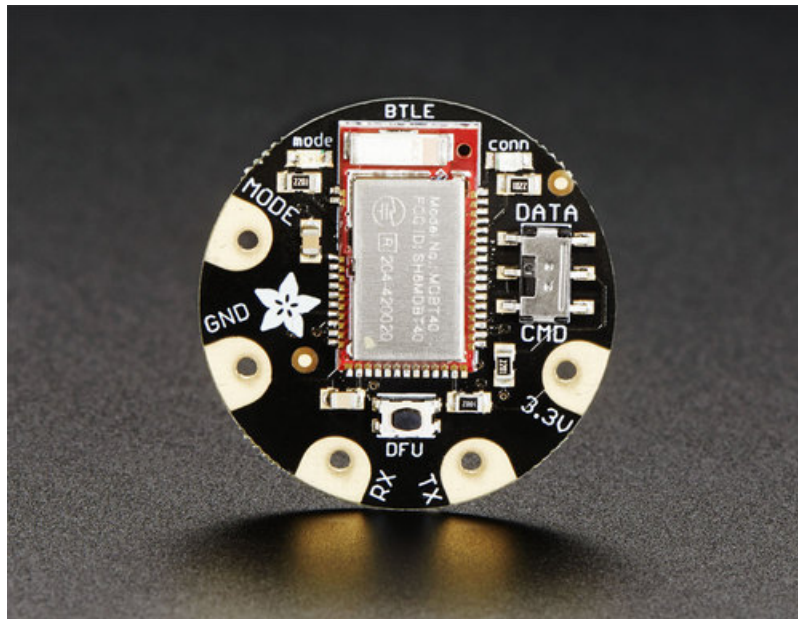




Adafruit Flora Bluefruit LE

Created by lady ada



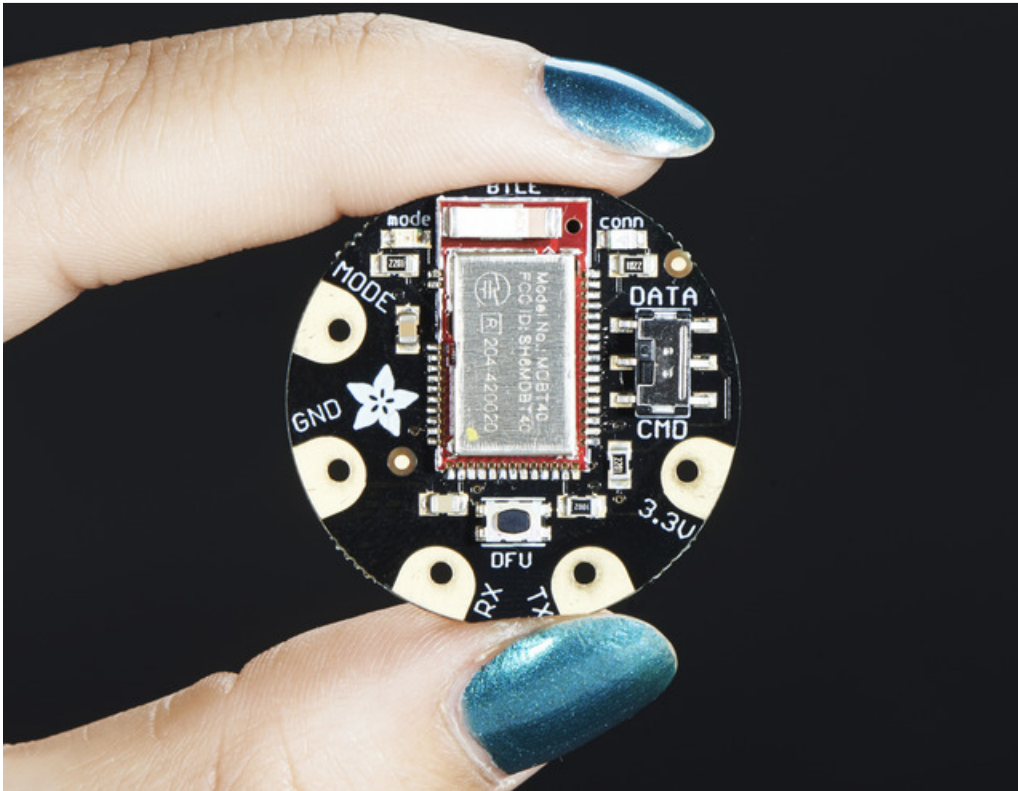
Last updated on 2018-08-22 03:48:18 PM UTC

Guide Contents

Guide Contents	2
Overview	4
Get started fast with the Bluefruit App	5
You can do a lot more too!	6
Pinouts	7
Power Pads	7
Data Pads	7
Other Pads	8
Reverse Side	9
Sewing	10
Factory Reset	11
Factory Reset via DFU Button	11
FactoryReset Sample Sketch	11
AT+FACTORYRESET	12
Factory Reset via F.RST Test Pad	12
Firmware Updates	14
Adafruit Bluefruit LE Connect	14
Installing Software	15
Example Code	16
Configuring for Flora	16
ATCommand	17
Opening the Sketch	17
Configuration	18
Running the Sketch	18
BLEUart	22
Opening the Sketch	22
Configuration	23
Running the Sketch	23
Controller	28
Opening the Sketch	28
Configuration	29
Running the Sketch	29
Using Bluefruit LE Connect in Controller Mode	30
Streaming Sensor Data	31
Control Pad Module	33
Color Picker Module	34
HIDKeyboard	36
Opening the Sketch	36
Configuration	37
Running the Sketch	37

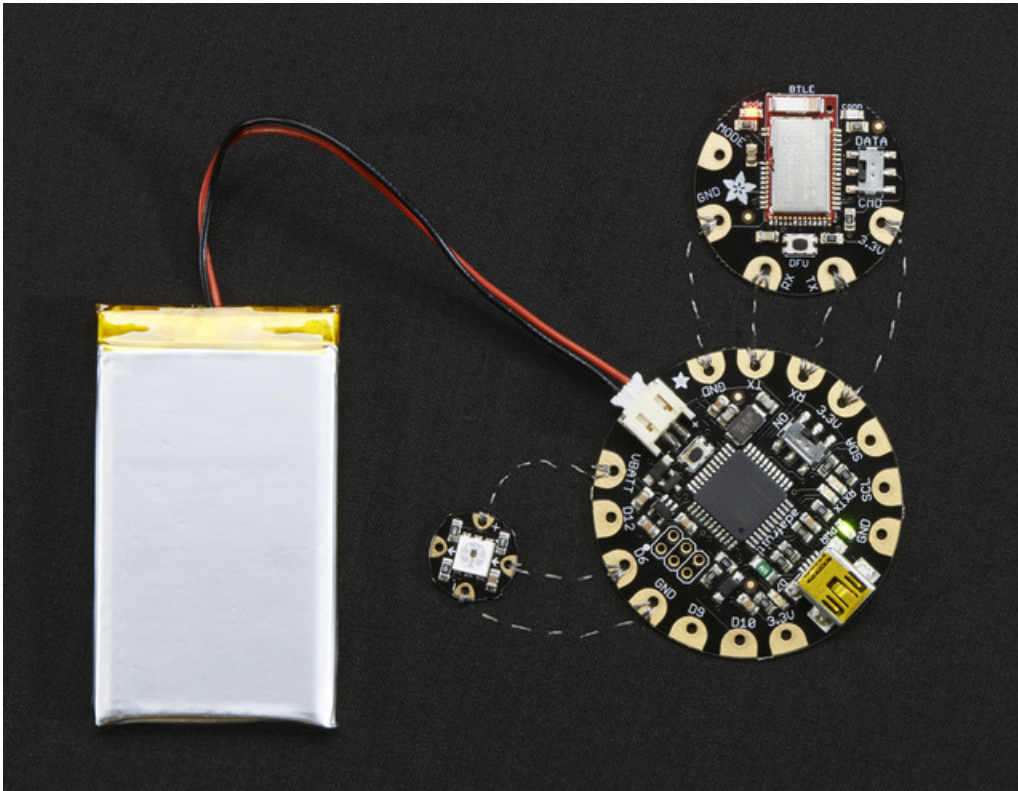
Bonding the HID Keyboard	38
Android	39
iOS	40
OS X	42
UriBeacon	44
Opening the Sketch	44
Configuration	45
Running the Sketch	45
HeartRateMonitor	47
Opening the Sketch	47
Configuration	48
If Using Hardware or Software UART	48
Running the Sketch	49
nRF Toolbox HRM Example	50
CoreBluetooth HRM Example	51
HALP!	53
When using the Bluefruit Micro or a Bluefruit LE with Flora/Due/Leonardo/Micro the examples dont run?	53
I can't seem to "Find" the Bluefruit LE!	53
Data Mode	55
Switching Command/Data Mode via +++	56
Command Mode	58
Hayes/AT Commands	58
UART Service	59
Downloads	60
Schematic	60
Fabrication print	60

Overview



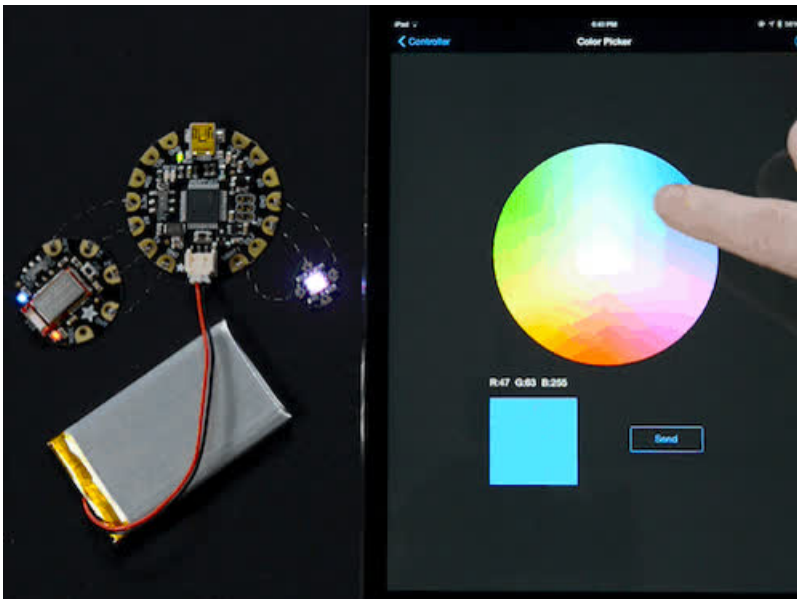
Would you like to add powerful and easy-to-use Bluetooth Low Energy to your wearable FLORA project? Heck yeah! With BLE now included in modern smart phones and tablets, its fun to add wireless connectivity. So what you really need is the new Adafruit Flora Bluefruit LE!

The Flora Bluefruit LE makes it easy to add Bluetooth Low Energy connectivity to your Flora. Sew 4 traces (or solder 4 wires) and Boom! Bluetooth Low Energy!



Get started fast with the Bluefruit App

Using our Bluefruit [iOS App \(https://adafru.it/iCi\)](https://adafru.it/iCi) or [Android App \(https://adafru.it/f4G\)](https://adafru.it/f4G), you can quickly get your interactive project prototyped by using your iOS or Android phone/tablet as a controller. We have a [color picker \(https://adafru.it/iCi\)](https://adafru.it/iCi), [quaternion/accelerometer/gyro/magnetometer or location \(GPS\) \(https://adafru.it/iCi\)](https://adafru.it/iCi), and an 8-button [control game pad \(https://adafru.it/iCi\)](https://adafru.it/iCi). After you connect to the Bluefruit, you can send commands wirelessly in under 10 minutes

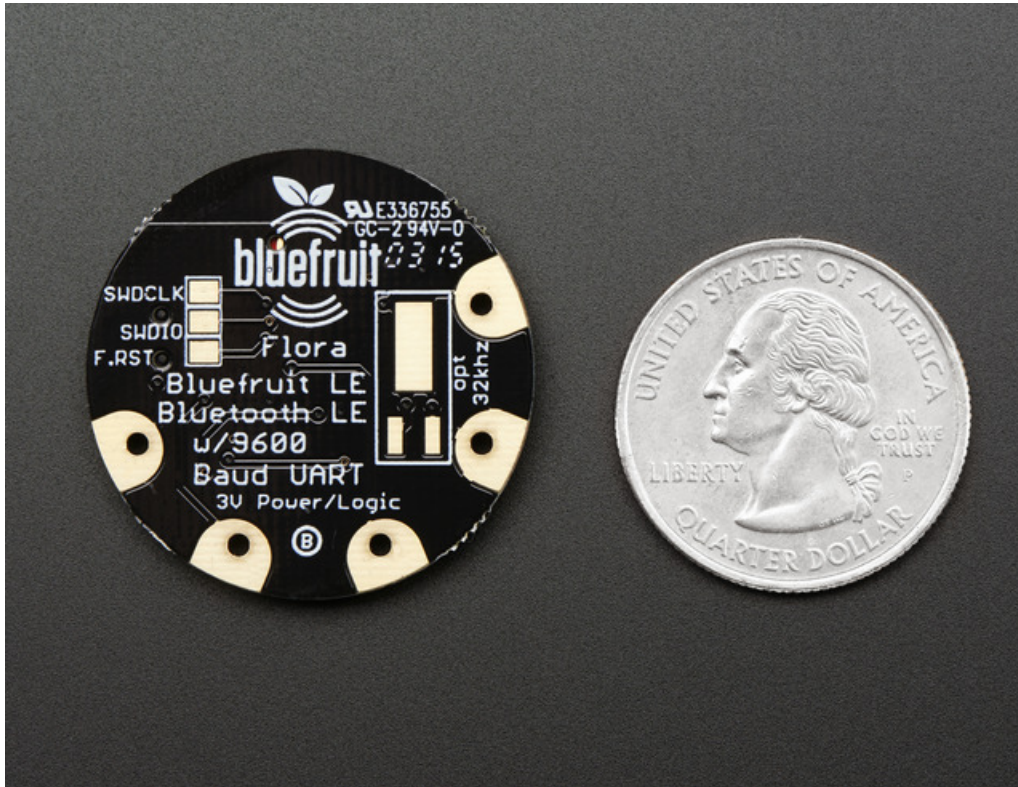


For advanced hackers, they'll be very happy to use the standard Nordic UART RX/TX connection profile. In this profile, the Bluefruit acts as a data pipe, that can 'transparently' transmit back and forth from your iOS or Android device. You

can use our [iOS App \(https://adafru.it/iCi\)](https://adafru.it/iCi) or [Android App \(https://adafru.it/f4G\)](https://adafru.it/f4G), or [write your own to communicate with the UART service \(https://adafru.it/iCF\)](https://adafru.it/iCF).

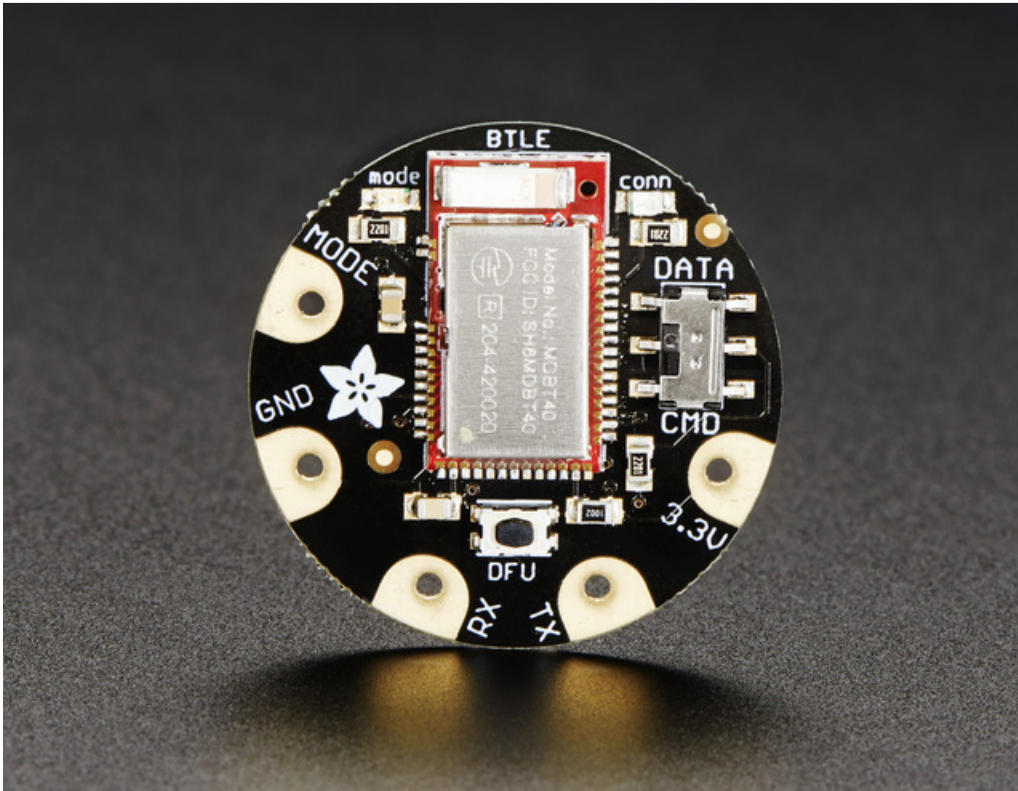
You can do a lot more too!

- [The Bluefruit can also act like an HID Keyboard \(https://adafru.it/iCJ\)](https://adafru.it/iCJ) (for devices that support BLE HID)
- [Can become a BLE Heart Rate Monitor \(https://adafru.it/iCK\)](https://adafru.it/iCK) (a standard profile for BLE) - you just need to add the pulse-detection circuitry
- [Turn it into a UriBeacon \(https://adafru.it/iCL\)](https://adafru.it/iCL), the Google standard for Bluetooth LE beacons. Just power it and the 'Friend will bleep out a URL to any nearby devices with the UriBeacon app installed.
- [Built in over-the-air bootloading capability so we can keep you updated with the hottest new firmware. \(https://adafru.it/iCM\)](https://adafru.it/iCM) Use any Android or iOS device to get updates and install them!



This is the same module and firmware as our [BLE UART Friend \(http://adafru.it/2479\)](http://adafru.it/2479) but in a nice rounded shape, so you can switch between the two and have the same working code. This Bluefruit LE does not have the hardware flow control pins so it is best used with a microcontroller with hardware serial support (like, y'know, the Flora!)

Pinouts



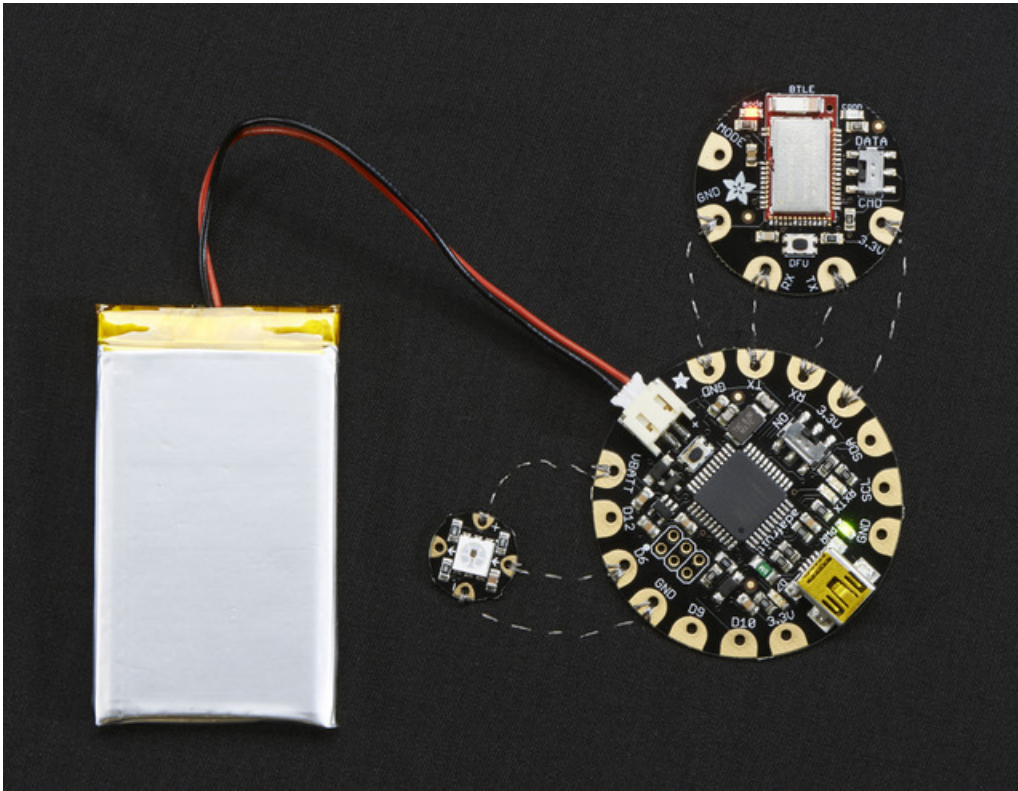
Power Pads

- **3.3V:** This is the power supply for the module, supply with 3.3V power supply input. You can dip down to maybe 2.7V and up to 3.6V or so, but regulated 3.3V is ideal
- **GND:** The common/GND pad for power and logic

Data Pads

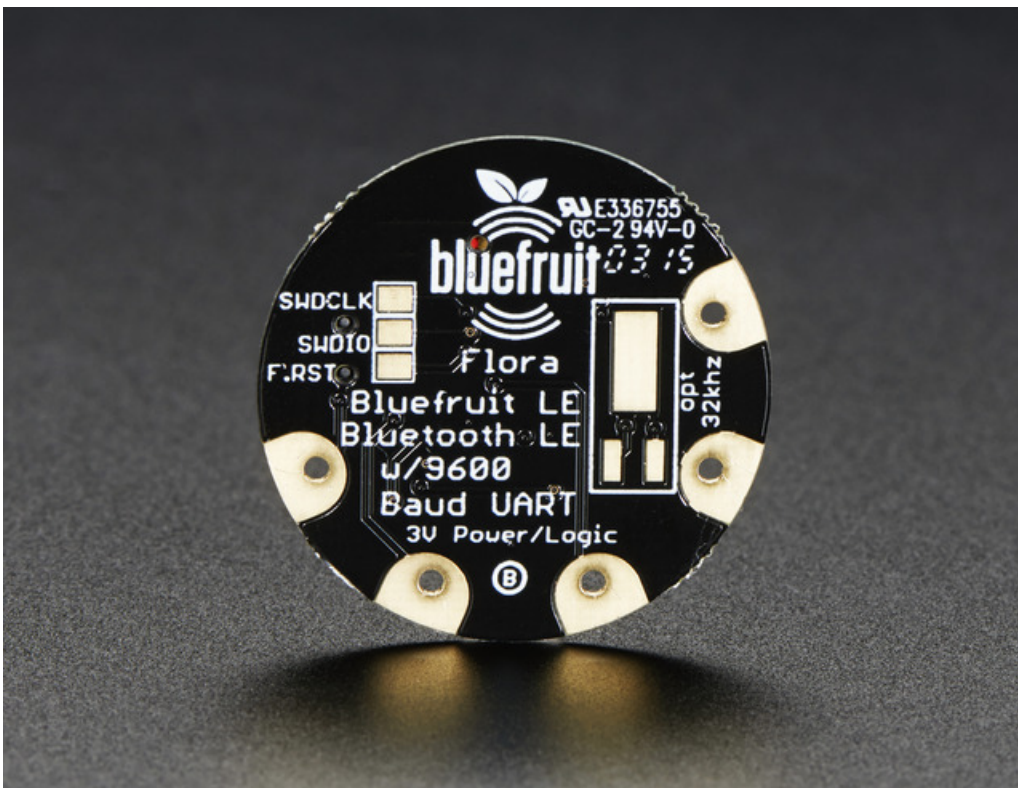
- **TX** - This is the UART Transmit pin *out* of the breakout (Bluefruit LE --> Flora RX), it's at 3.3V logic level.
- **RX** - This is the UART Receive pin *into* the breakout (Flora TX --> Bluefruit LE) it requires 3.3V logic level.

These 4 pads are in perfect order to connect directly to your Flora!



Other Pads

- MODE:** Mode Selection. The Bluefruit has two modes, Command and Data. You can keep this pin disconnected, and use the slide switch to select the mode. Or, you can control the mode by setting this pin voltage, it will override the switch setting! High = Command Mode, Low = UART/DATA mode. This pin requires 3.3V logic



Reverse Side

On the back we also have a few breakouts!

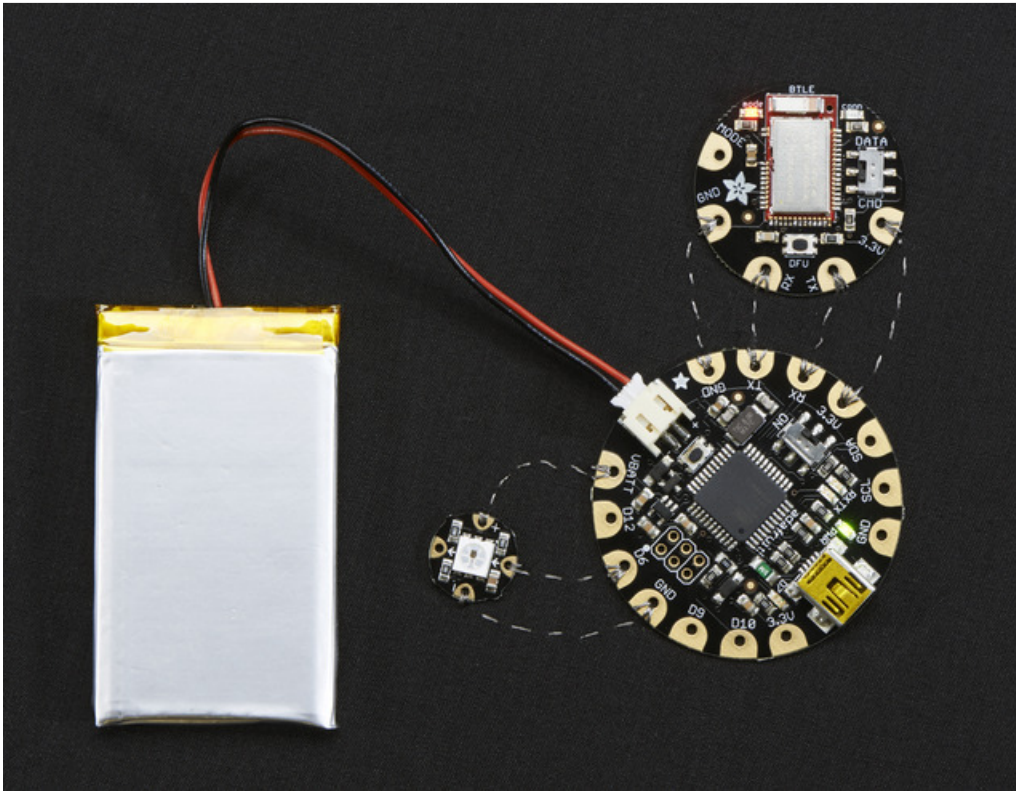
Opt. 32 KHz: If you're doing some funky low power work, [we wanted to give you the option of solderin in a 32khz oscillator.](https://adafru.it/f4U) (<https://adafru.it/f4U>) Our firmware doesn't support it *yet* but its there!

SWDCLK: This is the SWD clock pin, 3v logic - for advanced hackers!

SWDIO: This is the SWD data pin, 3v logic - for advanced hackers!

F.RST: This is the factory reset pin. When all else fails and you did something to really weird out your module, tie this pad to ground while powering up the module and it will factory reset. You should try the DFU reset method first tho (see that tutorial page)

Sewing



Sewing the Flora BLE to a Flora is very straight-forward: the four required pads line up so just sew those in short straight lines from one module to another. the **MODE** pad is not required for most uses so you can leave that unconnected.

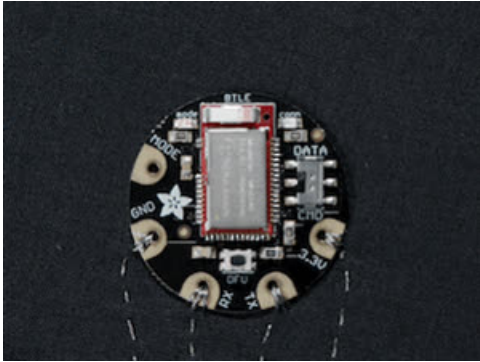
When you power it on, you should see the red LED on the BLE blink a few times and then pause & repeat, so you know that it has power!

Factory Reset

There are several methods that you can use to perform a factory reset on your Bluefruit LE module if something gets misconfigured, or to delete persistent changes like UriBeacon or advertising payload changes, etc.

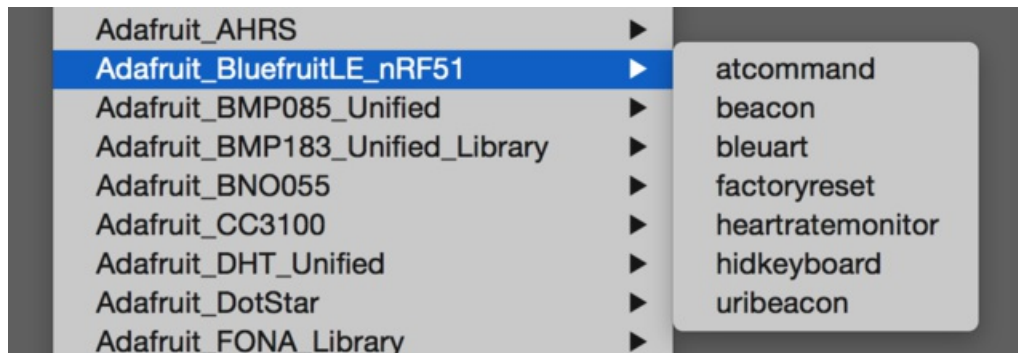
Factory Reset via DFU Button

Once the module is powered up, if you hold the DFU button down for **>5 seconds**, the blue LED on the module will start blinking and the device will perform a factory reset as soon as you release the DFU button.

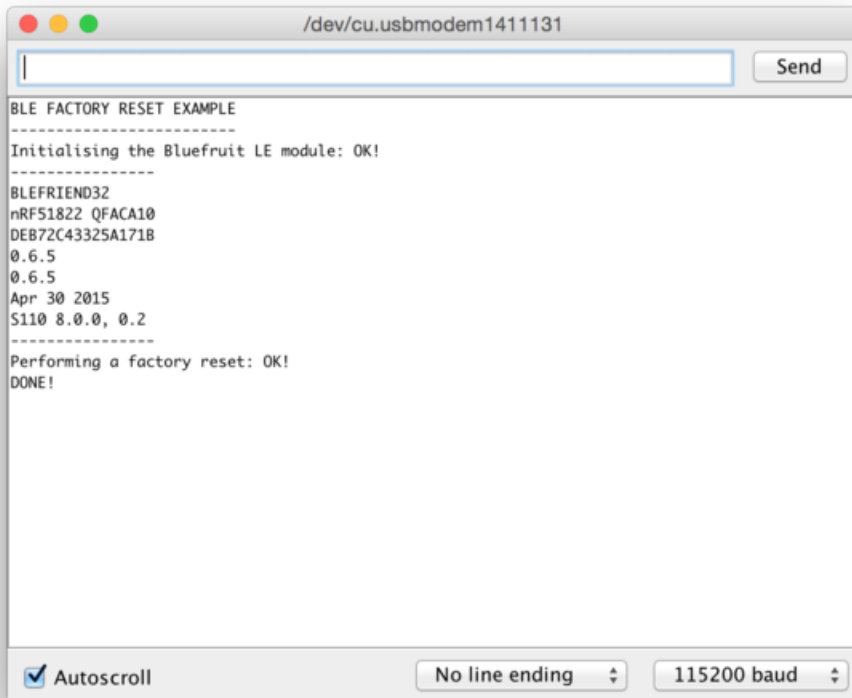


FactoryReset Sample Sketch

There is a FactoryReset sample sketch in the Adafruit Bluefruit LE library, which can be access in the **File > Examples > Adafruit_BluefruitLE_nRF51** folder:



Upload this sketch and open the Serial Monitor and it should perform a factory reset for you:



AT+FACTORYRESET

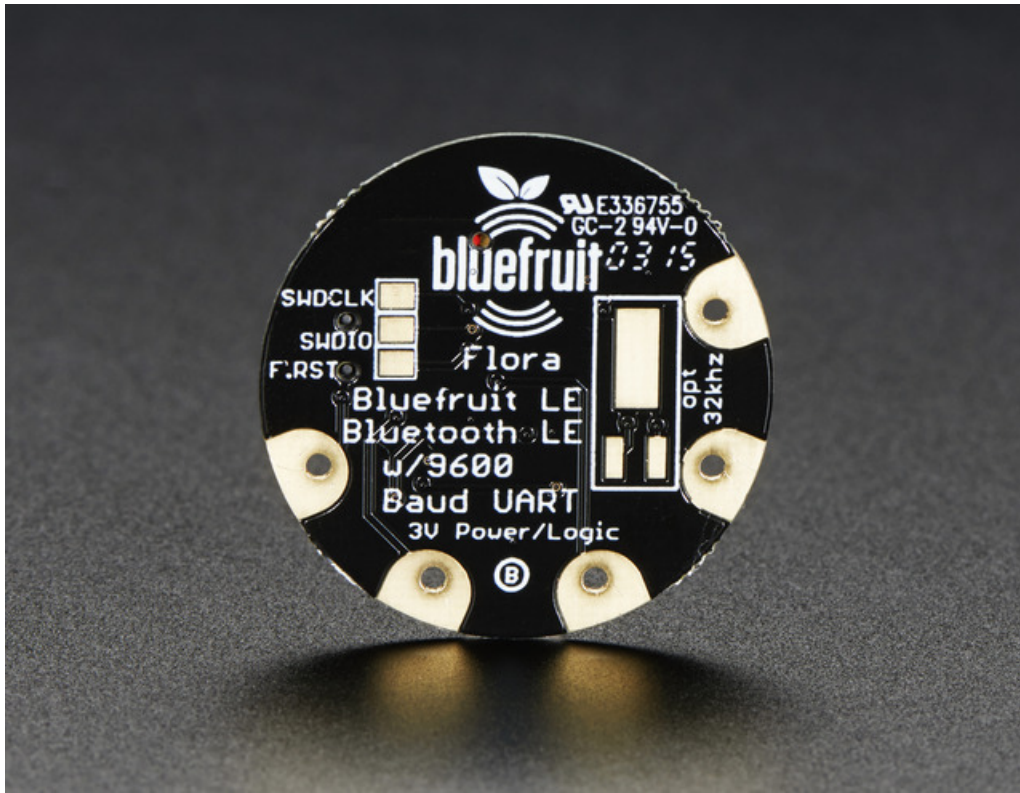
You can also perform a factory reset by sending the **AT+FACTORYRESET** command to your Bluefruit LE module in your favorite terminal emulator or using the **ATCommand** (<https://adafru.it/iCk>) example sketch. Make sure its in command mode first, then type in:

```
AT+FACTORYRESET
OK
```

This command will also cause the device to reset.

Factory Reset via F.RST Test Pad

On the back of the Bluefruit LE UART Friend board there is a test pad that exposes the Factory Reset pin on the modules (marked **F.RST**). Setting this pad low when the device is powered up will cause a factory reset at startup.



Firmware Updates

We're constantly working on the Bluefruit LE firmware to add new features, and keep up to date with what customers need and want.

To make sure you stay up to date with those changes, we've included an easy to use over the air updater on all of our nRF51 based Bluefruit LE modules.

Adafruit Bluefruit LE Connect

Updating your Bluefruit LE device to the latest firmware is as easy as installing [Adafruit's Bluefruit LE Connect application](https://adafru.it/f4G) (<https://adafru.it/f4G>) from the Google Play Store. (An updated iOS version with DFU support will be available shortly!)

Any time a firmware update is available, the application will propose to download the latest binaries and take care of all the details of transferring them to your Bluefruit device, as shown in the video below:

Installing Software

In order to try out our demos, you'll need to download the Adafruit BLE library for the nRF51 based modules such as this one (a.k.a. Adafruit_BluefruitLE_nRF51)

You can check out the code here at github, (<https://adafru.it/f4V>) but its likely easier to just download by clicking:

<https://adafru.it/f4W>

<https://adafru.it/f4W>

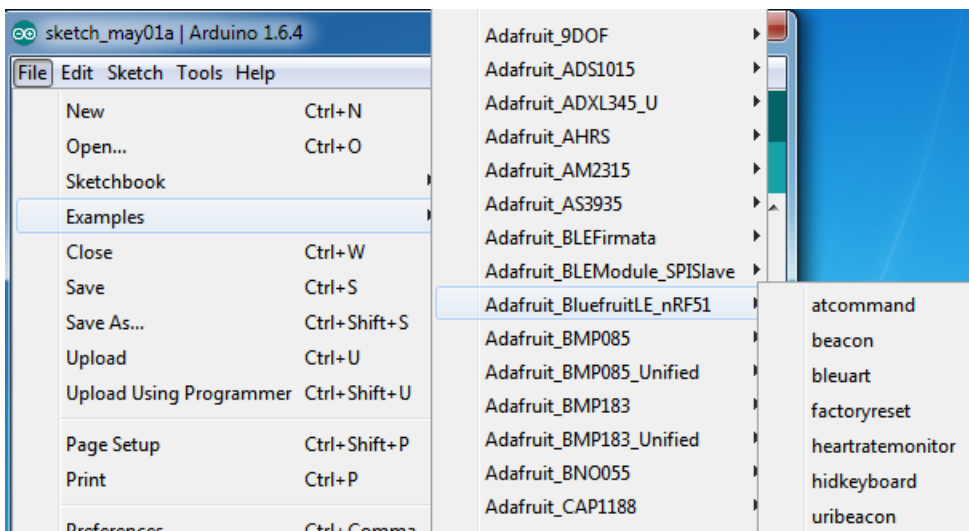
Rename the uncompressed folder **Adafruit_BluefruitLE_nRF51** and check that the **Adafruit_BluefruitLE_nRF51** folder contains **Adafruit_BLE.cpp** and **Adafruit_BLE.h** (as well as a bunch of other files)

Place the **Adafruit_BluefruitLE_nRF51** library folder your *arduinofolder/libraries/* folder. You may need to create the **libraries** subfolder if its your first library. Restart the IDE.

We also have a great tutorial on Arduino library installation at:

<http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use> (<https://adafru.it/aYM>)

After restarting, check that you see the library folder with examples:



Example Code

We have tons of examples that will get you going with the Bluefruit, and its really easy! Since we have one library for both wearable and non-wearable Bluefruit's, the example code is the same but may require some minor tweaks to adjust for Flora.

Most importantly, **the flora bluefruit is for use with Hardware Serial only** and does not have flow control pins. Also, **we expect you will not use the mode pad** so you can keep the switch set to CMD and then change into Data mode by sending +++ .

Configuring for Flora

For all of the examples, look near the top for a line like this:

```
///#define BLUEFRUIT_HWSERIAL_NAME      Serial1
```

and remove the // comment marks. Do the same for:

```
//Adafruit_BluefruitLE_UART ble(BLUEFRUIT_HWSERIAL_NAME, BLUEFRUIT_UART_MODE_PIN);
```

Next, remove the code for the SoftwareSerial type, find these lines and delete them:

```
/* Create the bluefruit object, either software serial... */  
  
SoftwareSerial bluefruitSS = SoftwareSerial(BLUEFRUIT_SWUART_TXD_PIN, BLUEFRUIT_SWUART_RXD_PIN);  
  
Adafruit_BluefruitLE_UART ble(bluefruitSS, BLUEFRUIT_UART_MODE_PIN,  
                               BLUEFRUIT_UART_CTS_PIN, BLUEFRUIT_UART_RTS_PIN);
```

Finally, unless you happen to be using the MODE pad, make sure **BLUEFRUIT_UART_MODE_PIN** is set to -1

```
#define BLUEFRUIT_UART_MODE_PIN      -1  // Not used with FLORA
```

Same with RTS and CTS, find these lines and set both to -1

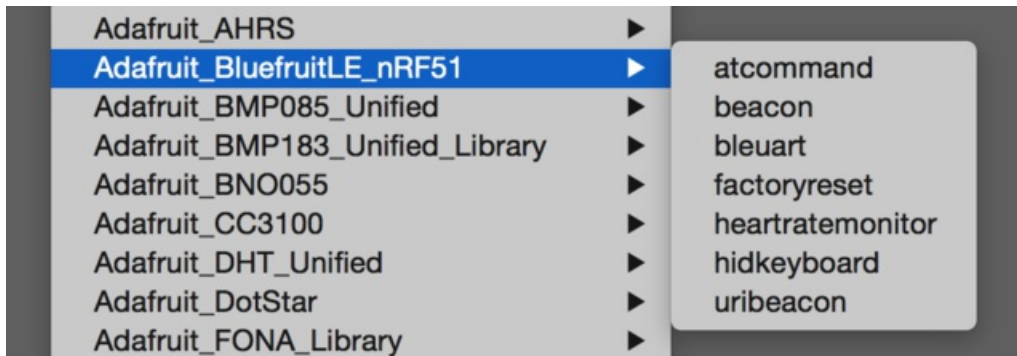
```
#define BLUEFRUIT_UART_CTS_PIN      -1  // Not used with FLORA  
#define BLUEFRUIT_UART_RTS_PIN      -1  // Not used with FLORA
```


ATCommand

The **ATCommand** example allows you to execute AT commands from your sketch, and see the results in the Serial Monitor. This can be useful for debugging, or just testing different commands out to see how they work in the real world. It's a good one to start with!

Opening the Sketch

To open the ATCommand sketch, click on the **File > Examples > Adafruit_BluefruitLE_nRF51** folder in the Arduino IDE and select **atcommand**:



This will open up a new instance of the example in the IDE, as shown below:



Configuration

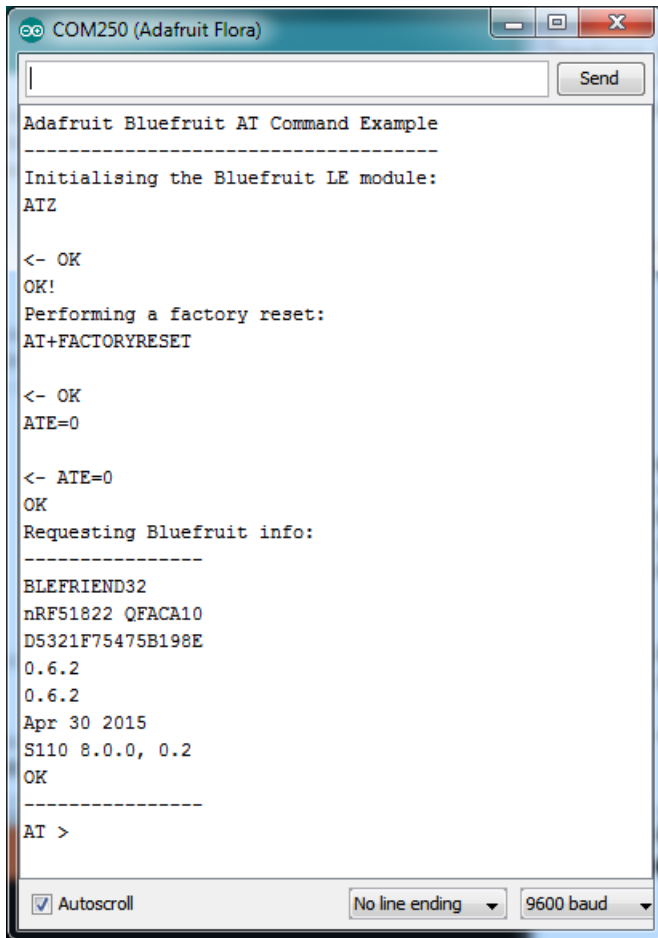
Check the **Configuration!** page earlier to set up the sketch for Software/Hardware UART or Software/Hardware SPI. The default is hardware SPI

If using software or hardware Serial UART:

- This tutorial does not need to use the MODE pin, make sure you have the mode switch in **CMD mode** if you do not configure & connect a MODE pin
- Don't forget to also connect the **CTS pin** on the Bluefruit to ground if you are not using it! (The Flora has this already done)

Running the Sketch

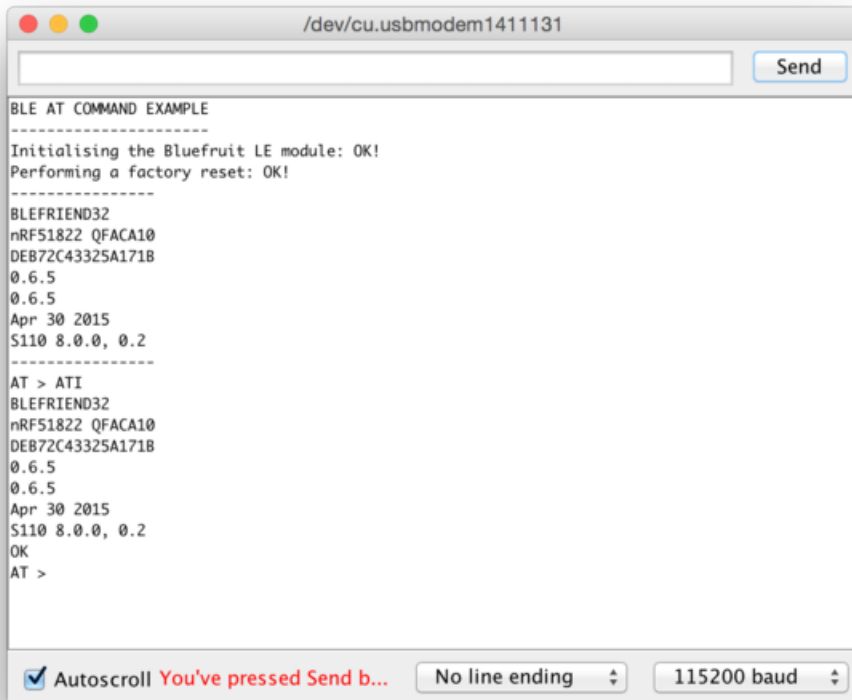
Once you upload the sketch to your board (via the arrow-shaped upload icon), and the upload process has finished, open up the Serial Monitor via **Tools > Serial Monitor**, and make sure that the baud rate in the lower right-hand corner is set to **115200**:



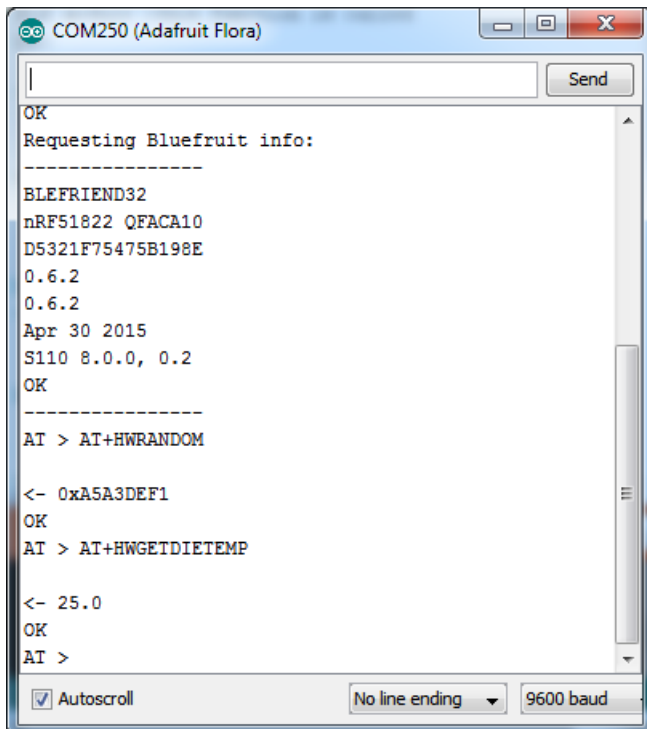
To send an AT command to the Bluefruit LE module, enter the command in the textbox at the top of the Serial Monitor and click the **Send** button:



The response to the AT command will be displayed in the main part of the Serial Monitor. The response from '**ATI**' is shown below:



You can do pretty much anything at this prompt, with the AT command set. Try **AT+HELP** to get a list of all commands, and try out ones like **AT+HWGETDIETEMP** (get temperature at the nRF51822 die) and **AT+HWRANDOM** (generate a random number)

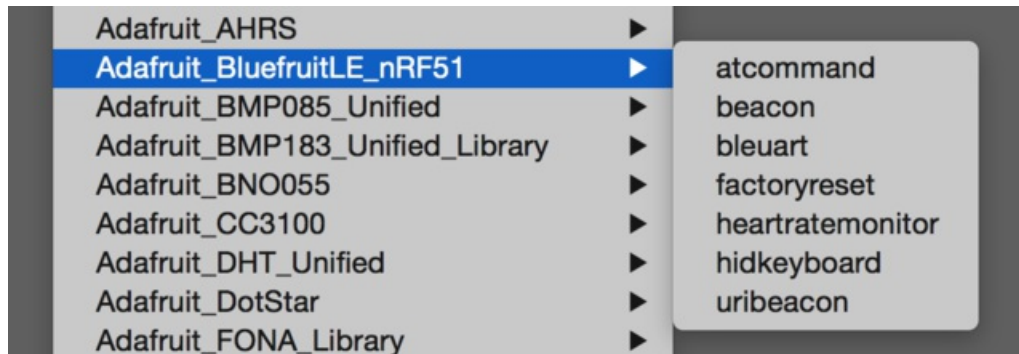


BLEUart

The **BLEUart** example sketch allows you to send and receive text data between the Arduino and a connected Bluetooth Low Energy Central device on the other end (such as your mobile phone using the **Adafruit Bluefruit LE Connect** application for [Android](https://adafru.it/f4G) (<https://adafru.it/f4G>) or [iOS](https://adafru.it/f4H) (<https://adafru.it/f4H>) in UART mode).

Opening the Sketch

To open the ATCommand sketch, click on the **File > Examples > Adafruit_BluefruitLE_nRF51** folder in the Arduino IDE and select **bleuart_cmdmode**:



This will open up a new instance of the example in the IDE, as shown below:

```
bleuart_cmdmode | Arduino 1.6.4
File Edit Sketch Tools Help
bleuart_cmdmode $
/*!
  @file    bleuart_cmdmode.ino
  @author  hathach, ktown (Adafruit Industries)

  This demo will show you how to send and receive data in COMMAND mode
  (without needing to put the module into DATA mode or using the MODE pin)
*/
#include <string.h>
#include <Arduino.h>
#include <SPI.h>
#include <SoftwareSerial.h>

#include "Adafruit_BLE.h"
#include "Adafruit_BLE_HWSPI.h"
#include "Adafruit_BluefruitLE_UART.h"

// If you are using Software Serial...
// The following macros declare the pins used for SW serial, you should
// use these pins if you are connecting the UART Friend to an UNO
#define BLUEFRUIT_SWUART_RXD_PIN    9    // Required for software serial!
#define BLUEFRUIT_SWUART_TXD_PIN    10   // Required for software serial!
#define BLUEFRUIT_UART_CTS_PIN      11   // Required for software serial!
#define BLUEFRUIT_UART_RTS_PIN      -1   // Optional, set to -1 if unused

// If you are using Hardware Serial
// The following macros declare the Serial port you are using. Uncomment this
// line if you are connecting the BLE to Leonardo/Micro or Flora
```

Configuration

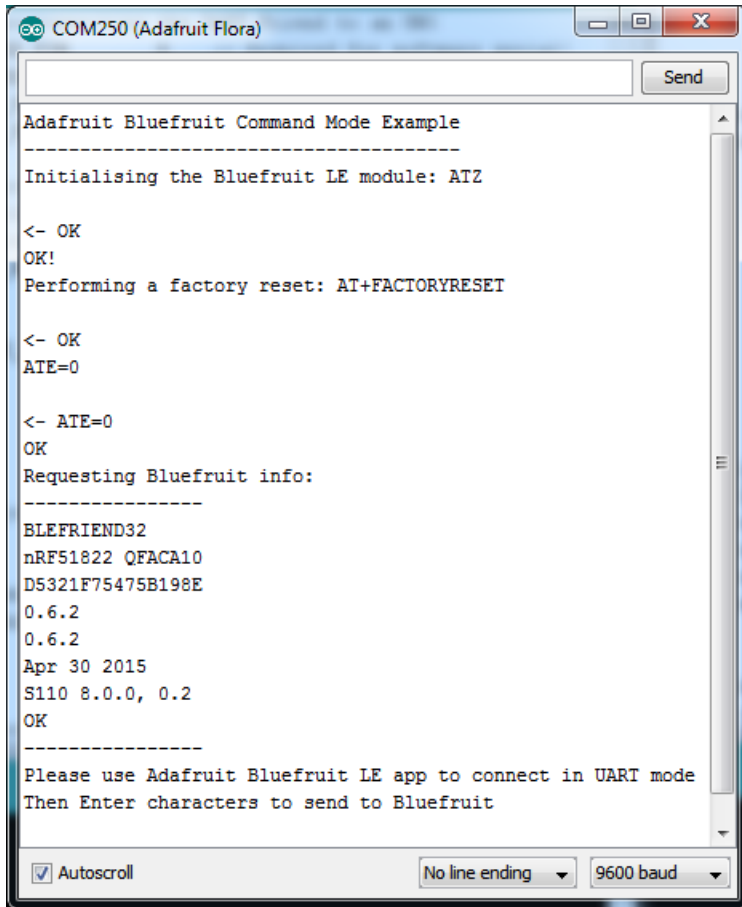
Check the **Configuration!** page earlier to set up the sketch for Software/Hardware UART or Software/Hardware SPI. The default is hardware SPI

If using software or hardware Serial UART:

- This tutorial does not need to use the MODE pin, make sure you have the mode switch in **CMD** mode if you do not configure & connect a MODE pin
- Don't forget to also connect the CTS pin on the Bluefruit to ground if you are not using it!(The Flora has this already done)

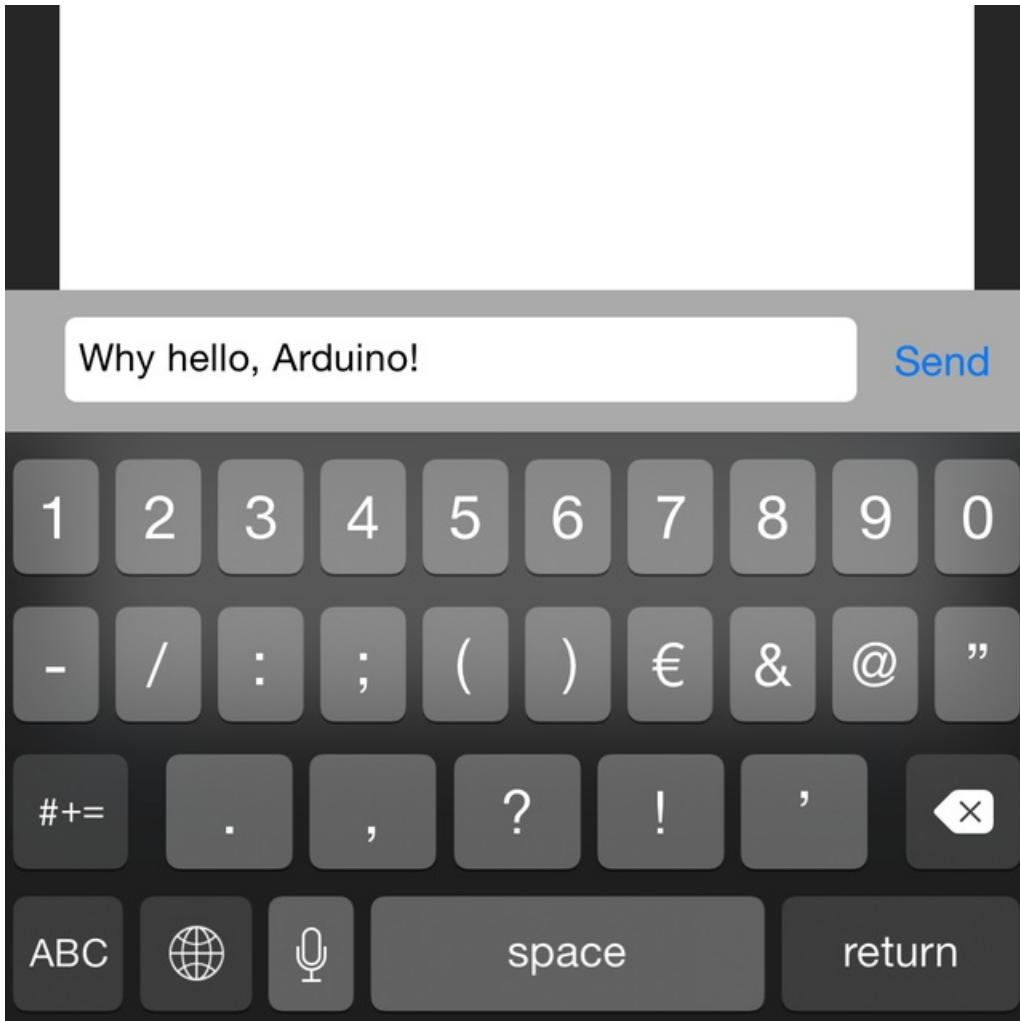
Running the Sketch

Once you upload the sketch to your board (via the arrow-shaped upload icon), and the upload process has finished, open up the Serial Monitor via **Tools > Serial Monitor**, and make sure that the baud rate in the lower right-hand corner is set to **115200**:

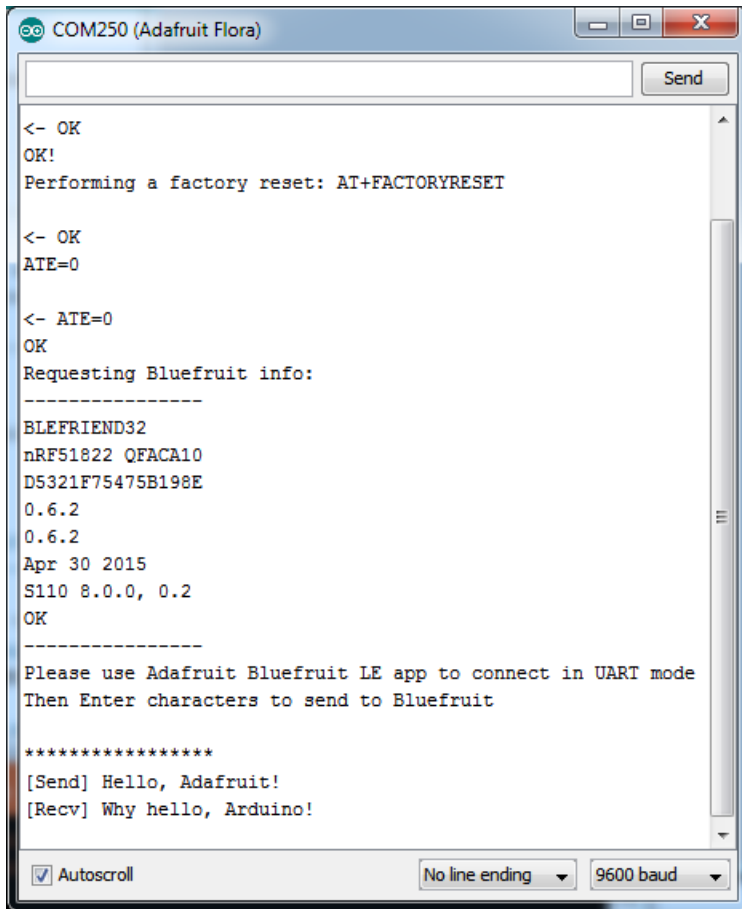


Once you see the request, use the App to connect to the Bluefruit LE module in **UART** mode so you get the text box on your phone

Any text that you type in the box at the top of the Serial Monitor will be sent to the connected phone, and any data sent from the phone will be displayed in the serial monitor:



The response text ('Why hello, Arduino!') can be seen below:



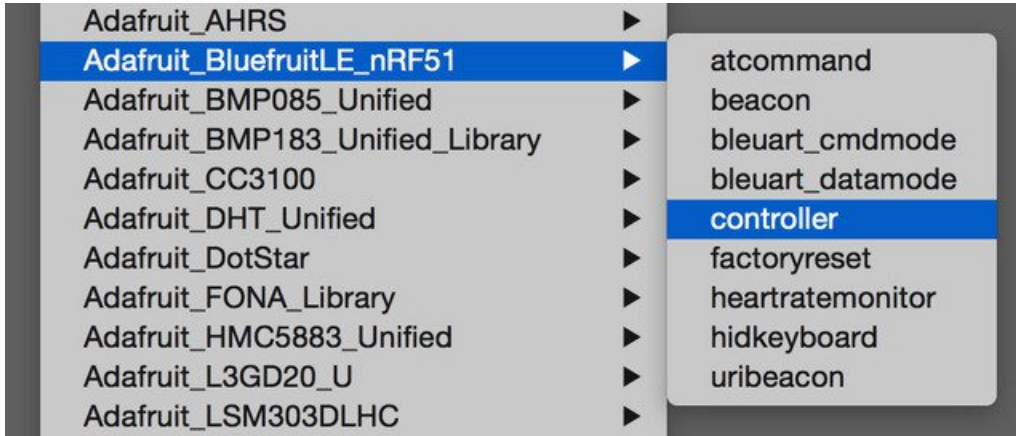
Controller

The **Controller** sketch allows you to turn your BLE-enabled iOS or Android device in a hand-held controller or an external data source, taking advantage of the wealth of sensors on your phone or tablet.

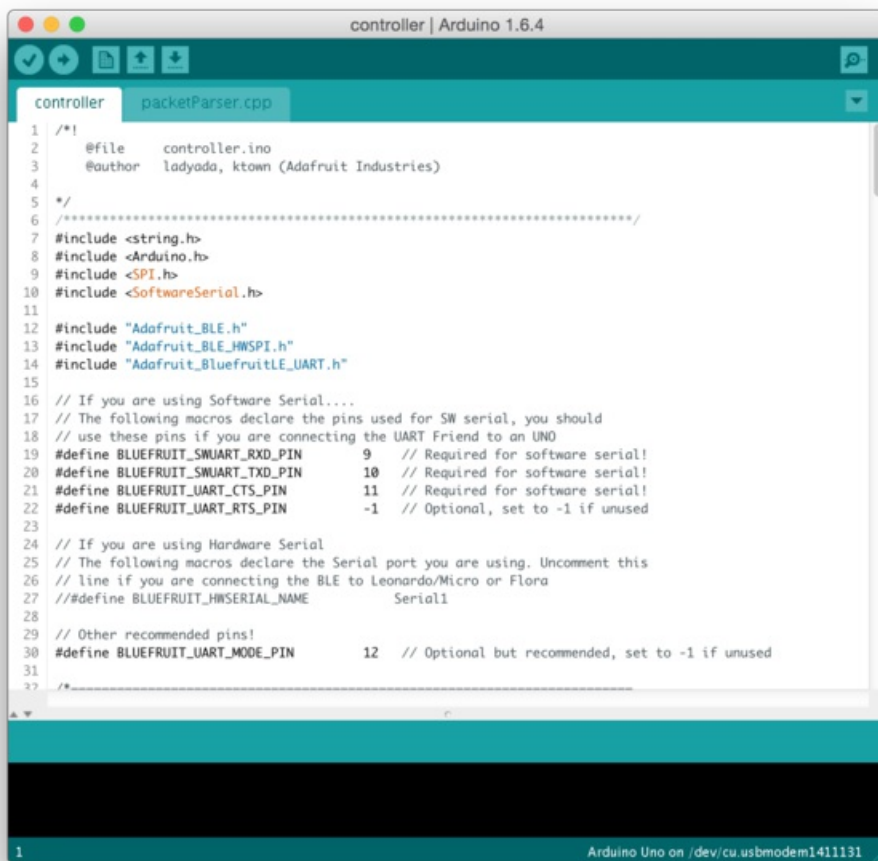
You can take accelerometer or quaternion data from your phone, and push it out to your Arduino via BLE, or get the latest GPS co-ordinates for your device without having to purchase (or power!) any external HW.

Opening the Sketch

To open the Controller sketch, click on the **File > Examples > Adafruit_BluefruitLE_nRF51** folder in the Arduino IDE and select **controller**:



This will open up a new instance of the example in the IDE, as shown below:



```
1  /*!
2   @file    controller.ino
3   @author  ladyada, ktown (Adafruit Industries)
4
5  */
6  //*****
7  #include <string.h>
8  #include <Arduino.h>
9  #include <SPI.h>
10 #include <SoftwareSerial.h>
11
12 #include "Adafruit_BLE.h"
13 #include "Adafruit_BLE_HWSPI.h"
14 #include "Adafruit_BluefruitLE_UART.h"
15
16 // If you are using Software Serial...
17 // The following macros declare the pins used for SW serial, you should
18 // use these pins if you are connecting the UART Friend to an UNO
19 #define BLUEFRUIT_SWUART_RXD_PIN    9 // Required for software serial!
20 #define BLUEFRUIT_SWUART_TXD_PIN    10 // Required for software serial!
21 #define BLUEFRUIT_UART_CTS_PIN      11 // Required for software serial!
22 #define BLUEFRUIT_UART_RTS_PIN      -1 // Optional, set to -1 if unused
23
24 // If you are using Hardware Serial
25 // The following macros declare the Serial port you are using. Uncomment this
26 // line if you are connecting the BLE to Leonardo/Micro or Flora
27 // #define BLUEFRUIT_HWSERIAL_NAME    Serial1
28
29 // Other recommended pins!
30 #define BLUEFRUIT_UART_MODE_PIN      12 // Optional but recommended, set to -1 if unused
31
32 */
```

Configuration

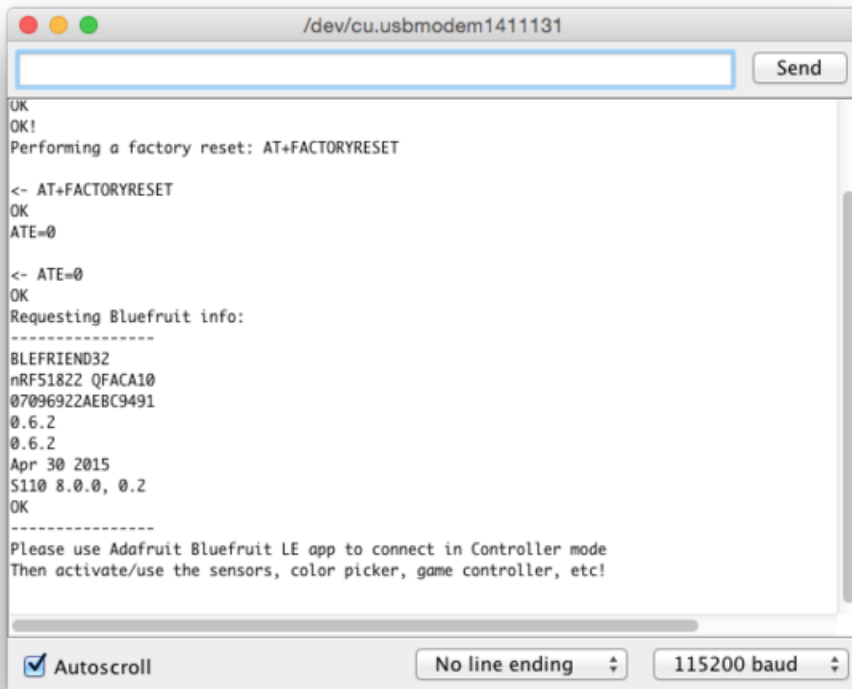
Check the Configuration! page earlier to set up the sketch for Software/Hardware UART or Software/Hardware SPI. The default is hardware SPI

If using software or hardware Serial UART:

- This tutorial will also be easier to use if you wire up the MODE pin, you can use any pin but our tutorial has pin 12 by default. You can change this to any pin. If you do not set the MODE pin then **make sure you have the mode switch in CMD mode**
- If you are using a Flora or otherwise don't want to wire up the Mode pin, set the BLUEFRUIT_UART_MODE_PIN to -1 in the configuration tab so that the sketch will use the +++ method to switch between Command and Data mode!
- Don't forget to also **connect the CTS pin on the Bluefruit to ground if you are not using it!** (The Flora has this already done)

Running the Sketch

Once you upload the sketch to your board (via the arrow-shaped upload icon), and the upload process has finished, open up the Serial Monitor via **Tools > Serial Monitor**, and make sure that the baud rate in the lower right-hand corner is set to **115200**:



Using Bluefruit LE Connect in Controller Mode

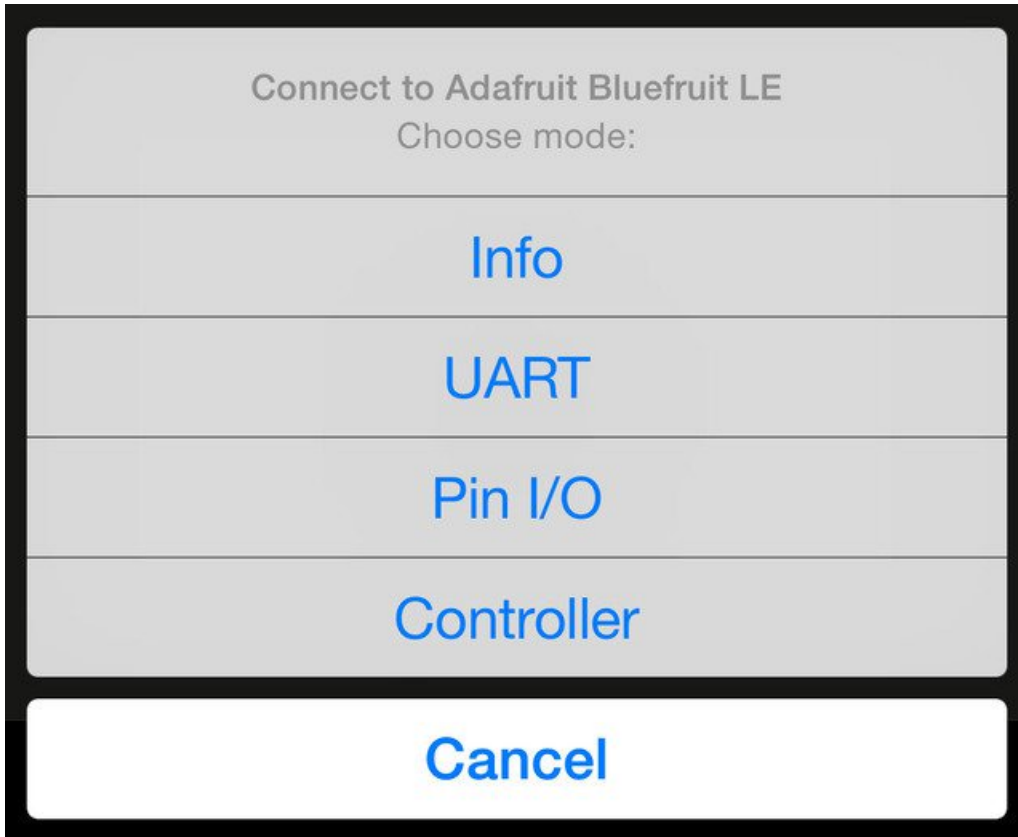
Once the sketch is running you can open Adafruit's Bluefruit LE Connect application (available for [Android \(https://adafru.it/f4G\)](https://adafru.it/f4G) or [iOS \(https://adafru.it/f4H\)](https://adafru.it/f4H)) and use the **Controller** application to interact with the sketch. (If you're new to Bluefruit LE Connect, have a look at our [dedicated Bluefruit LE Connect learning guide \(https://adafru.it/iCm\)](https://adafru.it/iCm).)

On the welcome screen, select the **Adafruit Bluefruit LE** device from the list of BLE devices in range:



Connect

Then from the activity list select **Controller**:



This will bring up a list of data points you can send from your phone or tablet to your Bluefruit LE module, by enabling or disabling the appropriate sensor(s).

Streaming Sensor Data

You can take Quaternion (absolute orientation), Accelerometer, Gyroscope, Magnetometer or GPS Location data from your phone and send it directly to your Arduino from the Controller activity.

By enabling the **Accelerometer** field, for example, you should see accelerometer data update in the app:

STREAM SENSOR DATA

Quaternion

OFF

Accelerometer

ON

x: 0.15683

y: -0.580338

z: -0.794373

Gyro

OFF

Magnetometer

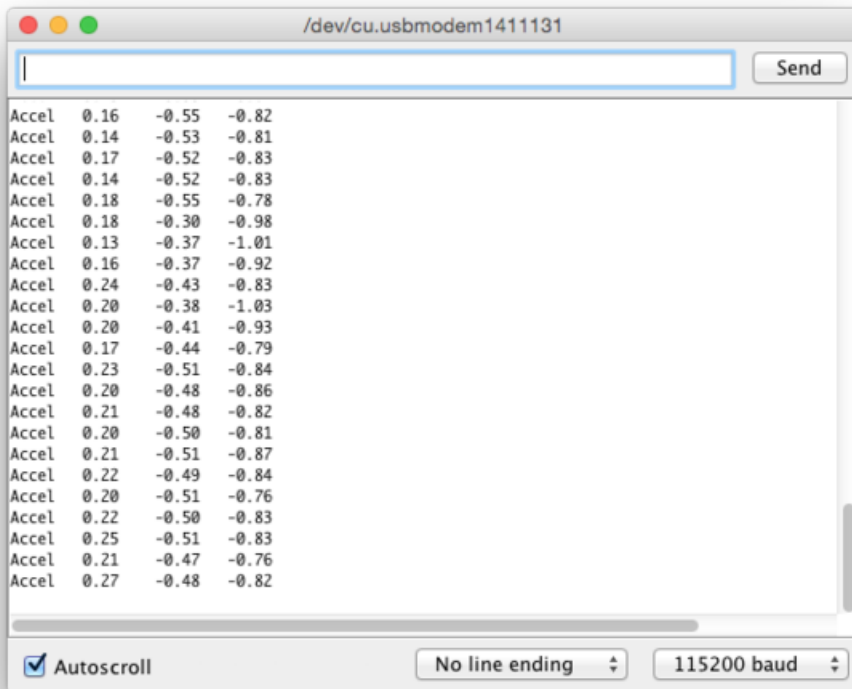
OFF

Location

OFF

The data is parsed in the example sketch and output to the Serial Monitor as follows:

```
Accel 0.20 -0.51 -0.76  
Accel 0.22 -0.50 -0.83  
Accel 0.25 -0.51 -0.83  
Accel 0.21 -0.47 -0.76  
Accel 0.27 -0.48 -0.82
```

Note that even though we only print 2 decimal points, the values are received from the App as a full 4-byte floating point.

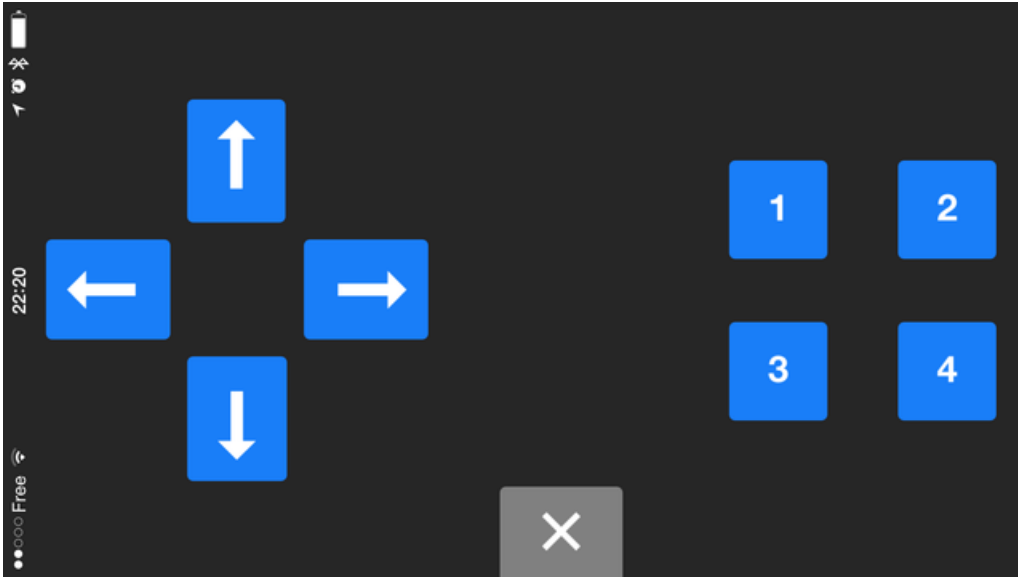
Control Pad Module

You can also use the **Control Pad Module** to capture button presses and releases by selecting the appropriate menu item:

Control Pad



This will bring up the Control Pad panel, shown below:



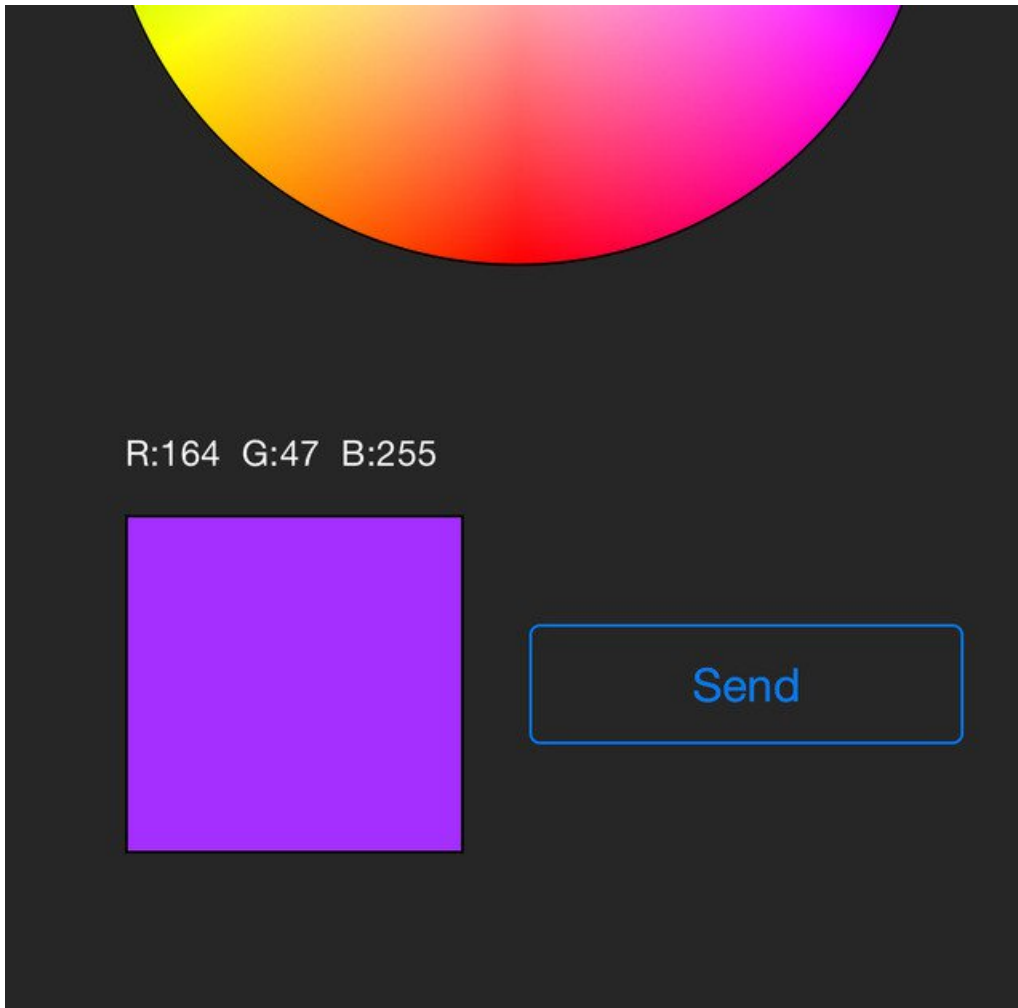
Button presses and releases will all be logged to the Serial Monitor with the ID of the button used:

```
Button 8 pressed
Button 8 released
Button 3 pressed
Button 3 released
```

Color Picker Module

You can also send RGB color data via the **Color Picker** module, which presents the following color selection dialogue:





This will give you Hexadecimal color data in the following format:

```
RGB #A42FFF
```

You can combine the color picker and controller sample sketches to make color-configurable animations triggered by buttons in the mobile app-- very handy for wearables! Download this combined sample code (configured for Feather but easy to adapt to FLORA, BLE Micro, etc.) to get started:

<https://adafru.it/kzF>

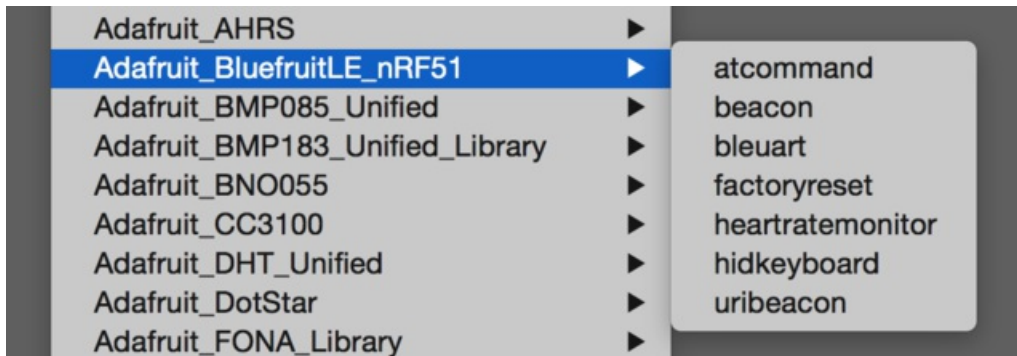
<https://adafru.it/kzF>

HIDKeyboard

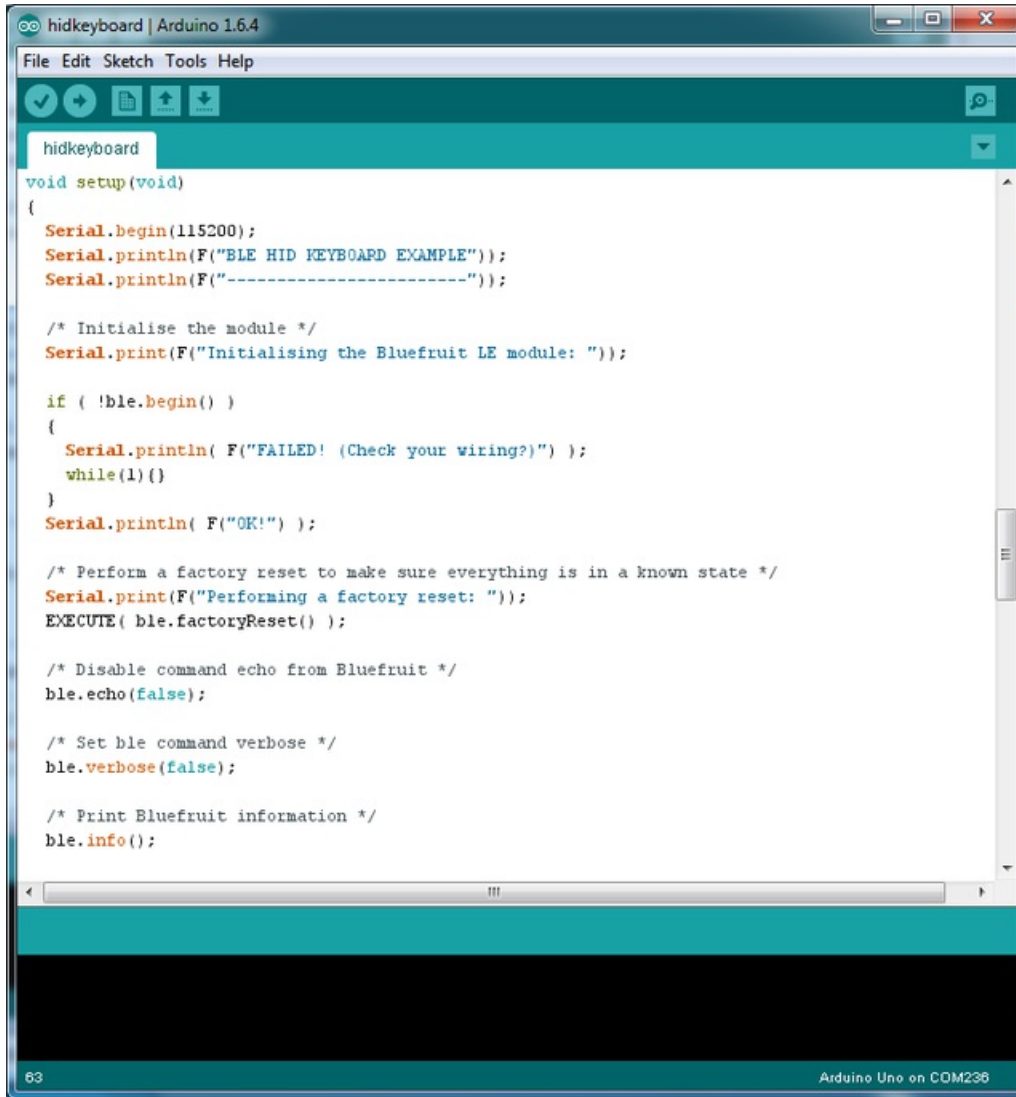
The **HIDKeyboard** example shows you how you can use the built-in HID keyboard AT commands to send keyboard data to any BLE-enabled Android or iOS phone, or other device that supports BLE HID peripherals.

Opening the Sketch

To open the ATCommand sketch, click on the **File > Examples > Adafruit_BluefruitLE_nRF51** folder in the Arduino IDE and select **hidkeyboard**:



This will open up a new instance of the example in the IDE, as shown below:



```
void setup(void)
{
  Serial.begin(115200);
  Serial.println(F("BLE HID KEYBOARD EXAMPLE"));
  Serial.println(F("-----"));

  /* Initialise the module */
  Serial.print(F("Initialising the Bluefruit LE module: "));

  if ( !ble.begin() )
  {
    Serial.println( F("FAILED! (Check your wiring?)") );
    while(1){}
  }
  Serial.println( F("OK!") );

  /* Perform a factory reset to make sure everything is in a known state */
  Serial.print(F("Performing a factory reset: "));
  EXECUTE( ble.factoryReset() );

  /* Disable command echo from Bluefruit */
  ble.echo(false);

  /* Set ble command verbose */
  ble.verbose(false);

  /* Print Bluefruit information */
  ble.info();
}
```

Configuration

Check the **Configuration!** page earlier to set up the sketch for Software/Hardware UART or Software/Hardware SPI. The default is hardware SPI

If using software or hardware Serial UART:

- This tutorial does not need to use the MODE pin, **make sure you have the mode switch in CMD mode!**
- Don't forget to also **connect the CTS pin on the Bluefruit to ground if you are not using it!** (The Flora has this already done)

Running the Sketch

Once you upload the sketch to your board (via the arrow-shaped upload icon), and the upload process has finished, open up the Serial Monitor via **Tools > Serial Monitor**, and make sure that the baud rate in the lower right-hand corner is set to **115200**:

```
COM250 (Adafruit Flora)
|
| Send
|
Adafruit Bluefruit HID Keyboard Example
-----
Initialising the Bluefruit LE module:
ATZ

<- OK
OK!
Performing a factory reset:
AT+FACTORYRESET

<- OK
ATE=0

<- ATE=0
OK
Requesting Bluefruit info:
-----
BLEFRIEND32
nRF51822 QFACA10
D5321F75475B198E
0.6.2
0.6.2
Apr 30 2015
S110 8.0.0, 0.2
OK
-----
Setting device name to 'Bluefruit Keyboard':
AT+GAPDEVNAME=Bluefruit Keyboard

<- OK
Enable Keyboard Service:
AT+BleKeyboardEn=On

<- OK
Performing a SW reset (service changes require a reset):
ATZ

<- OK

Go to your phone's Bluetooth settings to pair your device
then open an application that accepts keyboard input

Enter the character(s) to send:
- \r for Enter
- \n for newline
- \t for tab
- \b for backspace

keyboard >

 Autoscroll
No line ending
115200 baud
```

To send keyboard data, type anything into the textbox at the top of the Serial Monitor and click the **Send** button.

Bonding the HID Keyboard

Before you can use the HID keyboard, you will need to 'bond' it to your phone or PC. The bonding process establishes a permanent connection between the two devices, meaning that as soon as your phone or PC sees the Bluefruit LE module again it will automatically connect.

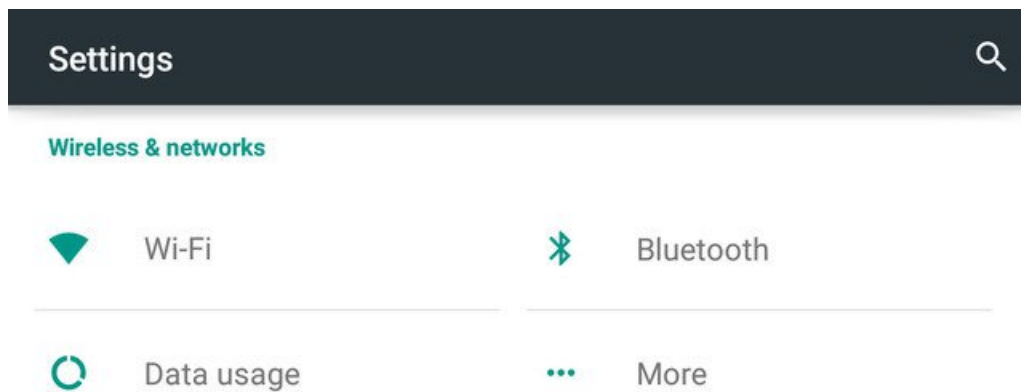
The exact procedures for bonding the keyboard will vary from one platform to another.

When you no longer need a bond, or wish to bond the Bluetooth LE module to another device, be sure to delete the bonding information on the phone or PC, otherwise you may not be able to connect on a new device!

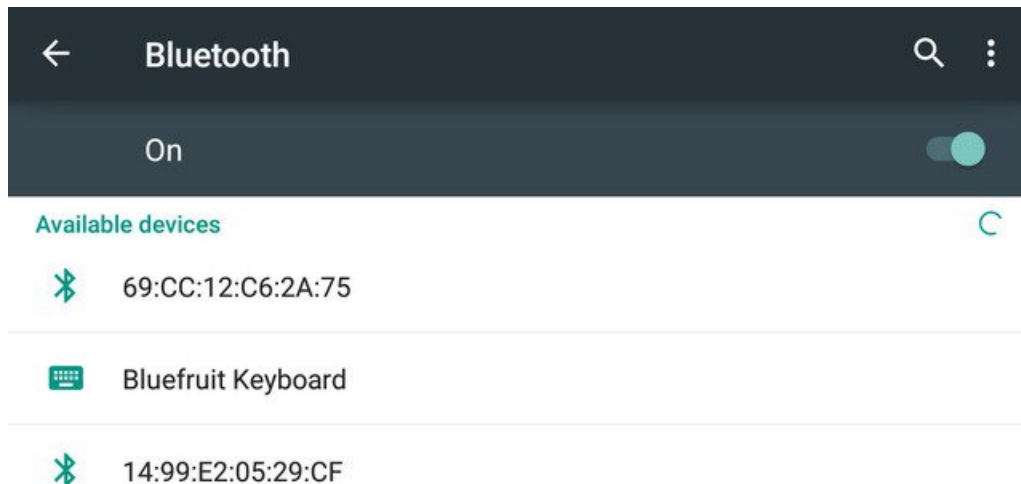
Android

To bond the keyboard on a Bluetooth Low Energy enabled Android device, go to the **Settings** application and click the **Bluetooth** icon.

These screenshots are based on Android 5.0 running on a Nexus 7 2013. The exact appearance may vary depending on your device and OS version.



Inside the Bluetooth setting panel you should see the Bluetooth LE module advertising itself as **Bluefruit Keyboard** under the 'Available devices' list:



Nexus 7 is visible to nearby devices while Bluetooth Settings is open.

Tapping the device will start the bonding process, which should end with the Bluefruit Keyboard device being moved

to a new 'Paired devices' list with 'Connected' written underneath the device name:

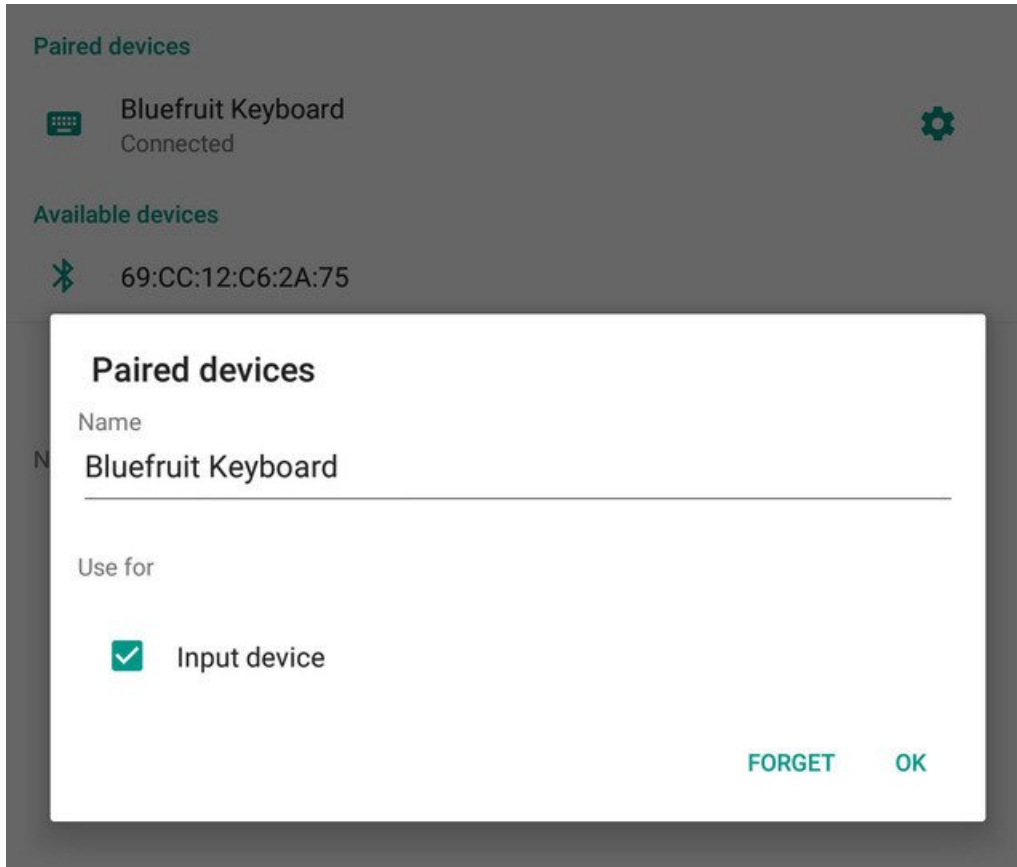
Paired devices



Bluefruit Keyboard
Connected



To delete the bonding information, click the gear icon to the right of the device name and then click the **Forget** button:



iOS

To bond the keyboard on an iOS device, go to the **Settings** application on your phone, and click the **Bluetooth** menu item.

The keyboard should appear under the **OTHER DEVICES** list:



Bluetooth



Now discoverable as “iPhone de Kevin”.

MY DEVICES

SONY:CMT-X5CD

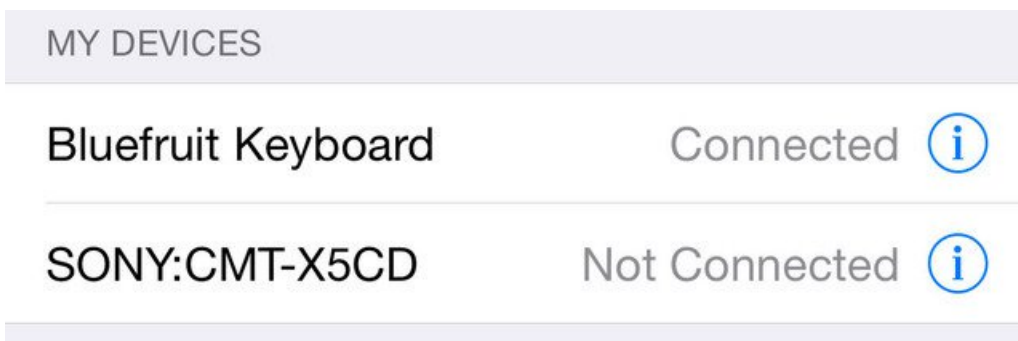
Not Connected 

OTHER DEVICES

Adafruit Bluefruit LE

To pair an Apple Watch with your iPhone, go to the [Apple Watch app](#).

Once the bonding process is complete the device will be moved to the **MY DEVICES** category, and you can start to use the Bluefruit LE module as a keyboard:



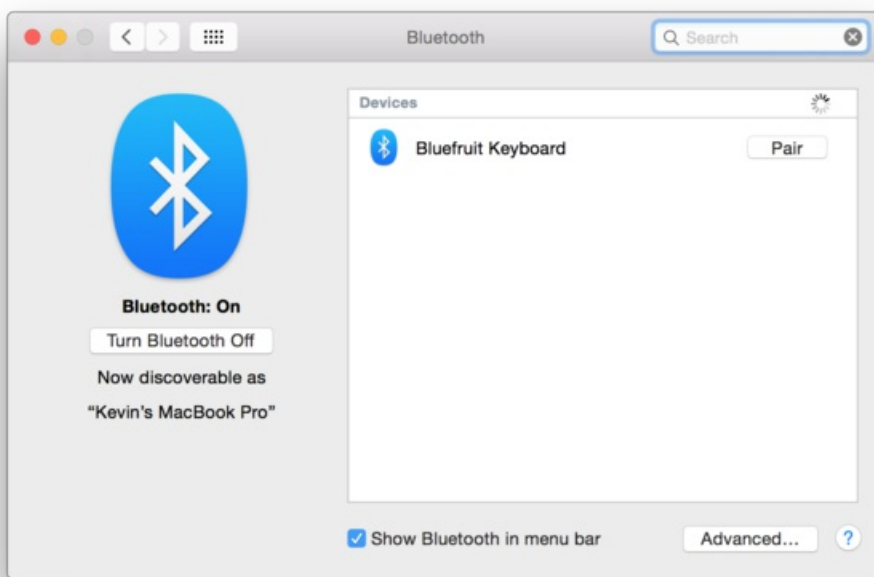
To unbond the device, click the 'info' icon and then select the **Forget this Device** option in the menu:

Bluetooth Bluefruit Keyboard

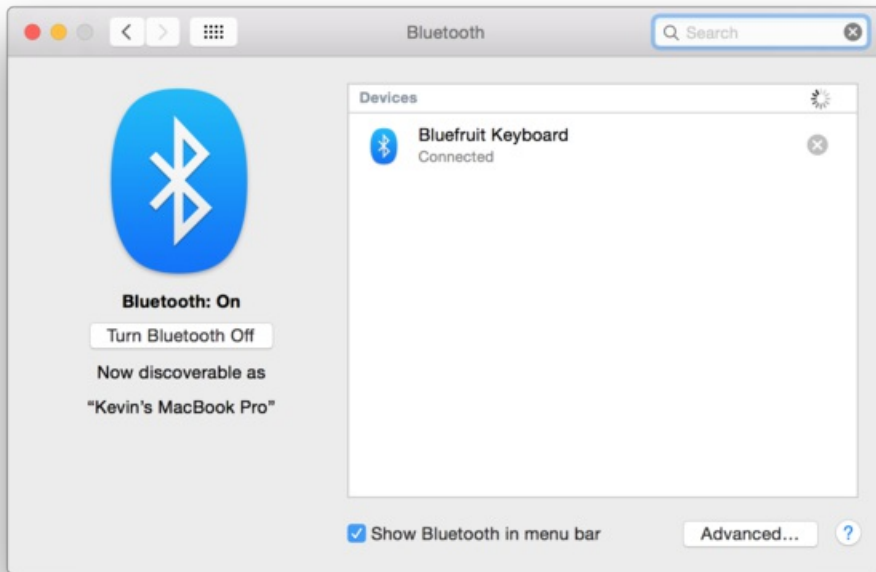
Forget This Device

OS X

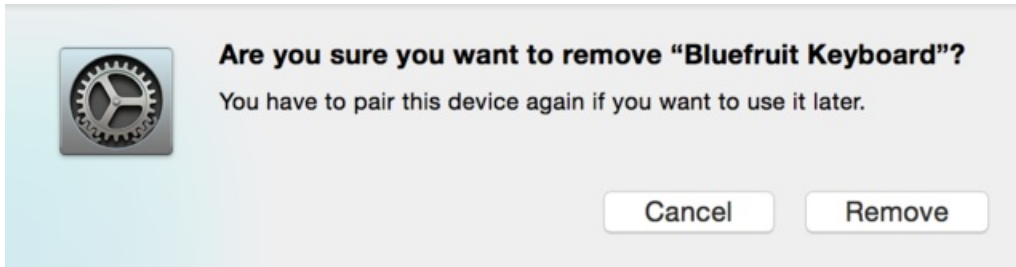
To bond the keyboard on an OS X device, go to the **Bluetooth Preferences** window and click the **Pair** button beside the **Bluefruit Keyboard** device generated by this example sketch:



To unbond the device once it has been paired, click the small 'x' icon beside **Bluefruit Keyboard**:



... and then click the **Remove** button when the confirmation dialogue box pops up:

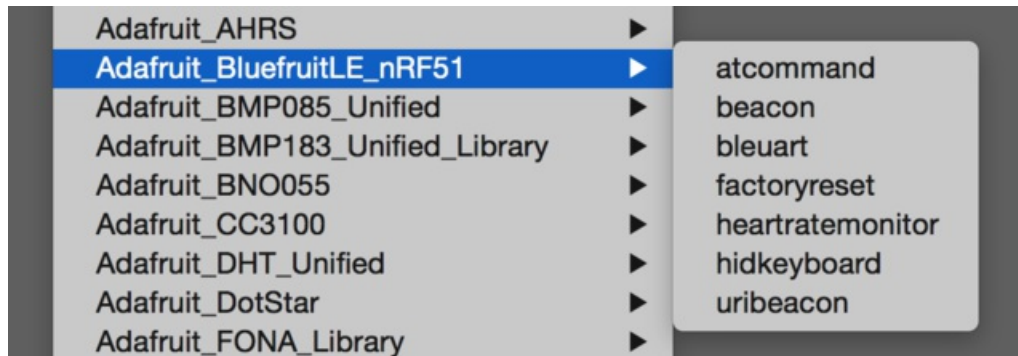


UriBeacon

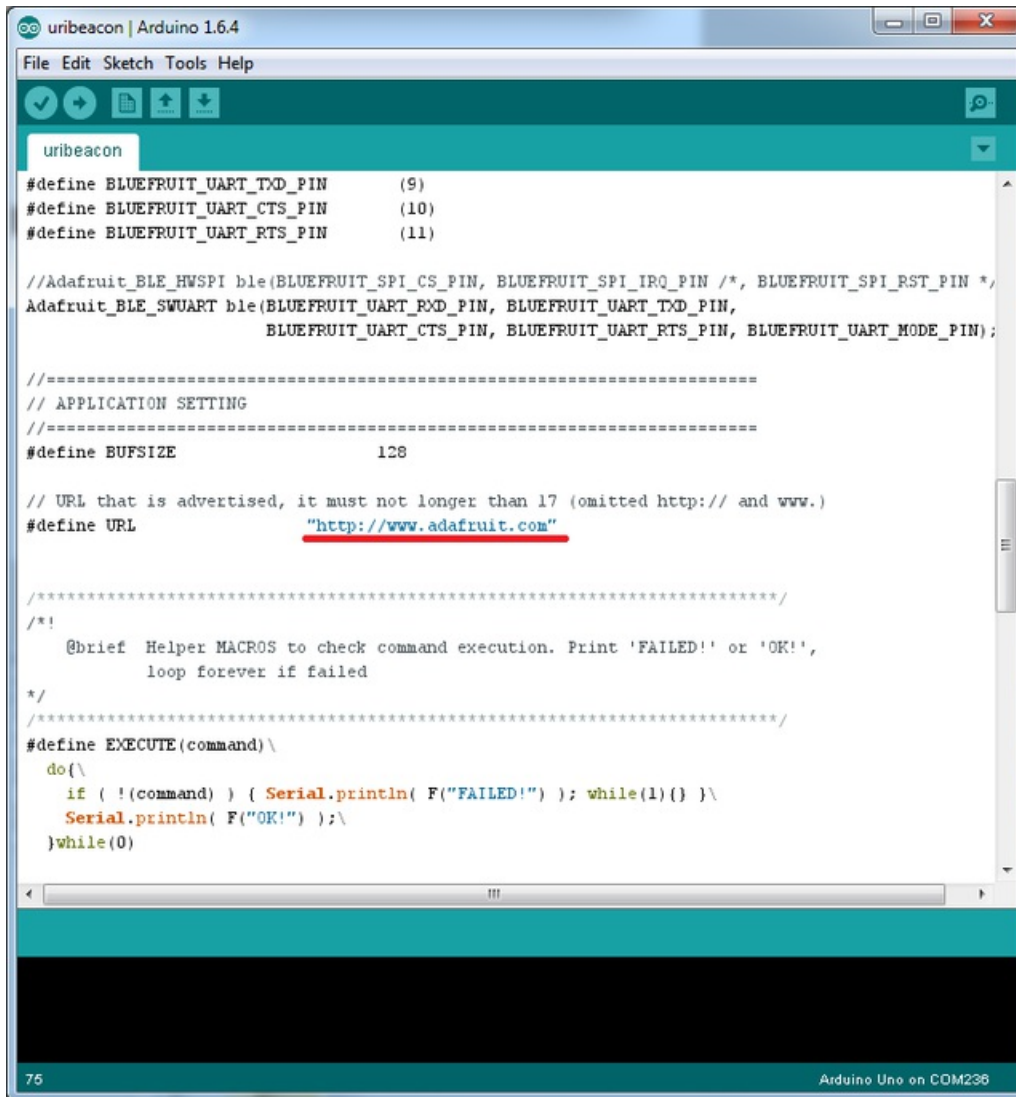
The **UriBeacon** example shows you how to use the built-in UriBeacon AT commands to configure the Bluefruit LE module as a UriBeacon advertiser, following Google's Physical Web [UriBeacon \(https://adafru.it/edk\)](https://adafru.it/edk) specification.

Opening the Sketch

To open the ATCommand sketch, click on the **File > Examples > Adafruit_BluefruitLE_nRF51** folder in the Arduino IDE and select **uribeacon**:



This will open up a new instance of the example in the IDE, as shown below. You can edit the URL that the beacon will point to, from the default <http://www.adafruit.com> or just upload as is to test



Configuration

Check the **Configuration!** page earlier to set up the sketch for Software/Hardware UART or Software/Hardware SPI. The default is hardware SPI

If using software or hardware Serial UART:

- This tutorial does not need to use the MODE pin, **make sure you have the mode switch in CMD mode** if you do not configure & connect a MODE pin
- Don't forget to also **connect the CTS pin on the Bluefruit to ground** if you are not using it!(The Flora has this already done)

Running the Sketch

Once you upload the sketch to your board (via the arrow-shaped upload icon), and the upload process has finished, open up the Serial Monitor via **Tools > Serial Monitor**, and make sure that the baud rate in the lower right-hand corner is set to **115200**:

```
COM250 (Adafruit Flora)
| Send
Adafruit Bluefruit UriBeacon Example
-----
Initialising the Bluefruit LE module:
ATZ

<- OK
OK!
Performing a factory reset:
AT+FACTORYRESET

<- OK
ATE=0

<- ATE=0
OK
Requesting Bluefruit info:
-----
BLEFRIEND32
nRF51822 QFACA10
D5321F75475B198E
0.6.2
0.6.2
Apr 30 2015
S110 8.0.0, 0.2
OK
-----
Setting uri beacon to Adafruit website: AT+BLEURIBEACON=http://www.adafruit.com

<- OK

Please use Google Physical Web application to test

 Autoscroll
No line ending
115200 baud
```

At this point you can open the Physical Web Application for [Android \(https://adafru.it/edi\)](https://adafru.it/edi) or for [iOS \(https://adafru.it/edj\)](https://adafru.it/edj), and you should see a link advertising Adafruit's website:

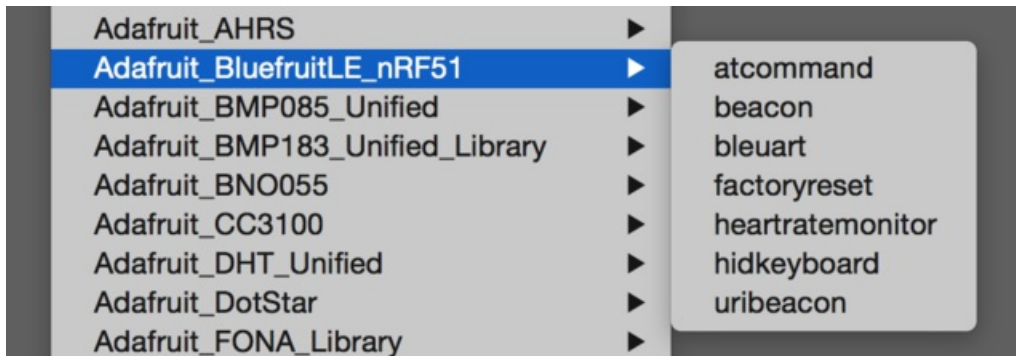


HeartRateMonitor

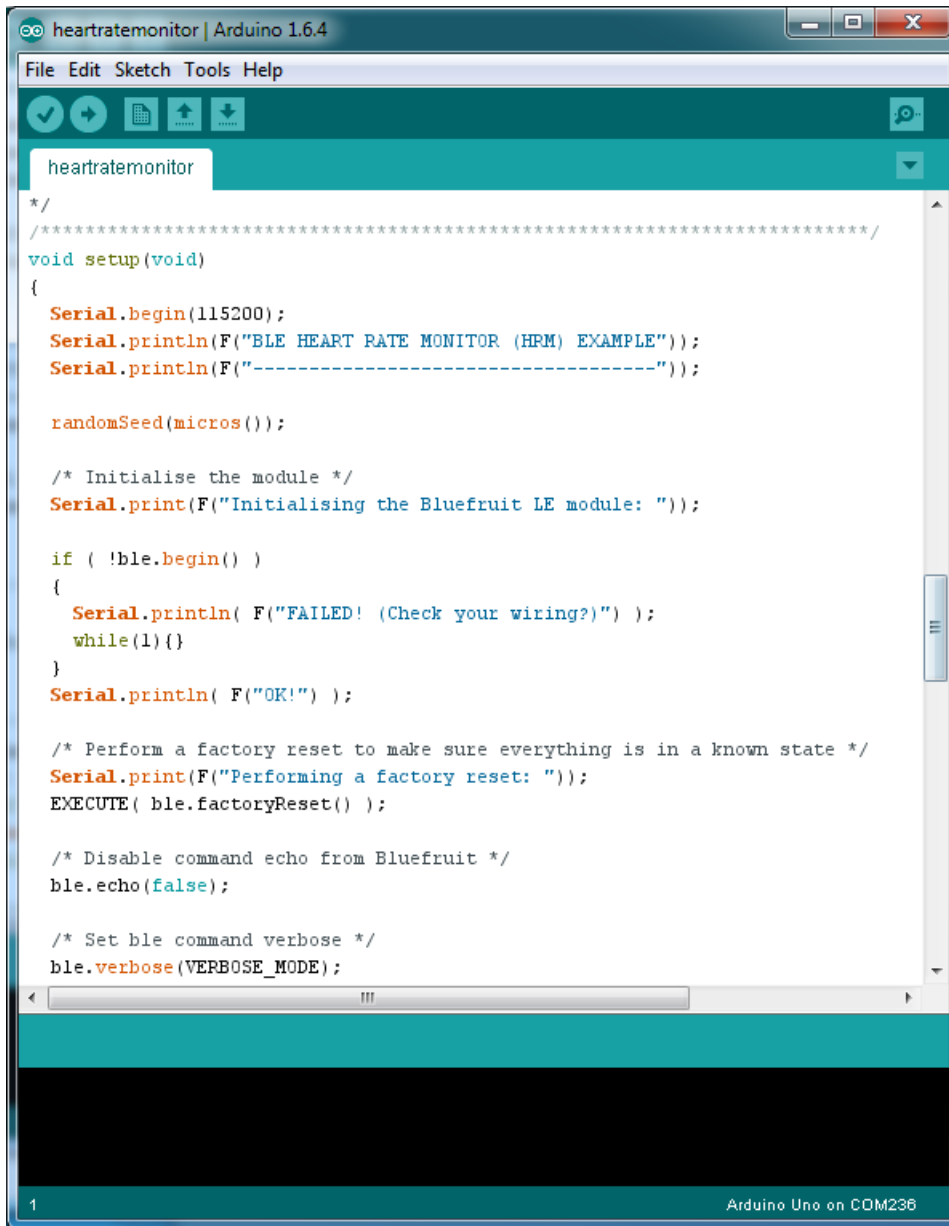
The **HeartRateMonitor** example allows you to define a new GATT Service and associated GATT Characteristics, and update the characteristic values using standard AT commands.

Opening the Sketch

To open the ATCommand sketch, click on the **File > Examples > Adafruit_BluefruitLE_nRF51** folder in the Arduino IDE and select **heartratemonitor**:



This will open up a new instance of the example in the IDE, as shown below:



Configuration

Check the **Configuration!** page earlier to set up the sketch for Software/Hardware UART or Software/Hardware SPI. The default is hardware SPI

If Using Hardware or Software UART

This tutorial does not need to use the MODE pin, **make sure you have the mode switch in CMD mode** if you do not configure & connect a MODE pin

This demo uses some long data transfer strings, so we recommend defining and connecting both CTS and RTS to pins, even if you are using hardware serial.

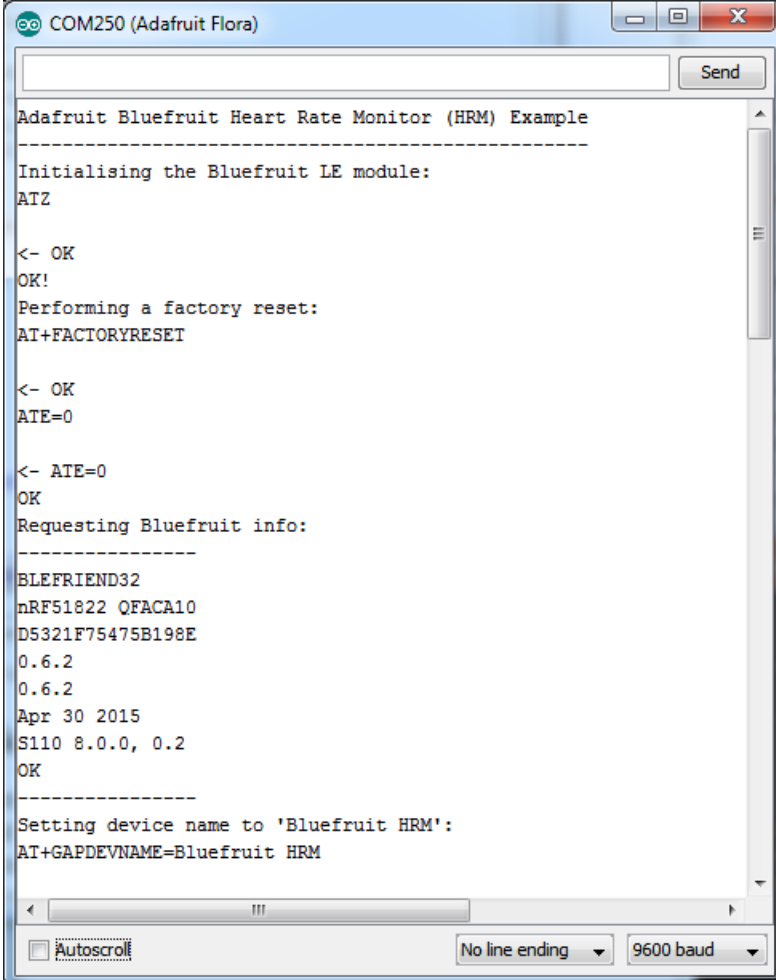
If you are using a Flora or just dont want to connect CTS or RTS, set the pin #define's to -1 and **Don't forget to also connect the CTS pin on the Bluefruit to ground!** (The Flora has this already done)

If you are using RTS and CTS, you can remove this line below, which will slow down the data transmission

```
// this line is particularly required for Flora, but is a good idea
// anyways for the super long lines ahead!
ble.setInterCharWriteDelay(5); // 5 ms
```

Running the Sketch

Once you upload the sketch to your board (via the arrow-shaped upload icon), and the upload process has finished, open up the Serial Monitor via **Tools > Serial Monitor**, and make sure that the baud rate in the lower right-hand corner is set to **115200**:



The screenshot shows the Serial Monitor window for a COM250 (Adafruit Flora) port. The window title is "COM250 (Adafruit Flora)". The output text is as follows:

```
Adafruit Bluefruit Heart Rate Monitor (HRM) Example
-----
Initialising the Bluefruit LE module:
ATZ

<- OK
OK!
Performing a factory reset:
AT+FACTORYRESET

<- OK
ATE=0

<- ATE=0
OK
Requesting Bluefruit info:
-----
BLEFRIEND32
nRF51822 QFACA10
D5321F75475B198E
0.6.2
0.6.2
Apr 30 2015
S110 8.0.0, 0.2
OK
-----
Setting device name to 'Bluefruit HRM':
AT+GAPDEVNAME=Bluefruit HRM
```

At the bottom of the window, there is a "Send" button, an "Autoscroll" checkbox, a "No line ending" dropdown menu, and a "9600 baud" dropdown menu.

```
COM250 (Adafruit Flora)
Setting device name to 'Bluefruit HRM':
AT+GAPDEVNAME=Bluefruit HRM

<- OK
Adding the Heart Rate Service definition (UUID = 0x180D):
AT+GATTADDSERVICE=UUID=0x180D

<- 1

<- OK
Adding the Heart Rate Measurement characteristic (UUID = 0x2A37):
AT+GATTADDCHAR=UUID=0x2A37, PROPERTIES=0x10, MIN_LEN=2, MAX_LEN=3, VALUE=00-40

<- 1

<- OK
Adding the Body Sensor Location characteristic (UUID = 0x2A38):
AT+GATTADDCHAR=UUID=0x2A38, PROPERTIES=0x02, MIN_LEN=1, VALUE=3

<- 2

<- OK
Adding Heart Rate Service UUID to the advertising payload: AT+GAPSETADVDATA=02-01-06-05-02-0d-18-0a-18

<- OK
Performing a SW reset (service changes require a reset): ATZ

<- OK

Updating HRM value to 82 BPM
AT+GATTCHAR=1,00-52

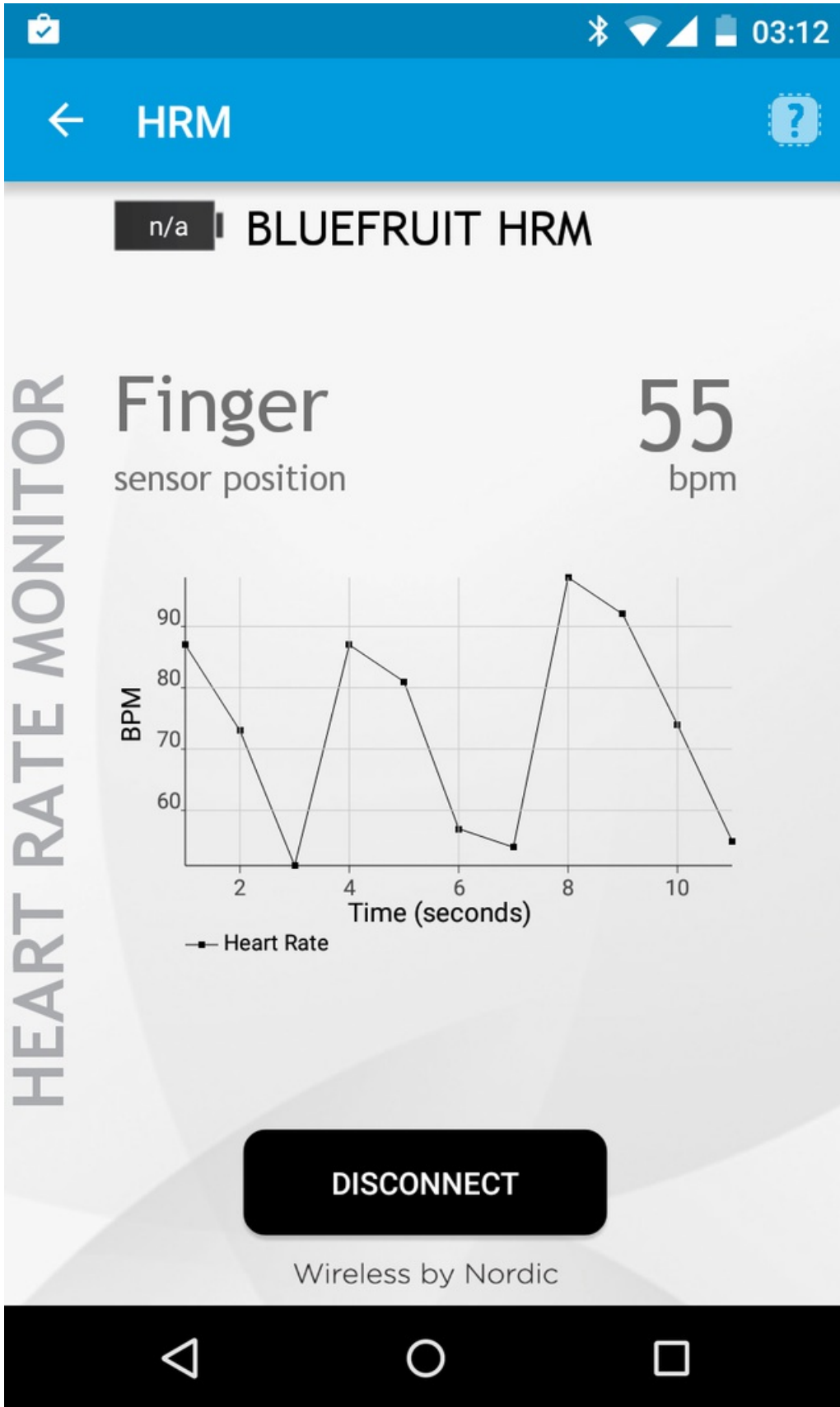
<- OK
Updating HRM value to 61 BPM
AT+GATTCHAR=1,00-3D

Autoscroll
No line ending
9600 baud
```

If you open up an application on your mobile device or laptop that support the standard [Heart Rate Monitor Service \(https://adafru.it/f4I\)](https://adafru.it/f4I), you should be able to see the heart rate being updated in sync with the changes seen in the Serial Monitor:

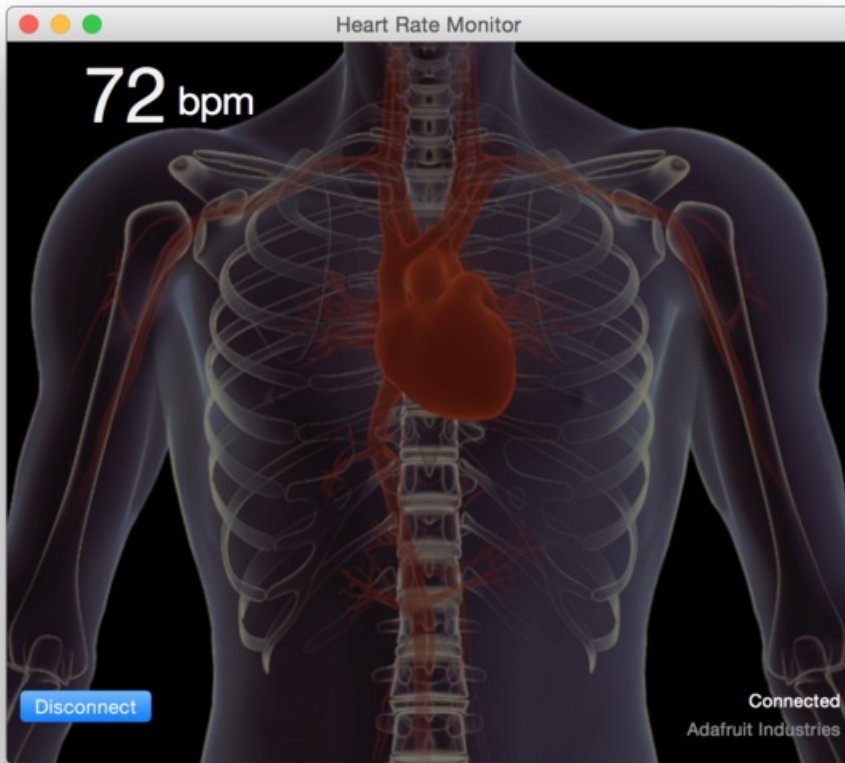
nRF Toolbox HRM Example

The image below is a screenshot from the free [nRF Toolbox \(https://adafru.it/e9M\)](https://adafru.it/e9M) application from Nordic on Android (also available on [iOS \(https://adafru.it/f4J\)](https://adafru.it/f4J)), showing the incoming Heart Rate Monitor data:



CoreBluetooth HRM Example

The image below is from a freely available [CoreBluetooth sample application](https://adafru.it/f4K) (<https://adafru.it/f4K>) from Apple showing how to work with Bluetooth Low Energy services and characteristics:



HALP!

When using the Bluefruit Micro or a Bluefruit LE with Flora/Due/Leonardo/Micro the examples dont run?

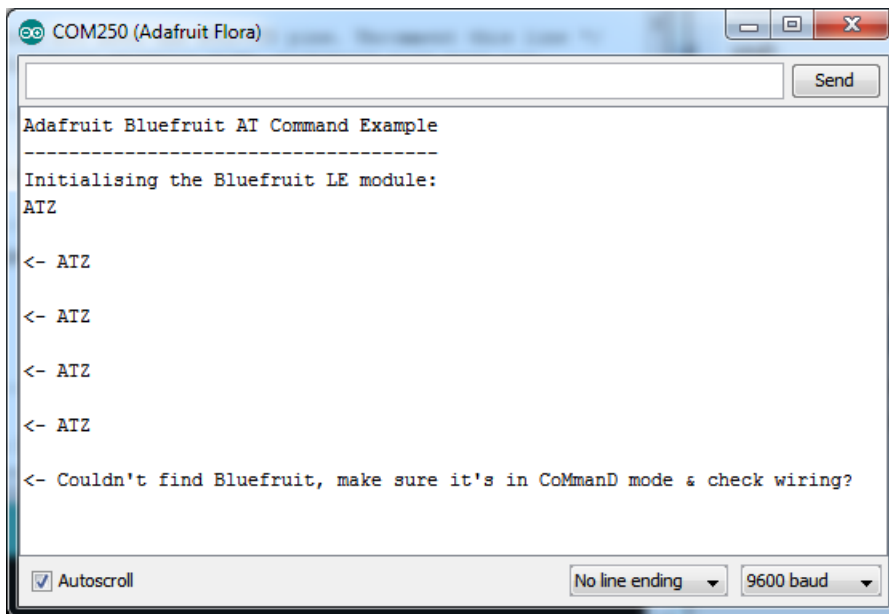
We add a special line to `setup()` to make it so the Arduino will halt until it sees you've connected over the Serial console. This makes debugging great but makes it so you cannot run the program disconnected from a computer.

Solution? Once you are done debugging, remove these two lines from `setup()`

```
while (!Serial);  
delay(500);
```

I can't seem to "Find" the Bluefruit LE!

Getting something like this?



For UART/Serial Bluefruits:

- Check you have the **MODE** switch in CMD and the MODE pin not wired to anything if it isnt used!
- If you are trying to control the **MODE** from your micro, make sure you set the MODE pin in the sketch
- Make sure you have **RXI** and **TXO** wired right! They are often swapped by accident
- Make sure **CTS** is tied to GND if you are using hardware serial and not using CTS
- Check the MODE red LED, is it blinking? If its blinking continuously, you might be in DFU mode, power cycle the module!
- If you are using Hardware Serial/Software Serial make sure you know which one and have that set up

If using SPI Bluefruit:

- Make sure you have all 5 (or 6) wires connected properly.
- If using hardware SPI, you need to make sure you're connected to the hardware SPI port, which differs depending on the main chipset.

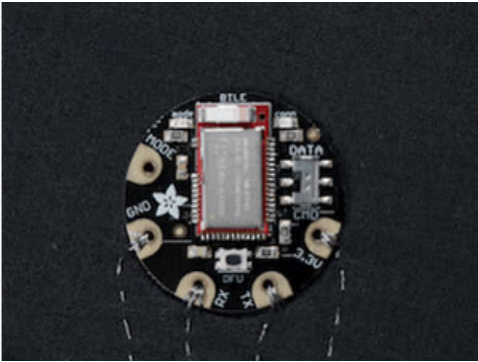
If using Bluefruit Micro:

- Make sure you change the **RESET** pin to #4 in any Config file. Also be sure you are using hardware SPI to connect!

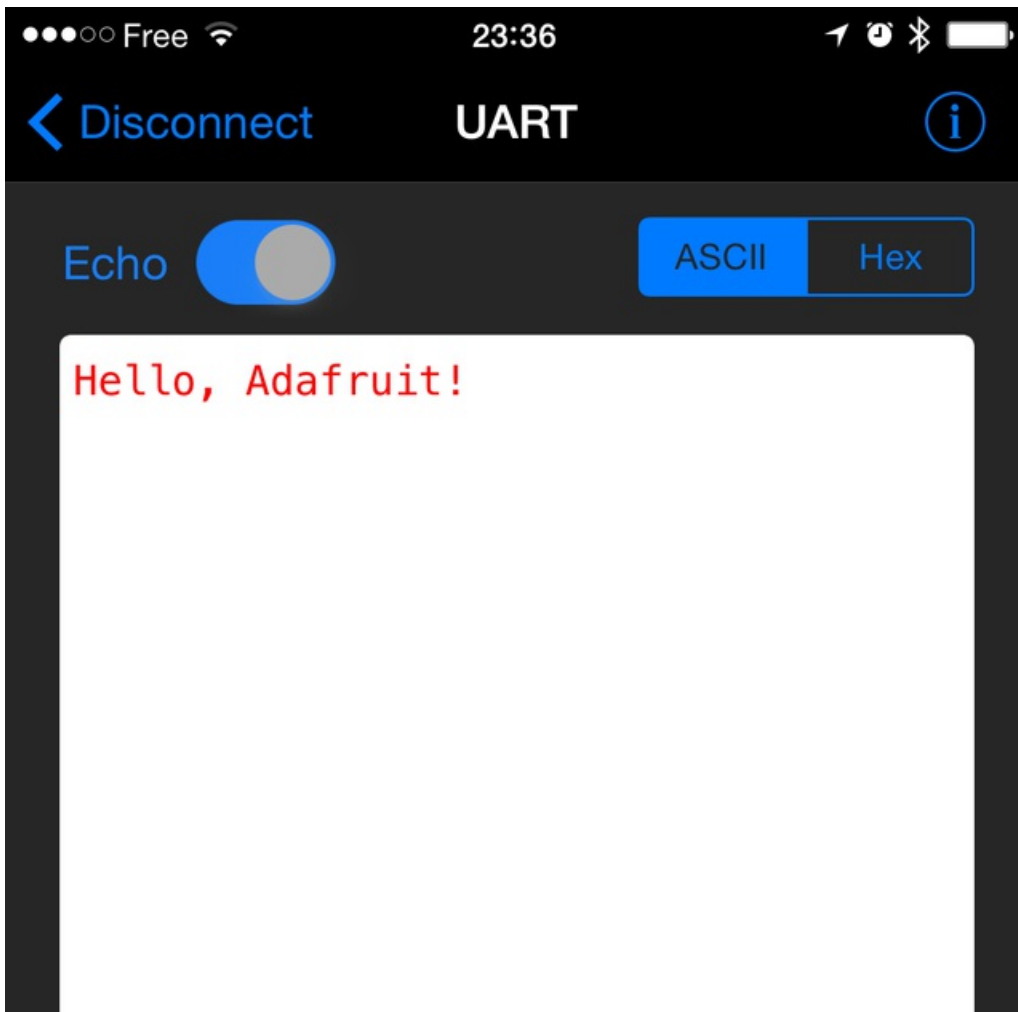
Data Mode

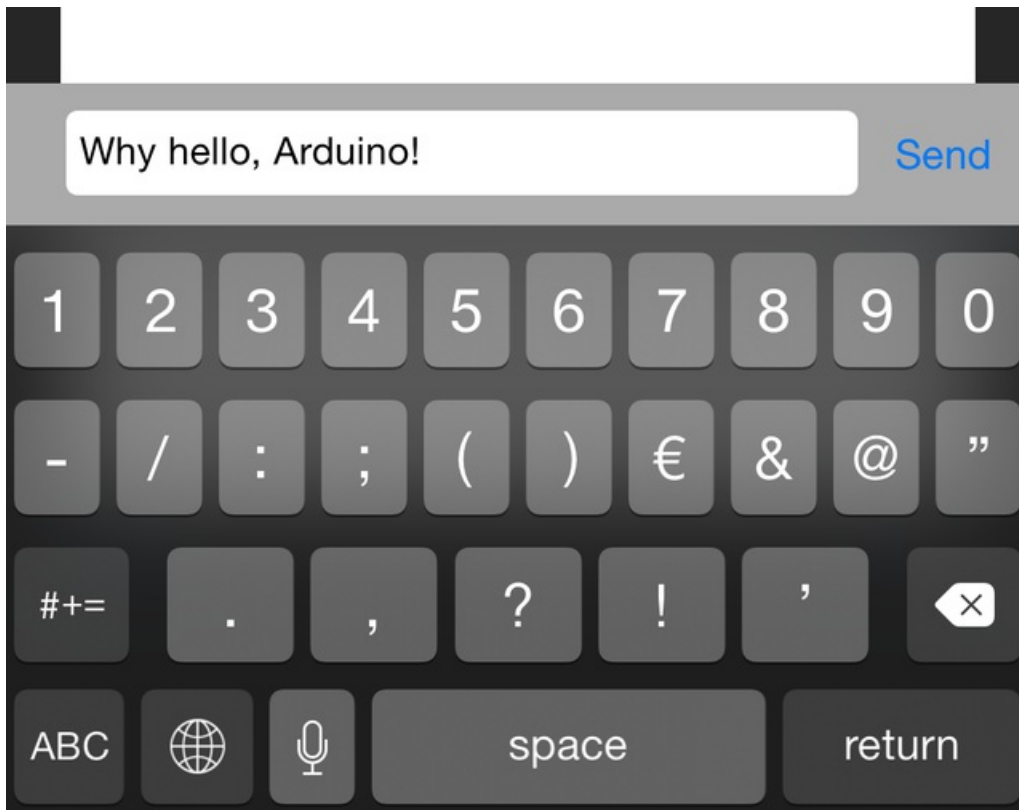
By placing the Flora BLEFriend module in 'UART Data' mode (set the mode selection switch to **DATA** or setting the MODE pin to ground) you can use the module as a 'transparent UART connection' to the Bluefruit app. This makes data transfer super simple. Data is sent to the app when any 9600 baud data is received on the **RX** pin and any data from the app is automatically transmitted via the **TX** pin to your Flora

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



You can then connect the the app in **UART** mode and send/receive data transparently





Switching Command/Data Mode via +++

On either side of the connection (via the Flora or in your mobile app), you can dynamically switch between command and data mode by sending the "+++\n" string, as detailed in the [+++ command summary \(https://adafru.it/iCN\)](https://adafru.it/iCN).

If you start in data mode, you can send text for example, with "+++\nATI\n+++\n", which will cause the Bluefruit LE module to switch to command mode, execute the ATI command, and then switch back to data mode.

The +++ command can be sent from either side, making it possible to execute commands from the mobile application as well as on the Bluefruit LE side.

```
# Start in Data Mode
> Hello, World! Data mode!

# Send command to switch modes
> +++

# Bluefruit LE module switches to CMD mode
# Send ATI command and wait for the response
> ATI
< BLEFRIEND
< nRF51822 QFAAG00
< B122AAC33F3D2296
< 0.6.2
< 0.6.2
< May 01 2015
< OK

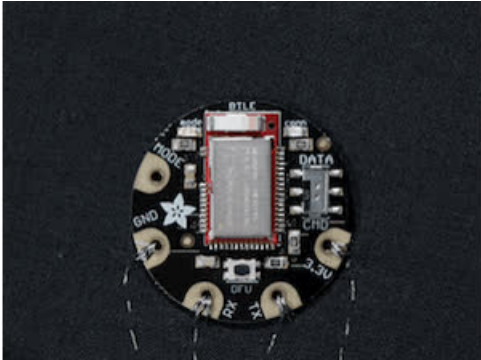
# Switch back to DATA mode
> +++
< OK

# We're back in data mode now
Welcome back!
```

Command Mode

By placing the Bluefruit LE module in 'Command' mode (set the mode selection switch to **CMD** or setting the MODE pin to a high voltage) you can enter a variety of Hayes AT style commands to configure the device or retrieve basic information about the module or 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 \(https://adafruit.it/ebJ\)](https://adafruit.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.

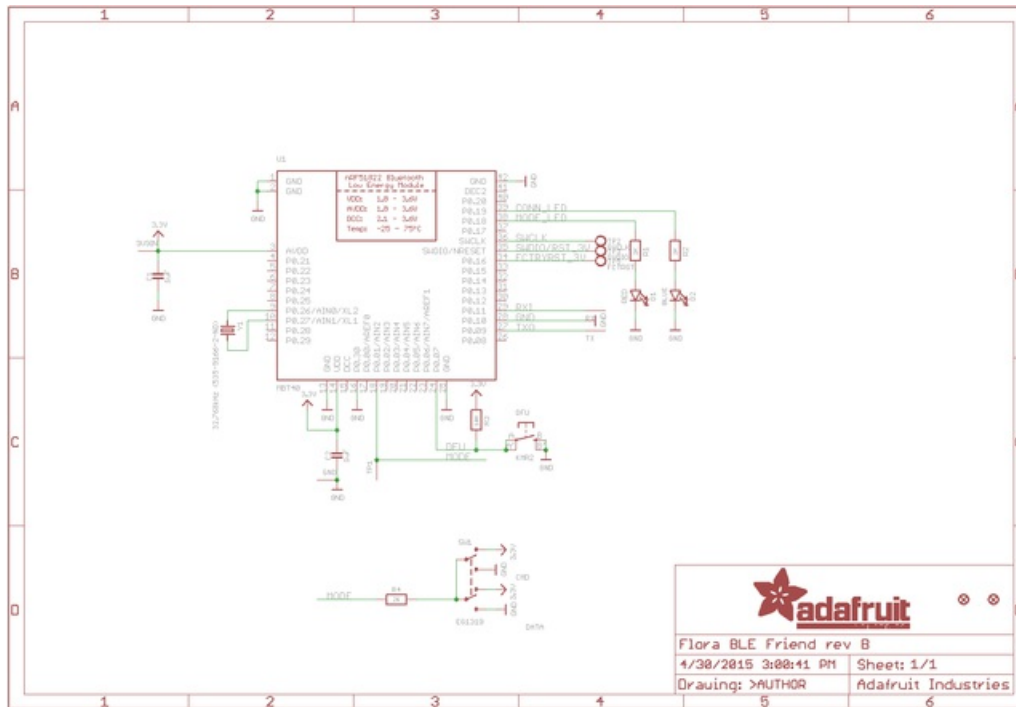
[We have a huge amount of commands and details on how to use them, so please do check out our list of possible commands over on the Bluefruit LE page \(https://adafruit.it/iCG\)](https://adafruit.it/iCG) (no need to duplicate all of the content here!)

Downloads

- [MDBT Datasheet \(https://adafru.it/oYE\)](https://adafru.it/oYE)
- [GitHub PCB files \(https://adafru.it/rDu\)](https://adafru.it/rDu)
- [Fritzing object in the Adafruit Fritzing Library \(https://adafru.it/aP3\)](https://adafru.it/aP3)

Schematic

Click to embiggen



Fabrication print

Dimensions in inches

