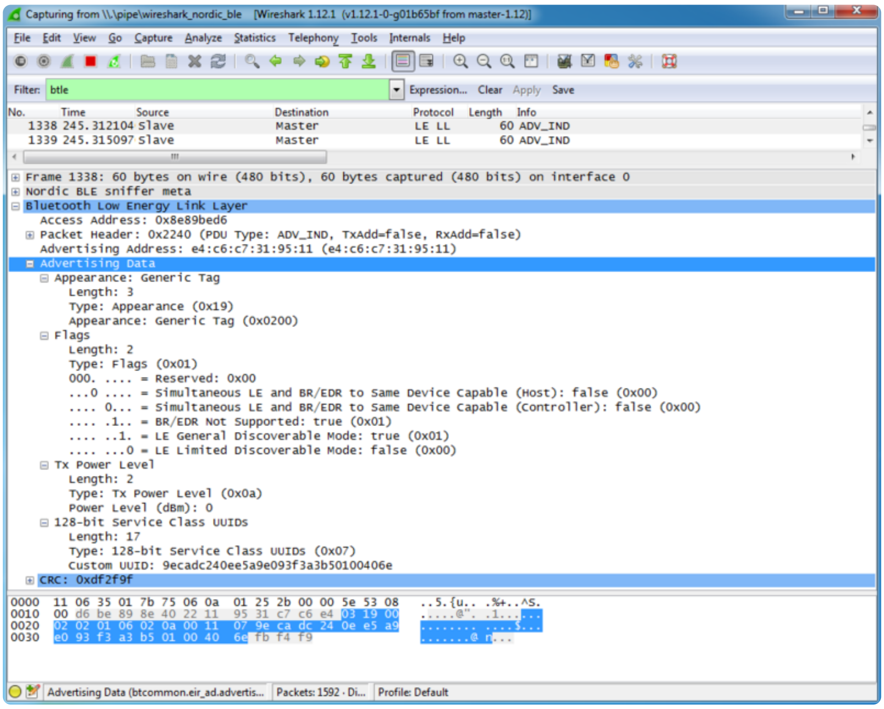
Working with Wireshark

Once Wireshark has loaded, you should see the advertising packets streaming out from the selected BLE device at a regular interval, as shown in the image below:



One of the key benefits of WireShark as an analysis tool is that it understands the raw packet formats and provides human-readable displays of the raw packet data.

The main way to interact with BLE data packets is to select one of the packets in the main window, and then expand the Bluetooth Low Energy Link Layer treeview item in the middle of the UI, as shown below:



Clicking on the Advertising Data entry in the treeview will highlight the relevant section of the raw payload at the bottom of the screen, but also provides human readable information about the payload that can save you a lot of time trying to debug or reverse engineer a device.

We can see, for example, that the device is advertising itself as a Bluetooth Low Energy only device ('BR/EDR Not Supported'), with a TX Power Level of 0dBm, and a single service is being advertised using a 128-bit UUID (the UART service in this case).

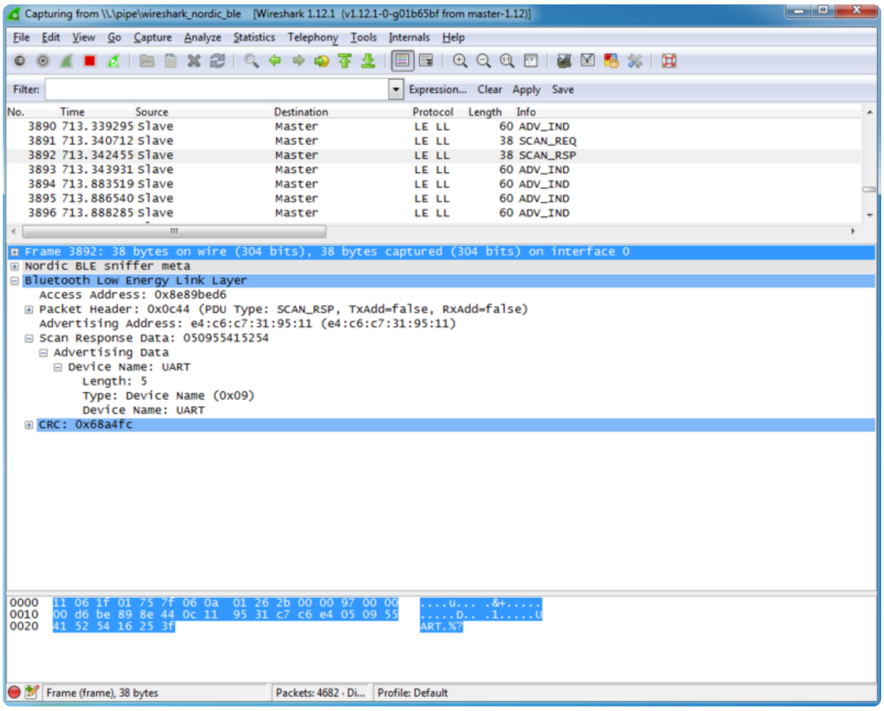
Capturing Exchanges Between Two Devices

If you wish to sniff data being exchanged between two BLE devices, you will need to establish a connection between the original device we selected above and a second BLE device (such as an iPhone or an Android tablet with BLE capabilities).

The nRF-Sniffer firmware is capable is listening the all of the exchanges that happen between these devices, but can not connect with a BLE peripheral or central device itself (it's a purely passive device).

Scan Response Packets

If you open up nRF UART on an Android or iOS device, and click the Connect button, the phone or tablet will start scanning for devices in range. One of the side effects of this scanning process is that you may spot a new packet in Wireshark on an irregular basis, the 'SCAN_REQ' and 'SCAN_RSP' packets:



The Scan Response is an optional second advertising packet that some Bluetooth Low Energy periperhals use to provide additional information during the advertising phase. The normal mandatory advertising packet is limited to 31 bytes, so the Bluetooth SIG includes the possibility to request a second advertising payload via the Scan Request.

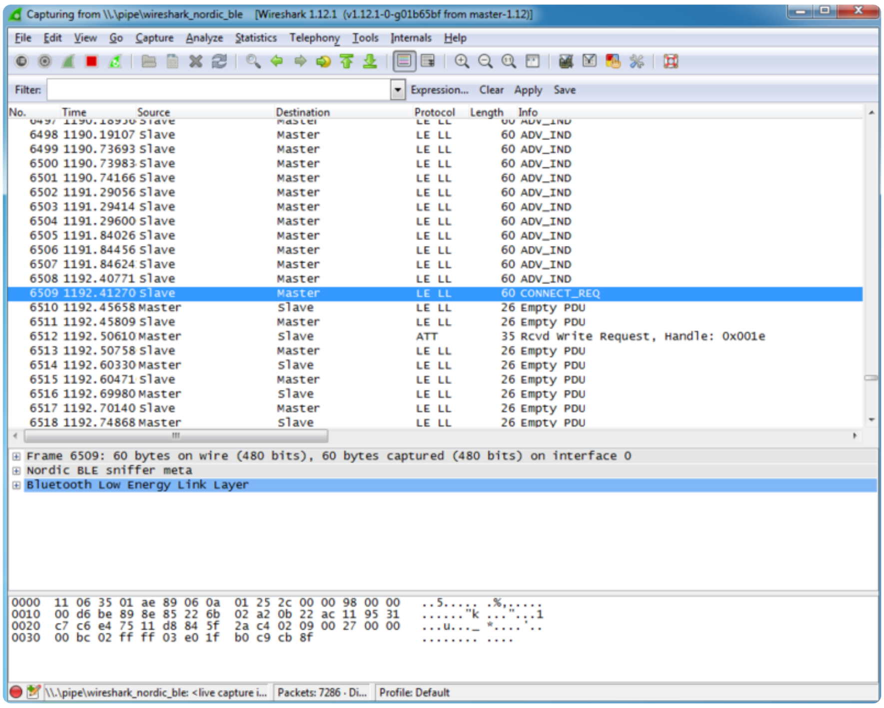
You can see both of these transactions in the image above, and the Device Name that is included in the Scan Response payload (since the 128-bit UART Service UUID takes up most of the free space in the main advertising packet).

For more information on Scan Responses and the advertising process in Bluetooth Low Energy see our [Introduction to Bluetooth Low Energy Guide]().

Connection Request

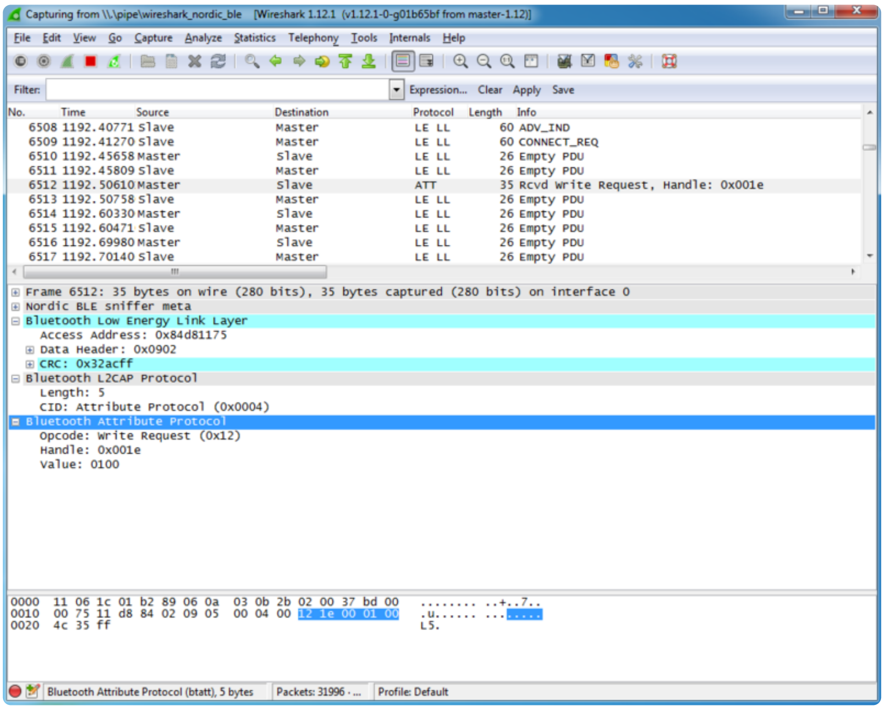
Once we click on the UART device in nRF UART, the two device will attempt to connect to each other by means of a Connection Request, which is initiated by the central device (the phone or tablet).

We can see this CONNECT_REQ in the timeline in the image below:



Write Request

Once the connection has been established, we can see that the nRF UART application tries to write data to the BLEFriend via a Write Request to handle '0x001E' (which is the location of an entry in the attribute table since everything in BLE is made up of attributes).



What this write request is trying to do is enable the 'notify' bit on the UART service's TX characteristic () (0x001E is the handle for the CCCD or 'Client Characteristic Configuration Descriptor ()'). This bit enables an 'interrupt' of sorts to tell the BLEFriend that we want to be alerted every time there is new data available on the characteristic that transmits data from the BLEFriend to the phone or tablet.

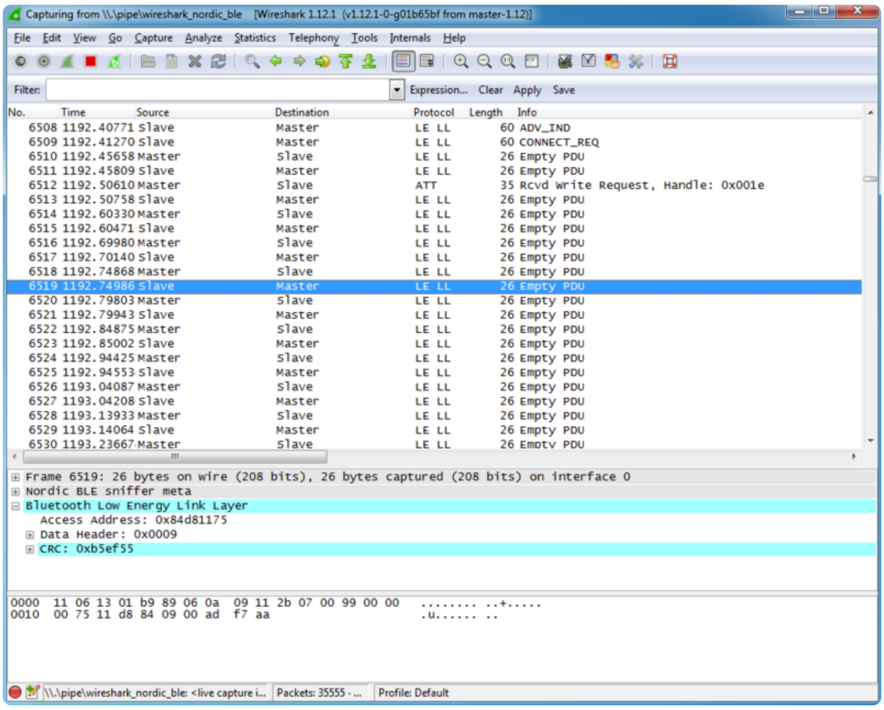
Regular Data Requests

At this point you will start to see a lot of regular Empty PDU requests. This is part of the way that Bluetooth Low Energy works.

Similar to USB, all BLE transaction are initiated by the bus 'Main', which is the central device (the tablet or phone).

In order to receive data from the bus secondary (the peripheral device, or the BLEFriend in this particular case) the central device sends a 'ping' of sorts to the peripheral at a delay known as the 'connection interval' (not to be confused with the one-time connection highlighted earlier in this tutorial).

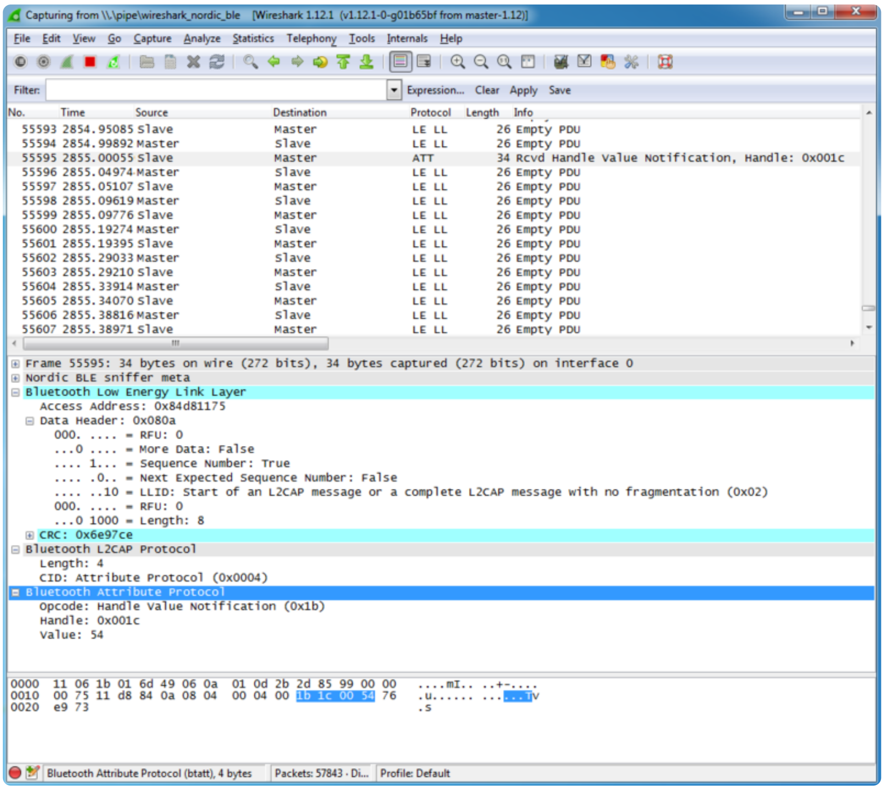
We can see pairs of transaction that happen at a reasonably consistent interval, but no data is exchanged since the BLEFriend (the peripheral) is saying 'sorry, I don't have any data for you':



Notify Event Data

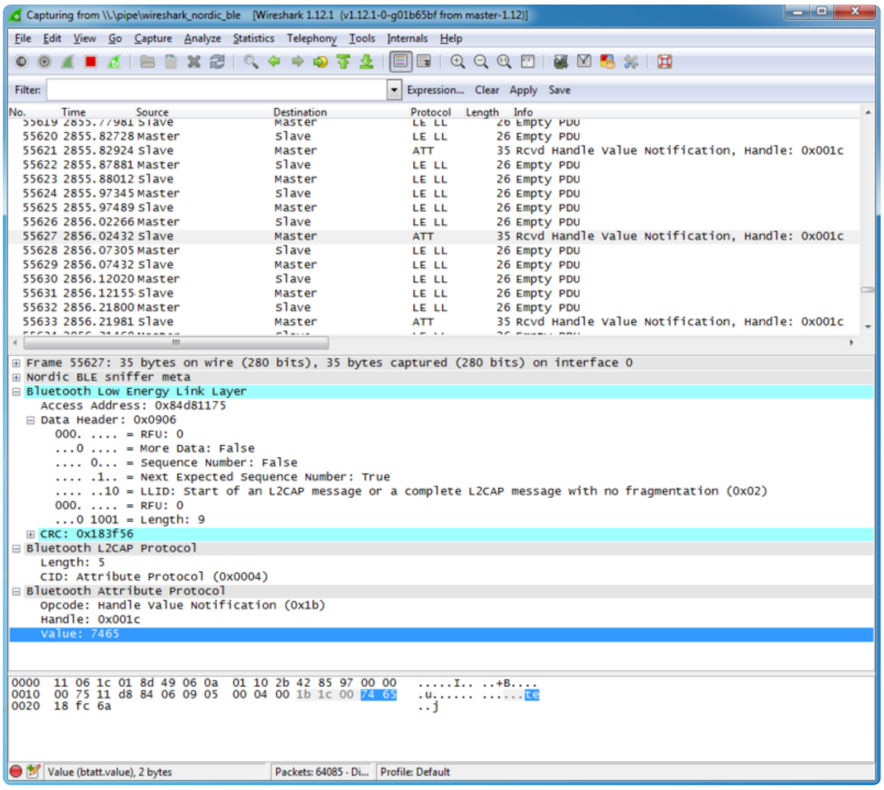
To see an actual data transaction, we simply need to enter some text in our terminal emulator SW which will cause the BLEFriend to send the data to nRF UART using the UART service.

Entering the string 'This is a test' in the terminal emulator, we can see the first packet being sent below (only the 'T' character is transmitted because the packets are sent out faster than we enter the characters into the terminal emulator):

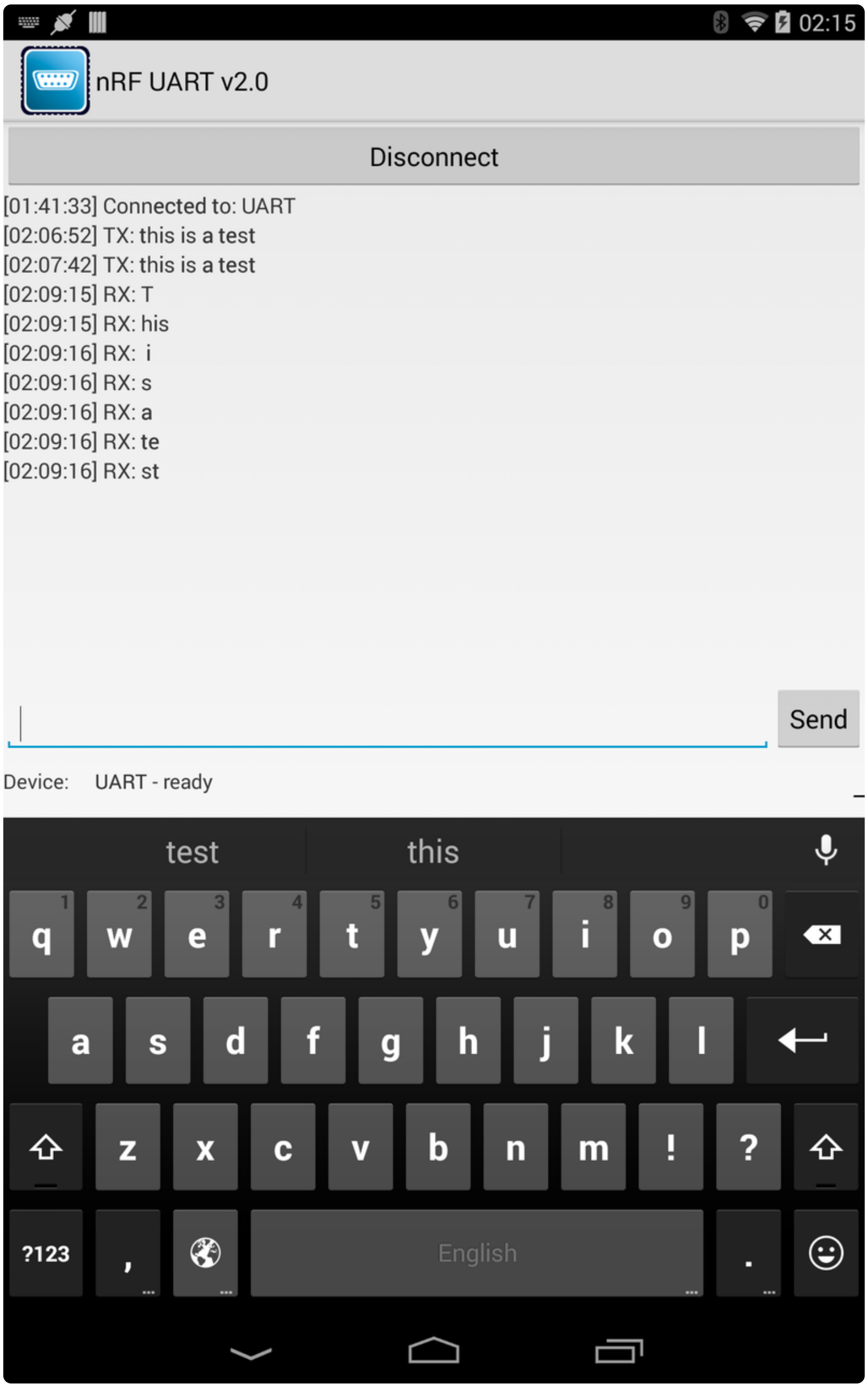


What this 4-byte 'Bluetooth Attribute Protocol' packet is actually saying is that attribute 0x001C (the location of the TX characteristic in the attribute table) has been updated, and the new value is '0x54', which corresponds to the letter 'T'.

Scrolling a bit further down we can see an example where more than one character was sent in a single transction ('te' in this case):



The results of this transaction in the nRF UART application can be seen below:



Closing Wireshark and nRF-Sniffer

When you're done debugging, you can save the session to a file for later analysis, or just close Wireshark right away and then close the nRF-Sniffer console window to end the debug session.

Moving Forward

A sniffer is an incredibly powerful and valuable tool debugging your own hardware, reverse engineering existing BLE peripherals, or just to learn the ins and outs of how Bluetooth Low Energy actually works on the a packet by packet level.

You won't learn everything there is to know about BLE in a day, but a good book on BLE, a copy of the Bluetooth 4.1 Core Specification and a sniffer will go a long way to teaching you most of the important things there is to know about BLE in the real world.

Using with Sniffer V2 (old)

This page is deprecated. It is being left here for reference and for anyone requiring a Python 2 setup.

In mid 2018, Nordic release new Bluetooth LE sniffer firmware - this firmware works way better with Wireshark.

As of August 2018 we are only selling Sniffers pre-prorgrammed with Firmware version 2

If you have a firmware V1 (packaging doesn't say firmware V2, or you bought before August 2018) see the previous sections!

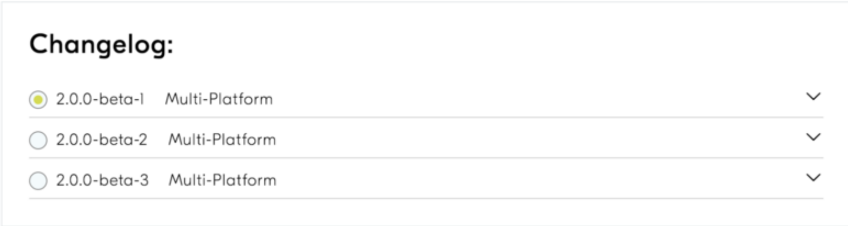
Nordic User Manual

You can grab the 'official' user manual from Nordic at https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF-Sniffer () we include a mirror of the v2.1 instructions below

nRF_Sniffer_User_Guide_v2.1.pdf

nRF Sniffer V2 Multi-Target Application

When downloading the desktop/Wireshark Sniffer tool, make sure to download BETA 1, which matches the firmware image below. You can select this version via the drop down selector shown below:



For convenience sake, the extcap folder contents for BETA 1 are also available for download here using the button below. See the user guide on how to install this in the correct location for Wireshark.

extcap.zip (BETA 1)

V2 Firmware

You need a J-Link or other SWD programming jig in order to install/change the sniffer firmware!

If by chance you have an nRF51822 board you want to load the firmware on, here's a hex that does not require the 32khz crystal (but does require the 16 mhz crystal)

This firmware is from the nrf_sniffer_2.0.0-beta-1_51296aa package

sniffer_pca10028_51296aa.hex

Again, we don't have a guide or tutorial on loading the firmware onto an nRF51. We don't have the original source code, that hex is from Nordic and they only release hex files

---

V2 Wireshark Usage (old)

This page is deprecated. It is being left here for reference and for anyone requiring a Python 2 setup.

For the V2 firmware, its recommended that you use Wireshark - the V1 had various methods such as a Python API but really they were all mediocre compared to the abilities of Wireshark

Install Wireshark

Start by installing Wireshark, a great cross-platform monitoring tool

Visit [https://www.wireshark.org/]() () and download the latest version of Wireshark for your operating system

When installing on windows, check the box to also install WinPcap

Install Wireshark Plugin

Next up, you'll need the Nordic plugin software. We need to work with a specific release - 2.0.0 Beta 1. For convenience sake, the extcap folder contents for BETA 1 are available for download using the button below.

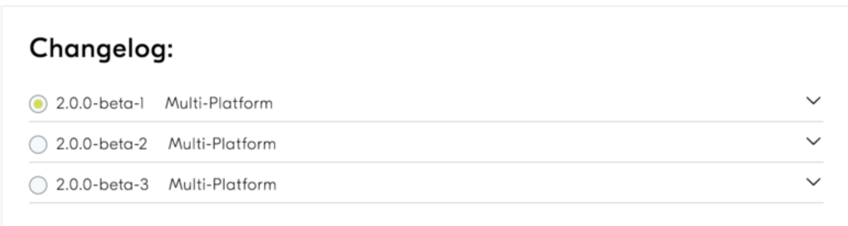
extcap.zip (BETA 1)

Download that zip file to your computer and see below for how to install the files into the necessary Wireshark folder.

For complete reference, the Nordic main page for the plugin software is here:

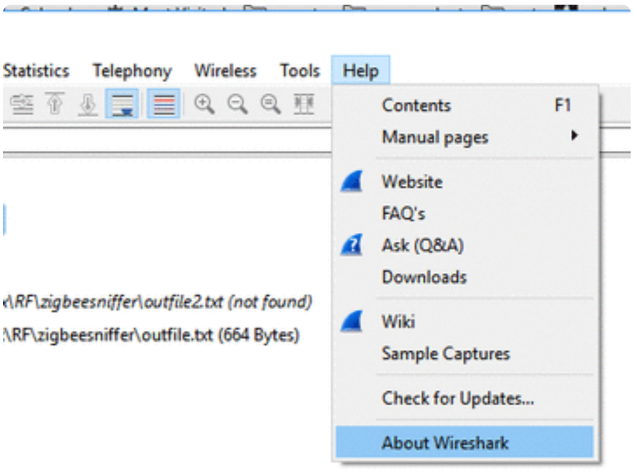
nRF Sniffer Software (reference only)

If you decide to go there, be sure to select the correct version for download.

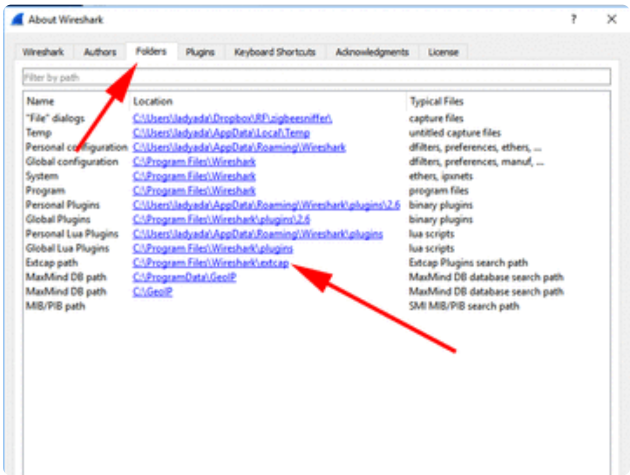


WARNING this file is huge - over 200MB! We suggest just using the extcap.zip file linked above.

Now to find the Wireshark folder location to unzip these files into.



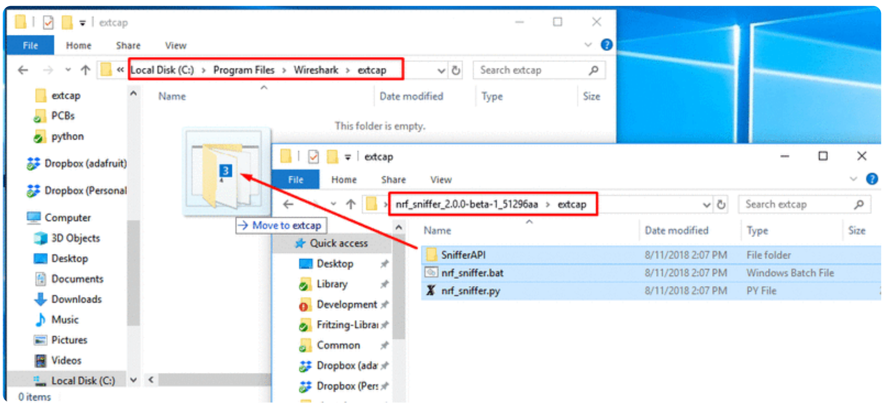
Open Wireshark, in the Help menu select About wireshark



In the Folders tab, find the extcap path

Open that directory up, then copy over the files within the extcap.zip extcap folder into the Wireshark extcap folder

In the end, your Wireshark/extcap directory should contain nrf_sniffer.bat, nrf_sniffer.py and SnifferAPI folder.



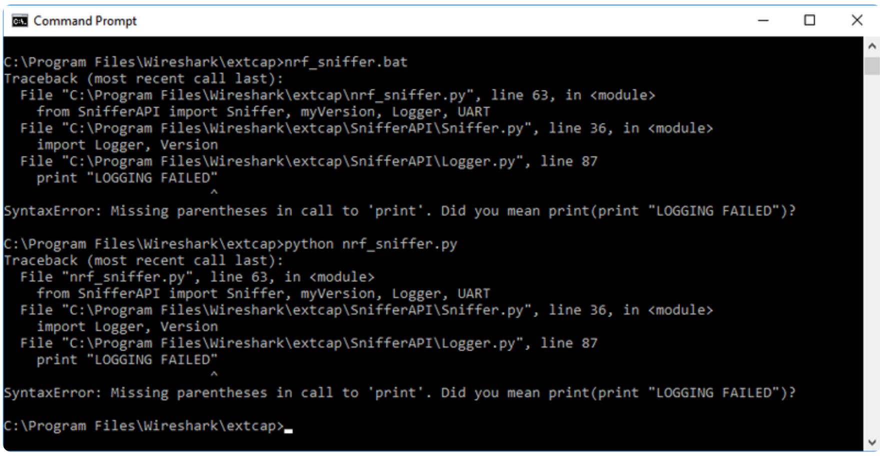
Now quit Wireshark so we can get things tested!

Dealing With Python 2 vs 3

Nordic's sniffer code is Python 2 only, so if you have Python 3 your default (which, by now, you probably do) you'll need to install Python 2.

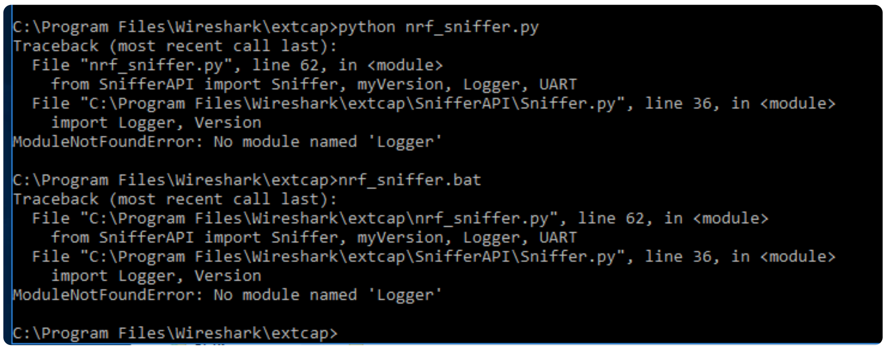
The best way to test is to go to the extcap directory in your terminal software and try running nrf_sniffer.bat (Windows) or python nrf_sniffer.py (Mac/Linux)

If you get the error on the print "LOGGING FAILED" line




```
C:\Program Files\Wireshark\extcap>nrf_sniffer.bat
Traceback (most recent call last):
  File "C:\Program Files\Wireshark\extcap\nrf_sniffer.py", line 63, in <module>
    from SnifferAPI import Sniffer, myVersion, Logger, UART
  File "C:\Program Files\Wireshark\extcap\SnifferAPI\Sniffer.py", line 36, in <module>
    import Logger, Version
  File "C:\Program Files\Wireshark\extcap\SnifferAPI\Logger.py", line 87
    print "LOGGING FAILED"
                         ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(print "LOGGING FAILED")?

C:\Program Files\Wireshark\extcap>python nrf_sniffer.py
Traceback (most recent call last):
  File "nrf_sniffer.py", line 63, in <module>
    from SnifferAPI import Sniffer, myVersion, Logger, UART
  File "C:\Program Files\Wireshark\extcap\SnifferAPI\Sniffer.py", line 36, in <module>
    import Logger, Version
  File "C:\Program Files\Wireshark\extcap\SnifferAPI\Logger.py", line 87
    print "LOGGING FAILED"
                         ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(print "LOGGING FAILED")?

C:\Program Files\Wireshark\extcap>
```

or the "No module named 'Logger'" error



```
C:\Program Files\Wireshark\extcap>python nrf_sniffer.py
Traceback (most recent call last):
  File "nrf_sniffer.py", line 62, in <module>
    from SnifferAPI import Sniffer, myVersion, Logger, UART
  File "C:\Program Files\Wireshark\extcap\SnifferAPI\Sniffer.py", line 36, in <module>
    import Logger, Version
ModuleNotFoundError: No module named 'Logger'

C:\Program Files\Wireshark\extcap>nrf_sniffer.bat
Traceback (most recent call last):
  File "C:\Program Files\Wireshark\extcap\nrf_sniffer.py", line 62, in <module>
    from SnifferAPI import Sniffer, myVersion, Logger, UART
  File "C:\Program Files\Wireshark\extcap\SnifferAPI\Sniffer.py", line 36, in <module>
    import Logger, Version
ModuleNotFoundError: No module named 'Logger'

C:\Program Files\Wireshark\extcap>
```

Then you'll need to trick the sniffer software into using Python 2

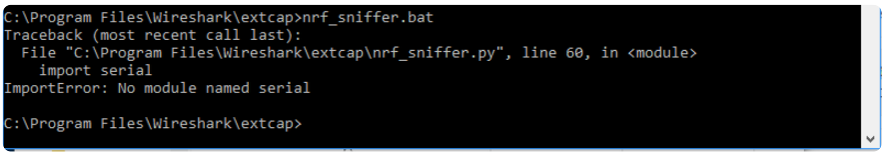
For Windows, at least, I installed Python 2.7 into C:\Python27 (the default) and then edited the nrf_sniffer.bat file to say:

```
@echo off
C:\Python27\python "%~dp0nrf_sniffer.py"  %*
```

note the explicit path!

Installing Dependancies

Once you get past that part, you can try rerunning the bat/py script and you may get other missing module errors like No module named serial

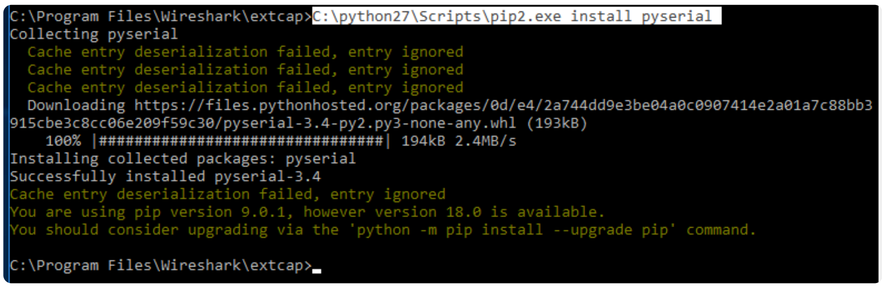


```
C:\Program Files\Wireshark\extcap>nrf_sniffer.bat
Traceback (most recent call last):
  File "C:\Program Files\Wireshark\extcap\nrf_sniffer.py", line 60, in <module>
    import serial
ImportError: No module named serial

C:\Program Files\Wireshark\extcap>
```

You'll need to install these with pip

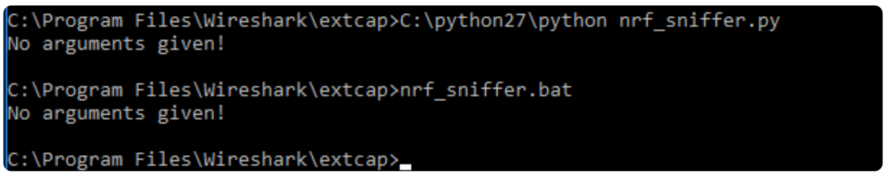
Warning! Because you have to use Python2 here, make sure you're using pip2 or on windows, use the full path C:\python27\Scripts\pip2.exe

e.g. C:\python27\Scripts\pip2.exe install pyserial



```
C:\Program Files\Wireshark\extcap>C:\python27\Scripts\pip2.exe install pyserial
Collecting pyserial
  Cache entry deserialization failed, entry ignored
  Cache entry deserialization failed, entry ignored
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/0d/e4/2a744dd9e3be04a0c0907414e2a01a7c88bb3
915cbe3c8cc06e209f59c30/pyserial-3.4-py2.py3-none-any.whl (193kB)
    100% |################################| 194kB 2.4MB/s
Installing collected packages: pyserial
Successfully installed pyserial-3.4
Cache entry deserialization failed, entry ignored
You are using pip version 9.0.1, however version 18.0 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Program Files\Wireshark\extcap>
```

Eventually you'll get No arguments given! which means the script is, at least, fully running

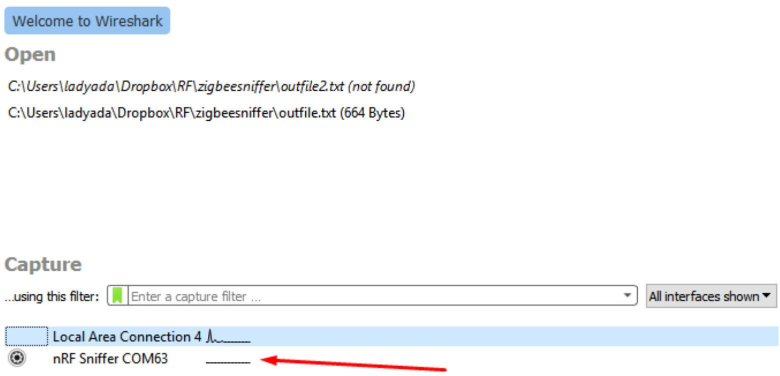


```
C:\Program Files\Wireshark\extcap>C:\python27\python nrf_sniffer.py
No arguments given!

C:\Program Files\Wireshark\extcap>nrf_sniffer.bat
No arguments given!

C:\Program Files\Wireshark\extcap>
```

Test Capture

OK finally once that works, start Wireshark again.

This time you'll see the nRF Sniffer capture device!



Double Click on that line to start the Capture!

Now go here to learn how to look at BLE packets with Wireshark

Windows Install Supplemental Information

Adafruit forums user @TomHildebrand put together a nice write up on their experience installing and setting up everything on Windows 10. It's generally the same info as above, but may have some more explicit info that is useful. Checkout it out here:

Adafruit BLE Sniffer installation on Windows Step-by-Step

Using with Sniffer V1 (old)

The original Bluetooth LE sniffer firmware from Nordic had some restrictions such as only being usable by Wireshark 1.

As of August 2018 we are only selling Sniffers pre-prorgrammed with Firmware version 2

However, we'll keep this documentation up in case its useful for people with old boards

You need a J-Link or other SWD programming jig in order to install/change the sniffer firmware!

If by chance you have an nRF51822 board you want to load the firmware on, here's a hex that does not require the 32khz crystal (but does require the 16 mhz crystal) ble-sniffer_nRF51822_1.0.1_1111_Sniffer_N

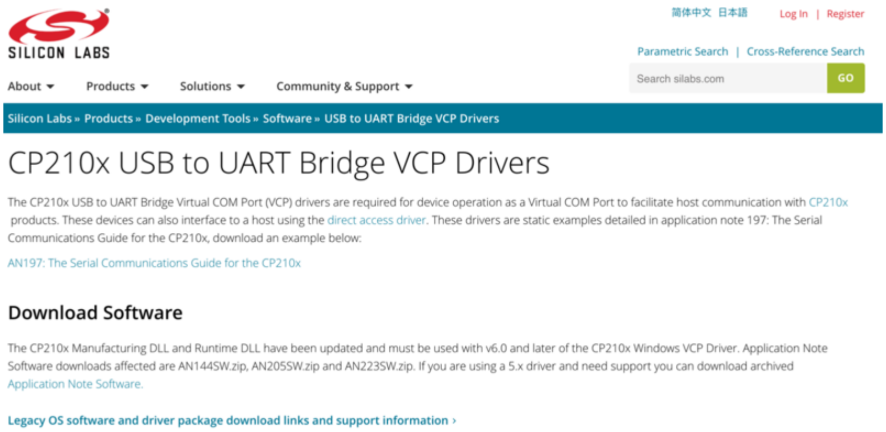
Again, we don't have a guide or tutorial on loading the firmware onto an nRF51. We don't have the original source code, that hex is from Nordic and they only release hex files

---

USB Driver Install

CP2104 Driver Requirements (Black Boards)

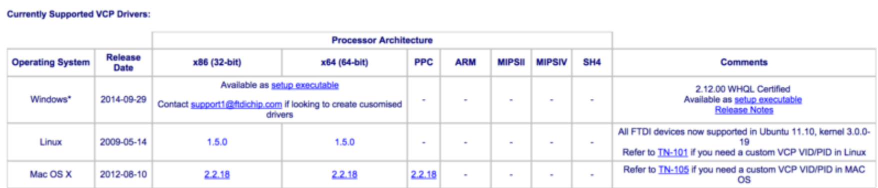
The latest version of the sniffer uses the CP2104 USB to Serial bridge and drops the SWD connector, allowing us to sell the boards at a significant discount compared to version 1.0. To use these boards, though, you will need to install the [CP2104 VCP driver from Silicon Labs]():



FTDI Driver Requirements (Blue Boards)

Before you can start talking to the sniffer, you'll need to install a standard FTDI driver for the FT231x located on the device.

Find the appropriate FTDI VCP installer on the [FTDI Driver Download Page](), install it on you system, and then insert the sniffer in any USB port on your system.



Currently Supported VCP Drivers:

		Processor Architecture							
Operating System	Release Date	x86 (32-bit)	x64 (64-bit)	PPC	ARM	MIPSII	MIPSIV	SH4	Comments
Windows*	2014-09-29	Available as setup executable Contact support1@ftdichip.com if looking to create cusomised drivers		-	-	-	-	-	2.12.00 WHQL Certified Available as setup executable Release Notes
Linux	2009-05-14	1.5.0	1.5.0	-	-	-	-	-	All FTDI devices now supported in Ubuntu 11.10, kernel 3.0.0-19 Refer to TN-101 if you need a custom VCP VID/PID in Linux
Mac OS X	2012-08-10	2.2.18	2.2.18	2.2.18	-	-	-	-	Refer to TN-105 if you need a custom VCP VID/PID in MAC OS

V1 Sniffer Software

This page is for the V1 Sniffer firmware only! If you have V2, check the other page - the process has changed between versions.

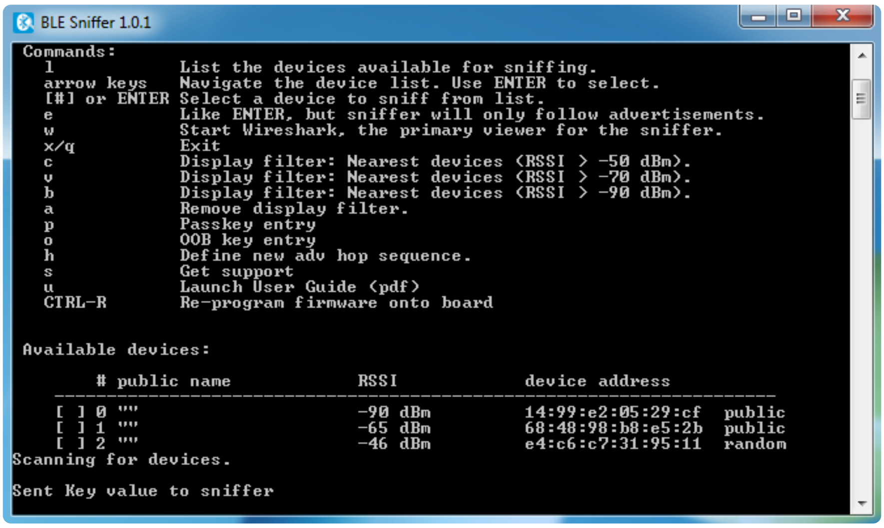
Using the Firmware V1 Sniffer

There are currently two ways to use the sniffer:

Nordic's nRF Sniffer Utility (Windows only)

If you are on Windows, the best user experience will be had by using the official Nordic nRFSniffer application, available as a download from Nordic Semiconductors after creating a 'My Pages' account, and registering your device using the product ID located on the Bluefruit LE Sniffer packaging.

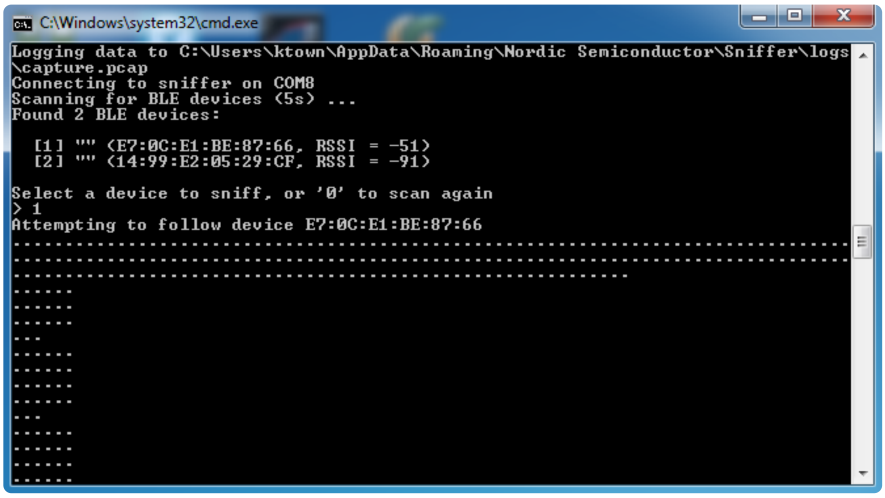
More information on using Nordic's nRF Sniffer application ().



Python API (Cross-Platform, no Registration)

If you are not using Windows, or don't wish to create a MyPages account, the alternative is to use a Python interface to communicate with the nRFSniffer firmware, which will log any traffic to a libpcap file that can be opened directly in Wireshark. This has been tested on OS X 10.10, Ubuntu 14.04 and Windows 7, but it currently doesn't support streaming data directly into Wireshark via named pipes (though this is possible with some platform-specific effort).

[More information on using the Python API ()](#).



V1 Nordic nRF Sniffer

This page is for the V1 Sniffer firmware only! If you have V2, check the other page - the process has changed between versions.

The following guide will walk you through downloading, installing and using the official nRF Sniffer application for Nordic Semiconductors.

Getting the Sniffer Utility

The Bluefruit LE Sniffer comes pre-flashed with the special sniffer firmware image, but you'll need to go to Nordic's website and download the nRF-Sniffer package to capture the data on Windows and push it out into Wireshark for packet by packet analysis.

Go to the nRF Sniffer product page downloads tab (), then download the latest V1 version of the utility, and unzip it.

Inside this downloaded file you'll find the sniffer executable, which will open up the command-line tool when you click on it.

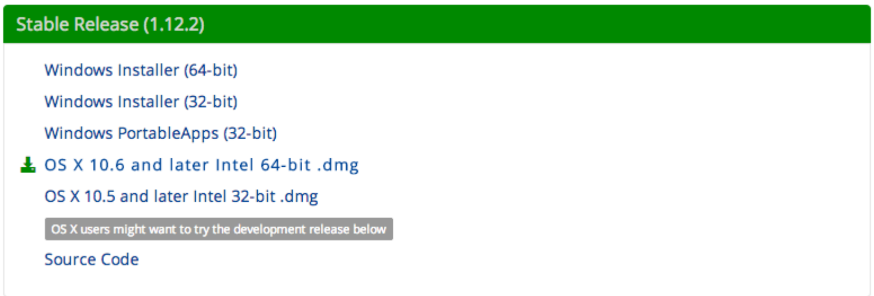
Getting Wireshark

In order to use the sniffer utility you'll also need to download Wireshark (), preferably verison 1.12.1 (the same one used in this tutorial).

You may need to explore the download mirrors, such as https://1.na.dl.wireshark.org/ () to find the download link since they dont have a direct v1 link

Simply select the 32-bit or 64-bit Windows Installer and install it on your machine using the default settings:

Make sure you are using a version greater than or equal to 1.12.1, but less than the most recent 2.x release family. The plugin is written with 1.12.x as a target.



Make sure that you install the libpcap library when installing Wireshark. Any log files captured by the python library are in libpcap format, and will require this library to work.

Running the Sniffer

Now that everything is installed, you can get started using the Bluefruit LE Sniffer and the sniffer bridge SW that pushes any sniffed data out into Wireshark ...

Select the Sniffer Target

The nRF-Sniffer can only sniff one device at a time, so the first step is getting the sniffer running and then selecting the device that you want to debug.

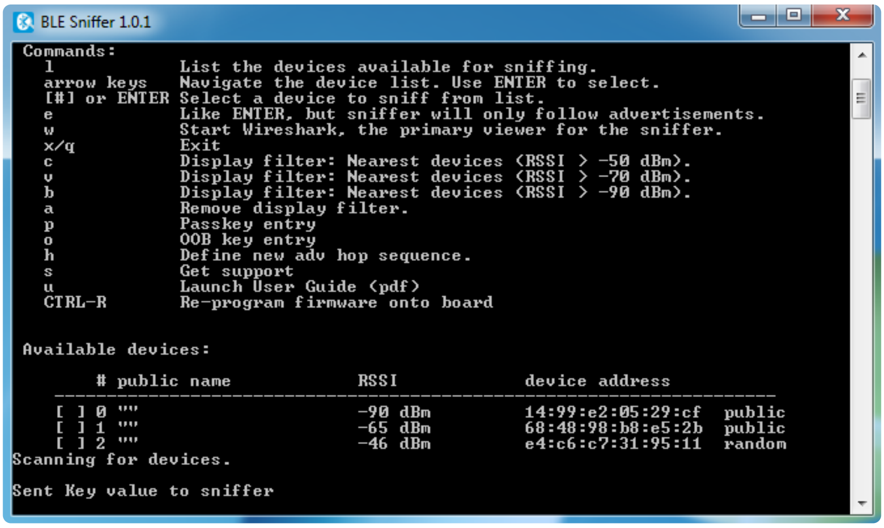
Start nRF-Sniffer by running the ble-sniffer_win executable (for example: ble-sniffer_win_1.0.1_1111_Sniffer.exe).

This will try to detect the device running the nRF-Sniffer firmware over a UART COM port.

If the board isn't detected right away type 'f' to erase any previous com port settings, or try removing and then re-inserting the sniffer while the console application is running.

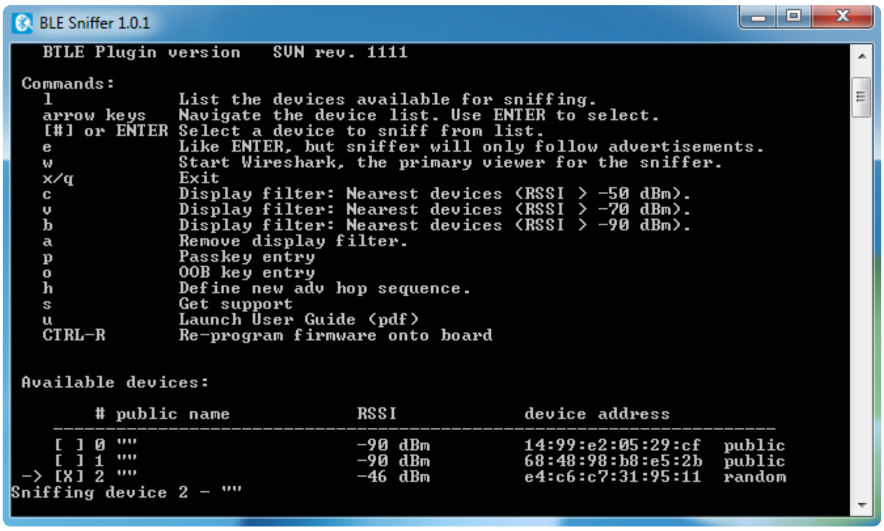
Once the sniffer is found, you should see a list of all BLE devices that were detected in listening range:

If you see a warning in the application about your firmware being out of date and requesting to update it, IGNORE THE WARNING. The Adafruit boards run a slightly modified version of the sniffer firmware, which causes the tool to think it is out of date.



In this particular case, we'll select device number 2, which is a BLEFriend running the standard UART firmware.

Type the device number you want to sniffer (in this case '2'), and you should see the device highlighted in the list, similar to the image below:



At this point you can type 'w', which will try to open wireshark and start pushing data out via a dedicate pipe created by the nRF-Sniffer utility.

Now go here to learn how to look at BLE packets with Wireshark

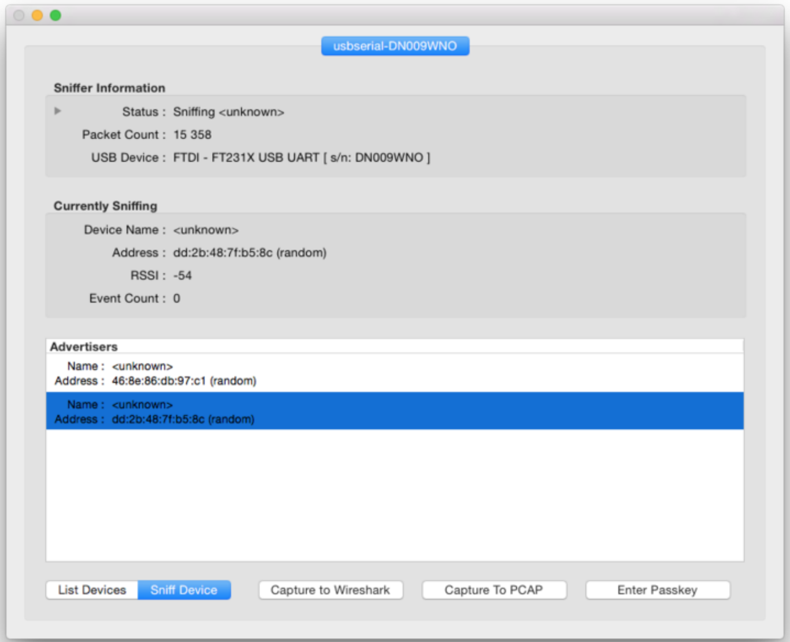
V1 OS X Support

This page is for the V1 Sniffer firmware only! If you have V2, check the other page - the process has changed between versions.

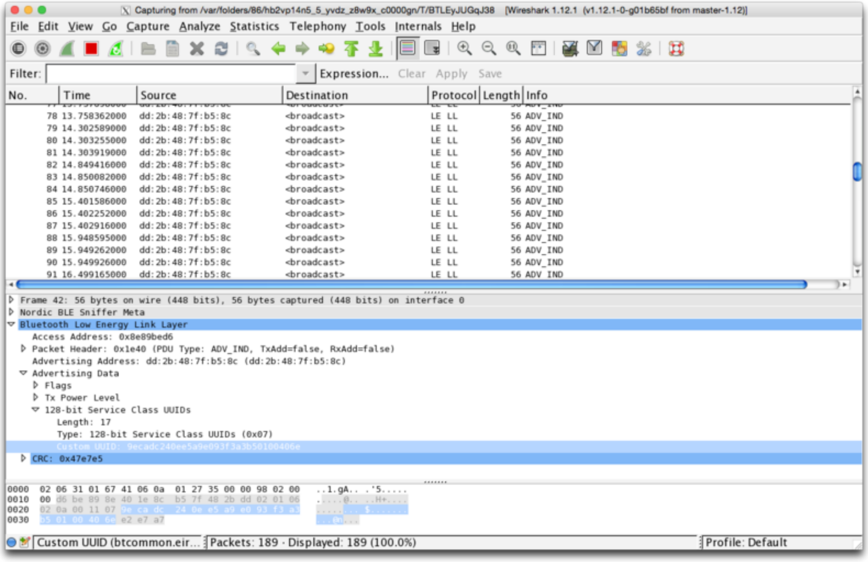
If you are running OS X 10.9 or higher, you can also use the sniffer on OS X using the nrf-ble-sniffer-osx () package from Roland King. (Make sure you have the latest version, as of 20 June 2015, which is now compatible with the FTDI chip used on the Adafruit board.)

Setup instructions are available on the wiki page () for the project.

Be sure to download Wireshark version 2.0.x NOT the new 2.2.7 that was released June 2017



Please note that there can be a long delay (30-60 seconds) before Wireshark shows up using the tool, due to the X11 startup time, etc.



If Wireshark doesn't show up and X11 has been installed correctly, try forcing X11 closed and trying a second time. The startup process can sometimes stall.

V1 Python API

This page is for the V1 Sniffer firmware only! If you have V2, check the other page - the process has changed between versions.

The Python interface requires a custom Wireshark library for Linux. We're currently working on adding support for this. Please use the Windows or OS X utility until the update is available.

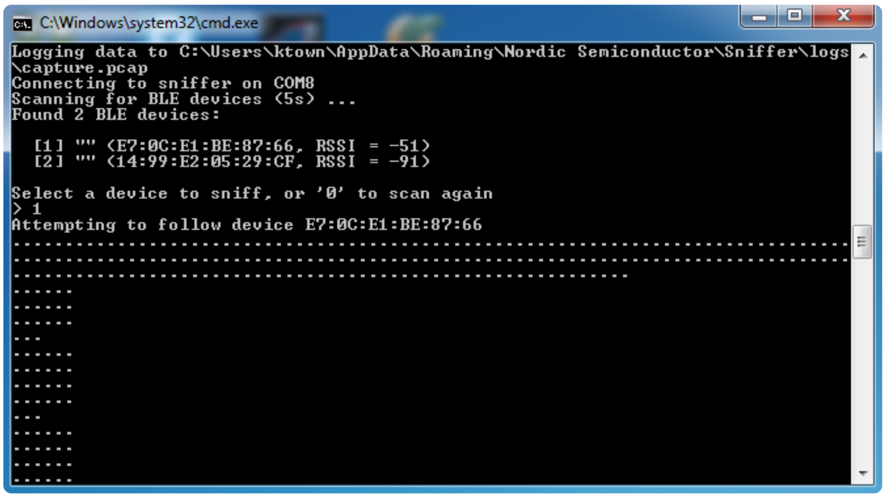
Nordic provides a Python API for their sniffer firmware that makes it possible for us to use the sniffer on any platform, and we've put together a basic wrapper for this API to help you get started.

We've tested this wrapper with Python 2.7 on the following platforms:

- OS X 10.10
- Windows 7 x64
- Ubuntu 14.04

To stream live data into Wireshark the way the official Windows app () from Nordic does you will need to compile a Wireshark utility that creates a name pipe that data gets pushed through.

To keep things simple, though, you can also just log sniffed traffic directly to a libpcap file, which can be opened directly in Wireshark when you are done, which is the easiest solution and what we'll be demonstrating here:



Requirements

To use the example we provide for the Python API, you will require the following utilities:

- Python 2.7.x () (we tested with 2.7.6)
- pySerial ()

If you're new to Python and pySerial, have a look at our Instaling Python and PySerial ( ) guide by Simon Monk.

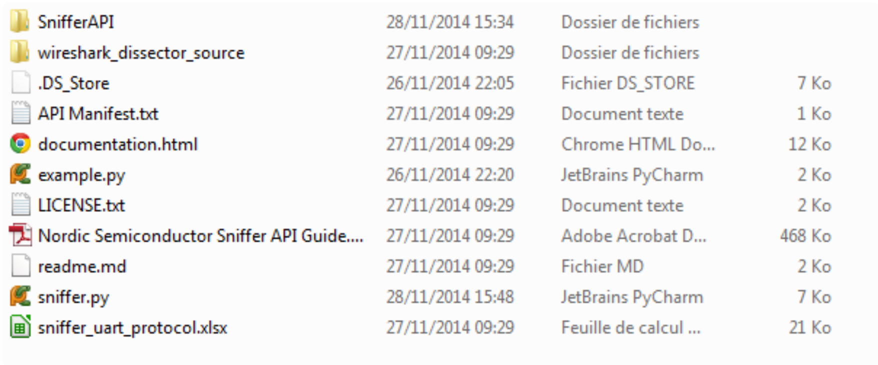
Download the API

Once you have Python and pySerial installed on your system, you will need to download a copy of the Python API.

The latest version of the API is always available on Github (), but you can also download a .zip file of the latest code directly using the button below:

Download the Python API from Github

Unzipping the file should give you a file structure resembling the image below:



SnifferAPI	28/11/2014 15:34	Dossier de fichiers	
wireshark_dissector_source	27/11/2014 09:29	Dossier de fichiers	
.DS_Store	26/11/2014 22:05	Fichier DS_STORE	7 Ko
API Manifest.txt	27/11/2014 09:29	Document texte	1 Ko
documentation.html	27/11/2014 09:29	Chrome HTML Do...	12 Ko
example.py	26/11/2014 22:20	JetBrains PyCharm	2 Ko
LICENSE.txt	27/11/2014 09:29	Document texte	2 Ko
Nordic Semiconductor Sniffer API Guide....	27/11/2014 09:29	Adobe Acrobat D...	468 Ko
readme.md	27/11/2014 09:29	Fichier MD	2 Ko
sniffer.py	28/11/2014 15:48	JetBrains PyCharm	7 Ko
sniffer_uart_protocol.xlsx	27/11/2014 09:29	Feuille de calcul ...	21 Ko

Using the sniffer.py Wrapper

To help you get started, we've made an easy to use wrapper called sniffer.py:

```
$ sudo python sniffer.py -h
usage: sniffer.py [-h] [-v] serialport

Interacts with the Bluefruit LE Friend Sniffer firmware

positional arguments:
  serialport     serial port location ('COM14', '/dev/tty.usbserial-DN009WNO',
                 etc.)

optional arguments:
  -h, --help     show this help message and exit
  -v, --verbose  verbose mode (all serial traffic is displayed)
```

It takes a single argument, the COM port location, which will be something like 'COM15' on Windows, '/dev/ttyACM*' on Linux, or '/dev/tty.usbserial*' on OS X.

Linux

To run the sniffer wrapper on Linux, enter the following command (changing the serial port as necessary):

```
$ sudo python sniffer.py /dev/ttyACM0
```

OS X

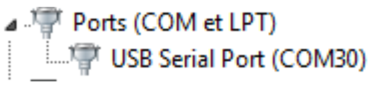
To run the sniffer wrapper on OS X, enter the following command (changing the serial port as necessary):

```
$ python sniffer.py /dev/tty.usbserial-DN009MP6
```

Windows

To run the sniffer wrapper on Windows, enter the following command (changing the serial port as necessary):

You can find the serial port used by the Bluefruit LE Sniffer by opening the Device Manager on your system and looking in the 'Ports' category:




```
python sniffer.py COM30
```

Scanning for Devices

If the wrapper was able to connect to the Bluefruit LE Sniffer, it will perform a 5 second scan for Bluetooth Low Energy devices in range, and ask you which device you want to listen to:

```
$ sudo python sniffer.py /dev/ttyACM0
[sudo] password for ktown:
Logging data to logs/capture.pcap
Connecting to sniffer on /dev/ttyACM0
Scanning for BLE devices (5s) ...
Found 2 BLE devices:

  [1] "" (E7:0C:E1:BE:87:66, RSSI = -52)
  [2] "" (14:99:E2:05:29:CF, RSSI = -94)

Select a device to sniff, or '0' to scan again
&gt;
```

Once you select a device, it will start scanning that specific device, and you will see an update every second of the number of packets 'sniffed' from the device (where each '.' represents a packet):

```
Select a device to sniff, or '0' to scan again
&gt; 1
Attempting to follow device E7:0C:E1:BE:87:66
...............................................................................................
............................
..............................
...........................
..............................
```

Locating the Log File

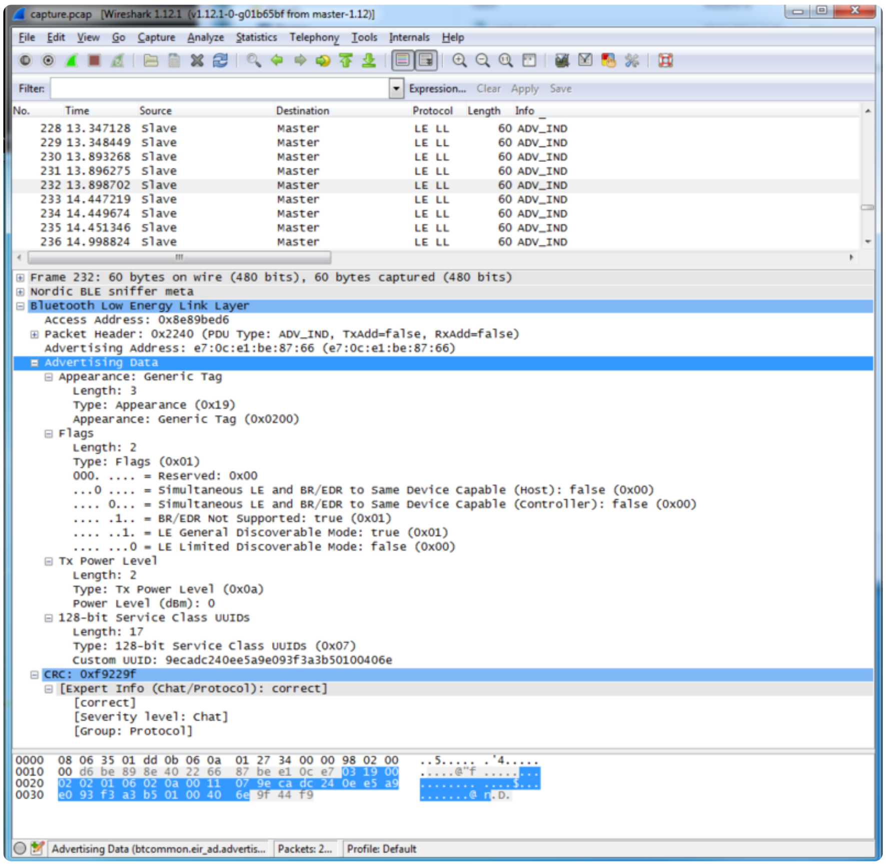
Once you've sniffed enough data, simply type CTRL+C to stop, and locate the libpcap log file at the path mentionned by the tool. This will normally be:

- Windows: 'C:\Users\ktown\AppData\Roaming\Nordic Semiconductor\Sniffer\logs\capture.pcap' (this will of course change based on your username)
- OS X/Linux: 'logs/capture.pcap' (relative to the location of the Python API)

Analyze Data in Wireshark

At this point, you simply need to open the capture.pcap file in Wireshark, and you can analyze the sniffed data!

The image below shows an advertising packet from a factory default Bluefruit LE Friend () board:



Note that the utility will start sniffing data as soon as you connect to the Bluefruit LE Sniffer, so early packets in the log file might contain advertising packets from other devices in range. It will only start filtering packets once you select a specific device via the selection dialogue.

For information on how to use Wireshark, have a look at the notes on the official nRF Sniffer utility (), which describes some of the packet types you might encounter working with Bluetooth Low Energy.

FAQs

I'm using the V2 (BETA 1) firmware, but can't seem to connect in Wireshark?

There are a number of possible issues here, mostly revolving around the fact that the system depends on a Python script piping data into Wireshark. You may find the following post from a user help to try to debug this, the cause being a potential conflict with multiple instances of Python on your system: https://forums.adafruit.com/viewtopic.php?f=53&t=146215#p726333 ()

When I connect to a Central device, I don't see any connection data, but when I disconnect I see the advertising packets again. How do I capture data with a connected peripheral?

This is a limitation of the sniffer firmware from Nordic. Advertising in Bluetooth Low Energy happens on three dedicated channels, each running at it's own frequency. For the sniffer to 'follow' the connection it needs to be looking at the right channel when the connection happens, and there is a 2/3 chance that it is looking at another channel at any given moment.

To capture the connection and see data exchanges post connection, you may need to connect several times until the channels are aligned between the sniffer and the BLE peripheral+central devices.

How do I convert between Sniffer and Bluefruit LE firmware using SWD?

Reflashing Bluefruit LE modules over SWD (ex. switching to the sniffer firmware and back) is at your own risk and can lead to a bricked device, and we can't offer any support for this operation! You're on your own here, and there are unfortunately 1,000,000 things that can go wrong, which is why we offer two separate Bluefruit LE Friend boards -- the sniffer and the normal Bluefruit LE Friend board with the non-sniffer firmware, which provides a bootloader with fail safe features that prevents you from ever bricking boards via OTA updates.

AdaLink (SWD/JTAG Debugger Wrapper)

Transitioning between the two board types (sniffer and Bluefruit LE module) is unfortunately not a risk-free operation, and requires external hardware, software and know-how to get right, which is why it isn't covered by our support team.

That said ... if you're determined to go down that lonely road, and you have a Segger J-Link () (which is what we use internally for production and development), or have already erased your Bluefruit LE device, you should have a look at AdaLink (), which is the tool we use internally to flash the four files required to restore a Bluefruit LE module. (Note: recent version of AdaLink also support the cheaper STLink/V2 (http://adafru.it/2548), though the J-Link is generally more robust if you are going to purchase a debugger for long term use.)

To go from the sniffer to Bluefruit LE firmware the mandatory Intel Hex files are available in the Bluefruit LE Firmware repo (). You will need to flash:

- An appropriate bootloader image
- An appropriate SoftDevice image
- The Bluefruit LE firmware image
- The matching signature file containing a CRC check so that the bootloader accepts the firmware image above (located in the same folder as the firmware image)

The appropriate files are generally listed in the version control .xml file () in the firmware repository.

If you are trying to flash the sniffer firmware (at your own risk!), you only need to flash a single .hex file, which you can find here (). The sniffer doesn't require a SoftDevice image, and doesn't use the fail-safe bootloader -- which is why changing is a one way and risky operation if you don't have a supported SWD debugger.

Adafruit_nF51822_Flasher

We also have an internal python tool available that sits one level higher than AdaLink (referenced above), and makes it easier to flash specific versions of the official firmware to a Bluefruit LE module. For details, see the Adafruit_nRF51822_Flasher () repo.

Why isn't the Firmware V1 plugin working in Wireshark?

The Sniffer Firmware V1 plugin was written for Wireshark 1.12.x and won't work with older versions of the tool or the new 2.x family. Be sure to download an appropriate version (for example 1.12.1, which is the version used in this guide).

Why am I being warned my Sniffer V1 firmware is out of data but updates fail?

The Adafruit board has a small difference compared to the original Nordic HW that Nordic wrote their sniffer firmware for. To keep the cost as low as possible, we don't populate the optional 32.768KHz RTC crystal on our boards, whereas it is present on the more expensive Nordic development kit.

Because the startup code in the sniffer firmware from Nordic uses this crystal, we had to request a custom version from Nordic that uses the internal 16MHz RC oscillator instead. When providing us the custom firmware, they changed the version number slightly, which is the reason for the warning message.

You can safely ignore the firmware update warning and use the device as normal, and in fact updating to a firmware from Nordic won't work unless you also solder the optional 32.768KHz crystal on the bottom of your PCB as well.

How can I check that the sniffer is outputting data?

If you think there is a problem with your sniffer, you should look at the LED closest to the black SWD connector box at the end of the board. It should flash every time Bluetooth Low Energy activity is detected when the serial port is open.

You can also open a Terminal Emulator (Putty, RealTerm, etc.) with the following settings, and you should see data coming out almost as soon as you plug the sniffer in:

- Baud Rate: 460800
- HW Flow Control: RTS + CTS Enabled

What is the difference between blue boards and black board)?

The Black boards (hardware v3) uses the much cheaper CP2104 USB to Serial bridge, and drops the SWD connector which had to be manually placed during the manufacturing process. This allows us to offer the sniffer board at a significant discount compared to the original, without sacrificing functionality that 99% of customers required. (The SWD pins are still available as pads on the bottom of the PCB if you need them!).

If you have a Blue board - you definitely have hardware version 1 and Firmware version 1

If you have a Black board - you definitely have hardware version 3. You may have firmware version 1 or version 2 depending on when you purchased it. Check your order receipt to know!

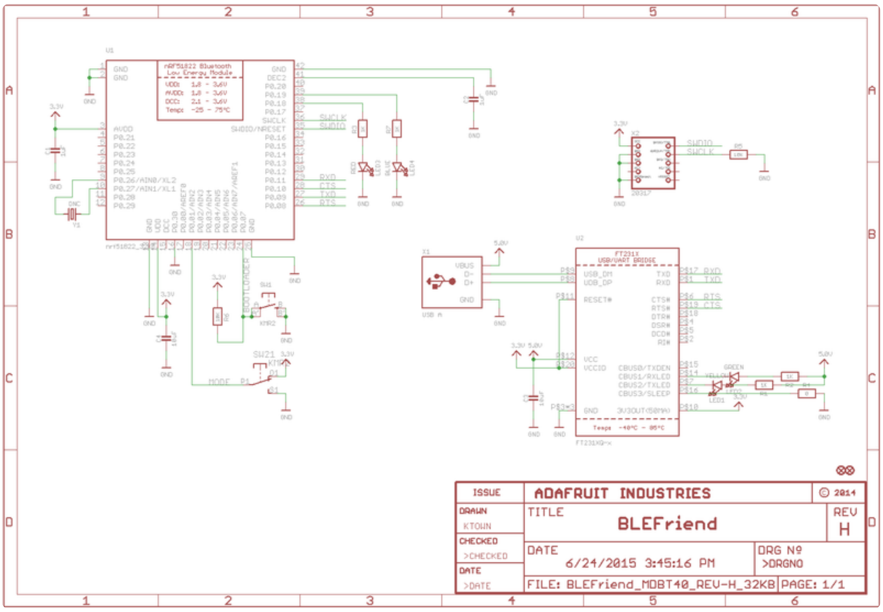
Downloads

Files

- EagleCAD PCB files on GitHub ()
- Bluetooth LE module Datasheet ()

Schematic C2104 Rev

Click to embiggen


ADAFRUIT INDUSTRIES
© 2014
ISSUE
DRAWN
KTOWN
CHECKED
>CHECKED
DATE
>DATE
TITLE
BLEFriend
REV
H
DATE
6/24/2015 3:45:16 PM
DRG Nº
>DRGNO
FILE: BLEFriend_MDBT40_REV-H_32KB
PAGE: 1/1