

#### Introducing the Adafruit Bluefruit LE Sniffer

Created by Kevin Townsend

Capturing from \\\pipe\wireshark_nordic_ble [Wireshark 1.12.1 (v1.12.1	2.1-0-g01b65bf from master-1.12)]	×
<u>File Edit View Go Capture Analyze Statistics Telephony Tools</u>	ols <u>I</u> nternals <u>H</u> elp	
◎ ◎ 🗶 ■ 🤬 🖿 🖄 🗶 🔍 🗢 🌳 🖗 💆	L   🗐 🗟   Q, Q, Q, 🔟   👹 🖄 🥵 %   💢	
Filter:	▼ Expression Clear Apply Save	
No. Time Source Destination	Protocol Length Info	
55593 2854.95085 Slave Master	LE LL 26 Empty PDU	
55594 2854.99892 Master Slave	LE LL 26 Empty PDU	
55595 2855.00055 slave Master	ATT 34 Rcvd Handle Value Notification, Handle: 0x001c	
55596 2855.04974 Master Slave	LE LL 26 Empty PDU	
55597 2855.05107 slave Master	LE LL 26 Empty PDU	
55598 2855.09619 Master Slave	LE LL 26 Empty PDU	
55599 2855.09776 Slave Master	LE LL 26 Empty PDU	
55600 2855.19274 Master Slave 55601 2855.19395 Slave Master	LE LL 26 Empty PDU LE LL 26 Empty PDU	
55602 2855.29033 Master Slave	LE LL 26 Empty PDU LE LL 26 Empty PDU	
55603 2855.29210 Slave Master	LE LL 26 Empty PDU	
55604 2855.33914 Master Slave	LE LL 26 Empty PDU	
55605 2855, 34070 Slave Master	LE LL 26 Empty PDU	
55606 2855, 38816 Master Slave	LE LL 26 Empty PDU	-
55607 2855.38971 Slave Master	LE LL 26 Empty PDU	-
< III.	•	
⊕ Frame 55595: 34 bytes on wire (272 bits), 34 byte	tes captured (272 bits) on interface 0	
Nordic BLE sniffer meta		
Bluetooth Low Energy Link Layer		
Access Address: 0x84d81175		
🗆 Data Header: 0x080a		
000 = RFU: 0		
0 = More Data: False		
1 = Sequence Number: True		
0 = Next Expected Sequence Number: Fa		
	or a complete L2CAP message with no fragmentation (0x02)	
000 = RFU: 0 0 1000 = Length: 8		
0 1000 = Length: 8		
Bluetooth L2CAP Protocol		
Length: 4		
CID: Attribute Protocol (0x0004)		
Bluetooth Attribute Protocol		
Opcode: Handle Value Notification (0x1b)		
Handle: 0x001c		
Value: 54		
0000 11 06 1b 01 6d 49 06 0a 01 0d 2b 2d 85 99 00		_
0010 00 75 11 d8 84 0a 08 04 00 04 00 1b 1c 00 54	54 76 .u	
0020 e9 73	.5	
		_
🛑 💅 Bluetooth Attribute Protocol (btatt), 4 bytes 🔰 Packets: 57843 · Di	Profile: Default	

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	5
Using with Sniffer V2 and Python3	5
BLE Sniffer Hardware	
Silicon Labs VCP Driver	
Python 3	
Python Serial Support	
Install Wireshark	
Install BLE Sniffer Plugin	
Final Check and Test Capture	
Next Steps	
Working with Wireshark	13
Working with Wireshark	
Capturing Exchanges Between Two Devices	
Scan Response Packets	
Connection Request	
Write Request	
Regular Data Requests	
Notify Event Data	
Closing Wireshark and nRF-Sniffer	
Moving Forward	
Using with Sniffer V2 (old)	21
Nordic User Manual	
nRF Sniffer V2 Multi-Target Application	
• V2 Firmware	
V2 Wireshark Usage (old)	22
Install Wireshark	
Install Wireshark Plugin	
Dealing With Python 2 vs 3	
Installing Dependancies	
Test Capture	
Windows Install Supplemental Information	
Using with Sniffer V1 (old)	27
USB Driver Install	28
CP2104 Driver Requirements (Black Boards)	
FTDI Driver Requirements (Blue Boards)	
V1 Sniffer Software	29
Using the Firmware V1 Sniffer	
Nordic's nRF Sniffer Utility (Windows only)	
Python API (Cross-Platform, no Registration)	
V1 Nordic nRF Sniffer	30
Getting the Sniffer Utility	
Getting Wireshark	

• Running the Sniffer

#### • Select the Sniffer Target

V1 OS X Support		
V1 Python API		
Distanting and states		

- Requirements
- Download the API
- Using the sniffer.py Wrapper
- Linux
- OS X
- Windows
- Scanning for Devices
- Locating the Log File
- Analyze Data in Wireshark

#### FAQs

Downloads		42

- Files
- Schematic C2104 Rev

33

34

39

## 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.

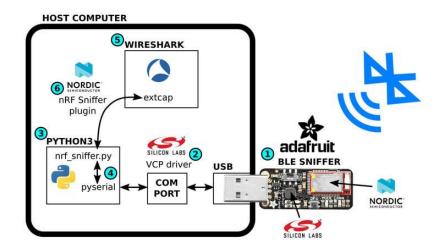


Since nRF-Sniffer is a passive solution that is simply scanning packets over the air, there is the possibility of missing packets using this tool (or any other passive sniffing solution). In order to capture as many packets as possible, be sure to run the sniffer on a USB bus that isn't busy and avoid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



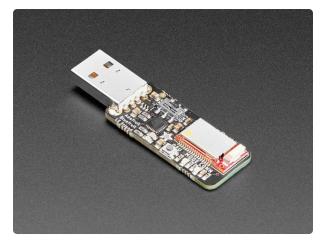
Here's a summary of all the parts needed:

- 1. The actual BLE sniffing hardware. This guide uses the Adafruit Bluefruit LE Sniffer with V2 firmware ().
- 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 <u>Virtual COM Port</u> driver () is needed.
- 3. The BLE sniffing plugin uses Python ().
- 4. To talk to the virtual com port from Python, the <u>pyserial module</u> () 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 <u>nRF Sniffer</u> 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

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...

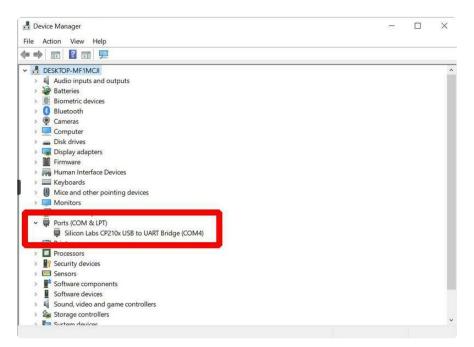
https://www.adafruit.com/product/2269

#### 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:



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.



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:

•	• •			The	Wiresh	ark Net	work A	analyzer		
<u>F</u> ile	<u>E</u> dit <u>V</u> i	ew <u>G</u> o	<u>C</u> apture	<u>A</u> nalyze	<u>S</u> tatist	cs Tele	phony	<u>W</u> ireless	Tools	<u>H</u> elp
	5	۱ 🜔		Q Q	$\langle \rangle$	2 1			0	1
A	oply a dis	olay filte	r <ctrl-< td=""><th>/&gt;</th><th></th><th></th><td></td><td></td><td></td><td>🚽 🗸</td></ctrl-<>	/>						🚽 🗸
	We	lcome	to Wires	nark						
	Caj	oture								
	usi	ng this f	ilter: 📕 E	nter a cap	ture filte	er			• A	Il interfaces shown -
		enol Loopbac any bluetoot nflog	k: lo h-monitor							
	•	DisplayP Random systemd SSH rem	ort AUX cl packet ge Journal Ex ote captu	ure: ciscoo nannel mo nerator: r kport: sdjo re: sshdun te capture	nitor cap andpkt urnal np		auxmo	n 		
	Lea	arn								
	Use	r's Guid	e · Wiki	Quest	tions an	d Answe	ers ·	Mailing L	ists.	
	You a	are runn	ing Wiresł	nark 3.2.3	(Git v3.2	.3 packa	ged as	3.2.3-1).		
Z	Ready to	load or	capture					No Packe	ts	Profile: Default

#### 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:

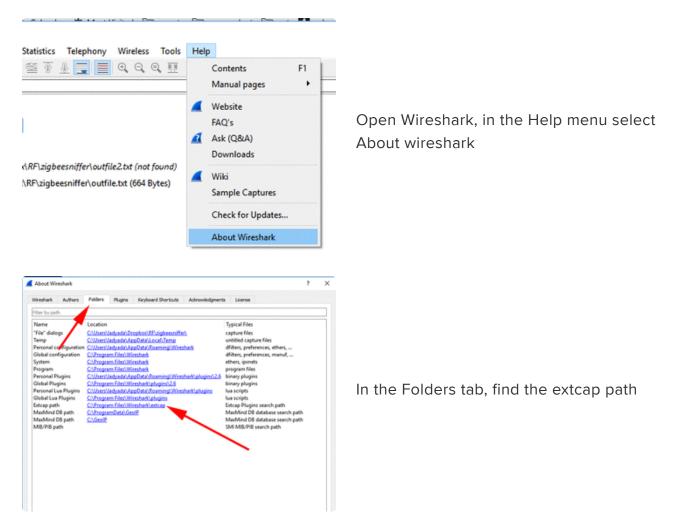
nRF Sniffer for Bluetooth LE

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

v4.x.x	Selected version 4.0.0 nrf_sniffer_for_bluetooth_le_4.0.0.zip	
	Changelog: • 4.0.0	~

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:

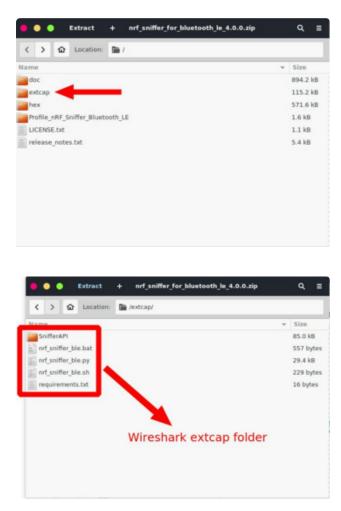


We'll refer to this folder location as the Wireshark extcap folder.

#### 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!

	The Wireshark	Network Analyzer		
e <u>E</u> dit <u>V</u> iew <u>G</u> o	Capture Analyze Statistics	Telephony Wireless	Tools Help	
I d 🛛 🎴	TR CQ()3			
Apply a display filter	<ctrl-></ctrl->			
Welcome to	Wireshark			
Capture				
using this filt	ter: 📕 Enter a capture filter		* All interfaces sh	iown -
enol				-
Loopback	i lo			
any				
bluetooth-	monitor			
nnog				
nRF Sniffe	r for Bluetooth LE: /dev/ttyUSB	0-3.2		
DisplayPor	rt AUX channel monitor capture	e: dpauxmon		
	acket generator: randpkt			
	ournal Export: sdjournal			
SSH remote	te capture: sshdump			-
Learn				
	· Wiki · Questions and A	anness a Malling I li	a ta	
	Wiki · Questions and A	-	sts	
	a test and the set of			
	g Wireshark 3.2.3 (Git v3.2.3 p	ackaged as 3.2.3-1).		
You are runnin		No Packets	P	ofile: Default
			s Pr	ofile: Default
You are runnin			s Pr	ofile: Default
You are runnin			s Pr	ofile: Default
You are runnin			s Pr	ofile: Default
You are runnin Ready to load or ca		No Packets		ofile: Default
You are runnin Ready to load or ci	apture	No Packets or Bluetooth LE: /dev	1tyU580-3.2	ofile: Default
You are runnin Ready to load or ci	apture Capturing from nRF Sniffer fr Capture Analyze Statistics	No Packets or Bluetooth LE: /dev	1tyU580-3.2	ofile: Default
You are runnin Ready to load or ci	apture Capturing from nRF Shiffer f Gapture Analyze Statistics	No Packets or Bluetooth LE: /dev	ttyUSB0-3.2 Tools Help	ofile: Default
You are runnin Ready to load or co le Edit View Go Apply a display fitte	Septuring from nRF Sniffer f Septuring from nRF Sniffer f Septure Analyze Statistics	No Packets or Bluetooth LE: /dev Telephony Wireless	ttyUSB0-3.2 Tools Help	•••
You are runnin Ready to load or co le Edit Yew Go Apply a display filte 3. Time 3.042 76.434719	apture Capturing from nBT Sniffer f gapture Analyze Statistics Capture D Gauta Statistics Capture C	No Packets	ttyUSB0-3.2 Tools Help I III Protocol Lengthinfo LE LL 63 ADV	
You are runnin Ready to load or cr lie Edit View So Acoly a display file 1002 79, 414719 1003 79, 414719	Capturing from this shiller fo Capture Analyze Statistics Capture Analyze Statistics Capture Analyze Statistics Capture Analyze Statistics Source Calify Source Calif	No Packets	ttyUSB0-3.2 Tools Help O m A II Protocol Lengthinfo LE LL 63 ADV	NONCOMN INC
You are runnin Ready to load or cr le Edit Yew Go Apply a display filte a. Tune 1962 76:413719 1963 75:518969 1964 76:625986	Capturing from nAF Sniffer fr gapture Analyze Statistics Concerning Concerning Concerning Source D Source D Source D Source D Source Statistics B Statistics B St	No Packet: or Bluetooth LE: /dev Telephony Wireless in in in in in in in in in iteration readcast readcast	ttyUSB0-3.2 Tools Help m m f ff Protocol Lengthinfo LE LL 63 ADV LE LL 63 ADV	NONCONN IND NONCONN IND NONCONN IND NONCONN IND
You are runnin Ready to lead or cr le Edit View So Edit View So Apply a display filte	Capituring from nRF Shiller fr Capiture Analyze Statistics Capiture Analyze Statistics Capiture Analyze Statistics Capiture Analyze Statistics Source D Statistics Capital Statistics Statistics Capital S	No Packets	ttyUS80-3.2 Tools Help Com C III Protocol Lengthinfo LE LL 63 ADV LE LL 63 ADV LE LL 63 ADV	NONCONN INC NONCONN INC NONCONN INC NONCONN INC
You are runnin Ready to load or cr le Edit Yew Go Apply a display filte a. Tune 1962 76:413719 1963 75:518969 1964 76:625986	Capturing from nAE Suffler f Sapture Analyze Statistics Capture Analyze Statistics	No Packet: or Bluetooth LE: /dev Telephony Wireless in in in in in in in in in iteration readcast readcast	TryUSBO3.2 Tools Help Trotocol Lengthinfd LE LL 63 ADV LE LL 63 ADV LE LL 63 ADV LE LL 63 ADV	NONCONN INC NONCONN INC NONCONN INC

1: 63 bytes on wire (504 bits), 63 bytes captured (504 bits) on interface /tmp/wi

rRF Sniffer for Blu\_apture in progress: Packets: 1068 · Displayed: 1068 (100.0%) Profile: Default

 0000
 9d
 06
 38
 01
 76
 05
 06
 0a
 01
 25
 48
 00
 00
 cc
 93
 01

 0010
 00
 d6
 ba
 80
 62
 25
 36
 63
 0e
 f6
 b1
 61
 1e
 ff
 06

 0020
 00
 01
 09
 20
 02
 18
 f6
 24
 7e
 8f
 bc
 26
 cc
 7d
 ad

 000
 01
 09
 20
 02
 18
 f6
 24
 7e
 8f
 bc
 25
 ad
 36
 34
 2a
 34
 bd
 2a
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 34
 <td

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.

	Equ	t <u>V</u> iev	v <u>G</u> o	Captu	re <u>A</u> n	alyze	Stati	istics	s Telephor	γ <u>W</u> ireles	s <u>T</u> ools <u>H</u> elp	2	
		6.0			2 8	Q	۲.	>	÷ ⊨ -			11	
l þi	ply i	a displ	ay filte	r «C	trl-/>								
<b>p</b> .		Time		So	urce			-	Destination		Protocol Le	ngth Info	

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('-')

#### 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

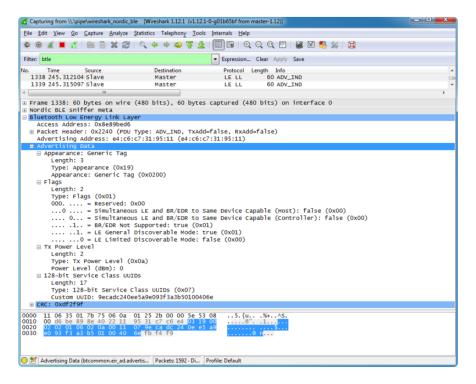
### Working with Wireshark

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:

		[Wireshark 1.12.1 (v1.12.1	-	master-1.12)]		
Eile <u>E</u> dit ⊻iew <u>G</u> o	<u>Capture</u> <u>Analyze</u> <u>S</u> t	atistics Telephony <u>I</u> ools	Internals Help			
D 🛈 📶 📕 🙋	BAX2	् 🗢 🔿 주 👱		Q Q 🖸 📅   🎬 🗎	1 🍢 🎇 🛛	
Filter: btle			<ul> <li>Expression</li> </ul>	Clear Apply Save		
o. Time	Source	Destination		Length Info		
210 38.0421390		Master	LE LL	60 ADV_IND		
211 38.0460030		Master	LE LL	60 ADV_IND		
212 38.0478100		Master	LE LL	60 ADV_IND		
213 38.5964040		Master	LE LL	60 ADV_IND		
214 38.5991960		Master	LE LL	60 ADV_IND		
215 38.6012380		Master	LE LL	60 ADV_IND		
216 39.1499950		Master	LE LL	60 ADV_IND		
217 39.1526170 218 39.1543130		Master	LE LL	60 ADV_IND		
218 39.1543130		Master	LE LL	60 ADV_IND		
219 39.69/1230		Master	LE LL LE LL	60 ADV_IND		
220 39.7004780		Master		60 ADV_IND		
221 39.7022140		Master	LE LL LE LL	60 ADV_IND		
223 40.2502480		Master	LE LL	60 ADV_IND 60 ADV_IND		
223 40.2502480		Master	LE LL	60 ADV_IND 60 ADV_IND		
225 40.8004240		Master Master	LE LL	60 ADV_IND 60 ADV IND		
226 40.8039290		Master	LE LL	60 ADV_IND		
227 40,8056880		Master	LE LL	60 ADV_IND		
228 41.3522920		Master	LE LL	60 ADV_IND		
229 41.3554480		Master	LE LL	60 ADV_IND		
230 41.3572300		Master	LE LL	60 ADV_IND		
230 41.33/2300	m	Mascer	LE LL	OU ADV_IND		
					-	
		bits), 60 bytes ca	ptured (480 b	vits) on interface	2 0	
Nordic BLE snif						
Bluetooth Low E	nergy Link Layer					
000 11 06 35 01		25 2b 00 00 26 6f		%+&0.		
		31 c7 c6 e4 03 19 9e ca dc 24 0e e5		· .1		
020 02 02 01 06	b5 01 00 40 66	fb f4 f9	a9			
020 02 02 01 06 030 e0 93 f3 a3						
020 02 02 01 06 030 e0 93 f3 a3						
020 02 02 01 06 030 e0 93 f3 a3						
020 02 02 01 06 030 e0 93 f3 a3						

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 OdBm, 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:

		12.1-0-g01b65bf from master-1.12)]	
ile <u>E</u> dit <u>V</u> iew <u>G</u> o <u>C</u> apture <u>A</u>	nalyze Statistics Telephony Ic	pols Internals Help	
D 🛛 🛋 🗖 🧖 🗎 🗎 🗎	🕷 🔁   🔍 🗢 🌳 🎝 春	노   🗐 🕞   오, 오, 환, 🖭   🕁 🗹 畅 %   💢	
ilter:		<ul> <li>Expression Clear Apply Save</li> </ul>	
o. Time Source	Destination	Protocol Length Info	
3890 713.339295 slave	Master	LE LL 60 ADV_IND	
3891 713.340712 slave	Master	LE LL 38 SCAN_REQ	
3892 713.342455 slave	Master	LE LL 38 SCAN_RSP	
3893 713.343931 slave	Master	LE LL 60 ADV_IND	
3894 713.883519 slave 3895 713.886540 slave	Master	LE LL 60 ADV_IND LE LL 60 ADV IND	
3895 713.888285 slave	Master	LE LL 60 ADV_IND	
	Mascel	LE LL OU ADV_IND	
			•
		tes captured (304 bits) on interface 0	
Nordic BLE sniffer meta			
Bluetooth Low Energy Li Access Address: 0x8e8			
Advertising Address: Scan Response Data: 0 Advertising Data Device Name: UART		5:c7:31:95:11)	
Scan Response Data: 0 Advertising Data	me (0x09)	5:c7:31:95:11)	
<ul> <li>Scan Response Data: 0</li> <li>Advertising Data</li> <li>Device Name: UART Length: 5</li> <li>Type: Device Na</li> </ul>	me (0x09)	5:c7:31:95:11)	
<ul> <li>Scan Response Data: 0</li> <li>Advertising Data</li> <li>Device Name: UART Length: 5</li> <li>Type: Device Na Device Name: UA</li> </ul>	me (0x09)	5:c7:31:95:11)	
Scan Response Data: 0 Advertising Data Device Name: UART Length: 5 Type: Device Na Device Name: UA	me (0x09)	s:c7:31:95:11)	
Scan Response Data: 0 Advertising Data Device Name: UART Length: 5 Type: Device Na Device Name: UA	me (0x09)	;c7:31:95:11)	
<ul> <li>Scan Response Data: 0</li> <li>Advertising Data</li> <li>Device Name: UART Length: 5 Type: Device Na Device Name: UA</li> </ul>	me (0x09)	;c7:31:95:11)	
Scan Response Data: 0 Advertising Data Device Name: UART Length: 5 Type: Device Na Device Name: UA	me (0x09)	s:c7:31:95:11)	
Scan Response Data: 0 Advertising Data Device Name: UART Length: 5 Type: Device Na Device Name: UA	me (0x09)	5:c7:31:95:11)	
<ul> <li>Scan Response Data: 0</li> <li>Advertising Data</li> <li>Device Name: UART Length: 5 Type: Device Na Device Name: UA</li> </ul>	me (0x09)	5:c7:31:95:11)	
Scan Response Data: 0 Advertising Data Device Name: UART Length: 5 Type: Device Na Device Name: UA	me (0x09)	s:c7:31:95:11)	
<ul> <li>Scan Response Data: 0</li> <li>Advertising Data</li> <li>Device Name: UART Length: 5</li> <li>Type: Device Na Device Name: UA</li> </ul>	me (0x09)	5:c7:31:95:11)	
Scan Response Data: 0 ⇒ Advertising Data ⇒ Device Name: UART Length: 5 Type: Device Na Device Name: UA ■ CRC: 0x68a4fc	50955415254 me (0x09) RT		
Scan Response Data: 0 advertising Data □ Device Name: UAR Units of the Units of Units □ CRC: 0x68a4fc	6 03 01 26 25 00 09 97	00.00	
□ Scan Response Data: 0 □ Advertising Data □ Device Name: UART Length: 5 Type: Device Name: UA Device Name: UART Device Name: UART 0 CRC: 0x68a4fC	50955415254 me (0x09) RT	00.00	
□ Scan Response Data: 0 □ Advertising Data □ Device Name: UART Length: 5 Type: Device Name: UA Device Name: UART Device Name: UART 0 CRC: 0x65a4fc	6 03 01 26 25 00 09 97	00 00 0 4+	
© Scan Response Data: 0 ⊂ Advertising Data ⊡ Device Name: UART Length: 5 Type: Device Name: UA @ CRC: 0x68a4fc 0000 11 06 1f 01 75 7f 0 010 00 06 be 89 84 40 0	6 03 01 26 25 00 09 97	00 00 0 4+	
© Scan Response Data: 0	6 03 01 26 25 00 09 97	00 00 0 4+	

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:

Capturing from \\.\pipe\wireshark_nordic_b	le [Wireshark 1.12.1 (v1.12	2.1-0-g01b65bf from r	naster-1.12)]	
<u>Eile Edit View Go</u> Capture Analyze	Statistics Telephony Ioc	ols Internals Help		
	0 🖕 🔿 🚮 🕇		Q Q 🖻 📓 M 🛼 🐝 🕽	an a
	~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~			38
Filter:		<ul> <li>Expression</li> </ul>	Clear Apply Save	
o. Time Source	Destination	Protocol	ength Info	
043\ TTAO'TOADO 214A6	mas Lei	LE LL	OU ADV_IND	
6498 1190.19107 Slave	Master	LE LL	60 ADV_IND	
6499 1190.73693 slave	Master	LE LL	60 ADV_IND	
6500 1190.73983 Slave	Master	LE LL	60 ADV_IND	
6501 1190.74166 Slave	Master	LE LL	60 ADV_IND	
6502 1191.29056 slave	Master	LE LL	60 ADV_IND	
6503 1191.29414 slave	Master	LE LL	60 ADV_IND	
6504 1191.29600 slave	Master	LE LL	60 ADV_IND	
6505 1191.84026 slave	Master	LE LL	60 ADV_IND	
6506 1191.84456 Slave	Master	LE LL	60 ADV_IND	
6507 1191.84624 Slave	Master	LE LL	60 ADV_IND	
6508 1192.40771 slave	Master	LE LL	60 ADV_IND	
6509 1192.41270 Slave	Master	LE LL	60 CONNECT_REQ	
6510 1192.45658 Master	slave	LE LL	26 Empty PDU	
6511 1192.45809 slave	Master	LE LL	26 Empty PDU	
6512 1192.50610 Master	slave	ATT	35 Rcvd Write Request, H	landle: 0x001e
6513 1192. 50758 slave	Master	LE LL	26 Empty PDU	
6514 1192.60330 Master	slave	LE LL	26 Empty PDU	
6515 1192.60471 slave	Master	LE LL	26 Empty PDU	
6516 1192.69980 Master 6517 1192.70140 Slave	Slave	LE LL	26 Empty PDU	
6517 1192.70140 STave 6518 1192.74868 Master	Master	LE LL	26 Empty PDU	
6518 1192.74868 Master	slave	LE LL	26 Empty PDU	
	(480 bits), 60 byte	es captured (48	0 bits) on interface 0	
	or			
Bruecooch Low Energy Link Lay	ei			
B Frame 6509: 60 bytes on wire B Nordic BLE sniffer meta B Bluetooth Low Energy Link Lay				80 bits), 60 bytes captured (480 bits) on interface 0
	01 25 2c 00 00 98 0		%,	
	02 a2 0b 22 ac 11 9 2a c4 02 09 00 27 0		k"1	
	2a C4 02 09 00 27 0 b0 c9 cb 8f	0 00u		
	31			
	ture i Dacketr: 7285 Di	Profile: Default		

#### 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).

	e [Wireshark 1.12.1 (v1.12.1		131C1 1.12/j	
ile <u>E</u> dit <u>V</u> iew <u>G</u> o <u>C</u> apture <u>A</u> nalyze g	tatistics Telephony <u>T</u> ools	Internals Help		
) () 🛋 📕 🔬 🗎 🗎 🗙 😕 (	् 🗢 🕈 😜 🗿 💈 🕹		0, 0, 🖸   🗃 🖻 🥵 🔆   💢	
ilter:		<ul> <li>Expression</li> </ul>	Clear Apply Save	
o. Time Source	Destination	Protocol Le	ength Info	
6508 1192.40771 slave	Master	LE LL	60 ADV_IND	
6509 1192.41270 slave	Master	LE LL	60 CONNECT_REQ	
6510 1192.45658 Master	slave	LE LL	26 Empty PDU	
6511 1192.45809 slave	Master	LE LL	26 Empty PDU	
6512 1192.50610 Master	slave	ATT	35 Rcvd Write Request, Handle: 0x001e	
6513 1192.50758 slave	Master	LE LL	26 Empty PDU	
6514 1192.60330 Master	slave	LE LL	26 Empty PDU	
6515 1192.60471 slave	Master	LE LL	26 Empty PDU	
6516 1192.69980 Master	slave	LE LL	26 Empty PDU	
6517 1192.70140 slave	Master	LE LL	26 Empty PDU	
m				,
Access Address: 0x84d81175 Data Header: 0x0902 CRC: 0x32acff Bluetooth L2CAP Protocol Length: 5 CID: Attribute Protocol (0x)	0004)			
Bluetooth Attribute Protocol				
Opcode: Write Request (0x12) Handle: 0x001e Value: 0100	)			
000 11 06 1c 01 b2 89 06 0a ( 010 00 75 11 d8 84 02 09 05 ( 020 4c 35 ff	03 0b 2b 02 00 37 bd 00 04 00 <u>12 1e 00 01</u>		···*··7··	

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 '<u>Client Characteristic</u> 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':

Capturing from \\.\pipe\wireshark_nordic_	ble [Wireshark 1.12.1 (v1.12.1	-0-g01b65bf from n	naster-1.12)]	
<u>File Edit View Go Capture Analyze</u>	Statistics Telephony Iools	Internals <u>H</u> elp		
● ● ∡ ■ ∡   ⊨   × 2	् 🗢 🕈 😜 🖥 🛂		Q. B. 🖻 📓 🕺 🥵 % 🛛 🧱	
Filter:		<ul> <li>Expression</li> </ul>	Clear Apply Save	
lo. Time Source	Destination	Protocol L	ength Info	
6508 1192.40771 slave	Master	LE LL	60 ADV_IND	
6509 1192.41270 slave	Master	LE LL	60 CONNECT_REQ	
6510 1192.45658 Master	slave	LE LL	26 Empty PDU	
6511 1192.45809 slave	Master	LE LL	26 Empty PDU	
6512 1192.50610 Master	slave	ATT	35 Rcvd Write Request, Handle: 0x00	1e
6513 1192.50758 slave	Master	LE LL	26 Empty PDU	
6514 1192.60330 Master	slave	LE LL	26 Empty PDU	
6515 1192.60471 slave	Master	LE LL	26 Empty PDU	
6516 1192.69980 Master	slave	LE LL	26 Empty PDU	
6517 1192.70140 slave	Master	LE LL	26 Empty PDU	
6518 1192.74868 Master	slave	LE LL	26 Empty PDU	
6519 1192.74986 Slave	Master	LE LL	26 Empty PDU	
6520 1192.79803 Master	Slave	LE LL	26 Empty PDU	
6521 1192.79943 slave	Master	LE LL	26 Empty PDU	
6522 1192.84875 Master	Slave	LE LL	26 Empty PDU	
6523 1192.85002 slave	Master	LE LL	26 Empty PDU	
6524 1192.94425 Master	slave	LE LL	26 Empty PDU	
6525 1192.94553 Slave	Master	LE LL	26 Empty PDU	
6526 1193.04087 Master	slave	LE LL	26 Empty PDU	
6527 1193.04208 slave	Master	LE LL	26 Empty PDU	
6528 1193.13933 Master	Slave	LE LL	26 Empty PDU	
6529 1193.14064 slave	Master	LE LL	26 Empty PDU	
6530 1193.23667 Master	Slave	LE LL	26 Empty PDU	
				,
Frame 6519: 26 bytes on wire Nordic BLE sniffer meta	(208 bits), 26 bytes	captured (20	8 bits) on interface O	
Bluetooth Low Energy Link La	10.0			
Access Address: 0x84d81175	yer			
Data Header: 0x0009				
CRC. OXDJETJJ				
0000 11 06 13 01 b9 89 06 0a	09 11 2b 07 00 99 00	00	+	
0010 00 75 11 d8 84 09 00 ad	f7 aa	.u	• • •	
🕽 💅 \\.\pipe\wireshark_nordic_ble: <live ca<="" td=""><td>pture i Packets: 35555 ·</td><td>Profile: Default</td><td></td><td></td></live>	pture i Packets: 35555 ·	Profile: Default		

#### 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):

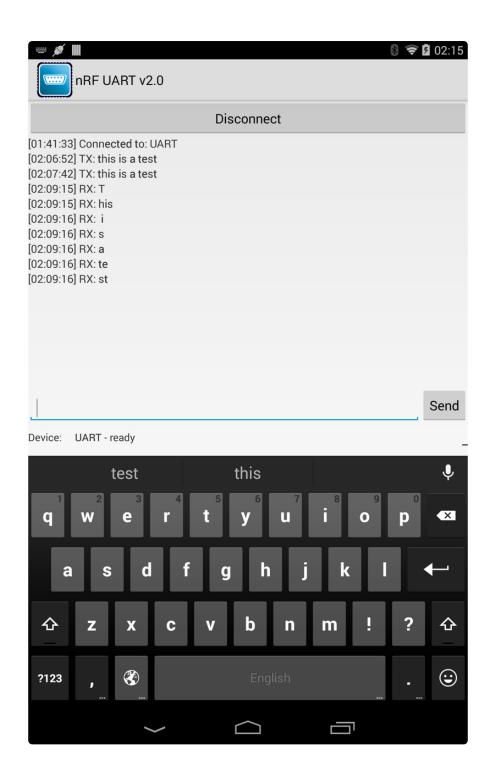
The For	it <u>V</u> iew <u>G</u> o <u>C</u> apture <u>A</u>	Analyze Statistics Telephony Tool	s Internals <u>H</u> elp		
• •	🛋 🔳 🔬 i 🖻 🗎	x ≳   < + + + • 7 4		. 0, 0, 🖭   👹 🖻 🥵 🔆   💢	
Filter:			<ul> <li>Expression</li> </ul>	. Clear Apply Save	
lo.	Time Source	Destination	Protocol	Length Info	
55593	2854.95085 slave	Master	LE LL	26 Empty PDU	
55594	2854.99892 Master	slave	LE LL	26 Empty PDU	
55595	2855.00055 slave	Master	ATT	34 Rcvd Handle Value Notification, Handle: 0x00	1c
	2855.04974 Master	slave	LE LL	26 Empty PDU	
	2855.05107 slave	Master	LE LL	26 Empty PDU	
	2855.09619 Master	slave	LE LL	26 Empty PDU	
	2855.09776 slave	Master	LE LL	26 Empty PDU	
	2855.19274 Master	slave	LE LL	26 Empty PDU	
	2855.19395 slave	Master	LE LL	26 Empty PDU	
	2855.29033 Master	slave	LE LL	26 Empty PDU	
	2855.29210 slave	Master	LE LL	26 Empty PDU	
	2855.33914 Master	Slave	LE LL	26 Empty PDU	
	2855.34070 slave	Master	LE LL	26 Empty PDU	
	2855.38816 Master	slave	LE LL	26 Empty PDU	
55607	2855.38971 Slave	Master	LE LL	26 Empty PDU	
Acc Dat	ic BLE sniffer meta tooth Low Energy Li cess Address: 0x840 ta Header: 0x080a 000 = RFU: 0	ink Layer d81175			
Acc Dat (	tooth Low Energy Li cess Address: 0x84 ta Header: 0x080 000 = RFU: 0 0 = More D: 10 = Sequend 10 = Next E: 000 = RFU: 0 0 1000 = Length: c: 0x6e97ce	ink Layer Jäll75 ata: False Ce Number: True spected Sequence Number: F Start of an L2CAP message ( : 8		L2CAP message with no fragmentation (0x02)	
Acc Dat ( C C Bluet	tooth Low Energy Licess Address: 0x840           cess Address: 0x840           ta Header: 0x080a           000 = RFU: 0           = More Di           = More Di           = Sequend	ink Layer Jäll75 ata: False Ce Number: True spected Sequence Number: F Start of an L2CAP message ( : 8		L2CAP message with no fragmentation (0x02)	
Acc Dat 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	tooth Low Energy Li cess Address: 0x84 ta Header: 0x080 000 = RFU: 0 0 = More D: 10 = Sequend 10 = Next E: 000 = RFU: 0 0 1000 = Length: c: 0x6e97ce	ink Layer dB1175 ata: False Ce Number: True spected Sequence Number: F. start of an L2CAP message ( : 8		L2CAP message with no fragmentation (0x02)	
Acc Dat ( CR Bluet Ler CII Bluet Ler CII Bluet Har	tooth Low Energy L1 cess Address: 0x844 ta Header: 0x080a erst address: 0x844 erst e	ink Layer 181175 ata: False ce Number: True Nyected Sequence Number: F. Start of an L2CAP message of an L2CAP message of the second second second second second scol (0x0004)		L2CAP message with no fragmentation (0x02)	
Acc Dat Dat CR Blue Blue CR Blue CR CR CR CR CR CR CR CR CR CR	tooth Low Energy L1 cess Address: 0x84 ta Header: 0x080a 000 = RFU: 0 0 More Di 1 = Sequenn 0 = Next Es 0 Next Es 0 0 LLID: 5 000 = RFU: 0 0 0 1000 = Length. 0: Attribute Protoco ngth: 4 0: Attribute Protoco ngth: 4 0: Attribute Protoco code: Handle Value ndle: 0x001c lue: 54	ink Layer JBI175 ata: False ce Number: True spected Sequence Number: F. Start of an L2CAP message ( : 8 bl col (0x0004) otocol Notification (0x1b)	or a complete	L2CAP message with no fragmentation (0x02)	

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):

		s Internals <u>H</u> elp	
			0, 0, 🗹   🗃 🗹 🥵 %   🉀
Filter		<ul> <li>Expression</li> </ul>	Clear Apply Save
lo. Time Source	Destination		ength Info
55620 2855.82728 Master	Slave	LE LL LE LL	26 Empty PDU 26 Empty PDU
55621 2855.82924 slave	Master	ATT	35 Rcvd Handle Value Notification, Handle: 0x001c
55622 2855.87881 Master	slave	LE LL	26 Empty PDU
55623 2855.88012 slave	Master	LE LL	26 Empty PDU
55624 2855.97345 Master	slave	LE LL	26 Empty PDU
55625 2855, 97489 slave	Master	LE LL	26 Empty PDU
55626 2856.02266 Master	slave	LE LL	26 Empty PDU
55627 2856.02432 slave	Master	ATT	35 Rcvd Handle Value Notification, Handle: 0x001c
55628 2856.07305 Master	slave	LE LL	26 Empty PDU
55629 2856.07432 slave	Master	LE LL	26 Empty PDU
55630 2856.12020 Master	slave	LE LL	26 Empty PDU
55631 2856.12155 slave	Master	LE LL	26 Empty PDU
55632 2856.21800 Master	slave	LE LL	26 Empty PDU
55633 2856.21981 slave	Master	ATT	35 Rcvd Handle Value Notification, Handle: 0x001c
	-1		AC PARAL AND A
□ Data Header: 0x0906 000 = RFU: 0 0 = More Data: F			
0 = Sequence Num 1. = Next Expecte 	nber: False ed Sequence Number: Th of an L2CAP message (0x0004)		L2CAP message with no fragmentation (0x02)
0 = Sequence Num 1 = Next Expecte 10 = LLID: Start 000 = RFU: 0 0101 = Length: 9 © CRC: 0x183F56 Bluetooth L2CAP Protocol Length: 5 CID: Attribute Protocol (C Bluetooth Attribute Protocol (C	nber: False ed Sequence Number: Th of an L2CAP message (0x0004)		L2CAP message with no fragmentation (0x02)
0 = Sequence Num 1. = Next Expecte 	nber: False dd Sequence Number: Ti of an L2CAP message (x00004) Fication (0x1b)	or a complete	
0 = Sequence Num 1 = Next Expecte 10 = LLID: Start 000 = RFU: 0 0101 = Length: 9 Bluetooth LZCAP Protocol Eluetooth LZCAP Protocol CID: Attribute Protocol Opcode: Handle value Notif Handle: 0x001c	nber: False de Sequence Number: Ti of an L2CAP message (x0004) fication (0x1b) 01 10 2b 42 85 97 01	or a complete :	L2CAP message with no fragmentation (0x02)

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 <u>https://www.nordicsemi.com/</u> <u>eng/Products/Bluetooth-low-energy/nRF-Sniffer</u> () 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:

Changelog:	
• 2.0.0-beta-1 Multi-Platform	~
2.0.0-beta-2 Multi-Platform	~
2.0.0-beta-3 Multi-Platform	~

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 <a href="https://www.wireshark.org/">https://www.wireshark.org/</a> () 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:

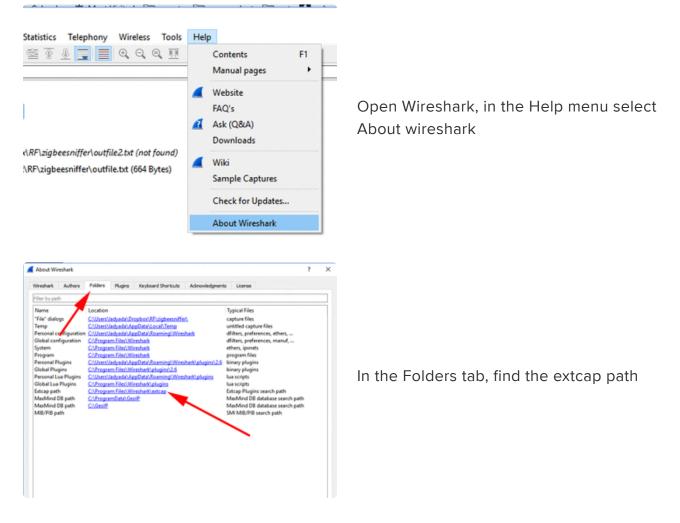


If you decide to go there, be sure to select the correct version for download.

2.0.0-beta-1 Multi-Platform	$\sim$
2.0.0-beta-2 Multi-Platform	~
2.0.0-beta-3 Multi-Platform	~

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 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.p y and SnifferAPI folder.



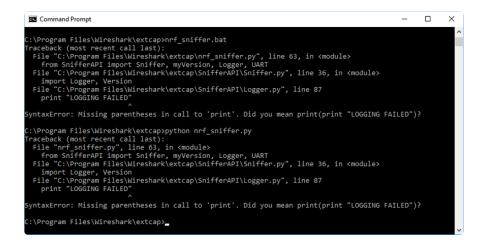
Now quit Wireshark so we can get things tested!

## 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



or the "No module named 'Logger'" error



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" %*
```

## 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



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%  ###################################
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 installupgrade pip' command.
C:\Program Files\Wireshark\extcap>

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



### Test Capture

OK finally once that works, start Wireshark again.

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

Welcome to Wireshark	
Open	
C:\Users\ladyada\Dropbox\RF\zigbeesniffer\outfile2.txt (not found)	
C:\Users\ladyada\Dropbox\RF\zigbeesniffer\outfile.txt (664 Bytes)	
Capture	
using this filter: 📙 Enter a capture filter	▼ All interfaces shown ▼
Local Area Connection 4 J	
nRE Sniffer COM63	

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)

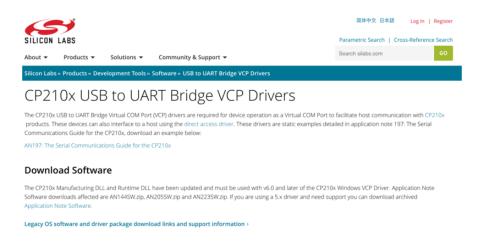


Again, we don't have a guide or tutorial on loading the firmware onto an nRF51. We don't have the original source code, that hex is from Nordic and they only release hex files

## **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 <u>CP2104 VCP</u> 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 <u>se</u> Contact <u>support1@ftdichip.com</u> driv	if looking to create cusomised	-	-				2.12.00 WHQL Certified Available as <u>setup executable</u> <u>Release Notes</u>
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 <u>TN-101</u> if you need a custom VCP VID/PID in Linux
Mac OS X	2012-08-10	2.2.18	2.2.18	<u>2.2.18</u>	-	-	-	-	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.

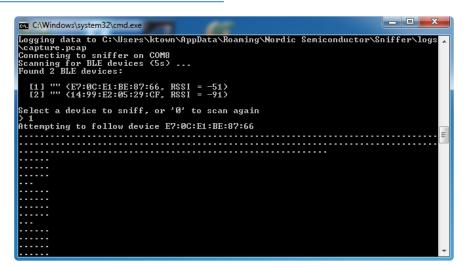
More information on using Nordic's nRF Sniffer application ().

arrow keys [#] or ENTER e w x/q c v b a p b a p b a p o b h s u	Start Wireshark, Exit Display filter: Display filter: Remove display f Passkey entry 00B key entry 0	vice list. Use Ef to sniff from 1 sniffer will on the primary vir Nearest devices Nearest devices Stearest devices ilter. Nop sequence. le (pdf)	TER to select.	
# public	: name	RSSI	device address	
[ ] 0 ''' [ ] 1 ''' [ ] 2 ''' Scanning for deu	vices.	-90 dBm -65 dBm -46 dBm	14:99:e2:05:29:cf 68:48:98:b8:e5:2b e4:c6:c7:31:95:11	public
Sent Key value t	o sniffer			-

#### 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 <u>nRF Sniffer product page downloads tab</u> (), then download the latest V1 version of the utility, and unzip it.

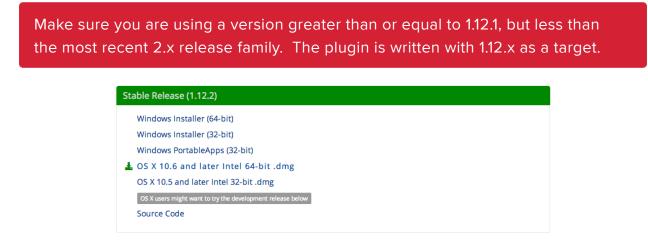
Inside this downloaded file you'll find the sniffer executable, which will open up the command-line tool when you click on it.

## **Getting Wireshark**

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

You may need to explore the download mirrors, such as  $\frac{https://1.na.dl.wireshark.org/}{1}$  () 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 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.

BLE Sniffer 1.0.1				
Commands: 1 arrow keys [#] or ENTER e w x/q c v b a p o h s u CTRL-R Available devic	Start Wireshark, Exit Display filter: Display filter: Remove display f Passkey entry 00B key entry 0	ice list. Use EN to sniff from li sniffer will onl the primary vie Nearest devices Nearest devices ilter. op sequence. e (pdf)	TER to select.	
# public	: name	RSSI	device address	
[ ] 0 "" [ ] 1 "" [ ] 2 "" Scanning for dev	vices.	-65 dBm	14:99:e2:05:29:cf 68:48:98:b8:e5:2b e4:c6:c7:31:95:11	public public random
Sent Key value t	o sniffer			-

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:

BLE Sniffer 1.0.1	version SUN rev	. 1111		
Commands: 1 arrow keys [#1] or ENTER e w ×/q c v b a p p b a p c C TRL-R Available devia	Start Wireshark, Exit Display filter: Display filter: Display filter: Remove display f Passkey entry OOB key entry OOB key entry Define new adv H Get support Launch User Guid Re-program firmw	<pre>ice list. Use Ef to sniff from 1: sniffer will on the primary vie Nearest devices Nearest devices Nearest devices ilter.</pre>	TER to select.	E.
# public		RSSI	device address	
[ ] 0 ''' [ ] 1 ''' -> [X] 2 ''' Sniffing device	2 - ""	-90 dBm -90 dBm -46 dBm		public public random

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.

## 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	5
released June 2017						

				usbseri	al-DN009	WNO			
Sniff	er Informa	tion							
►	Stat	tus: Sniffing <u< td=""><td>nknown&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td></u<>	nknown>						
	Packet Co	unt: 15 358							
	USB Dev	ice : FTDI - FT2	31X USB L	JART [ s/n: DN	009WNO ]				
Curr	ently Sniffi	ng							
	Device Na	me: <unknown< td=""><td>&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td></unknown<>	>						
	Addre	ess : dd:2b:48:7	f:b5:8c (ra	ndom)					
	RS	SSI: -54							
	Event Co	unt: 0							
Adve	tisers								
	me: <unkn ess: 46:8e:</unkn 	own> 86:db:97:c1 (ran	dom)						
	me: <unkn iss: dd:2b</unkn 	own> :48:7f:b5:8c (rand	iom)						
List	Devices	Sniff Device	Cap	oture to Wiresh	ark	Capture To F	CAP	Enter Passkey	

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.

	<u>Capture Analyze Stati</u>	stics Telephony Tool		[Wireshark 1.12.1 (v1.12.1-0-g01b65bf from master-1.12)]	
00 🛋 🗖 🙋	🖿 🗎 🗙 😂   9	l 🔶 🌳 🤏 🗿 🛃		0, 🖭   🕁 🔟 🎦 🏂   🔛	
Filter:		<ul> <li>Expression.</li> </ul>	. Clear Apply Save		
No. Time	Source	Destination	Protocol Leng	th Info	
78 13.758362000	dd:2b:48:7f:b5:8c	<pre><broadcast></broadcast></pre>	LE LL	56 ADV_IND	
79 14.302589000		 dcast>	LE LL	56 ADV_IND	
80 14.303255000		 dcast>	LE LL	56 ADV_IND	
81 14.303919000		 dcast>	LE LL	56 ADV_IND	
82 14.849416000		 dcast>	LE LL	56 ADV_IND	
83 14.850082000		 dcast>	LE LL	56 ADV_IND	
84 14.850746000		 dcast>	LE LL	56 ADV_IND	
85 15.401586000		 dcast>	LE LL	56 ADV_IND	
86 15.402252000		 dcast>	LE LL	56 ADV_IND	
87 15.402916000		 dcast>	LE LL	56 ADV_IND	
88 15.948595000		<pre><broadcast></broadcast></pre>	LE LL	56 ADV_IND	
89 15.949262000		 dcast> 	LE LL	56 ADV_IND	
90 15.949926000		<pre><broadcast> <broadcast></broadcast></broadcast></pre>	LE LL	56 ADV_IND	
91 16.499165000	dd:2b:48:7f:b5:8c	<pre>  droadcast&gt;</pre>	LE LL	56 ADV IND	
	n wire (448 bits), 56 b	ytes captured (448 bits)	on interface 0		
Nordic BLE Sniffer M Bluetooth Low Energy Access Address: 00 b Packet Header: 0x Advertising Addres Tadvertising Data b Flags b Tx Power Level v 128-bit Service Length: 17 Type: 128-bit	teta Link Layer (4689bed6 Le40 (PDU Type: ADV_IND, ss: dd:2b:48:7f:b5:8c (d Class UUIDs Service Class UUIDs (6x	TxAdd=false, RxAdd=falt d:2b:48:7f:b5:8c) x07)	on interface 0		
Nordic BLE Sniffer ► Bluetooth Low Energy Access Address: 0 ▷ Packet Header: 0x: Advertising Addres ♥ Advertising Data ▷ Flags ▷ Tx Power Level ♥ 128-bit Service Length: 17 Type: 128-bit	keta r Link Layer Ge895bed6 Le40 (PDU Type: ADV_IND, ss: dd:2b:48:7f:b5:8c (d Class UUIDs	TxAdd=false, RxAdd=falt d:2b:48:7f:b5:8c) x07)	on interface 0		
Nordic BLE Sniffer P Bluetooth Low Energy Access Address: 0 ▷ Packet Header: 0x Advertising Addre ♥ Advertising Data ▷ Flags ▷ Tx Power Level ▷ 128-bit Service Length: 17 Type: 128-bit Crites (BDD) ▷ CRC: 0x47e7e5	eta (IIIA: Layer (809306d6 Le40 (POU Type: ADV_IND, ss: dd:2b:40:7f:b5:8c (d Class UUIDs Service Class UUIDs (0) Pecado 2000 Service (13:10)	TxAdd=false, RxAdd=fals d:2b:48:7f:b5:8c) (07)	e)		
Nordic BLE Sniffer P Bluetooth Love Energy Access Address: 0; D Packet Header; 0x; Advertising Addre Advertising Data D Flags D Tx Power Level V 128-bit Service Length: 17 Type: 128-bit Crite Sliffer D CR: 0x47e765 D00 02 06 31 01 67 4	Meta Link Layer de050ed6 Led0 (PDU Type: ADV_IND, ss: dd:2b:40:7f:b5:8c (d Class UUIDs Service Class UUIDs (d) 1 06 0a 01 27 35 00 00 1 06 0a 01 27 35 00 00	TxAdd=false, RxAdd=fals d:2b:40:7f:b5:8c) (1004866 00 00 001.gA	on interface 0		

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.

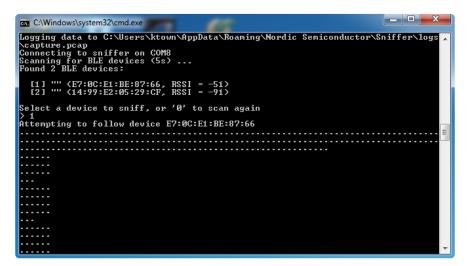
Nordic provides a Python API for their sniffer firmware that makes it possible for us to use the sniffer on any platform, and we've put together a basic wrapper for this API to help you get started.

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 <u>official Windows app</u> () 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 <u>Instaling Python and PySerial</u> ( ) 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 resembing the image below:

InifferAPI	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
尾 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
尾 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:

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
>
```

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
> 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:

File	pture.pcap [Wiresh	nark 1.12.1 (v1.12.1-0-	g01b65bf from master-1.12)		-			
_	Edit View Go	Capture Analyze	tatistics Telephony <u>T</u> ool	s Internals <u>H</u> elp				
0	• 🔺 🔳 🔬	🖹 🕌 🗶 🔁	् 🗢 🗢 😜 👍 🛓		ର୍ଷ୍ 🖭   🎬	8 🛃 💥	Ħ	
Filter	:			<ul> <li>Expression</li> </ul>	Clear Apply Save			
0.	Time	Source	Destination	Protocol	ength Info			
	228 13.347128		Master	LE LL	60 ADV_IND			
	229 13.348449		Master	LE LL	60 ADV_IND			
	230 13.893268		Master	LE LL	60 ADV_IND			
	231 13.896275		Master	LE LL	60 ADV_IND			
	232 13.898702 233 14.447219		Master Master	LE LL	60 ADV_IND 60 ADV_IND			
	234 14.449674		Master	LE LL	60 ADV_IND			
	235 14.451346		Master	LE LL	60 ADV_IND			
	236 14.998824		Master	LE LL	60 ADV_IND			
								+
Fr	ame 232: 60 b	ovtes on wire (4	180 bits), 60 bytes	captured (480	bits)			
NO	ordic BLE snif	fer meta						
		nergy Link Lay	in .					
		ss: 0x8e89bed6		4 6-1 August	6-1)			
			Type: ADV_IND, TXAd		=raise)			
	Advertising D		e1:be:87:66 (e7:0c:	el:De:8/:00)				
	Appearance:	Generic Tag						
	Length: 3	3						
		pearance (0x19)						
		ce: Generic Tag	(0x0200)					
	🗆 Flags							
	Length: 2							
		ags (0x01)	.00					
		= Reserved: 0	S LE and BR/EDR to :	Samo Dovico Ca	pable (Host): fr	100 (000	2	
			S LE and BR/EDR to :					
			Supported: true (Ox		pable (Controlle	r): rais	e (0x00)	
			Discoverable Mode: 1					
			Discoverable Mode: 1	false (0x00)				
	TX Power Le		Discoverable Mode: 1	false (0x00)				
	Tx Power Le Length: 2	evel	Discoverable Mode:	false (0x00)				
	Length: 2 Type: Tx	evel 2 Power Level (0)		false (0x00)				
	Length: 2 Type: Tx Power Lev	evel 2 Power Level (0) /el (dBm): 0	x0a)	false (0x00)				
	Length: 2 Type: Tx Power Lev ⊡ 128-bit Ser	evel 2 Power Level (0 /el (dBm): 0 rvice Class UUII	x0a)	false (0x00)				
	Length: 2 Type: Tx Power Lev ⊡ 128-bit Ser Length: 1	evel 2 Power Level (02 vel (dBm): 0 vvice Class UUIG L7	x0a) OS	false (0x00)				
	Length: 2 Type: Tx Power Lev ⊡ 128-bit Ser Length: 1 Type: 128	evel Power Level (0) vel (dBm): 0 vvice Class UUII 17 3-bit service C	xOa) DS lass UUIDS (OxO7)					
	Length: 2 Type: Tx Power Lev E 128-bit Ser Length: 1 Type: 128 Custom UU	evel Power Level (0: vel (dBm): 0 vvice Class UUII 7 3-bit Service C JID: 9ecadc240ed	x0a) OS					
	Length: 2 Type: Tx Power Lev E 128-bit Ser Length: 1 Type: 128 Custom UL CRC: 0xf9229f	evel Power Level (0) vel (dBm): 0 vvice Class UUII 7 3-bit Service C JID: 9ecadc240ee	x0a) D5 lass UUID5 (0x07) e5a9e093f3a3b501004					
E	Length: 2 Type: Tx Power Lew 128-bit Ser Length: 1 Type: 128 Custom UL CRC: 0xf9229f □ [Expert Inf	evel Power Level (0) vel (dBm): 0 vvice Class UUII 7 3-bit Service C JID: 9ecadc240e fo (Chat/Protoco	x0a) D5 lass UUID5 (0x07) e5a9e093f3a3b501004					
	Length: 2 Type: Tx Power Lev E 128-bit Ser Length: 1 Type: 128 Custom UL CRC: 0xf9229f [Expert Inf [correct]	evel 2 Power Level (0; rel (dBm): 0 vice Class UUII 17 3-bit Service C UID: 9ecadc240ee 5 6 (Chat/Protoco	x0a) D5 lass UUID5 (0x07) e5a9e093f3a3b501004					
	Length: 2 Type: Tx Power Lev E 128-bit Ser Length: 1 Type: 128 Custom UL CRC: 0xf9229f [Expert Inf [correct]	evel Power Level (0: /el (dBm): 0 vice Class UUII /7 3-bit Service C UID: 9ecadc240ee fo (Chat/Protoco / level: Chat]	x0a) D5 lass UUID5 (0x07) e5a9e093f3a3b501004					
₿	Length: 2 Type: Tx Power Lev E 128-bit Ser Length: 1 Type: 128 Custom UL CRC: 0xf9229f E [Expert Inf [correct] [Severity [Group: P	avel Power Level (0) vel (ddm): 0 vice class UUII J. 	KOQ) DS 1ass UUIDS (0x07) 25a9e093f3a3b501004 0]): correct]	06e				
0000	Length: 2 Type: Tx Power Lev 128-bit Ser Length: 1 Type: 128 Custom UL CRC: 0xf9229f Expert Inf [correct] [Severity [Group: P	<pre>vel Power Level (0) vel (dsm): 0 vvice class uuu T bit service C bit service C c to (chat/Protocol v level: chat] dd 0b 06 0a (d to 0</pre>	<pre>k0a) 35 lass UUIDS (0x07) 55a9e093f3a3b501004( 51): correct] 11 27 24 00 00 08 00</pre>	06e	· · '4 <u></u>			
0000	Length: 2 Type: Tx Power Lev 128-bit Ser Length: 1 Type: 128 Custom UL CRC: 0xf9229f Expert Inf [correct] [Severity [Group: P	<pre>vel Power Level (0) vel (dsm): 0 vvice class uuu T bit service C bit service C comparison of the service C co</pre>	<pre>k0a) 35 lass UUIDS (0x07) 55a9e093f3a3b501004( 51): correct] 11 27 24 00 00 08 00</pre>	06e	<sup>1</sup> 4			
E	Length: 2 Type: Tx Power Lev 128-bit Ser Length: 1 Type: 128 Custom UL CRC: 0xf9229f Expert Inf [correct] [Severity [Group: P	<pre>vel Power Level (0) vel (dsm): 0 vvice class uuu T bit service C bit service C comparison of the service C co</pre>	KOQ) DS 1ass UUIDS (0x07) 25a9e093f3a3b501004 0]): correct]	06e	· · · · · · · · · · · · · · · · · · ·			
D 000 010 020	Length: 2 Type: Tx Power Lev 128-bit Ser Length: 1 Type: 128 Custom UL CRC: 0xf9229f Expert Inf [correct] [Severity [Group: P	<pre>vel Power Level (0) vel (dsm): 0 vvice class uuu T bit service C bit service C comparison of the service C co</pre>	<pre>k0a) 35 lass UUIDS (0x07) 55a9e093f3a3b501004( 51): correct] 11 27 24 00 00 08 00</pre>	06e	f			
E 0000	Length: 2 Type: Tx Power Lev 128-bit Ser Length: 1 Type: 128 Custom UL CRC: 0xf9229f Expert Inf [correct] [Severity [Group: P	<pre>vel Power Level (0) vel (dsm): 0 vvice class uuu T bit service C bit service C comparison of the service C co</pre>	<pre>k0a) 35 lass UUIDS (0x07) 55a9e093f3a3b501004( 51): correct] 11 27 24 00 00 08 00</pre>	06e	f			

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 <u>notes on the official nRF</u> <u>Sniffer utility</u> (), 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

