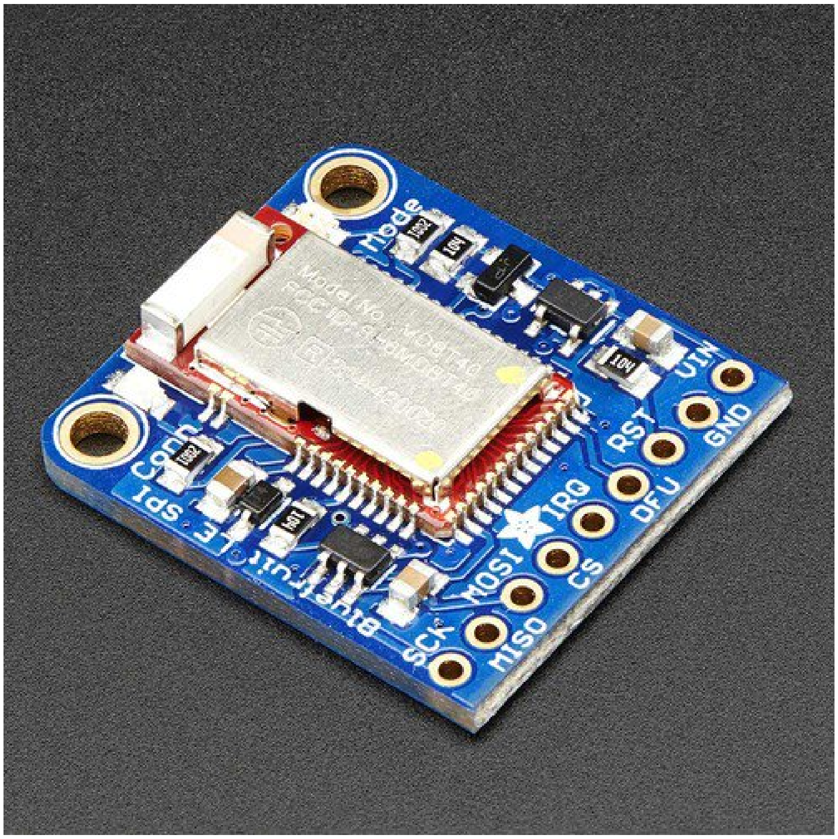
Chipsmall
深圳市津利帝科技有限公司
Geelead Technology Limited
Chipsmall
采购部

Introducing the Adafruit Bluefruit LE SPI Friend

Created by Kevin Townsend



Last updated on 2017-01-26 06:45:47 PM UTC

Guide Contents

Guide Contents 2

Introduction 9

Why Use Adafruit's Module? 10

Technical Specifications 10

Pinouts 11

Power Pins 11

SPI Pins 11

Other Pins 11

Reverse Side Breakouts 12

Assembly 13

- Prepare the header strip: 13
- Add the breakout board: 13
- And Solder! 14

Wiring 15

Default Pinout 15

Changing the Default Pinout 15

Software 17

Configuration! 18

Which board do you have? 18

- Bluefruit Micro or Feather 32u4 Bluefruit 18
- Feather M0 Bluefruit LE 18
- Bluefruit LE SPI Friend 19
- Bluefruit LE UART Friend or Flora BLE 19

Configure the Pins Used 19

- Common settings: 20
- Software UART 20
- Hardware UART 20
- Mode Pin 20
- SPI Pins 20
- Software SPI Pins 20

Select the Serial Bus 20

- UART Based Boards (Bluefruit LE UART Friend & Flora BLE) 20
- SPI Based Boards (Bluefruit LE SPI Friend) 21

ATCommand 22
Opening the Sketch 22
Configuration 22
Running the Sketch 23
BLEUart 25
Opening the Sketch 25
Configuration 25
Running the Sketch 26
HIDKeyboard 30
Opening the Sketch 30
Configuration 30
Running the Sketch 31
Bonding the HID Keyboard 31
Android 32
iOS 33
OS X 34
Controller 36
Opening the Sketch 36
Configuration 36
Running the Sketch 37
Using Bluefruit LE Connect in Controller Mode 37
Streaming Sensor Data 38
Control Pad Module 39
Color Picker Module 40
HeartRateMonitor 42
Opening the Sketch 42
Configuration 42
If Using Hardware or Software UART 43
Running the Sketch 43
nRF Toolbox HRM Example 44
CoreBluetooth HRM Example 45
UriBeacon 47

Opening the Sketch 47
Configuration 47
Running the Sketch 48
HALP! 49
AT Commands 50
Test Command Mode '=?' 50
Write Command Mode '=xxx' 50
Execute Mode 50
Read Command Mode '?' 51
Standard AT 52
AT 52
ATI 52
ATZ 52
ATE 52
+++ 53
General Purpose 54
AT+FACTORYRESET 54
AT+DFU 54
AT+HELP 54
AT+NVMWRITE 54
AT+NVMREAD 55
AT+MODESWITCHEN 55
Hardware 56
AT+BAUDRATE 56
AT+HWADC 56
AT+HWGETDIETEMP 56
AT+HWGPIO 57
AT+HWGPIOMODE 57
AT+HWI2CSCAN 58
AT+HWVBAT 58
AT+HWRANDOM 58
AT+HWMODELED 58

AT+UARTFLOW 59
BLE Generic 60
AT+BLEPOWERLEVEL 60
AT+BLEGETADDRTYPE 60
AT+BLEGETADDR 60
AT+BLEGETPEERADDR 61
AT+BLEGETRSSI 61
BLE Services 62
AT+BLEUARTTX 62
TX FIFO Buffer Handling 62
AT+BLEUARTTXF 63
AT+BLEUARTRX 63
AT+BLEUARTFIFO 64
AT+BLEKEYBOARDEN 64
AT+BLEKEYBOARD 64
AT+BLEKEYBOARDCODE 65
Modifier Values 65
AT+BLEHIDEN 65
AT+BLEHIDMOUSEMOVE 66
AT+BLEHIDMOUSEBUTTON 66
AT+BLEHIDCONTROLKEY 67
AT+BLEHIDGAMEPADEN 67
AT+BLEHIDGAMEPAD 68
AT+BLEMIDIEN 68
AT+BLEMIDIRX 68
AT+BLEMIDITX 69
AT+BLEBATTEN 69
AT+BLEBATTVAL 69
BLE GAP 70
AT+GAPCONNECTABLE 70
AT+GAPGETCONN 70
AT+GAPDISCONNECT 70
AT+GAPDEVNAME 70

AT+GAPDELBONDS 71
AT+GAPINTERVALS 71
AT+GAPSTARTADV 72
AT+GAPSTOPADV 72
AT+GAPSETADVDATA 72
BLE GATT 74
GATT Limitations 74
AT+GATTCLEAR 74
AT+GATTADDSERVICE 74
AT+GATTADDCHAR 75
AT+GATTCHAR 76
AT+GATTLIST 77
AT+GATTCHARRAW 78
Debug 79
AT+DBGMEMRD 79
AT+DBGNVMRD 79
AT+DBGSTACKSIZE 79
AT+DBGSTACKDUMP 79
History 82
Version 0.7.7 82
Version 0.7.0 82
Version 0.6.7 83
Version 0.6.6 83
Version 0.6.5 84
Version 0.6.2 84
Version 0.5.0 84
Version 0.4.7 85
Version 0.3.0 85
UART Service 86
Characteristics 86
- TX (0x0002) 86
- RX (0x0003) 86
Software Resources 87

Bluefruit LE Client Apps and Libraries 87
Bluefruit LE Connect (http://adafru.it/f4G) (Android/Java) 87
Bluefruit LE Connect (http://adafru.it/f4H) (iOS/Swift) 87
Bluefruit LE Connect for OS X (http://adafru.it/o9F) (Swift) 87
Bluefruit LE Command Line Updater for OS X (http://adafru.it/pLF) (Swift) 88
Deprecated: Bluefruit Buddy (http://adafru.it/mCn) (OS X) 88
ABLE (http://adafru.it/ijB) (Cross Platform/Node+Electron) 89
Bluefruit LE Python Wrapper (http://adafru.it/fQF) 89
Debug Tools 90
AdaLink (http://adafru.it/fPq) (Python) 90
Adafruit nRF51822 Flasher (http://adafru.it/fVL) (Python) 90
Factory Reset 91
Factory Reset via DFU Pin 91
FactoryReset Sample Sketch 91
AT+FACTORYRESET 92
Factory Reset via FCTR Test Pad 92
DFU Updates 94
Adafruit Bluefruit LE Connect 94
SDEP (SPI Data Transport) 95
SDEP Overview 95
SPI Setup 95
SPI Hardware Requirements 95
IRQ Pin 95
SDEP Packet and SPI Error Identifier 95
Sample Transaction 95
SDEP (Simple Data Exchange Protocol) 96
Endianness 96
Message Type Indicator 96
SDEP Data Transactions 96
Message Types 96
Command Messages 96
Response Messages 97
Alert Messages 98
Standard Alert IDs 98
Error Messages 98
Standard Error IDs 99

Existing Commands 99

SDEP AT Wrapper Usage 99

BLE FAQ 101

Bluefruit LE Connect (Android) 102

Nordic nRF Toolbox 102

Adafruit_nRF51822_Flasher 102

Device Recovery 107

How to Recover a Bluefruit Board 107

1. Force DFU Mode at Startup 107

2. Update the Bluefruit Firmware 107

BLEFRIEND32 Firmware (UART, 32KB SRAM) 107

BLESPIFRIEND Firmware (SPI) 107

3. Flash a Test Sketch 107

4. Perform a Factory Reset 107

Still Having Problems? 108

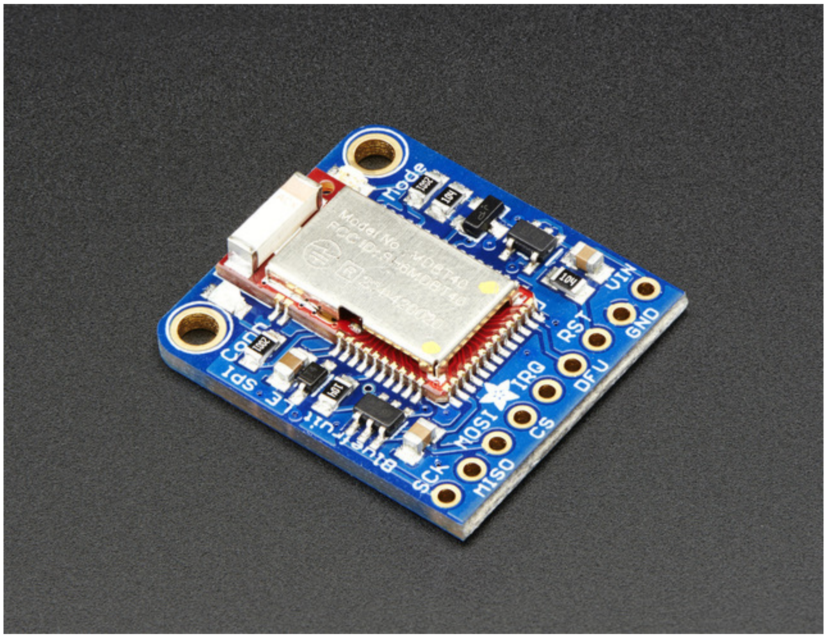
Downloads 109

Files 109

Schematic 109

Board Layout 109

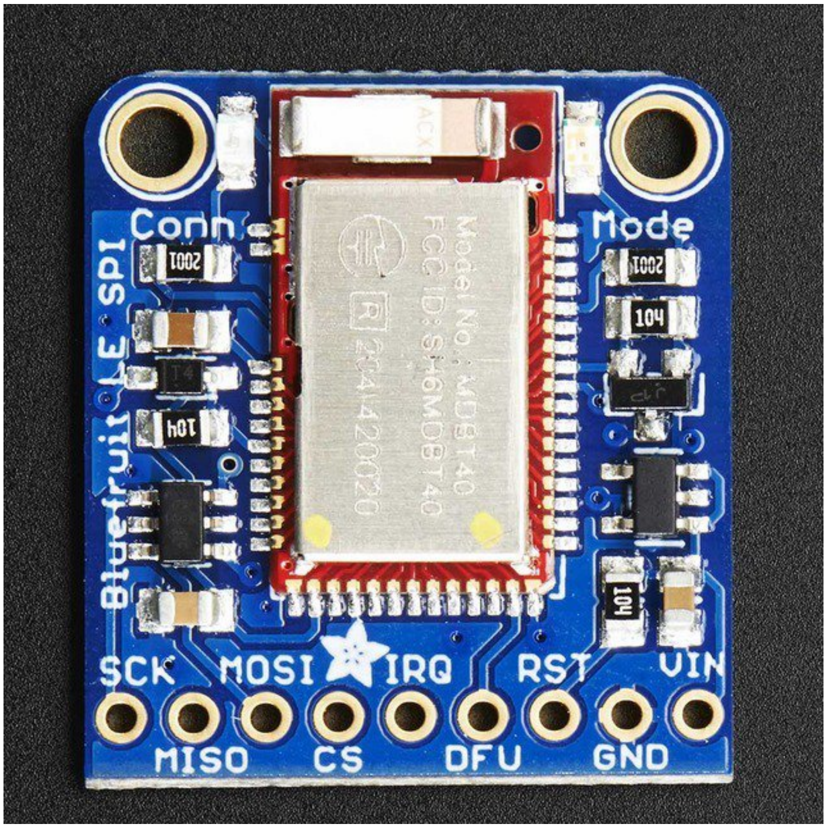
Introduction



Would you like to add powerful and easy-to-use Bluetooth Low Energy to your robot, art or other electronics 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 Bluefruit LE SPI Friend!

The Bluefruit LE SPI Friend (http://adafru.it/2633) makes it easy to add Bluetooth Low Energy connectivity to anything that supports SPI communication. Connect to your Arduino or other microcontroller using the common four-pin SPI interface (MISO, MOSI, SCK and CS/SSEL) plus a 5th GPIO pin for interrupts (to let the Arduino know when data or a response is ready).

If you like Serial communication more than SPI, we also have a version that can talk UART(http://adafru.it/iCh)



This multi-function module can do quite a lot! For most people, they'll be very happy to use the standard Nordic UART RX/TX 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 (http://adafru.it/iCi) or Android App (http://adafru.it/f4G) to get started sending data from your Arduino to your phone quickly and painlessly.

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.

One of the biggest advantages of the Adafruit Bluefruit LE family is that**we wrote all of the firmware running on the devices ourselves from scratch.**

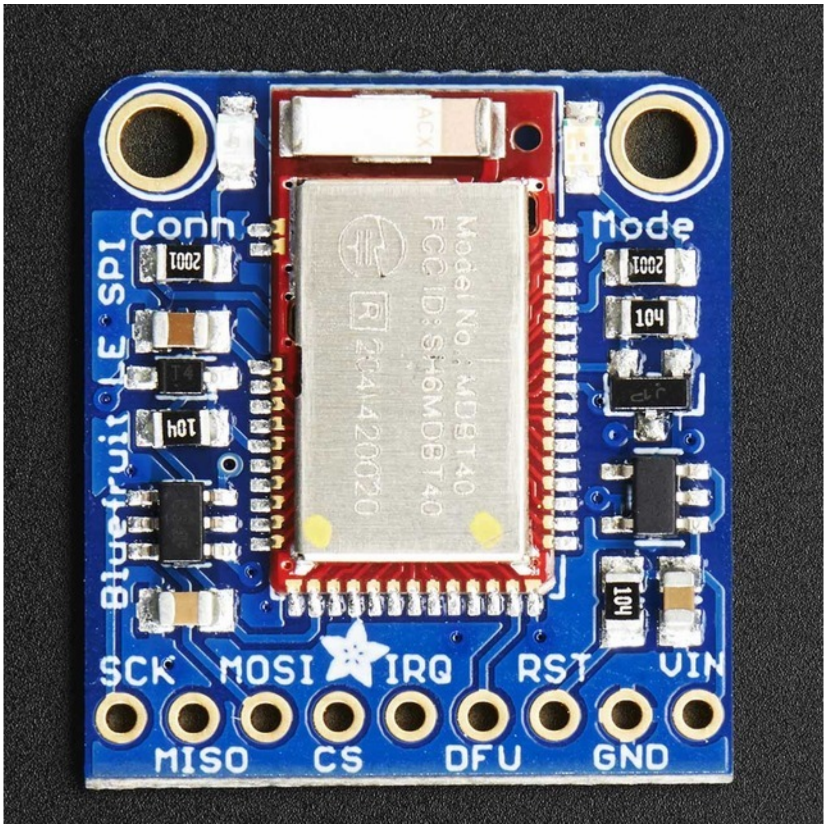
We control every line of code that runs on our modules ... and so we aren't at the mercy of any third party vendors who may or may not be interested in keeping their code up to date or catering to our customer's needs.

Because we control everything about the product, we add features that are important to*our* customers, can solve any issues that do come up without begging any 3rd parties, and we can even change Bluetooth SoCs entirely if the need ever arises!

Technical Specifications

- ARM Cortex M0 core running at 16MHz (nRF51822)
- 256KB flash memory
- 32KB SRAM
- Transport: SPI at 4MHz with HW IRQ (5 pins required)
- 5V-safe inputs (Arduino Uno friendly, etc.)
- On-board 3.3V voltage regulation
- Bootloader with support for safe OTA firmware updates
- Easy AT command set to get up and running quickly

Pinouts



Power Pins

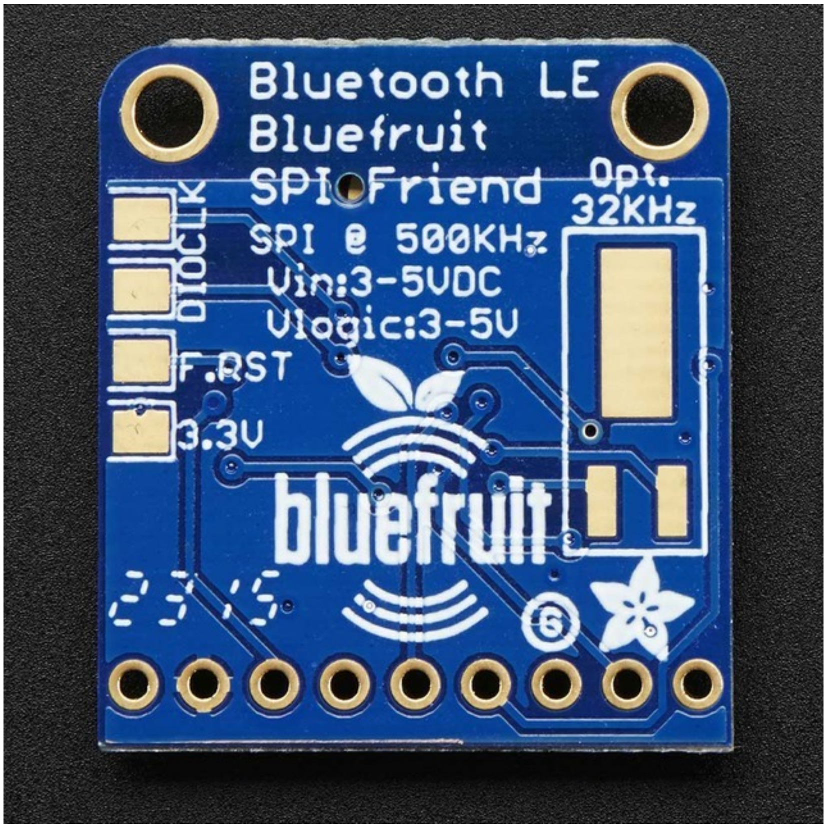
- **VIN**: This is the power supply for the module, supply with 3.3-16V power supply input. This will be regulated down to 3.3V to run the chip
- **GND**: The common/GND pin for power and logic

SPI Pins

- **SCK**: This is the serial clock pin, connected to SCK on your Arduino or MCU
- **MISO**: This is the Master In Slave Out SPI pin (nRF51 -> Arduino communication)
- **MOSI**: This is the Master Out Slave In SPI pin (Arduino -> nRF51 communication)
- **CS**: This is the Chip Select SPI pin, which is used to indicate that the SPI device is currently in use.
- **IRQ**: This is the nRF51 -> Arduino 'interrupt' pin that lets the Arduino or MCU know when data is available on the nRF51, indicating that a new SPI transaction should be initiated by the Arduino/MCU.

Other Pins

- **DFU**: Setting this pin low when you power the device up will force the Bluefruit LE module to enter a special firmware update mode to update the firmware over the air. Once the device is powered up, this pin can *also* be used to perform a factory reset. Wire the pin to GND for >5s until the two LEDs start to blink, then release the pin (set it to 5V or logic high) and a factory reset will be performed.
- **RST**: Holding this pin low then releasing it (or wiring it HIGH) will cause a system reset



Reverse Side Breakouts

On the back we also have a few breakouts!

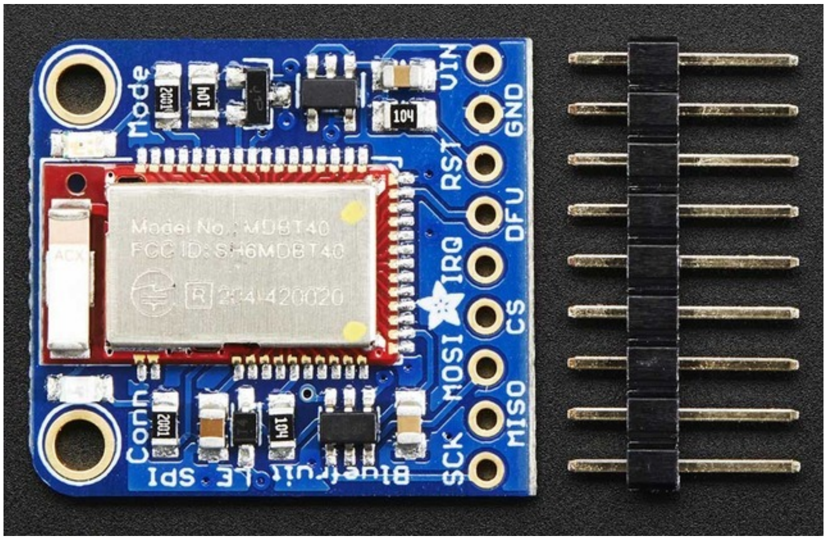
CLK: This is the SWD clock pin (SWCLK), 3v logic - for advanced hackers!

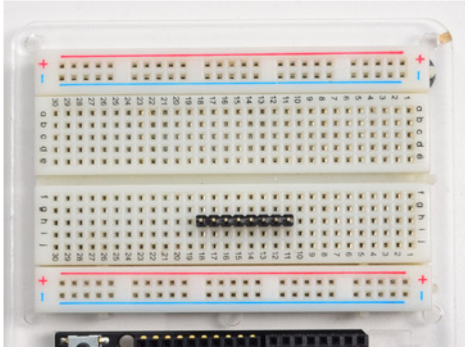
DIO: This is the SWD data pin (SWDIO), 3v logic - for advanced hackers!

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 though (see that tutorial page).

3.3V: This is the output from the 3V regulator, for testing and also if you really need regulated 3V, up to 250mA available

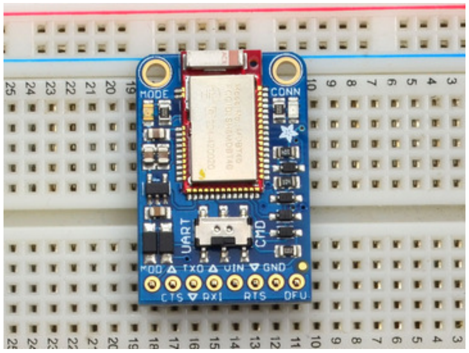
Assembly



- 

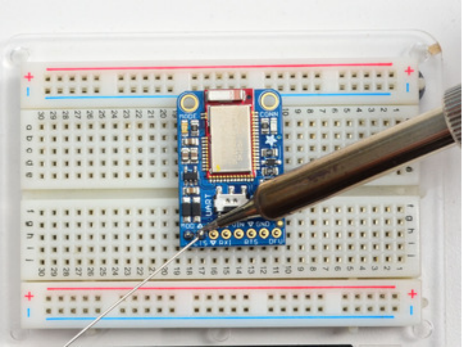
  ### Prepare the header strip:

  Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**

- 

  ### Add the breakout board:

  Place the breakout board over the pins so that the short pins poke through the breakout pads

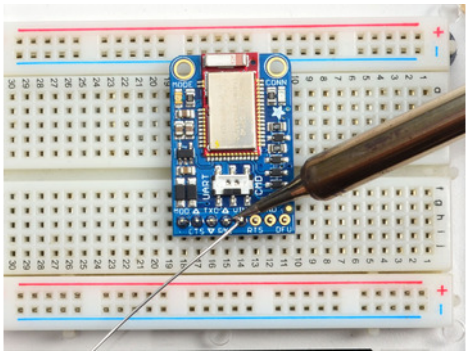
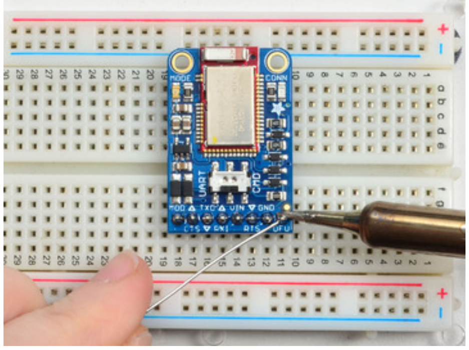
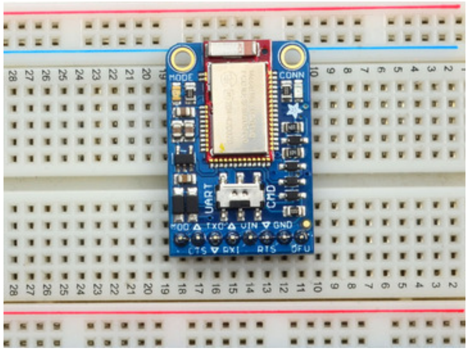
- 

  ## And Solder!

  Be sure to solder all pins for reliable electrical contact.

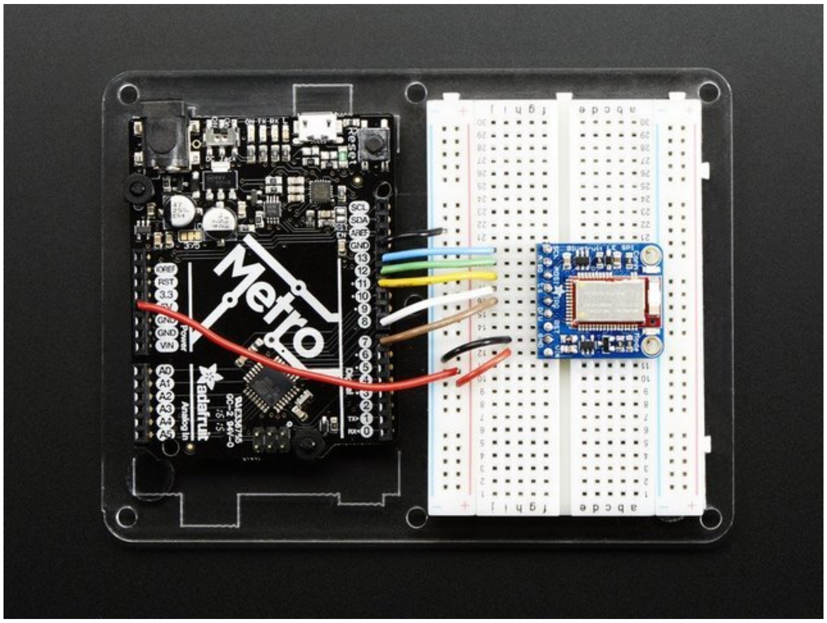
  Solder the longer power/data strip first

  (For tips on soldering, be sure to check out our[Guide to Excellent Soldering](http://adafru.it/aTk) (http://adafru.it/aTk)).

- 
- 
- 

  You're done! Check your solder joints visually and continue onto the next steps

Wiring



Default Pinout

In order to follow along with the default tutorial wiring, the Bluefruit LE SPI Friend should be connected to your Uno or Metro board using the following pins:

Bluefruit LE SPI Friend	Metro/Uno
SCK	13
MISO	12
MOSI	11
CS	8
IRQ	7
RST (Optional)	4
VIN	5V
GND	GND

We'll be using **hardware serial** by default, that uses the UNO's hardware pins #13, #12 and #11. You can also use software SPI so you don't have to locate the hardware SPI pins!

In order for the Bluefruit LE SPI Friend to be 5V-Safe, the VIN pin must be connected to 5V on the Arduino. If you wish to run the board with 3.3V logic, you can optionally connect VIN to 3.3V, but this should not be done on a 5V Arduino.

Changing the Default Pinout

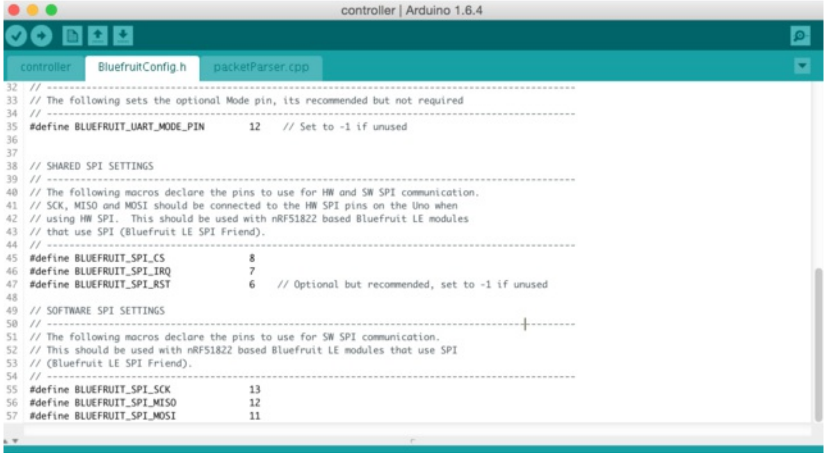
The examples sketches all assume the above pinouts by default. If you wish to change the location of the CS, IRQ or RST pins, open the **BluefruitConfig.h** file in the example folder, and change the pin to an appropriate value(See the Software section of this tutorial (http://adafru.it/iCj) for instructions on installing the library):

```
#define BLUEFRUIT_SPI_CS          8
#define BLUEFRUIT_SPI_IRQ         7
#define BLUEFRUIT_SPI_RST          4
```

If you want to use **software** (bitbang) SPI, you can change the SCK, MISO and MOSI pins using the following macros in the same file:

```
#define BLUEFRUIT_SPI_SCK             13
#define BLUEFRUIT_SPI_MISO            12
#define BLUEFRUIT_SPI_MOSI            11
```

The **BluefruitConfig.h** file can be found in a dedicated tab, as shown below:



```
// ----------------------------------------------------------------------------------------------
// The following sets the optional Mode pin, its recommended but not required
// ----------------------------------------------------------------------------------------------
#define BLUEFRUIT_UART_MODE_PIN        12    // Set to -1 if unused


// SHARED SPI SETTINGS
// ----------------------------------------------------------------------------------------------
// The following macros declare the pins to use for HW and SW SPI communication.
// SCK, MISO and MOSI should be connected to the HW SPI pins on the Uno when
// using HW SPI.  This should be used with nRF51822 based Bluefruit LE modules
// that use SPI (Bluefruit LE SPI Friend).
// ----------------------------------------------------------------------------------------------
#define BLUEFRUIT_SPI_CS               8
#define BLUEFRUIT_SPI_IRQ              7
#define BLUEFRUIT_SPI_RST              6    // Optional but recommended, set to -1 if unused

// SOFTWARE SPI SETTINGS
// ----------------------------------------------------------------------------------------------
// The following macros declare the pins to use for SW SPI communication.
// This should be used with nRF51822 based Bluefruit LE modules that use SPI
// (Bluefruit LE SPI Friend).
// ----------------------------------------------------------------------------------------------
#define BLUEFRUIT_SPI_SCK              13
#define BLUEFRUIT_SPI_MISO             12
#define BLUEFRUIT_SPI_MOSI             11
```

For all the example code, we have at the top of the sketch a few different ways you can communicate with the Bluefruit LE: hardware serial, software serial, hardware SPI and software SPI.

For the SPI Bluefruit, you cannot use serial. However, you can choose between hardware and software SPI.

If you want to use hardware SPI, uncomment this chunk of code (and comment out the other three options)

```
/* ...hardware SPI, using SCK/MOSI/MISO hardware SPI pins and then user selected CS/IRQ/RST */
Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
```

If you want to use software/bitbang SPI, uncomment the following definition. You can then use any 6 pins (or 5, if you dont want to use RST)

```
/* ...software SPI, using SCK/MOSI/MISO user-defined SPI pins and then user selected CS/IRQ/RST */
Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_SCK, BLUEFRUIT_SPI_MISO,
                             BLUEFRUIT_SPI_MOSI, BLUEFRUIT_SPI_CS,
                             BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
```

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,(http://adafru.it/f4V) but its likely easier to just download by clicking:

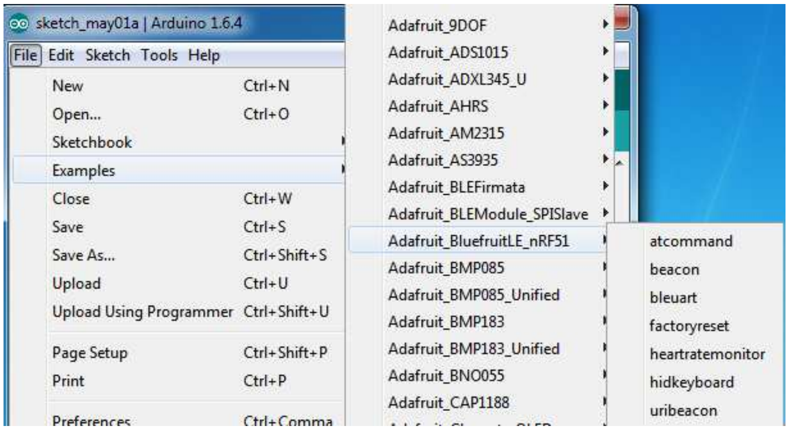
Download Adafruit_BluefruitLE_nRF51
http://adafru.it/f4W

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 ***arduinosketchfolder*/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 (http://adafru.it/aYM)

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

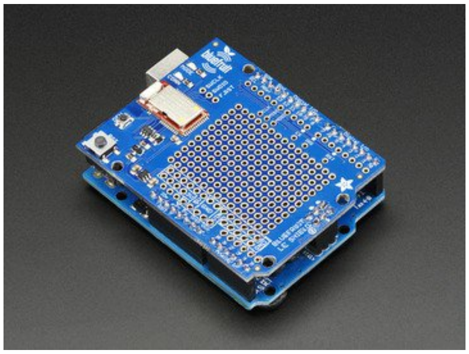


Configuration!

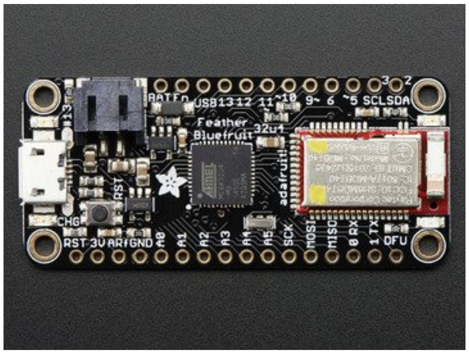
Before you start uploading any of the example sketches, you'll need to CONFIGURE the Bluefruit interface - there's a lot of options so pay close attention!

Which board do you have?

There's a few products under the Bluefruit name:

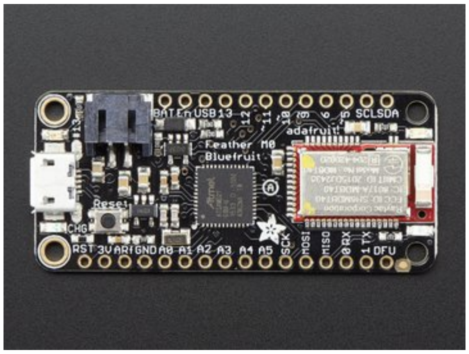
- 

  If you are using the Bluefruit LE Shield then you have an**SPI-connected NRF51822** module. You can use this with**Atmega328** (Arduino UNO or compatible)**, ATmega32u4** (Arduino Leonardo, compatible) or **ATSAMD21** (Arduino Zero, compatible) and possibly others.
  Your pinouts are **Hardware SPI, CS = 8, IRQ = 7, RST = 4**

- 
  Feather
  Bluefruit
  32u4
  BAT En USB 13 12 11 10 9 6 5 SCL SDA
  RST 3V AR GND A0 A1 A2 A3 A4 A5 SCK MOSI MISO 0RX 1TX DFU
  CHG
  adafruit

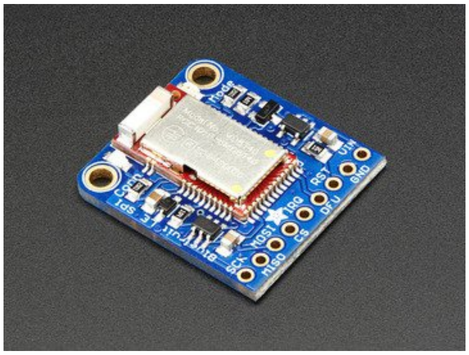
  ## Bluefruit Micro or Feather 32u4 Bluefruit

  If you have a Bluefruit Micro or Feather 32u4 Bluefruit LE then you have an **ATmega32u4** chip with **Hardware SPI**, CS = **8**, IRQ = **7**, RST = **4**

- 
  Feather M0
  Bluefruit
  adafruit
  BAT En USB 13 12 11 10 9 6 5 SCL SDA
  RST 3V AR GND A0 A1 A2 A3 A4 A5 SCK MOSI MISO 0RX 1TX DFU
  Reset
  CHG

  ## Feather M0 Bluefruit LE

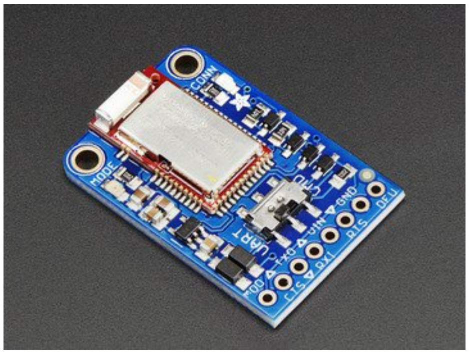
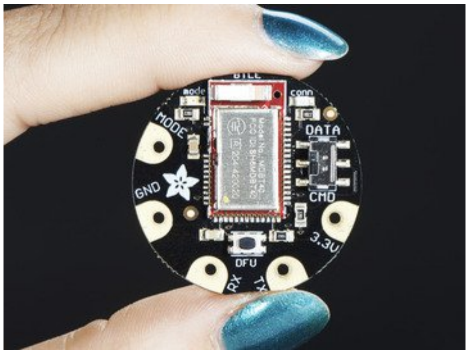
  If you have a Feather M0 Bluefruit LE then you have an**ATSAMD21** chip with **Hardware SPI**, CS = **8**, IRQ = **7**, RST = **4**

- 

Bluefruit LE SPI Friend

If you have a stand-alone module, you have a bit of flexibility with wiring however we strongly recommend **Hardware SPI**, CS = **8**, IRQ = **7**, RST = **4**

You can use this with just about any microcontroller with 5 or 6 pins

- 
- 

Bluefruit LE UART Friend or Flora BLE

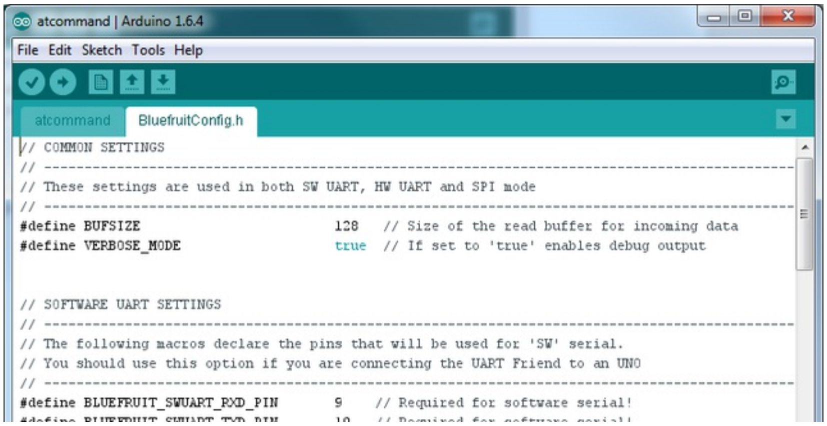
If you have a stand-alone UART module you have some flexibility with wiring. However we suggest **hardware UART** if possible. You will likely need to use the flow control **CTS** pin if you are not using hardware UART. There's also a **MODE** pin

You can use this with just about any microcontroller with at least 3 pins, but best used with a Hardware Serial/UART capable chip!

Configure the Pins Used

You'll want to check the Bluefruit Config to set up the pins you'll be using for UART or SPI

Each example sketch has a secondary tab with configuration details. You'll want to edit and save the sketch to your own documents folder once set up.



```
// COMMON SETTINGS
// ----------------------------------------------------------------------------------------------
// These settings are used in both SW UART, HW UART and SPI mode
// ----------------------------------------------------------------------------------------------
#define BUFSIZE                        128   // Size of the read buffer for incoming data
#define VERBOSE_MODE                   true  // If set to 'true' enables debug output


// SOFTWARE UART SETTINGS
// ----------------------------------------------------------------------------------------------
// The following macros declare the pins that will be used for 'SW' serial.
// You should use this option 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!
```

Common settings:

You can set up how much RAM to set aside for a communication buffer and whether you want to have full debug output. Debug output is 'noisy' on the serial console but is handy since you can see all communication between the micro and the BLE

```
// ----------------------------------------------------------------------------------------------
// These settings are used in both SW UART, HW UART and SPI mode
// ----------------------------------------------------------------------------------------------
#define BUFSIZE                        128   // Size of the read buffer for incoming data
#define VERBOSE_MODE                   true  // If set to 'true' enables debug output
```

Software UART

If you are using Software UART, you can set up which pins are going to be used for RX, TX, and CTS flow control. Some microcontrollers are limited on which pins can be used! Check the SoftwareSerial library documentation for more details

```
// SOFTWARE UART SETTINGS
#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
```

Hardware UART

If you have Hardware Serial, there's a 'name' for it, usually Serial1 - you can set that up here:

```
// HARDWARE UART SETTINGS
#ifdef Serial1    // this makes it not complain on compilation if there's no Serial1
  #define BLUEFRUIT_HWSERIAL_NAME      Serial1
#endif
```

Mode Pin

For both hardware and software serial, you will likely want to define the MODE pin. There's a few sketches that dont use it, instead depending on commands to set/unset the mode. Its best to use the MODE pin if you have a GPIO to spare!

```
#define BLUEFRUIT_UART_MODE_PIN        12    // Set to -1 if unused
```

SPI Pins

For both Hardware and Software SPI, you'll want to set the**CS** (chip select) line, **IRQ** (interrupt request) line and if you have a pin to spare,**RST** (Reset)

```
// SHARED SPI SETTINGS
#define BLUEFRUIT_SPI_CS               8
#define BLUEFRUIT_SPI_IRQ              7
#define BLUEFRUIT_SPI_RST              4    // Optional but recommended, set to -1 if unused
```

Software SPI Pins

If you don't have a hardware SPI port available, you can use any three pins...its a tad slower but very flexible

```
// SOFTWARE SPI SETTINGS
#define BLUEFRUIT_SPI_SCK              13
#define BLUEFRUIT_SPI_MISO             12
#define BLUEFRUIT_SPI_MOSI             11
```

Refer to the table above to determine whether you have SPI or UART controlled Bluefruits!

Select the Serial Bus

Once you've configured your pin setup in the BluefruitConfig.h file, you can now check and adapt the example sketch.

The Adafruit_BluefruitLE_nRF51 library supports four different serial bus options, depending on the HW you are using:**SPI** both hardware and software type, and **UART** both hardware and software type.

UART Based Boards (Bluefruit LE UART Friend & Flora BLE)

This is for Bluefruit LE UART Friend & Flora BLE boards. You can use*either* software serial or hardware serial. Hardware serial is higher quality, and less risky with respect to losing data. However, you may not have hardware serial available! Software serial does work just fine with flow-control and we do have that available at the cost of a single GPIO pin.

For software serial (Arduino Uno, Adafruit Metro) you should uncomment the software serial contructor below, and make sure the other three options (hardware serial & SPI) are commented out.

```
// Create the bluefruit object, either software serial...uncomment these lines
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);
```

For boards that require hardware serial (Adafruit Flora, etc.), uncomment the hardware serial constructor, and make sure the other three options are commented out

```
/* ...or hardware serial, which does not need the RTS/CTS pins. Uncomment this line */
Adafruit_BluefruitLE_UART ble(BLUEFRUIT_HWSERIAL_NAME, BLUEFRUIT_UART_MODE_PIN);
```

SPI Based Boards (Bluefruit LE SPI Friend)

For SPI based boards, you should uncomment the hardware SPI constructor below, making sure the other constructors are commented out:

```
/* ...hardware SPI, using SCK/MOSI/MISO hardware SPI pins and then user selected CS/IRQ/RST */
Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
```

If for some reason you can't use HW SPI, you can switch to software mode to bit-bang the SPI transfers via the following constructor:

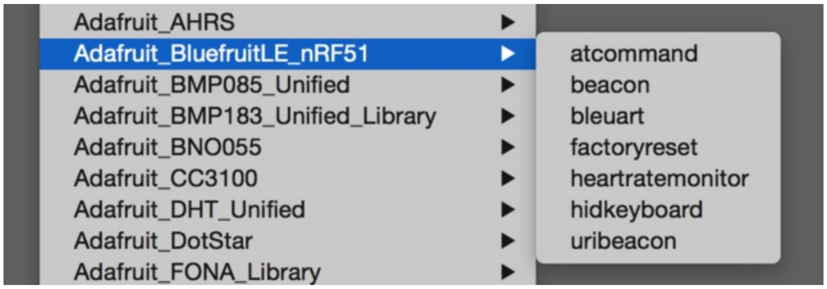
```
/* ...software SPI, using SCK/MOSI/MISO user-defined SPI pins and then user selected CS/IRQ/RST */
Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_SCK, BLUEFRUIT_SPI_MISO,
                             BLUEFRUIT_SPI_MOSI, BLUEFRUIT_SPI_CS,
                             BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
```

ATCommand

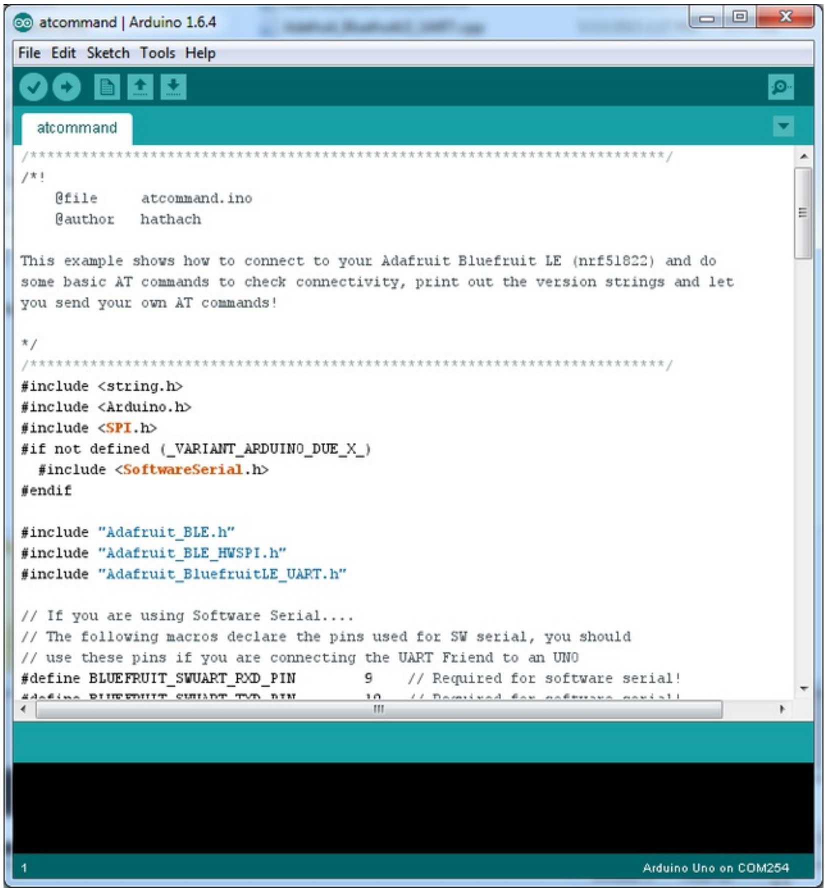
The **ATCommand** example allows you to execute AT commands from your sketch, and see the results in the Serial Monitor. This can be useful for debugging, or just testing different commands out to see how they work in the real world. It's a good one to start with!

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:



```
/******************************************************************************/
/*!
    @file     atcommand.ino
    @author   hathach

This example shows how to connect to your Adafruit Bluefruit LE (nrf51822) and do
some basic AT commands to check connectivity, print out the version strings and let
you send your own AT commands!

*/
/******************************************************************************/
#include <string.h>
#include <Arduino.h>
#include <SPI.h>
#if not defined (_VARIANT_ARDUINO_DUE_X_)
  #include <SoftwareSerial.h>
#endif

#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!
```

Configuration

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

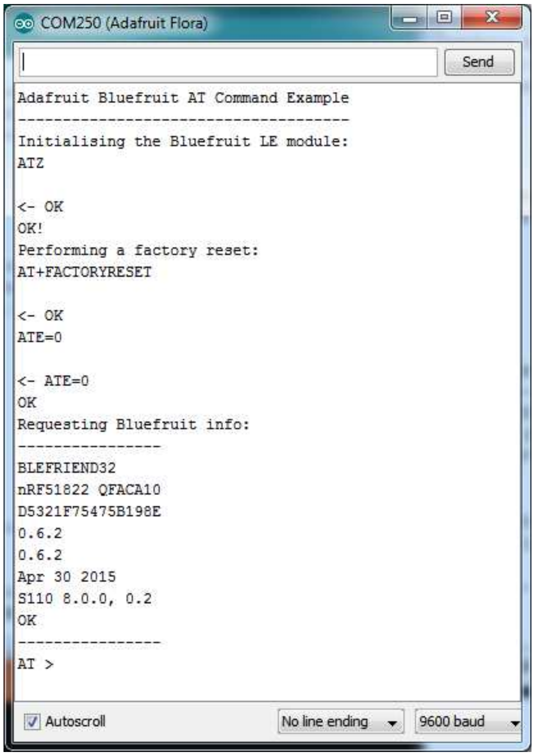
If using software or hardware Serial UART:

- 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)

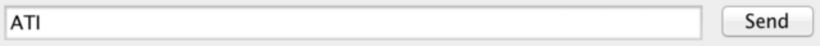
Running the Sketch

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**:



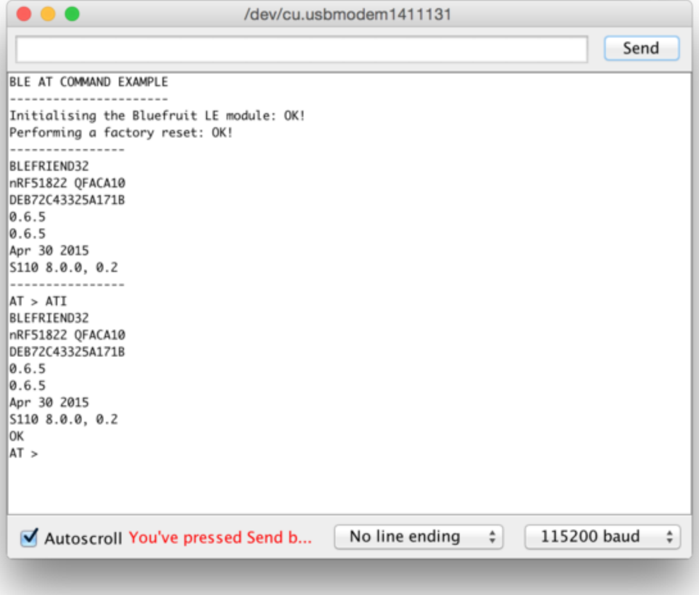
```
Adafruit Bluefruit AT Command Example
-------------------------------------
Initialising the Bluefruit LE module:
ATZ

<- 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
----------------
AT >
```

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:



```
ATI
```

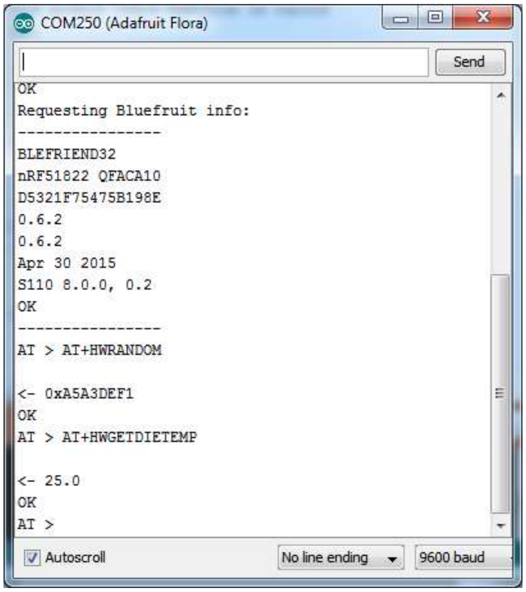
The response to the AT command will be displayed in the main part of the Serial Monitor. The response from**ATI**' is shown below:



```
BLE AT COMMAND EXAMPLE
----------------------
Initialising the Bluefruit LE module: OK!
Performing a factory reset: OK!
----------------
BLEFRIEND32
nRF51822 QFACA10
DEB72C43325A171B
0.6.5
0.6.5
Apr 30 2015
S110 8.0.0, 0.2
----------------
AT > ATI
BLEFRIEND32
nRF51822 QFACA10
DEB72C43325A171B
0.6.5
0.6.5
Apr 30 2015
S110 8.0.0, 0.2
OK
AT >
```

☑ Autoscroll You've pressed Send b... No line ending 115200 baud

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)



```
OK
Requesting Bluefruit info:
----------------
BLEFRIEND32
nRF51822 QFACA10
D5321F75475B198E
0.6.2
0.6.2
Apr 30 2015
S110 8.0.0, 0.2
OK
----------------
AT > AT+HWRANDOM

<- 0xA5A3DEF1
OK
AT > AT+HWGETDIETEMP

<- 25.0
OK
AT >
```

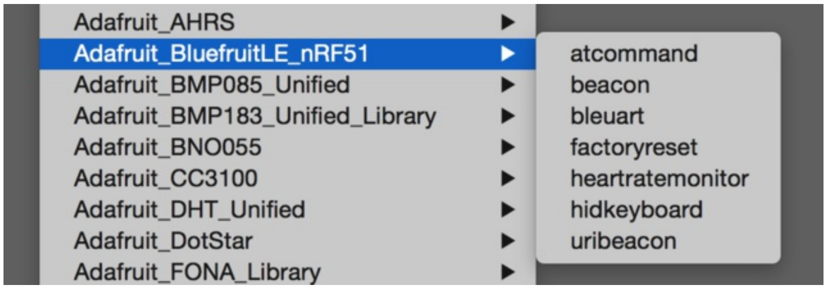
☑ Autoscroll No line ending 9600 baud

BLEUart

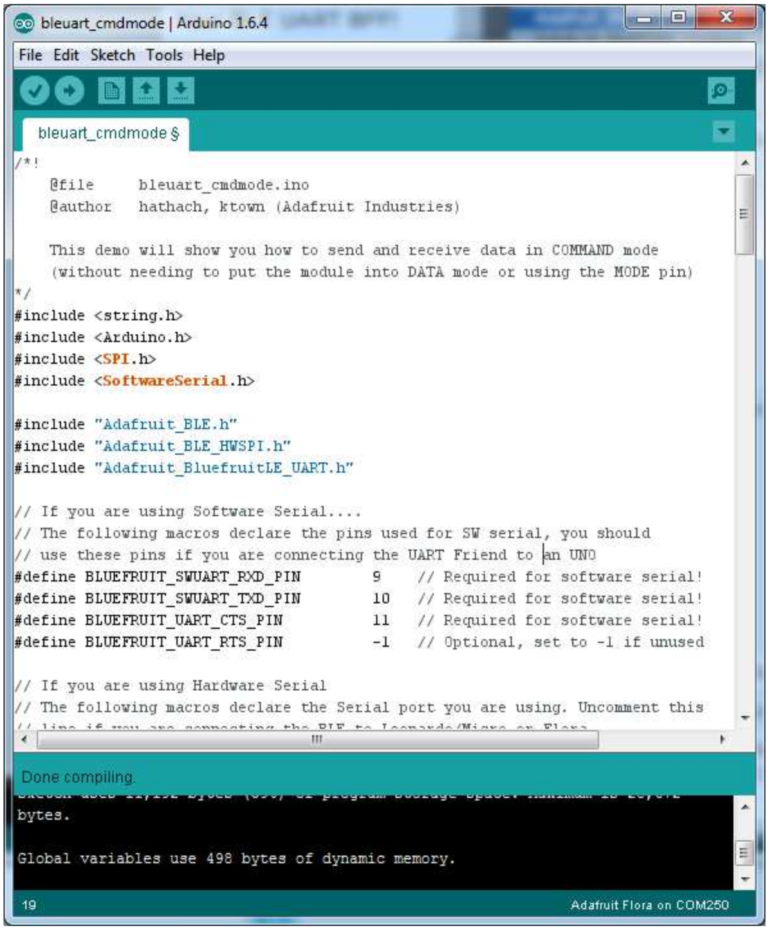
The **BLEUart** example sketch allows you to send and receive text data between the Arduino and a connected Bluetooth Low Energy Central device on the other end (such as you mobile phone using the **Adafruit Bluefruit LE Connect** application for Android (http://adafru.it/f4G) or iOS (http://adafru.it/f4H)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



```
/*!
    @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
```

```
Done compiling.
bytes.

Global variables use 498 bytes of dynamic memory.
```

Configuration

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