Contact us

Introducing the Adafruit Bluefruit LE Friend

Created by Kevin Townsend

Last updated on 2017-05-02 08:36:44 PM UTC

Guide Contents

Introduction

The BLEFriend makes it easy to get any USB enabled device talking to your BLE enabled phone or tablet using a standard USB CDC connection.

Please note that there are three versions of this board. An older v1.0 blue PCB which uses 16KB SRAM parts and can run firmware 0.5.0 and lower. A newer v2.0 black PCB that uses the latest 32KB parts and can run old firmware plus version 0.6.2 and higher, based on the FTDI bridge and with an SWD connector. A cost-optimized v3.0 board that uses a CP2104 USB chip and drops the SWD connector. v3.0 can run all of the same firmware as v2.0, and is the latest board available.

In it's simplest form, it works on the same principle as a common USB/Serial adapter (the FTDI Friend (http://adafru.it/dQa), for example!). Any data that you enter via your favorite terminal emulator on your development machine will be transferred over the air to the connected phone or tablet, and vice versa.

Why Not Just Use a BLE USB Dongle?

Good question! You can get a Bluetooth 4.0 USB dongle (http://adafru.it/ecb) from the store already, and it's a useful tool to have (particularly on the Raspberry Pi or BBB), but that won't solve some problems out of the box.

To start with, Bluez (Linux) has a decent learning curve (and doesn't run on OS X if you're a Mac user). Windows 7 doesn't even support Bluetooth Low Energy, and on OS X you'll have to sort through the native Bluetooth APIs and development tools that require an annual paid license and specific license terms to access. There isn't a standard, open source, cross-platform way to talk BLE today.

With the BLEFriend, you can be up and running in under and hour on just about anything with a USB port, with easy migration across platforms anc operating systems. It's not perfect (it's currently a peripheral mode only solution), but it's the easiest way you'll find to get any USB device talking to your iOS or Android device.

So it's a Fancy Pants Wireless UART Adapter?

The board is capable of much more than simulating a basic UART bridge (and this is still early days for the Bluefruit LE board family)! Thanks to an easy to learn AT command set (http://adafru.it/iCO), you can also create you own basic GATT Services (http://adafru.it/iCP) and Characteristics, simulate Beacons (http://adafru.it/iCu), and change the way that the device advertises itself for other Bluetooth Low Energy devices to see.

To make sure that your device stays up to date and can benefit from the latest Bluefruit LE firmware from Adafruit, you can also update the firmware on your BLEFriend over the air (http://adafru.it/iCQ) using any supported iOS or Android device.

You can even pick up a sniffer edition of the board that comes pre-flashed with special firmware that turns your BLEFriend into a low cost Bluetooth Low Energy sniffer (http://adafru.it/iCR), capturing data and pushing it out to Wireshark. We currently offer this as a seperate product, though, since the firmware isn't compatible with the over the air bootloader used on the standard products, but we'll address this in the future with a tutorial for J-Link owners, allowing you to switch between modes using your SWD debugger.

Why Use Adafruit's Module?

There are plenty of BLE modules out there, with varying quality on the HW design as well as the firmware. We always try to keep the bar as high as possible at Adafruit, and one of the biggest advantages of the BLEFriend and the entire Bluefruit LE family is that **we wrote all of the firmware running on the devices ourelves from scratch.**

We control every line of code that runs on our modules ... and so we aren't at the mercy of any third party vendors who may or may not be interested in keeping their code up to date or catering to our customer's needs.

Because we control everything about the product, we add features that are important to*our* customers, benefit from being able to use the latest Bluetooth specifications, can solve any issues that do come up without having to persuade a half-hearted firmware engineer on the other side of the planet, and we can even change Bluetooth SoCs entirely if the need ever arises!

Getting Started

If you just want to get up and running quickly, this is the right guide for you, and the QuickStart section should have you up and running in no time.

If you're new to Bluetooth Low Energy, and want to get a high level overview of how data is organized and how devices communicate with each other, you might want to have a look at our Introduction to Bluetooth Low Energy (http://adafru.it/iCS) learning guide, or buy Getting Started with Bluetooth Low Energy (http://adafru.it/1978) from the store.*

* Full disclosure: co-written by me :)

QuickStart Guide

The BLEFriend board is designed to be easy to use and get started with.

In most circumstance, the only thing you'll need is an FTDI driver, and a terminal emulator to start working with BLE from your development machine.

This guide will explain some of the differentoperating modes (http://adafru.it/iCT) that the BLEFriend board can be configure to operate in, how to setup your terminal emulator (http://adafru.it/iCU) to start talking to the BLEFriend, and a basic example ofsending bi-directional data (http://adafru.it/iCV)between the BLEFriend and a BLE-enabled phone or tablet.

The BLEFriend is (of course) capable of much more than basic UART data exchanges! You can use it to create custom GATT services and characteristics. You can emulate a Beacon (http://adafru.it/iCu) for indoor navigation purposes. You canupdate the firmware on your device (http://adafru.it/iCQ) over the air, or even develop your own firmware thanks to the SWD pins on the bottom of the PCB if you possess a HW debugger like the Segger J-Link.

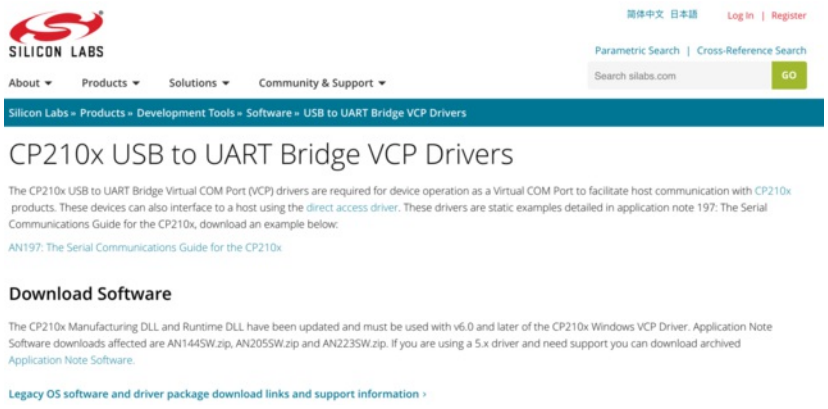
Depending on the version of the board you have or whether or not you have access to a J-Link, you can even use the BLEFriend as a powerful Bluetooth Low Energy sniffer (http://adafru.it/iCW) to debug or reverse engineer existing BLE devices, and write your own custom applications or drivers for existing HW!

Have a look through this quick start guide to familiarise yourself with the basics, though, and you should be able to quickly move on to more advanced topics in no time!

HW Setup

v3.0 Software Requirements

The cost-optimized v3.0 board uses the less expensive CP2104 USB to Serial bridge, which requires that you install the Silicon Labs VCP drivers (http://adafru.it/vrf) on your system before using the board.



简体中文 日本語 Log In | Register

Parametric Search | Cross-Reference Search

Search silabs.com GO

About Products Solutions Community & Support

Silicon Labs » Products » Development Tools » Software » USB to UART Bridge VCP Drivers

CP210x USB to UART Bridge VCP Drivers

The CP210x USB to UART Bridge Virtual COM Port (VCP) drivers are required for device operation as a Virtual COM Port to facilitate host communication with CP210x products. These devices can also interface to a host using the direct access driver. These drivers are static examples detailed in application note 197: The Serial Communications Guide for the CP210x, download an example below:

AN197: The Serial Communications Guide for the CP210x

Download Software

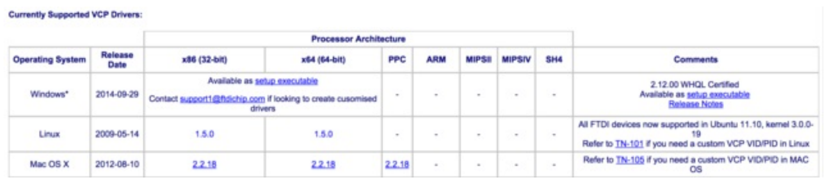
The CP210x Manufacturing DLL and Runtime DLL have been updated and must be used with v6.0 and later of the CP210x Windows VCP Driver. Application Note Software downloads affected are AN144SW.zip, AN205SW.zip and AN223SW.zip. If you are using a 5.x driver and need support you can download archived Application Note Software.

Legacy OS software and driver package download links and support information ›

v1.0 and v2.0 Software Requirements

Setting up the BLEFriend is super easy. All you need to start talking to the device is a standard FTDI driver for the FT231x located on the device.

Find the appropriate FTDI VCP installer on the FTDI Driver Download Page (http://adafru.it/aJv), install it on you system, and then insert the BLEFriend in any USB port on your system.



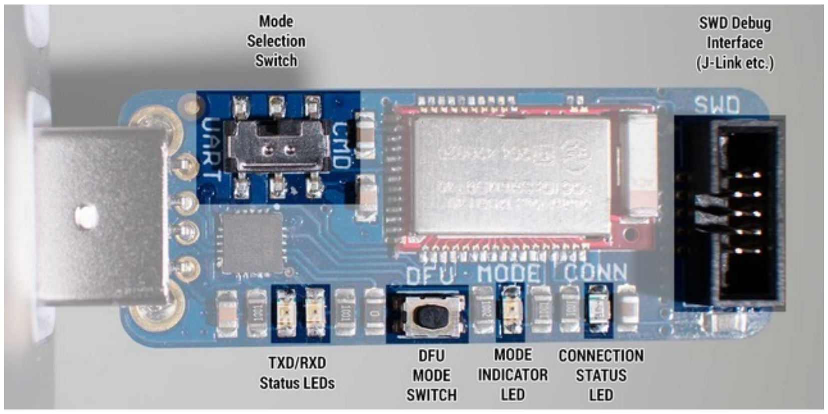
Currently Supported VCP Drivers:

		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

HW Layout

There are a few items on the BLEFriend you should be familiar with before you start working with it. To help you get started quickly, we've highlighted them in the image below:

This image applies to v2.0. The latest cost-optimized v3.0 release drops the SWD connector, and leaving the SWD pins on available on the bottom of the PCB.



Mode Selection Switch

This switch can be moved between 'CMD' (Command Mode) and 'UART' (Data Mode), which will change the way that the device behaves in your terminal emulator.

For more information on these two operating modes, see theOperating Modes (http://adafru.it/iCT) page in this learning guide.

TXD/RXD Status LEDs

These two LEDs are provided primarily for debug purposes to help you visualise the incoming and outgoing characters over the USB CDC interface.

DFU Mode Switch

Holding this switch down when you insert the device into the USB port will cause the device to enter a special 'DFU' mode, which allows you to update the firmware over the air.

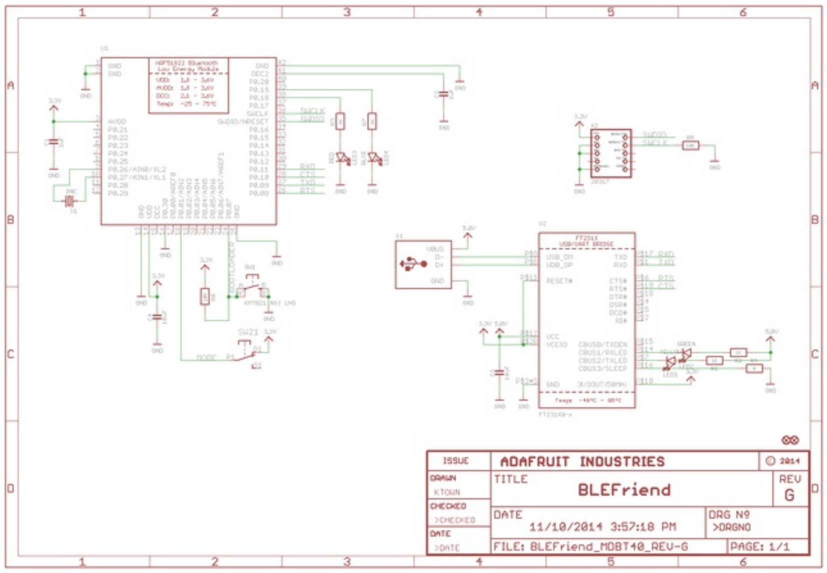
For more information on DFU mode see theField Updates (http://adafru.it/iCQ) page.

Mode Indicator LED

This LED is used to indicate the mode that the device is currently operating in (Data, Command or DFU).

Connection Status LED

This LED will be enabled when the BLEFriend has successfully established a connection with another BLE device, and it useful for debugging purposes.



Operating Modes

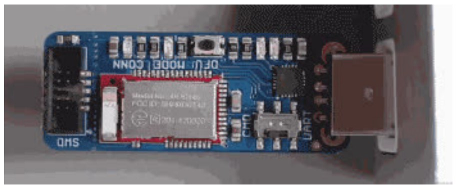
BLEFriend modules can be operated in one of three modes:

Data Mode

Data mode makes use of the BLE UART Service, and turns the BLEFriend into a HW UART bridge between a BLE Central device (your phone or tablet) and your PC or USB-enabled device.

To use data mode, simply connect your BLEFriend module to the USB port on your PC, mode the mode selection switch to **UART** and start sending or receiving data at 9600 bps using your favorite terminal software.

If the MODE LED blinks twice followed by a three second delay you are in **Data Mode:**

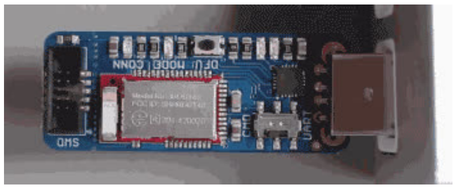


Command Mode

Command mode is used to send configuration commands to the module or retrieve information about the module itself or the device connected on the other side of the BLE connection.

To use command mode, make sure that the mode selection switch is set to **CMD**, and enter a valid Hayes AT style command using your favorite terminal emulator at 9600 bps (for example 'ATI' to display some basic info about the module).

If the MODE LED blinks three times followed by a three second delay you are in **Command Mode**:



For more information on this operating mode see the dedicated Command Mode (http://adafru.it/iCO) page in this learning guide.

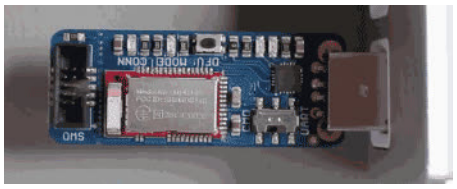
DFU Mode

DFU Mode (which stands for 'device firmware upgrade' or sometimes 'device field update') is a special mode that allows you to update the firmware on the BLEFriend over the air using dedicated DFU applications available on iOS and Android.

This allows you to update your device with the latest BLEFriend firmware from Adafruit without having to purchase an external HW debugger like the Segger J-Link (http://adafru.it/e9G).

To enter DFU mode hold down the **DFU** button while inserting the BLEFriend into the USB port.

If the LED blinks at a constant rate, you know that you are in DFU mode:



For more information on DFU mode see the dedicate Device Field Update (DFU) page (http://adafru.it/iCQ).

Terminal Settings

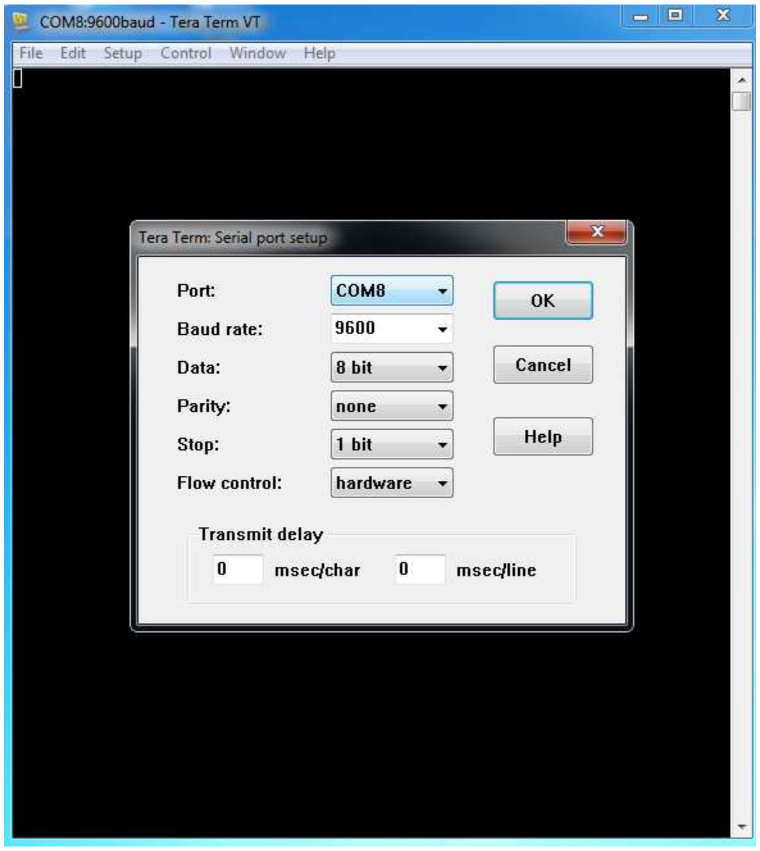
Since the BLEFriend board uses UART for the data transport, you will need to configure your favorite terminal emulator software with the appropriate settings to send or receive data over UART.

The BLEFriend is configured to run with the following settings:

- **9600 bps**
- **HW flow control** (CTS+RTS)
- **8n1** (8-bit data, no parity, 1 stop bit)

TerraTerm (Windows)

If you are using Windows we recommend usingTeraTerm (http://adafru.it/e9l), which should be configured as follows (via the 'Setup > Serial Port' menu):

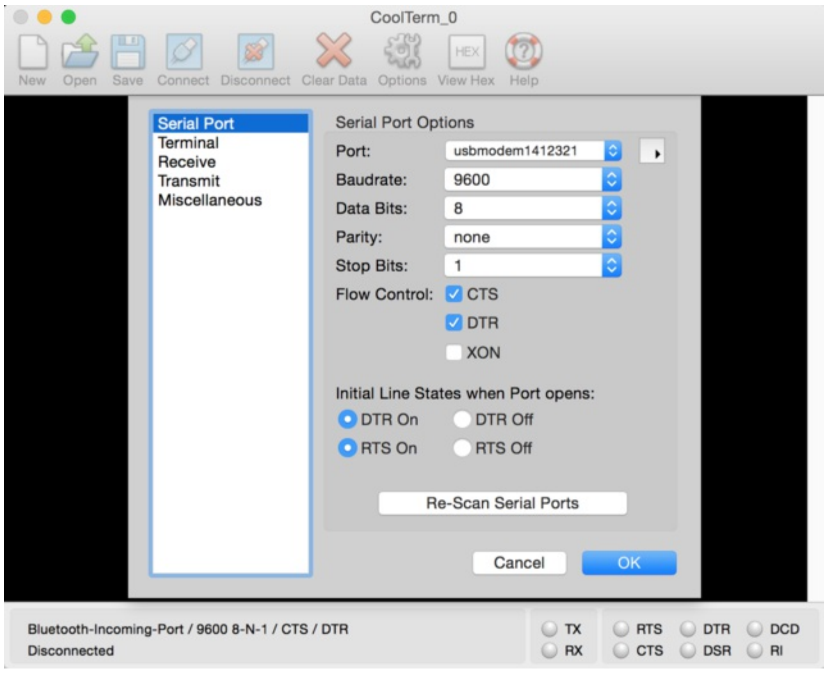

COM8:9600baud - Tera Term VT
File Edit Setup Control Window Help
Tera Term: Serial port setup
OK
Cancel
Help

Port: COM8
Baud rate: 9600
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: hardware
Transmit delay
0 msec/char 0 msec/line

CoolTerm (OS X)

If you are using OS X, we recommend usingCoolTerm (http://adafru.it/e9J), a free and reasonably fully featured terminal emulator package.

You'll need to set the following configuration options in Coolterm to communicate with the Bluefruit LE Pro module:



Serial Port Options
Port: usbmodem1412321
Baudrate: 9600
Data Bits: 8
Parity: none
Stop Bits: 1
Flow Control: [✓ CTS] [✓ DTR] XON

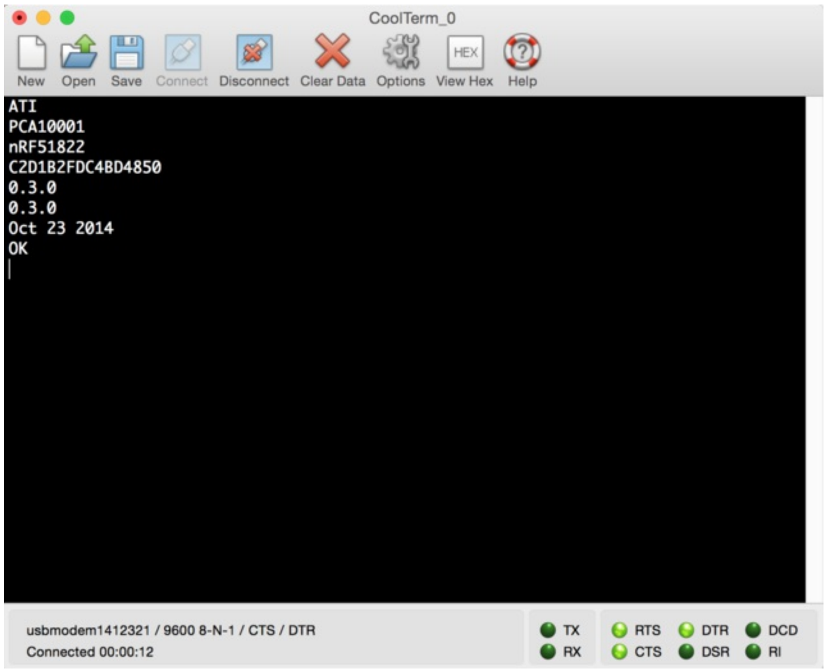
Initial Line States when Port opens:
[✓ DTR On] DTR Off
[✓ RTS On] RTS Off

Bluetooth-Incoming-Port / 9600 8-N-1 / CTS / DTR
Disconnected

Make sure you click the 'Connect' button to open the connection in CoolTerm, and 'Disconnect' to close the connection when you are done.

Testing the Terminal Config

To make sure the connection is properly configured, you can set the device in**Command Mode** (the LED should blink three times followed by a short delay in this mode), and enter the **ATI** command, as shown below (which will provide some basic information about the BLE module):



```
ATI
PCA10001
nRF51822
C2D1B2FDC4BD4850
0.3.0
0.3.0
Oct 23 2014
OK
```

usbmodem1412321 / 9600 8-N-1 / CTS / DTR
Connected 00:00:12

Make sure you are in CMD mode or you won't get any echo or AT command replies!

UART Test

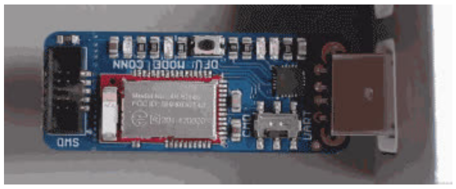
To test out the BLE UART service that is used to transfer data between the BLEFriend and your Bluetooth Low Energy enabled mobile device, you can used the **nRF UART** application from Nordic Semiconductors for iOS (http://adafru.it/dd7) or Android (http://adafru.it/dd6).

This tutorial uses the Android version of nRF UART but the iOS UI is reasonably similar.

BLEFriend Configuration

Set the BLEFriend to **DATA MODE** by placing the mode selector switch (near the USB connector) to the **UART** position.

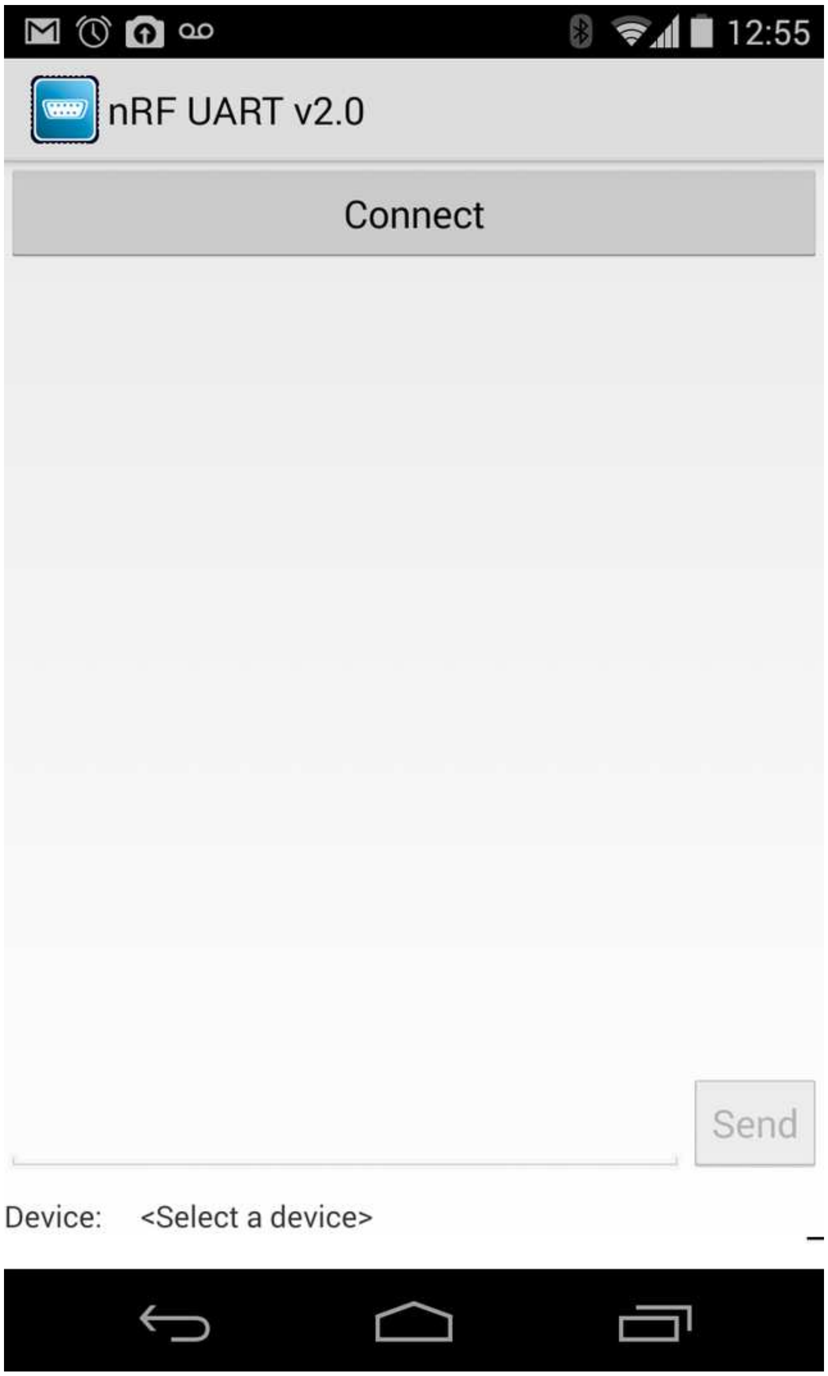
The 'MODE' indicator LED will blink two times and then pause for three seconds when you are in DATA mode:



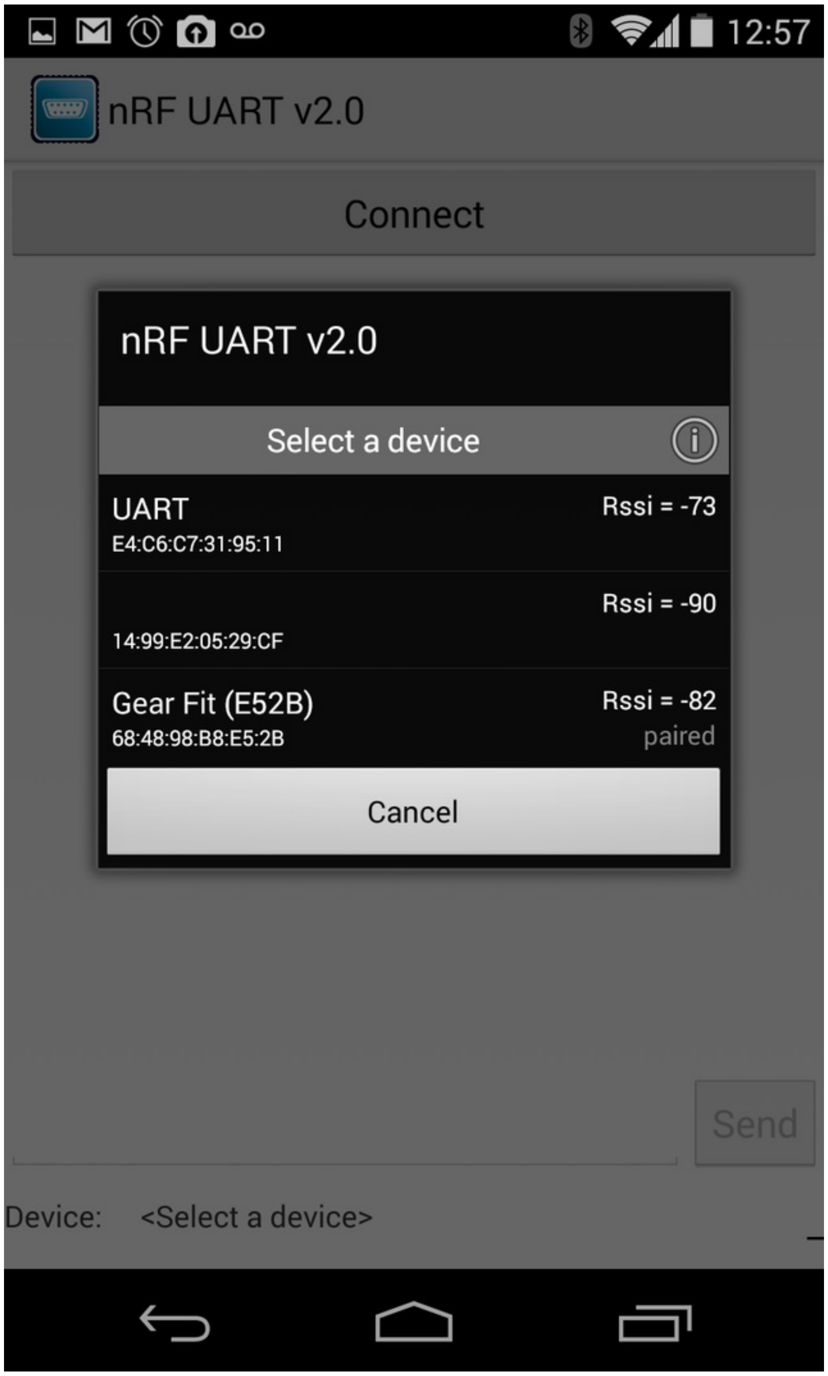
Using your favorite terminal emulator (http://adafru.it/iCU), connect to the BLEFriend at **9600 baud** with HW flow control enabled.

nRF UART Configuration

Open the nRF UART application on Android or iOS, and click the 'Connect' button

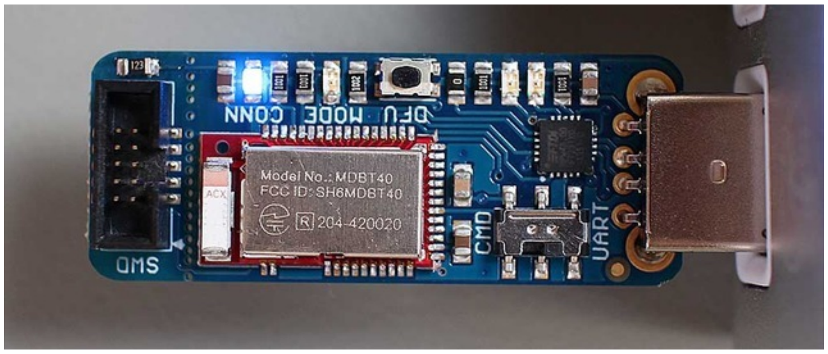


After clicking on 'Connect' you should see a list of BLE peripherals in range:



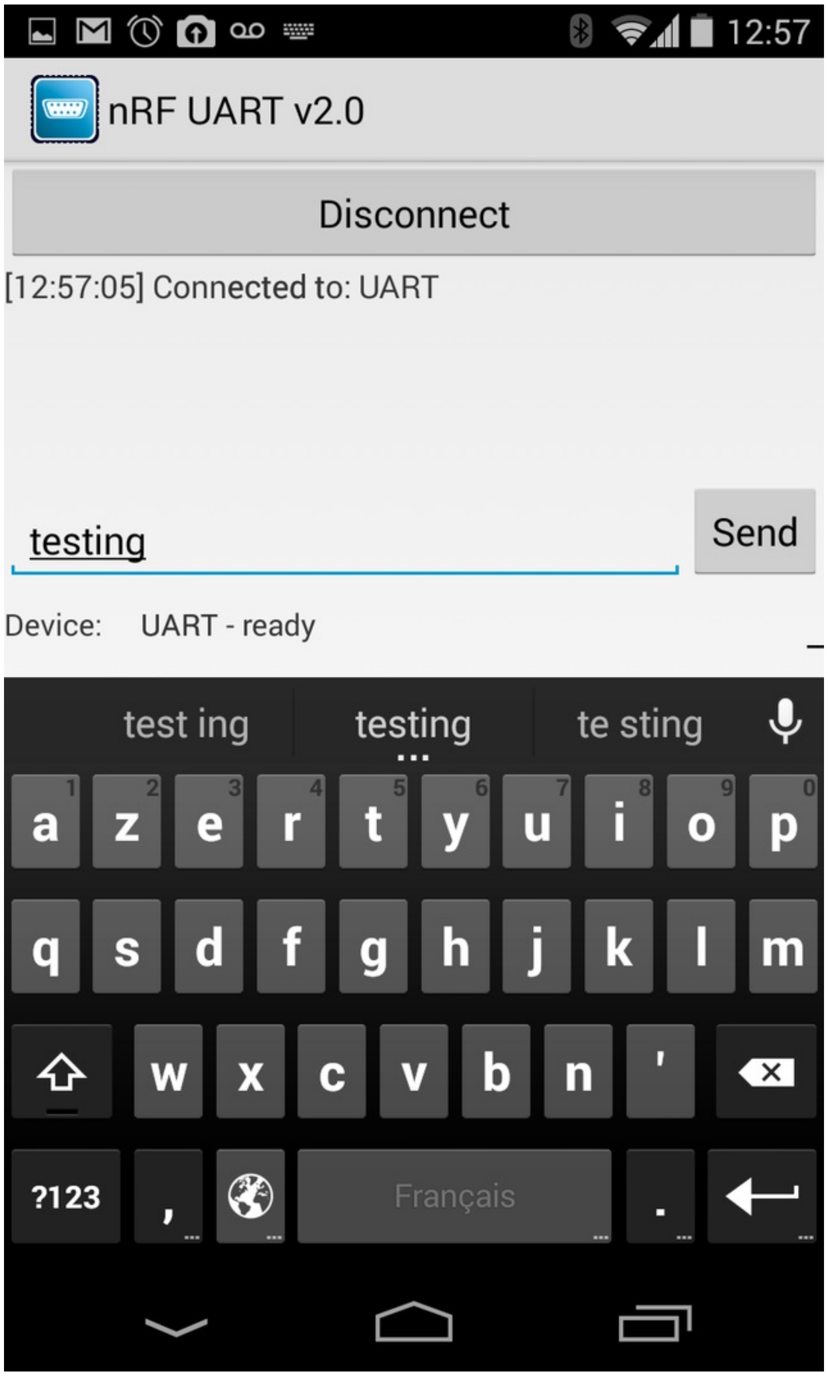
Select the '**UART**' device by clicking on it from the popup selection dialogue, and a connection will be established between the BLEFriend and your phone or tablet.

NOTE: You know that you are connected to another device when the**CONN** LED is lit up on the BLEFriend:



Once you see the 'Connected To: UART' message in nRF UART, you can start sending and receiving data between the two devices using the mobile app or the terminal emulator.

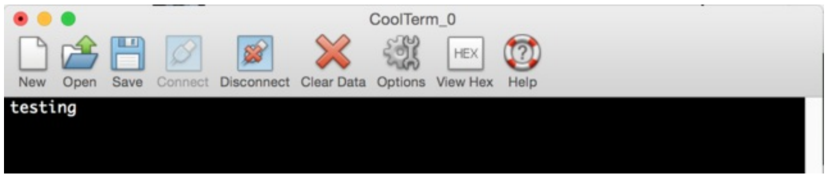
Sending data from one device should cause the text to appear on the other end:



[12:57:05] Connected to: UART

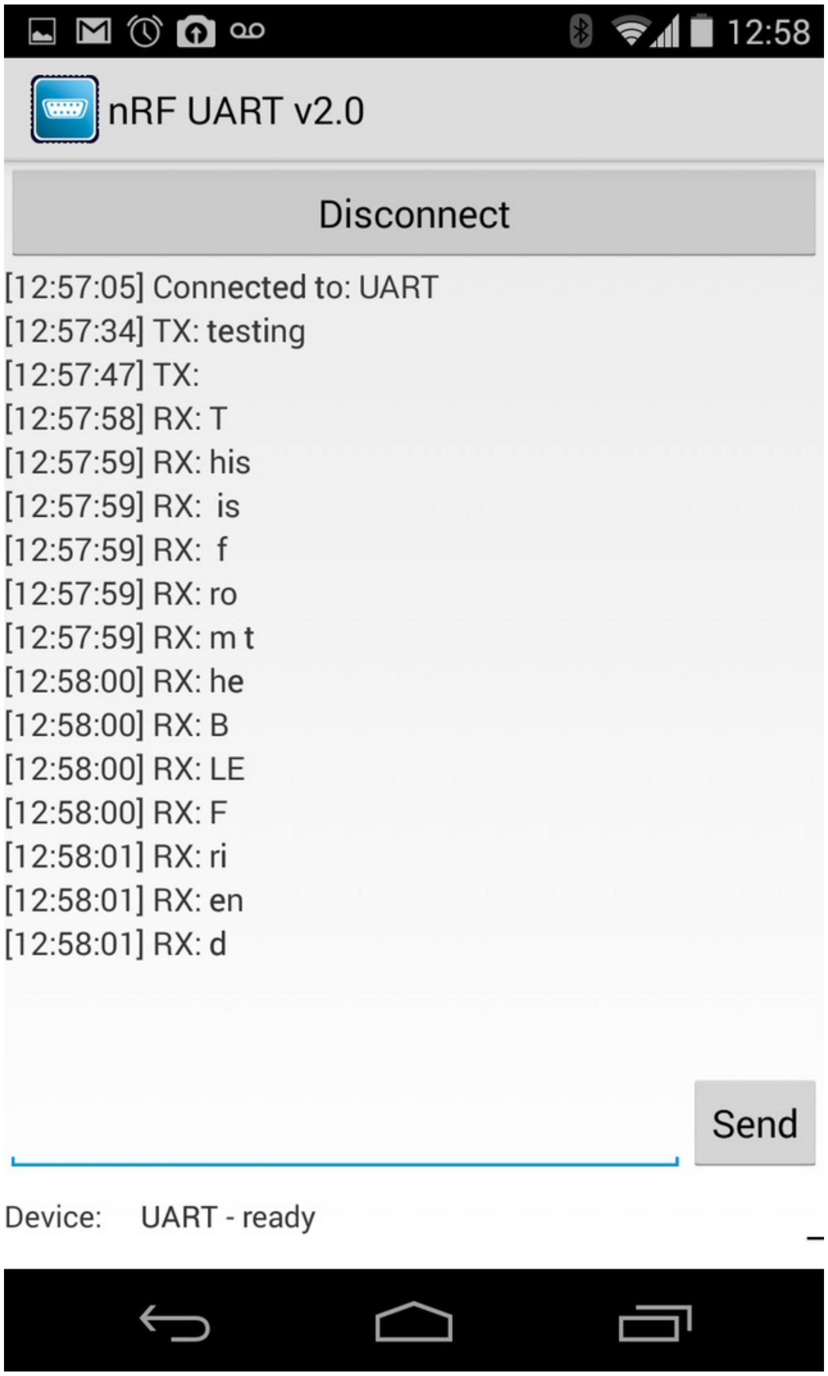
testing

On the BLEFriend side we can see the 'testing' string that was sent from nRF UART in the terminal emulator below:



```
testing
```

Typing data into the terminal emulator will cause it to appear in the nRF UART application, as shown below:



```
[12:57:05] Connected to: UART
[12:57:34] TX: testing
[12:57:47] TX:
[12:57:58] RX: T
[12:57:59] RX: his
[12:57:59] RX:  is
[12:57:59] RX:  f
[12:57:59] RX: ro
[12:57:59] RX: m t
[12:58:00] RX: he
[12:58:00] RX: B
[12:58:00] RX: LE
[12:58:00] RX: F
[12:58:01] RX: ri
[12:58:01] RX: en
[12:58:01] RX: d
```

Device: UART - ready

The incoming data is broken up into small packets because it is transferred in chunks of up to 20 bytes per payload.

All data transfers in BLE occur at a rate controlled by the BLE central device (the phone or tablet). Every n milliseconds, the phone will ping the BLE peripheral (the BLEFriend) and check if there is any data available.

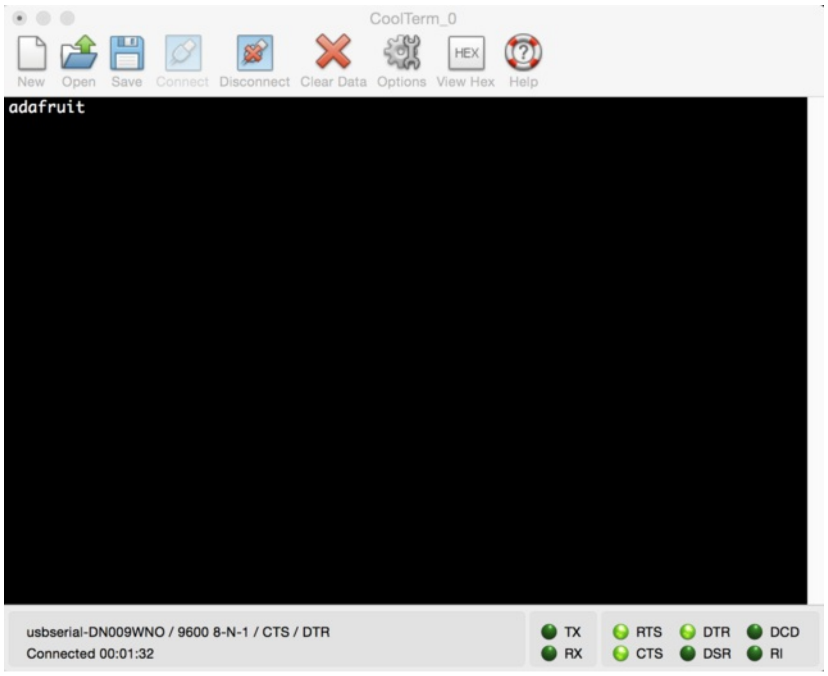
If anything is found in the UART FIFO, for example, it will be sent and data will start to accumulate again in the FIFO until the next connection event.

Sample Video

Download the screen capture below to see nRF UART in action, first receiving 'this is a test' from the BLEFriend (typed in on the terminal emulator), and then sending 'adafruit' out to the BLEFriend, where is will appear in the terminal emulator.

nRFUART_640.mp4
http://adafru.it/ea5

Depending on which terminal emulator SW you use, you would see something like this on your development machine running a similar demo yourself:



```
adafruit
```

usbserial-DN009WNO / 9600 8-N-1 / CTS / DTR
Connected 00:01:32

Factory Reset

As of version 0.5.0+ of the firmware, you can perform a factory reset by holding the DFU button down for 10s until the blue CONNECTED LED lights up, and then releasing the button.

If you have any problems with your BLEFriend module, you can perform a factory reset by entering command mode (set the mode selection switch to 'CMD'), and entering the following command in your terminal emulator:

```
AT+FACTORYRESET
```

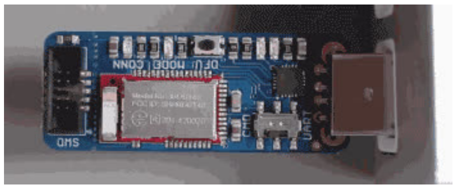
This will erase the non volatile memory section where the config data is stored and reset everything to factory default values, and perform a system reset.

When you see the 'OK' response the device should be ready to use.

Command Mode

By placing the BLEFriend module in 'Command' mode (set the mode selection switch to **CMD**) you can enter a variety of Hayes AT style commands to configure the device or retrieve basic information about the module of BLE connection.

You can determine if you are in Command Mode by looking at the mode LED. It should blink three times followed by a three second pause, as shown below:



Hayes/AT Commands

When operating in command mode, the Bluefruit LE Pro modules use a Hayes AT-style command set (http://adafru.it/ebJ) to configure the device.

The advantage of an AT style command set is that it's easy to use in machine to machine communication, while still being somewhat user friendly for humans.

Test Command Mode '=?'

'Test' mode is used to check whether or not the specified command exists on the system or not.

Certain firmware versions or configurations may or may not include a specific command, and you can determine if the command is present by taking the command name and appending **'=?'** to it, as shown below

```
AT+BLESTARTADV=?
```

If the command is present, the device will reply with **'OK'**. If the command is not present, the device will reply with **'ERROR'**.

```
AT+BLESTARTADV=?
OK\r\n
AT+MISSINGCMD=?
ERROR\r\n
```

Write Command Mode '=xxx'

'Write' mode is used to assign specific value(s) to the command, such as changing the radio's transmit power level using the command we used above.

To write a value to the command, simple append an **'='** sign to the command followed by any paramater(s) you wish to write (other than a lone **'?'** character which will be interpretted as tet mode):

```
AT+BLEPOWERLEVEL=-8
```

If the write was successful, you will generally get an **'OK'** response on a new line, as shown below:

```
AT+BLEPOWERLEVEL=-8
OK\r\n
```

If there was a problem with the command (such as an invalid parameter) you will get an **'ERROR'** response on a new line, as shown below:

```
AT+BLEPOWERLEVEL=3
ERROR\r\n
```

Note: This particular error was generated because '3' is not a valid value for the AT+BLEPOWERLEVEL command. Entering '-4', '0' or '4' *would* succeed since these are all valid values for this command.

Execute Mode

'Execute' mode will cause the specific command to 'run', if possible, and will be used when the command name is entered with no additional parameters.

```
AT+FACTORYRESET
```

You might use execute mode to perform a factory reset, for example, by executing the AT+FACTORYRESET command as follows:

```
AT+FACTORYRESET
OK\r\n
```

NOTE: Many commands that are means to be read will perform the same action whether they are sent to the command parser in 'execute' or 'read' mode. For example, the following commands will produce identical results:

```
AT+BLEGETPOWERLEVEL
-4\r\n
OK\r\n
AT+BLEGETPOWERLEVEL?
-4\r\n
OK\r\n
```

If the command doesn't support execute mode, the response will normally be **'ERROR'** on a new line.

Read Command Mode '?'

'Read' mode is used to read the current value of a command.

Not every command supports read mode, but you generally use this to retrieve information like the current transmit power level for the radio by appending a **'?'** to the command, as shown below:

```
AT+BLEPOWERLEVEL?
```

If the command doesn't support read mode or if there was a problem with the request, you will normally get an **'ERROR'** response.

If the command read was successful, you will normally get the read results followed by **'OK'** on a new line, as shown below:

```
AT+BLEPOWERLEVEL?
-4\r\n
OK\r\n
```

Note: For simple commands, 'Read' mode and 'Execute' mode behave identically.