



Micro:bit with Arduino

Created by lady ada

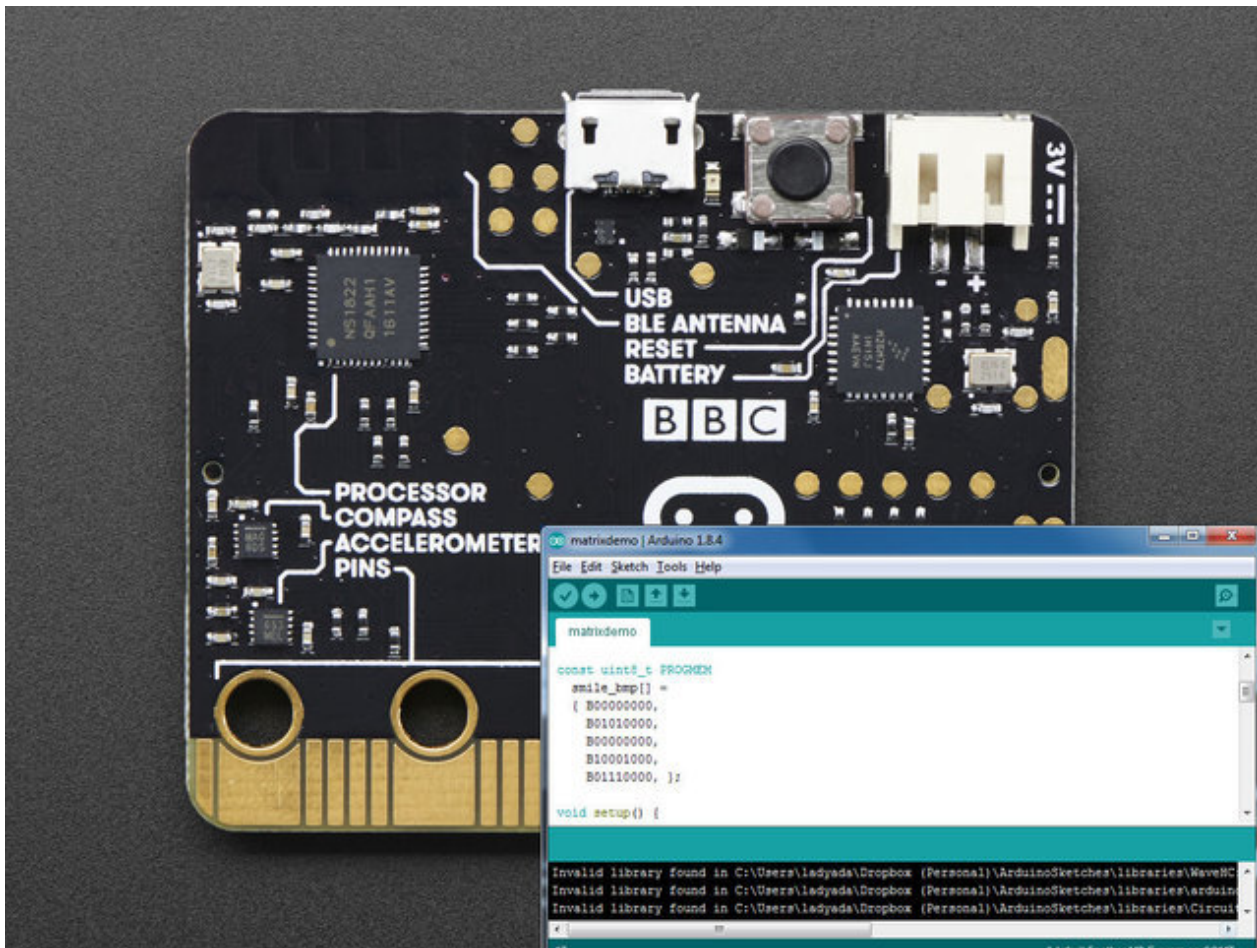


Last updated on 2017-09-27 05:26:28 PM UTC

Guide Contents

Guide Contents	2
Overview	3
BBC micro:bit	4
BBC micro:bit Go Bundle	4
Install board and blink!	5
Install Windows 7 Driver	5
Download Arduino IDE	5
Add NRF5x Board Support	5
Select Board and Upload	6
Buttons	9
Other GPIO	11
Accelerometer and Magnetometer	13
Magnetometer	13
Accelerometer	15
Adafruit Libraries	18
Download BLE Peripheral library	18
Download Adafruit GFX library	18
Download Adafruit_Microbit library	19
LED Matrix	21
Bluetooth UART	23
Install Library & Example Code	23
Bluetooth Connection	25
Bluetooth Plotter	29
Install Library & Example Code	29
Bluetooth Controller	33
Install Library & Example Code	33
Logging Temperature to Adafruit IO	36
Create a Microbit Temperature Feed	36
Temperature Logger Sketch	36
Test UART Mode	38

Overview



Did you know that the Arduino IDE can be used to program the micro:bit? Now you have yet another way to use this cool board! Learn how to set up Arduino to program your micro:bit, blink some LEDs, read the internal temperature sensor, send and receive data over Bluetooth - even log data to Adafruit.IO!

The micro:bit is a small nRF51-powered learning platform for kids - you can use it with Microsoft [MakeCode](https://adafru.it/zrb) (<https://adafru.it/zrb>) (drag-n-drop block programming or Javascript), [micropython](https://adafru.it/zrc) (<https://adafru.it/zrc>), or mbed. But we really like using the Arduino IDE, especially since there's thousands of existing projects you can use and adapt. Also, you get to have much more advanced projects since you won't run out of memory (as you would with micropython) and you can write just about any code you want (with MakeCode you are more constrained to what has already been provided for you, a trade-off for ease-of-use).

Pick up a microbit and follow along on how you can do some pretty advanced things with your 'bit!



BBC micro:bit

PRODUCT ID: 3530

The British Invasion is here! No, not music...microcontrollers! New to the USA is the newest and easiest way to learn programming and electronics - the BBC micro:bit . Designed...

<https://adafruit.it/yEP>

\$14.95

IN STOCK

Your browser does not support the video tag.

BBC micro:bit Go Bundle

PRODUCT ID: 3362

The British Invasion is here! No, not music...microcontrollers! New to the USA is the newest and easiest way to learn programming and electronics - the BBC micro:bit . Designed...

<https://adafruit.it/yEO>

\$16.50

IN STOCK

Install board and blink!

Install Windows 7 Driver

If you are running Windows 7 you need to install this driver.

If you are on Mac, Win 10+ or Linux it is not require! Skip this step

[Download mBed serial driver](https://adafru.it/pNa)

<https://adafru.it/pNa>

Download Arduino IDE

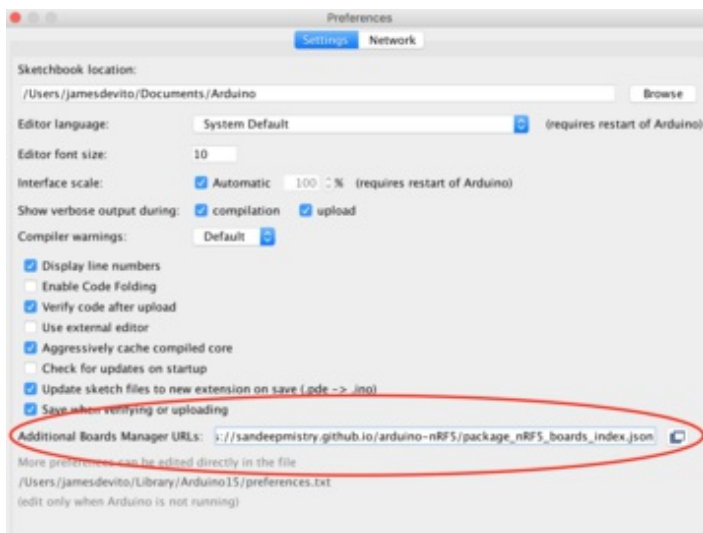
You will need to use the Desktop IDE. Make sure you are running the latest version.

[Download Arduino IDE](https://adafru.it/fvm)

<https://adafru.it/fvm>

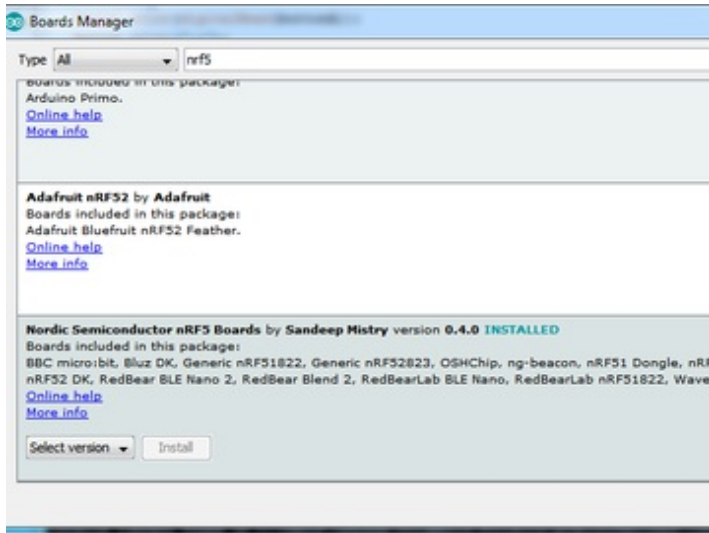
Add NRF5x Board Support

The microbit uses the nRF51 which is not 'natively' supported. But its easy to add support!



In Arduino, go to Preferences and add https://sandeepmistry.github.io/arduino-nRF5/package_nRF5_boards_index.json into the Additional Board Manager URL text box.

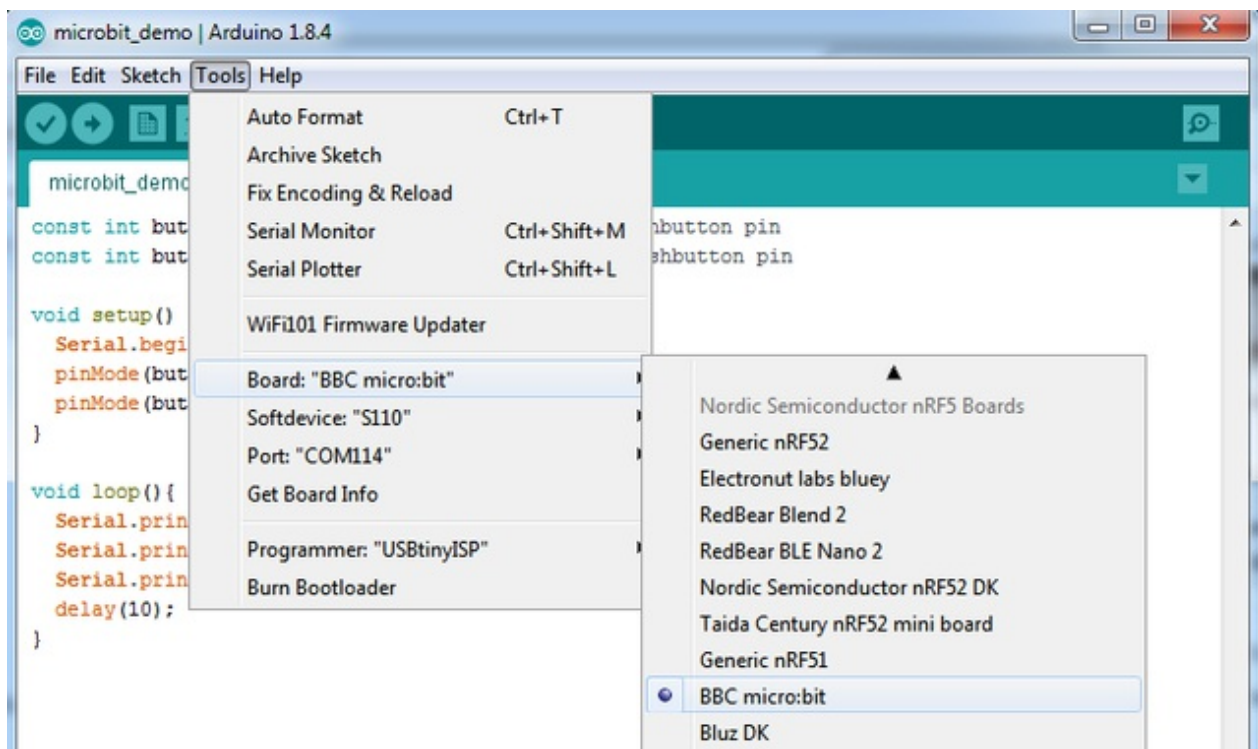
If this is not your first, make sure to separate URLs with a comma.



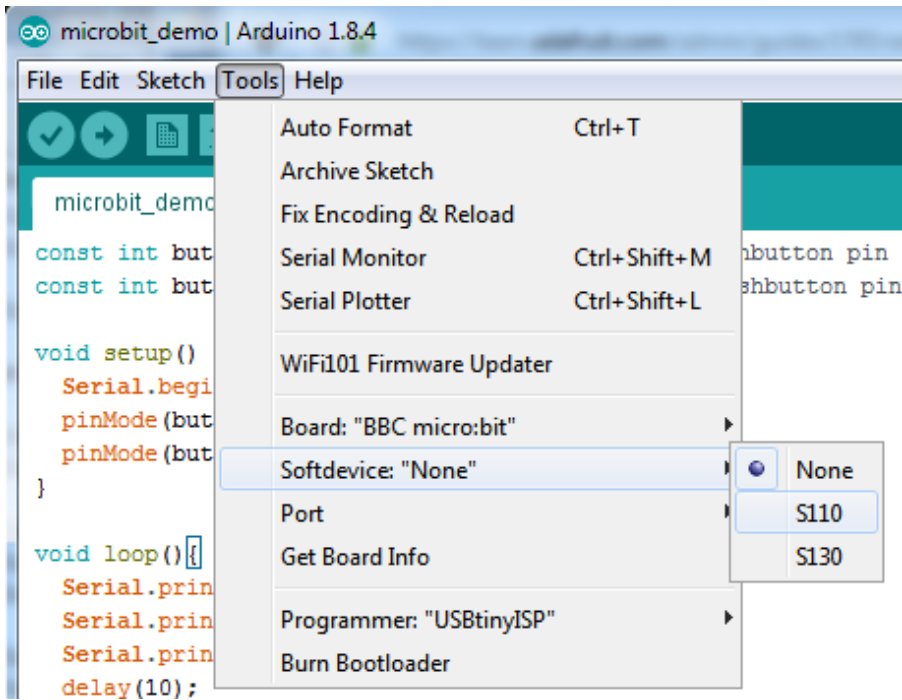
Open **Tools>Board>Boards Manager** from the menu bar, search for **nRF5** and install "**Nordic Semiconductor nRF5 Boards**" by Sandeep Mistry

Select Board and Upload

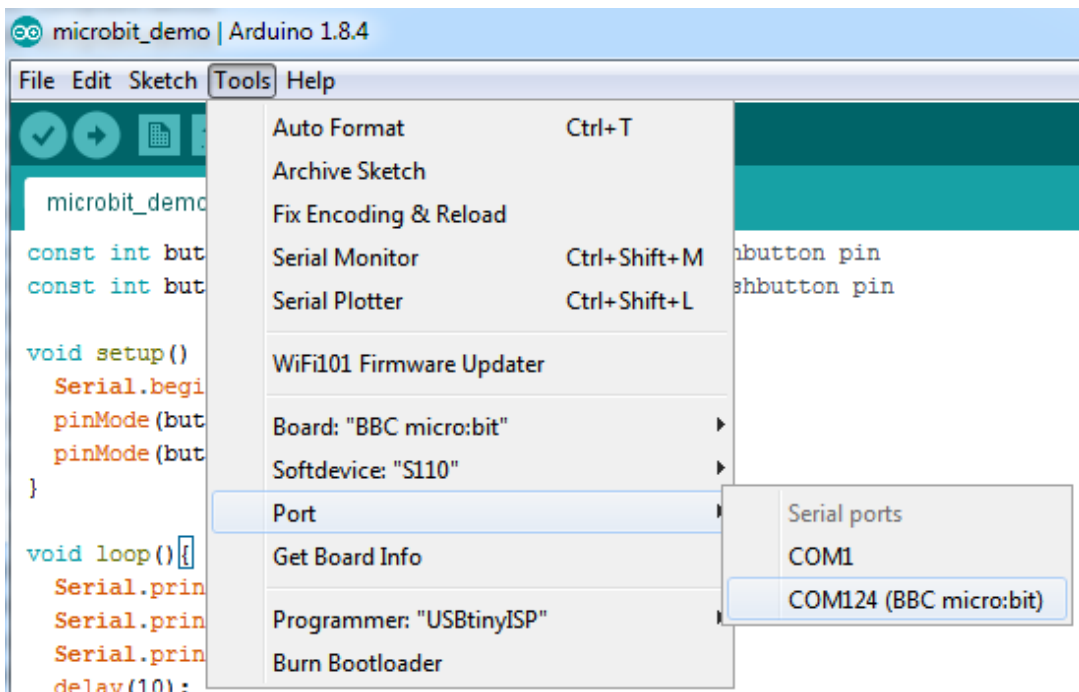
Select BBC micro:bit from the Boards menu



Set SoftDevice to S110



And set the Port to the microbit



And create a new sketch with this blink demo

```
const int COL1 = 3; // Column #1 control
const int LED = 26; // 'row 1' led

void setup() {
  Serial.begin(9600);
```

```

Serial.println("microbit is ready!");

// because the LEDs are multiplexed, we must ground the opposite side of the LED
pinMode(COL1, OUTPUT);
digitalWrite(COL1, LOW);

pinMode(LED, OUTPUT);
}

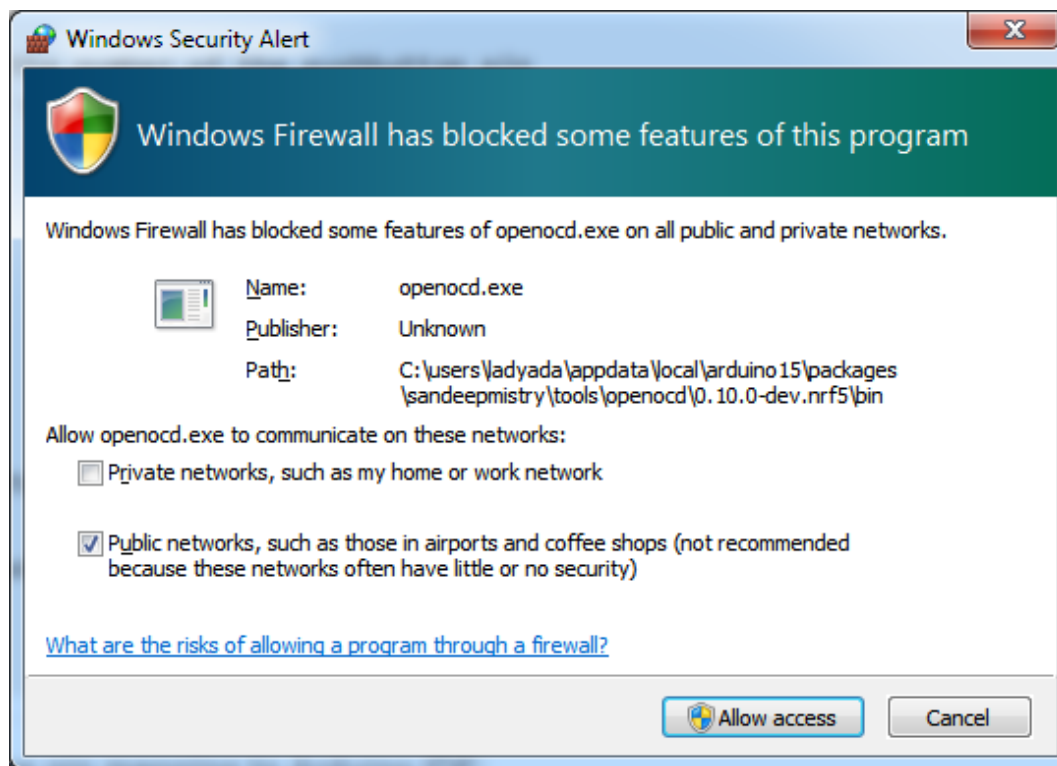
void loop(){
  Serial.println("blink!");

  digitalWrite(LED, HIGH);
  delay(500);
  digitalWrite(LED, LOW);
  delay(500);
}

```

Click **Upload!**

If you get a warning about **openocd** - approve access so it can upload the code



You should see the top left LED blinking!

Buttons

Create a new sketch and upload the following code to read the button presses in the Serial Console

```
const int buttonA = 5;  // the number of the pushbutton pin
const int buttonB = 11; // the number of the pushbutton pin

void setup() {
  Serial.begin(9600);

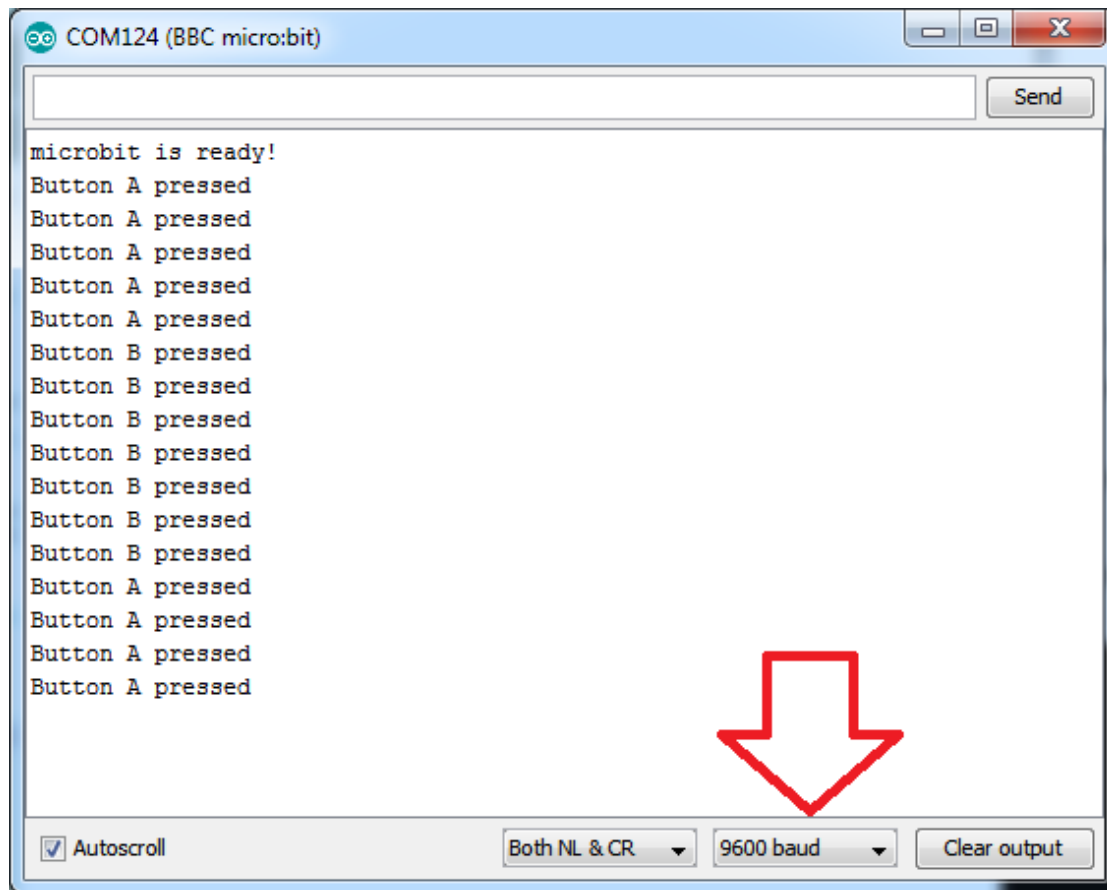
  Serial.println("microbit is ready!");

  pinMode(buttonA, INPUT);
  pinMode(buttonB, INPUT);
}

void loop(){
  if (!digitalRead(buttonA)) {
    Serial.println("Button A pressed");
  }
  if (!digitalRead(buttonB)) {
    Serial.println("Button B pressed");
  }
  delay(10);
}
```

Open up the Serial console at **9600 baud** and **press the reset button on the back of the microbit** to restart the software

You will see the welcome message and then try pressing some buttons!



Other GPIO

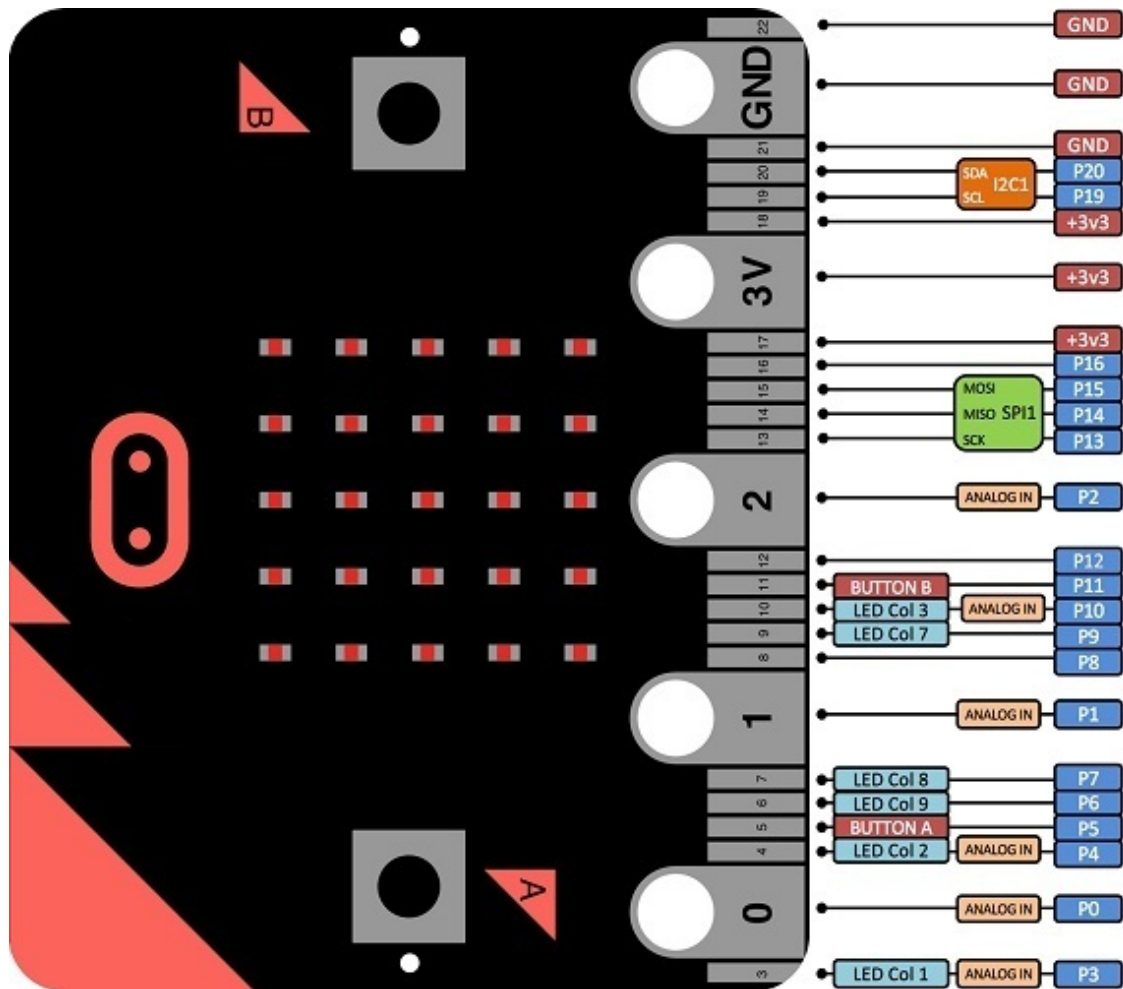
You can easily get to GPIO #0 #1 and #2 using large alligator clips. With a microbit breakout, you can access the other 16 or so.

Note that many of these pins are *multiplexed* with the LED matrix so if you decide to use that, you can't use them for analog inputs or digital I/O

- Pin #0 - large pad - analog in
- Pin #1 - large pad - analog in
- Pin #2 - large pad - analog in
- Pin #3 - analog in, also used for LED matrix
- Pin #4 - analog in, also used for LED matrix
- Pin #5 - also used for Button A
- Pin #6 - also used for LED matrix
- Pin #7 - also used for LED matrix
- Pin #8
- Pin #9 - also used for LED matrix
- Pin #10 - analog in, also used for LED matrix
- Pin #11 - also used for button B
- Pin #12
- Pin #13 - also available as SPI clock
- Pin #14 - also available as SPI MISO
- Pin #15 - also available as SPI MOSI
- Pin #16
- Pin #19 - also available as I2C clock
- Pin #20 - also available as I2C data

So really, if you're using the buttons and LEDs, you can use pins: #0, #1, #2, #8, #12, #13, #14, #15, #16, #19 and #20. Which is still a good amount!

[blynk.cc \(https://adafru.it/yLE\)](https://adafru.it/yLE) has a great graphic of the pin mapping to Arduino IDE:



Accelerometer and Magnetometer

Magnetometer

Lets start with the magnetometer chip, the MAG3110

[MAG3110 Datasheet](#)

<https://adafru.it/z4B>

We can talk to the chip using an Arduino library

[You can download Sparkfun's library by clicking the button below!](#)(<https://adafru.it/z4C>)

[Download Sparkfun MAG3110 breakout library](#)

<https://adafru.it/z4D>

[And read our guide on how to install libraries](#)(<https://adafru.it/nEO>)

Restart the IDE. Now you can [upload some examples](#) (<https://adafru.it/z4E>). I suggest starting with the [Basic example](#) (<https://adafru.it/z4F>) which is replicated below

```
/* *****  
 * SparkFun_MAG3110_Basic  
 * Triple Axis Magnetometer Breakout - MAG3110  
 * Hook Up Guide Example  
 *  
 * Utilizing Sparkfun's MAG3110 Library  
 * A basic sketch that reads x y and z readings  
 * from the MAG3110 sensor  
 *  
 * George B. on behalf of SparkFun Electronics  
 * Created: Sep 22, 2016  
 * Updated: n/a  
 *  
 * Development Environment Specifics:  
 * Arduino 1.6.7  
 *  
 * Hardware Specifications:  
 * SparkFun MAG3110  
 * Bi-directional Logic Level Converter  
 * Arduino Micro  
 *  
 * This code is beerware; if you see me (or any other SparkFun employee) at the  
 * local, and you've found our code helpful, please buy us a round!  
 * Distributed as-is; no warranty is given.  
 * *****/
```

```

#include <SparkFun_MAG3110.h>

MAG3110 mag = MAG3110(); //Instantiate MAG3110

void setup() {
  Serial.begin(9600);

  mag.initialize(); //Initializes the mag sensor
  mag.start();      //Puts the sensor in active mode
}

void loop() {

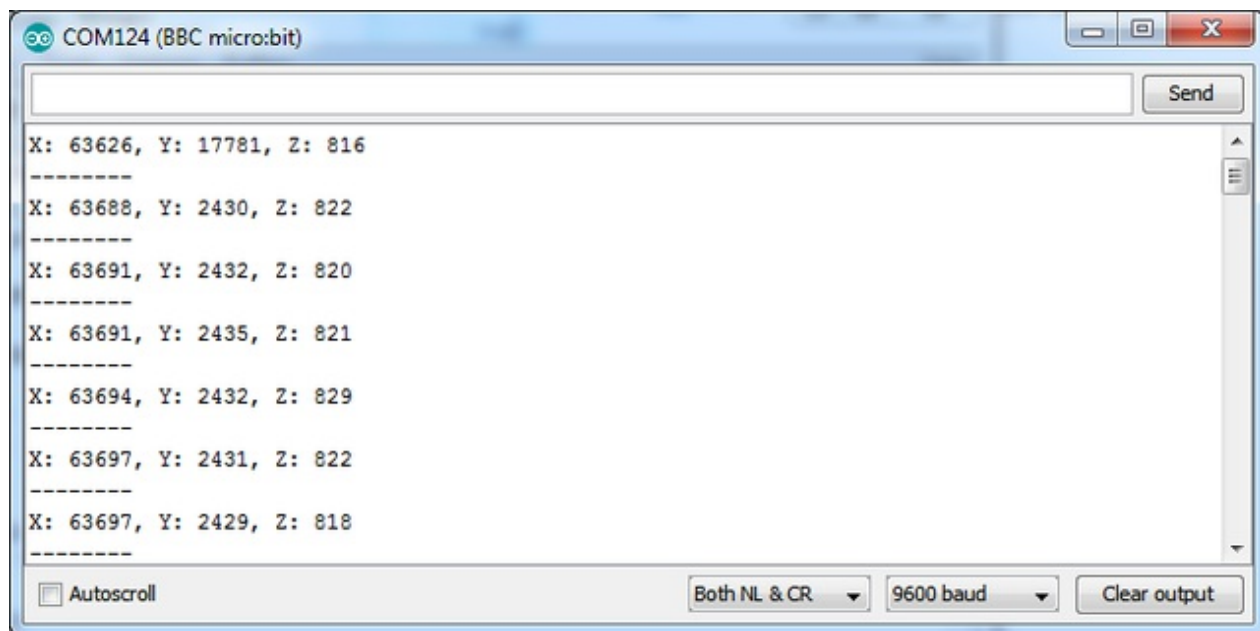
  int x, y, z;
  //Only read data when it's ready
  if(mag.dataReady()) {
    //Read the data
    mag.readMag(&x, &y, &z);

    Serial.print("X: ");
    Serial.print(x);
    Serial.print(", Y: ");
    Serial.print(y);
    Serial.print(", Z: ");
    Serial.println(z);

    Serial.println("-----");
  }
}

```

Upload this to the microbit to see the following raw data:



Note that the magnetometer is not calibrated, so you'll get different numbers on XYZbut when

you twist and rotate the microbit the numbers should move up and down a bit! (This is why magnetometers must be calibrated)

Accelerometer

The microbit has an onboard 3-axis accelerometer as well!

You can use this **akafugu MMA8653** to communicate with it:

[MMA8653.zip](#)

<https://adafru.it/z5a>

Install like other libraries!

Next up, run this example code:

```
/*
 * MMA845XQ test code
 * (C) 2012 Akafugu Corporation
 *
 * This program is free software; you can redistribute it and/or modify it under the
 * terms of the GNU General Public License as published by the Free Software
 * Foundation; either version 2 of the License, or (at your option) any later
 * version.
 *
 * This program is distributed in the hope that it will be useful, but WITHOUT ANY
 * WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
 * PARTICULAR PURPOSE. See the GNU General Public License for more details.
 */

#include "Wire.h"
#include "MMA8653.h"

MMA8653 accel;

void setup() {
  Serial.begin(9600);

  Serial.println("microbit accel test");

  accel.begin(false, 2); // 8-bit mode, 2g range
}

void loop() {
  accel.update();
  Serial.print(accel.getX()); Serial.print(" ");
  Serial.print(accel.getY()); Serial.print(" ");
```

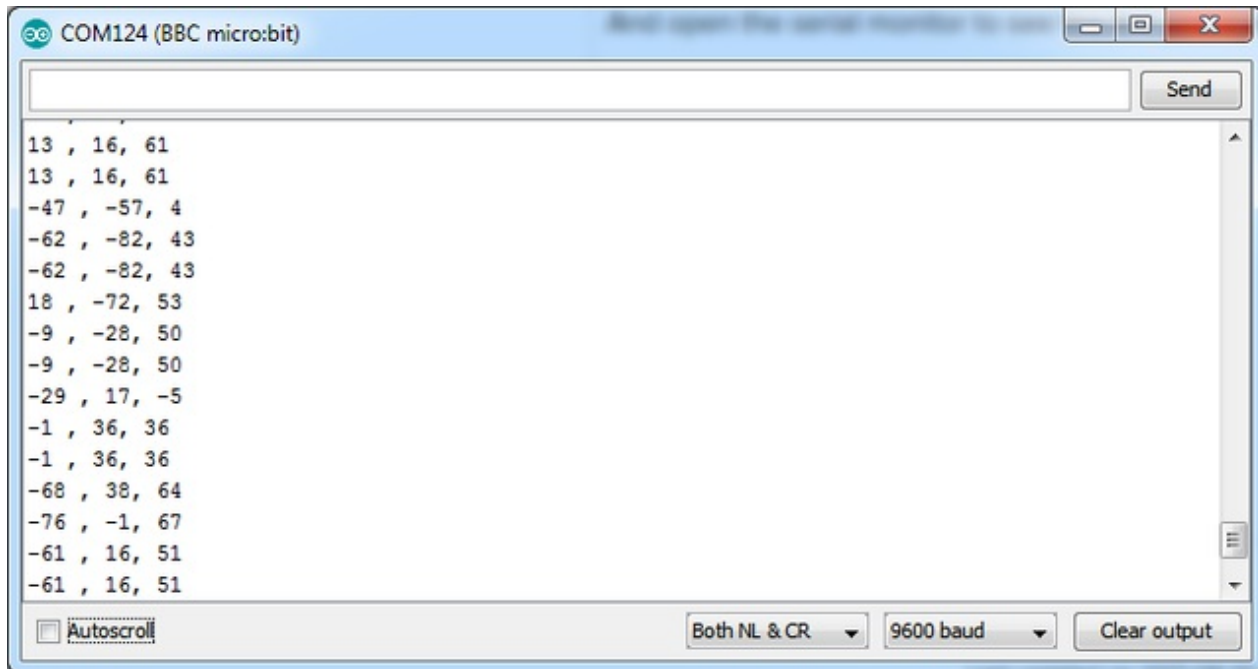
```

Serial.println(accel.getZ());

delay(100);
}

```

And open the serial monitor to see the X Y and Z acceleration data points!



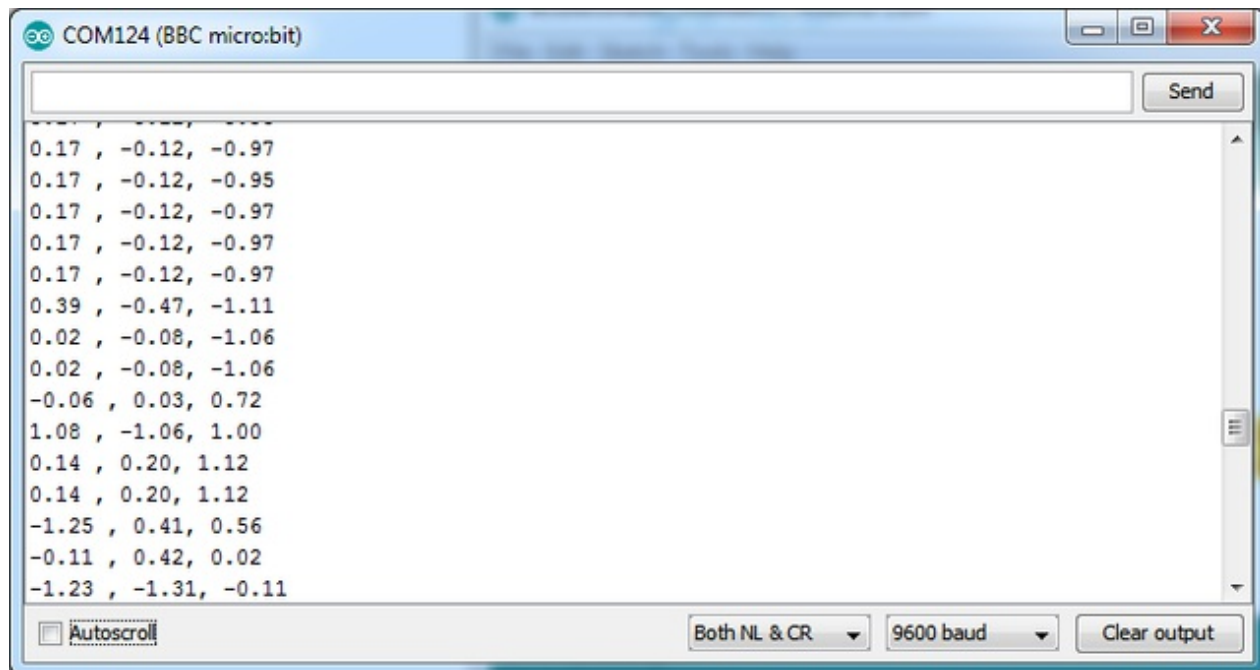
This library is pretty old and incomplete so at this time you can only use it in 8-bit mode. If you want to get the data in **g**'s use this for the loop:

```

void loop() {
  accel.update();
  Serial.print((float)accel.getX() * 0.0156); Serial.print(" , ");
  Serial.print((float)accel.getY() * 0.0156); Serial.print(" , ");
  Serial.println((float)accel.getZ() * 0.0156);

  delay(100);
}

```



Adafruit Libraries

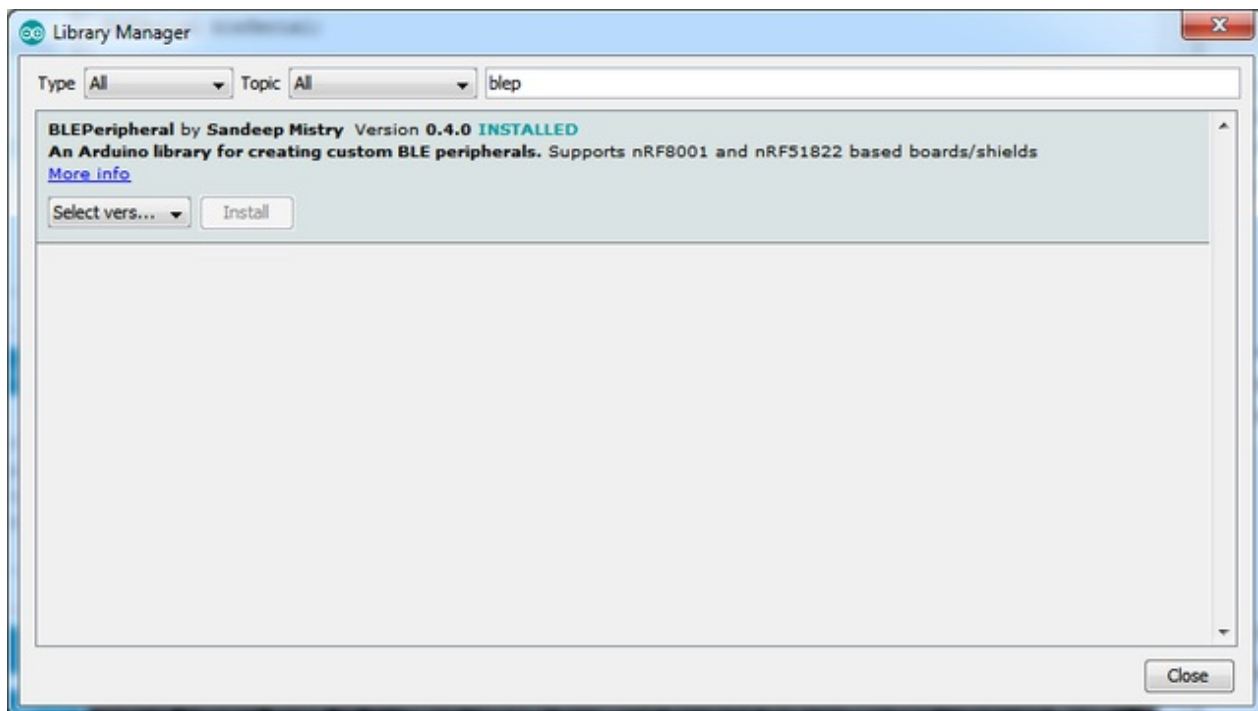
Once you want to get any more complex stuff going, you'll need a helper library to manage stuff like the internal temperature sensor, LED matrix, or Bluetooth connection.

To make your life easier, we've written up a wrapper library that manages all this stuff for you.

You'll also need to install some helpers:

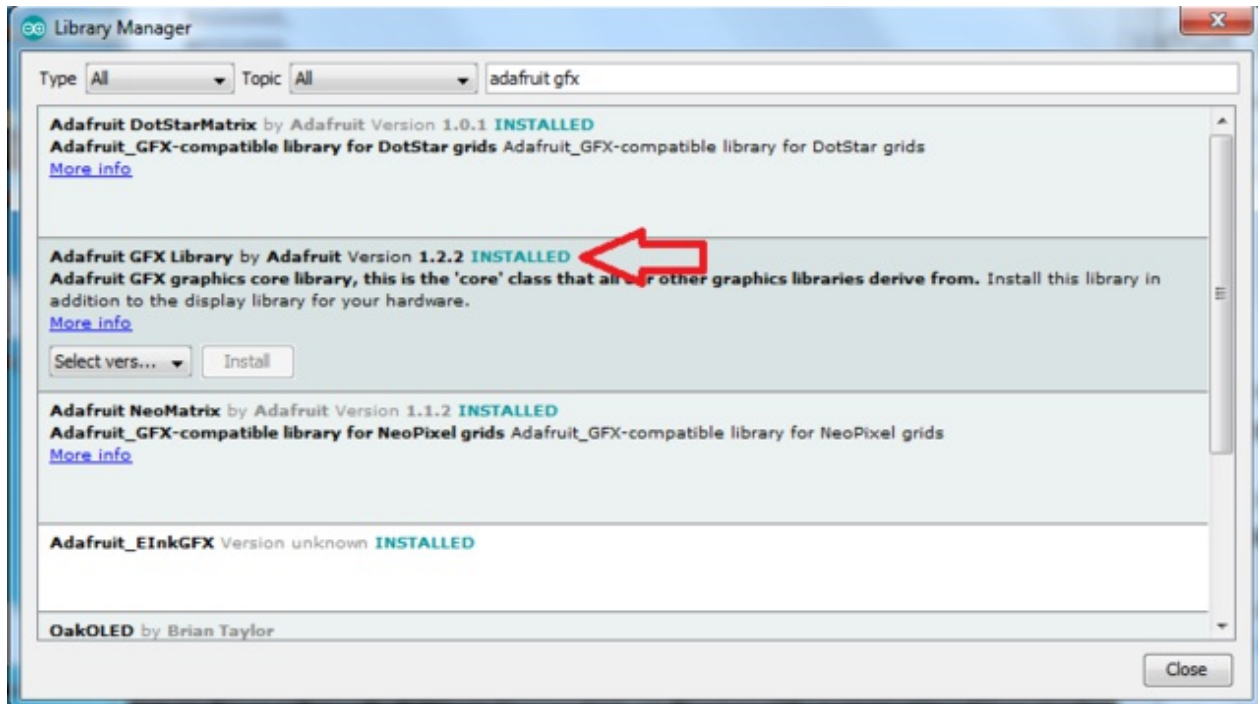
Download BLE Peripheral library

In the Arduino library manager, install the **BLE Peripheral library**:



Download Adafruit GFX library

In the Arduino library manager, install the **Adafruit GFX library**:



Download Adafruit_Microbit library

To use the LED matrix or Bluetooth connection, you will need to download Adafruit_Microbit from our github repository. [You can do that by visiting the github repo \(https://adafru.it/zqC\)](https://adafru.it/zqC) and manually downloading or, easier, just click this button to download the zip:

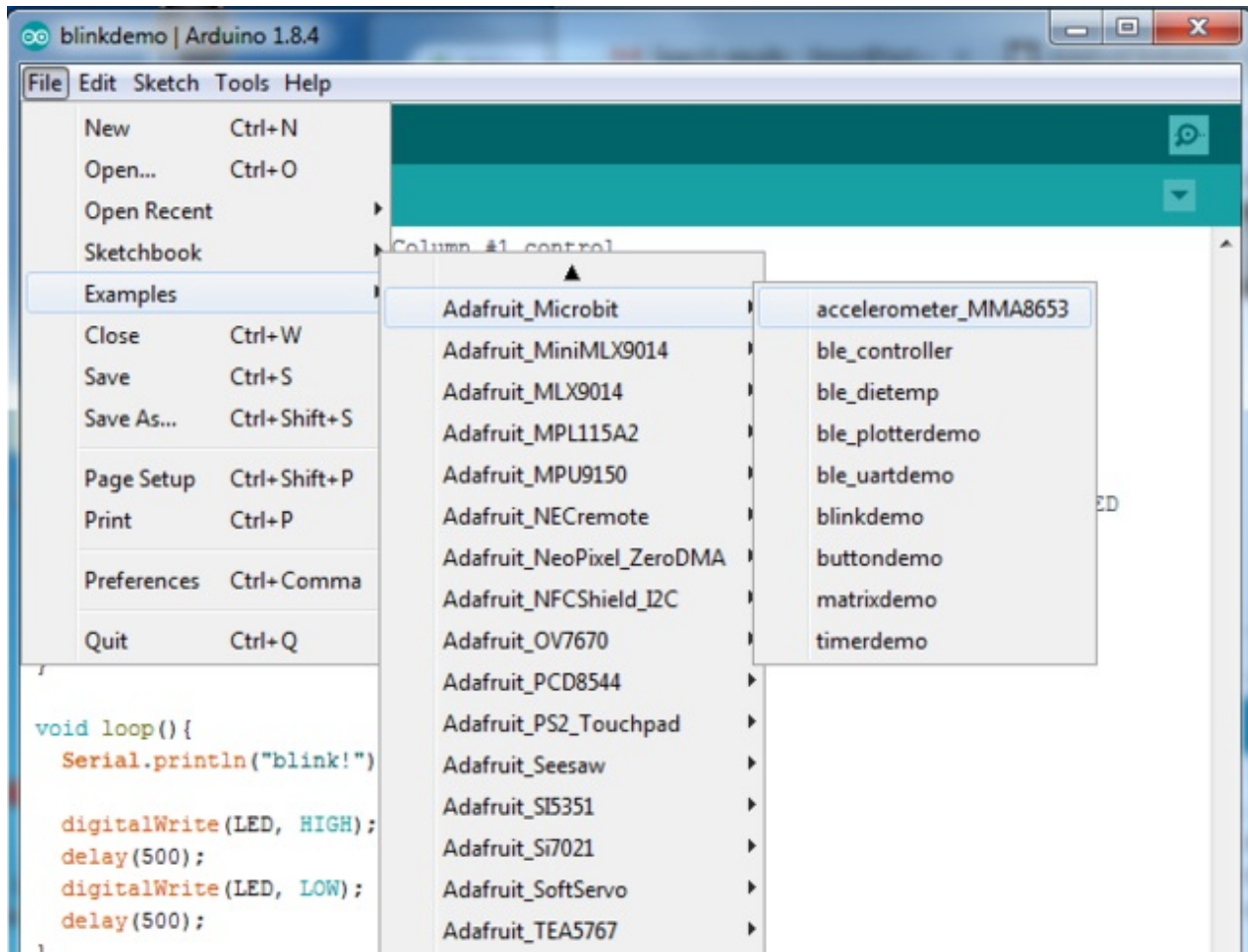
[Download Adafruit Microbit Library](https://adafru.it/zqD)
<https://adafru.it/zqD>

Rename the uncompressed folder **Adafruit_Microbit** and check that the **Adafruit_Microbit** folder contains **Adafruit_Microbit.cpp** and **Adafruit_Microbit.h**

Place the **Adafruit_Microbit** 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>)

Once you've re-started the Arduino IDE you should see the library examples appear in the **File->Examples->Adafruit_Microbit** menu

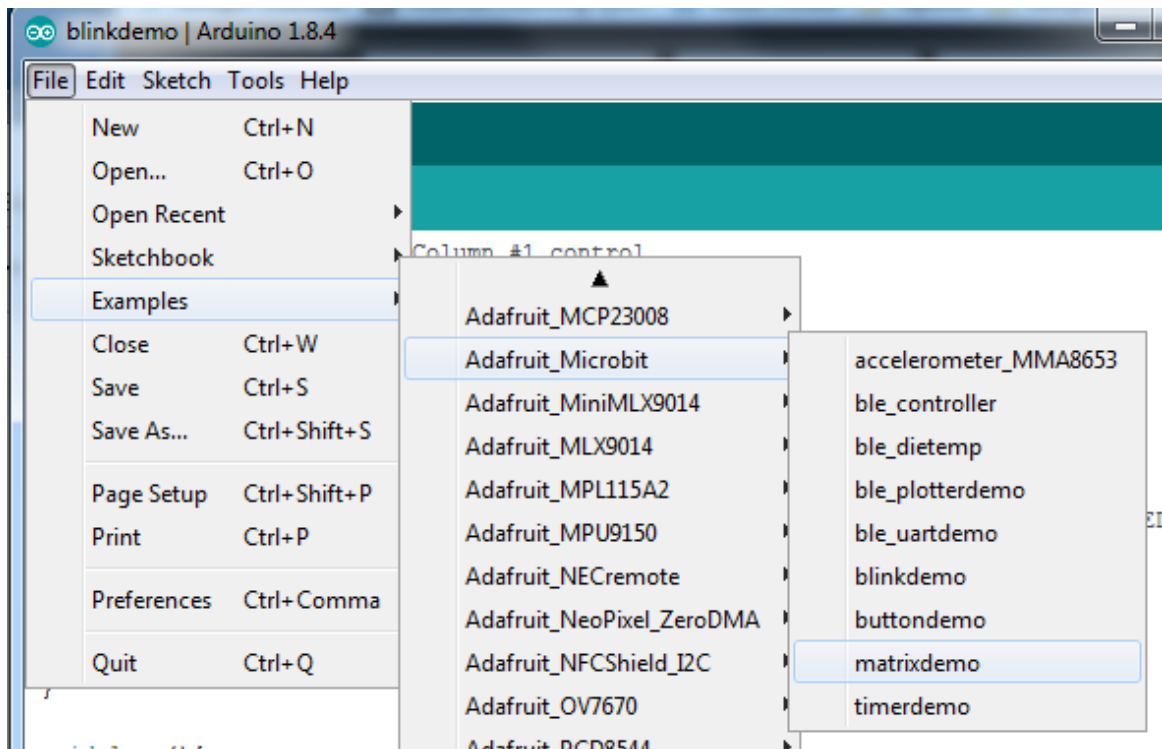


LED Matrix

the LED matrix is a 25-LED multiplexed array. You can't just set the LED's on or off, they must be scanned through continuously to keep them lit. To make this easy for you we've added support to the **Adafruit_Microbit** library - it even uses a timer (Timer 2) to manage all the multiplexing for you.

First up, install the Adafruit helper libraries (<https://adafru.it/zqF>)

Open up the **matrixdemo** sketch and upload it to your bit



The library manages all the refreshing, the drawing routines are all handled by **Adafruit_GFX** which we have documented in detail here <https://learn.adafruit.com/adafruit-gfx-graphics-library/> (<https://adafru.it/zra>)

Note that the screen is small so even though the GFX library can do a lot, there's not a ton of space for complex shapes. We do have a small and simple font you can use that will fit in the 5-pixel-tall display, called **TomThumb.h**

```
// setup font for later!
microbit.setFont(&TomThumb);
microbit.setTextWrap(false);
microbit.setTextColor(LED_ON);
```

To scroll text you'll need to set the cursor 'outside' the 5x5 boundary, like so:

```
// scroll some text
String myMessage = "HELLO WORLD";
for (int i = 5; i > (((int)myMessage.length()-1) * -5) ; i--) {
  microbit.setCursor(i, 5);
  microbit.clear();
  microbit.print(myMessage);
  delay(150);
}
```

For bitmaps, the screen is 5x5 but to make the math easier on everyone, you can store bitmaps in 8-bit-wide structures and just ignore the right-most 3 bits in each 8-bit word:

```
smile_bmp[] =
{ B00000000,
  B01010000,
  B00000000,
  B10001000,
  B01110000, };
```

Bluetooth UART

The main chip has a bluetooth LE radio built in, which is how you can make cool wireless projects!

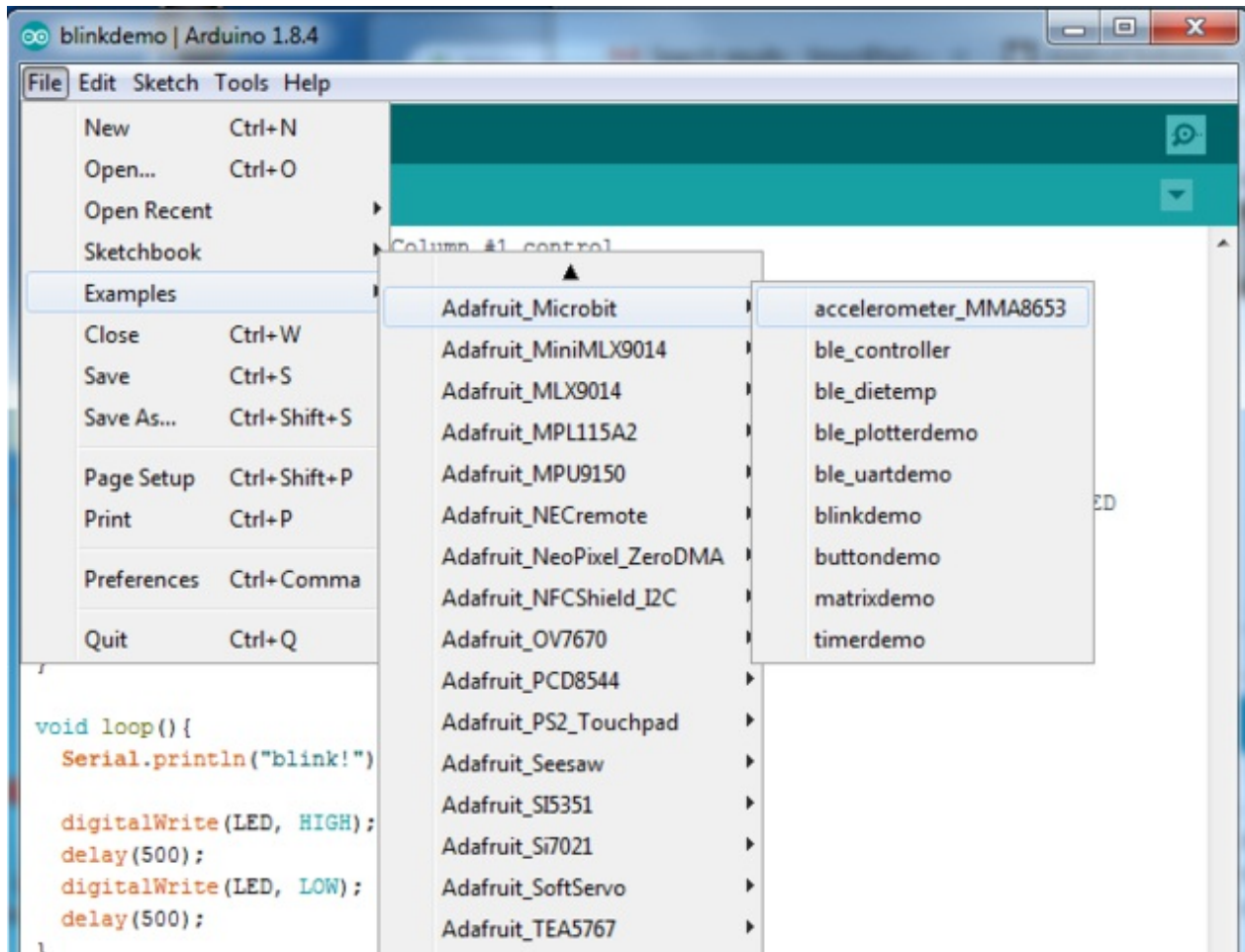
You can use the radio with our Adafruit Bluefruit Connect app without too much difficulty! You can [download **Bluefruit Connect** in both the iOS App store](https://adafru.it/ddu) (<https://adafru.it/ddu>) and [Android app stores](https://adafru.it/f4G) (<https://adafru.it/f4G>)

[Learn more about our app over at the Connect guide.](https://adafru.it/wsD) (<https://adafru.it/wsD>) we'll assume you've read thru it so you have a rough idea how it works

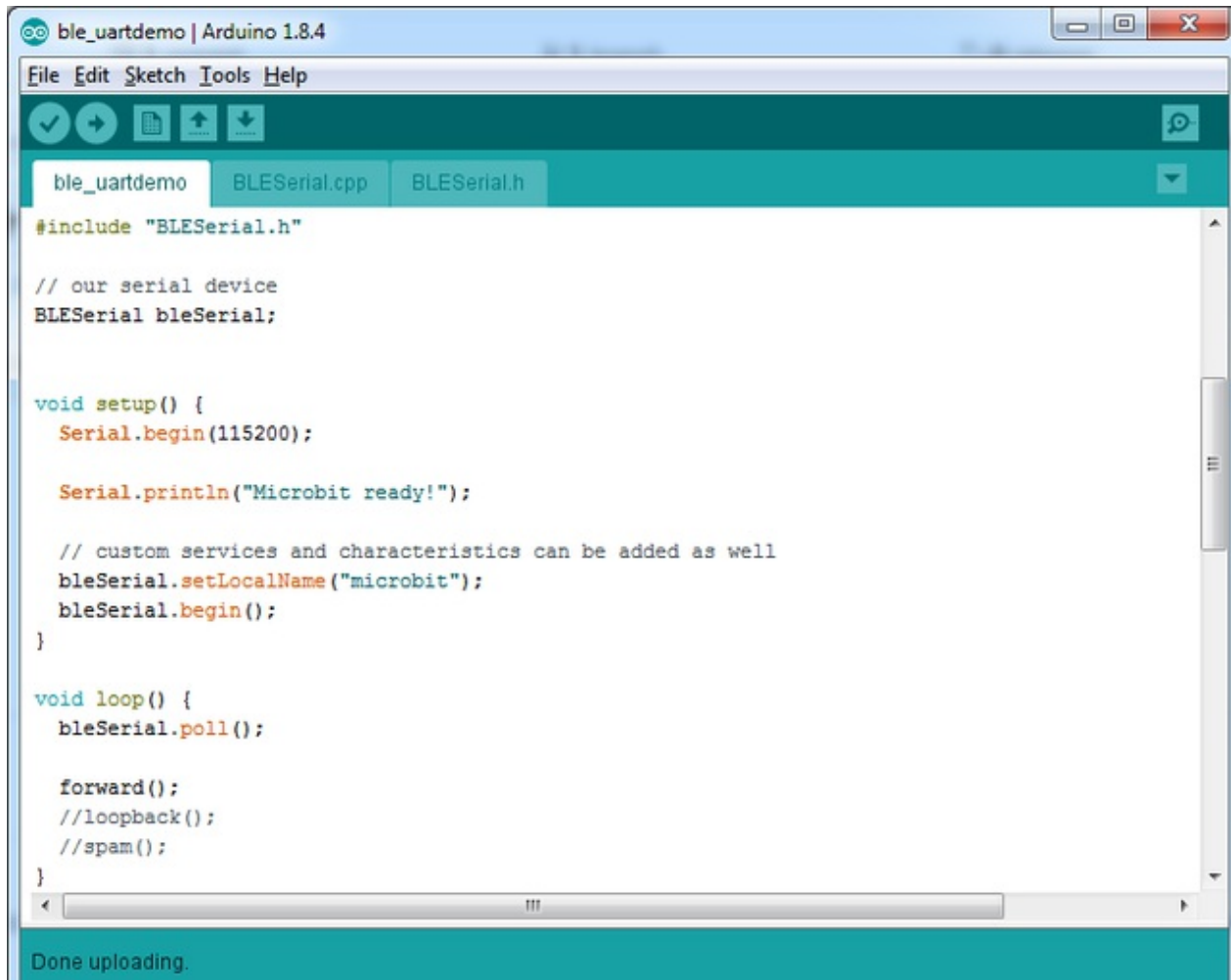
Install Library & Example Code

[First up, install the Adafruit helper library](https://adafru.it/zqF) (<https://adafru.it/zqF>) and friends

You can find our BLE demos in the examples menu:



Load up the BLE UART demo to start



Find these three lines:

```
forward();
//loopback();
//spam();
```

and change them to:

```
//forward();
loopback();
spam();
```

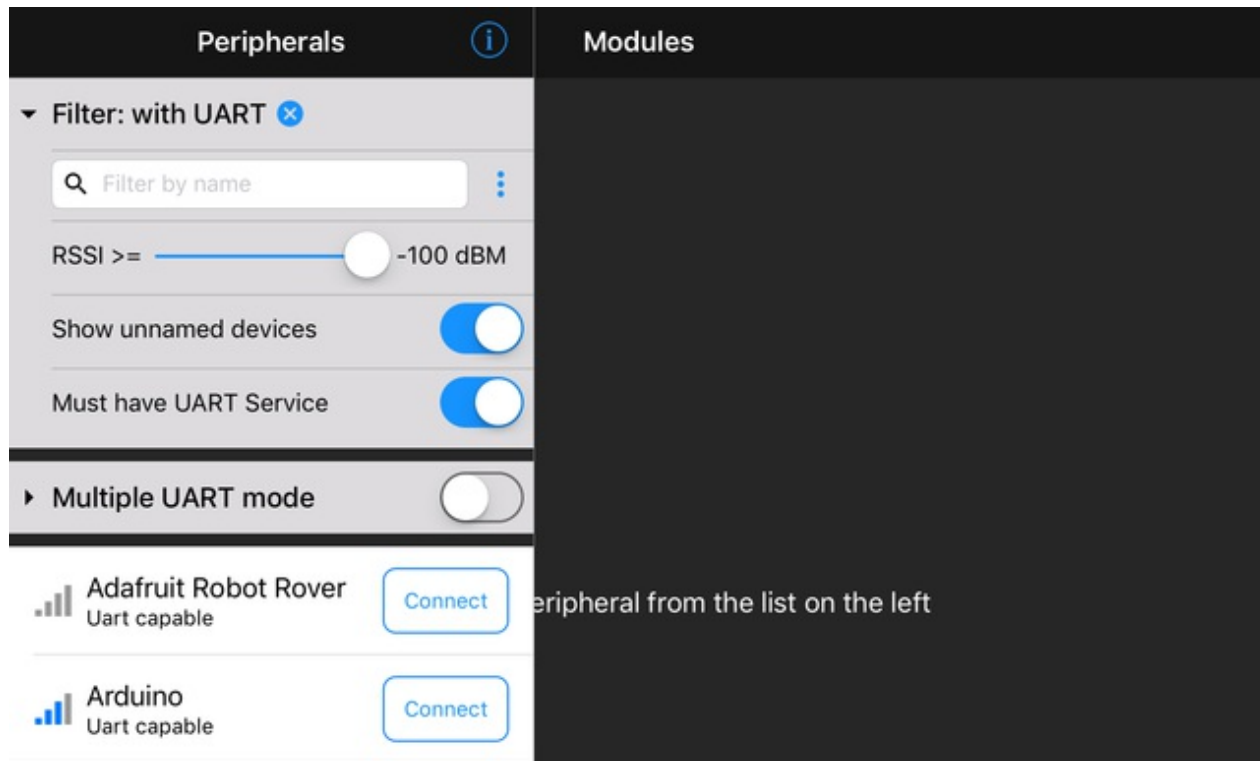
This will turn on auto-transmitting data once a second which will make testing easier. Then upload the sketch

Bluetooth Connection

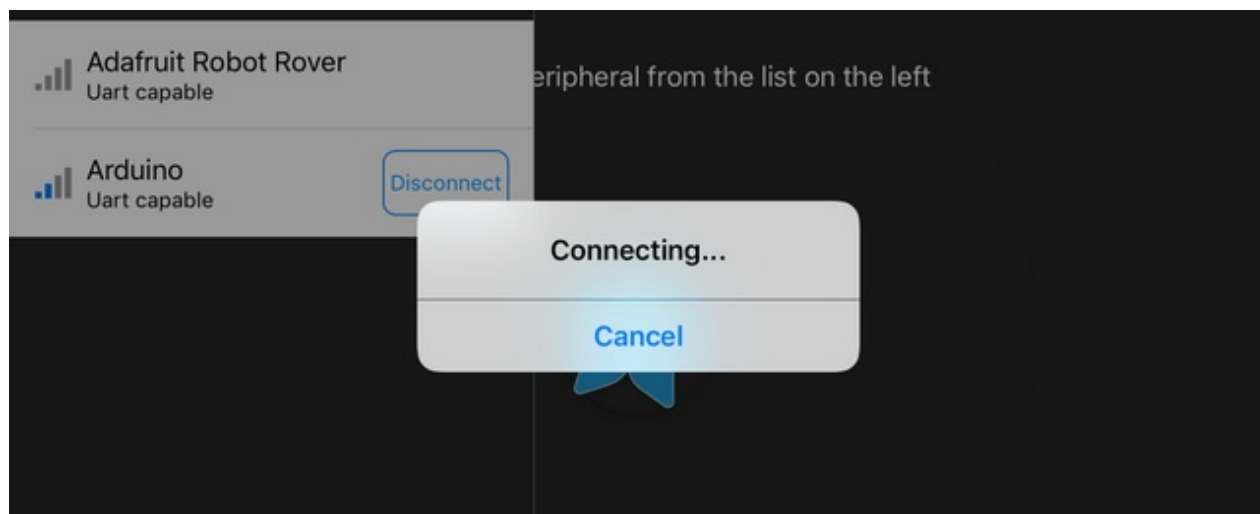
Once you have the sketch on the microbit, open up the Adafruit Bluefruit Connect app.

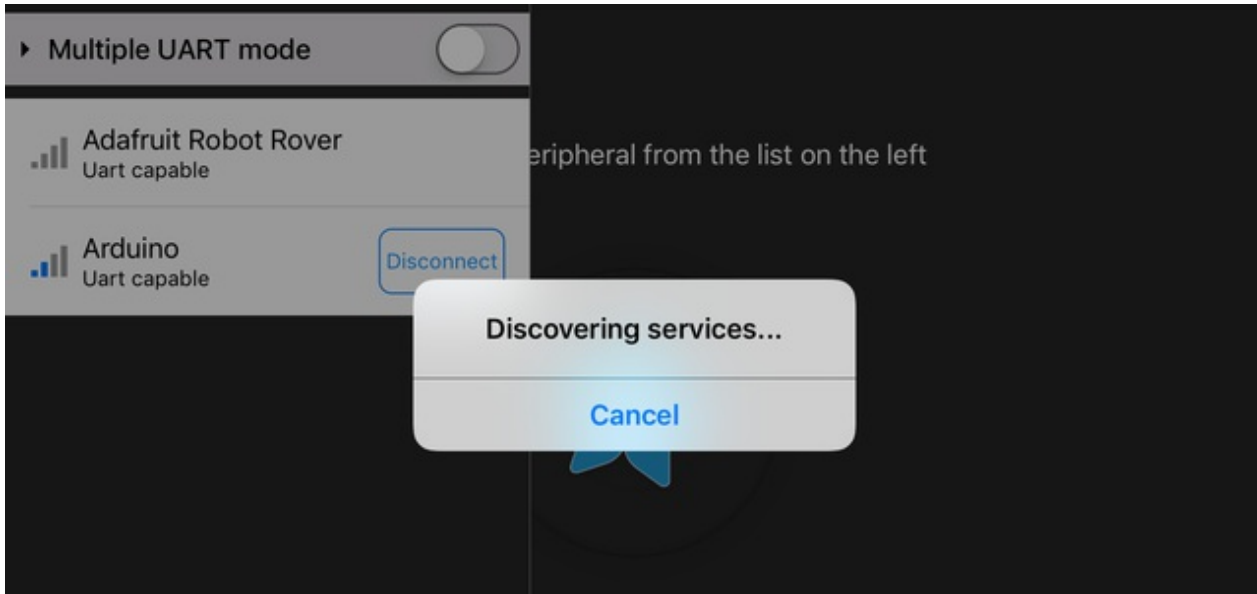
On the left there's a menu you can open. Select the microbit, it might be named **UART** or

Arduino

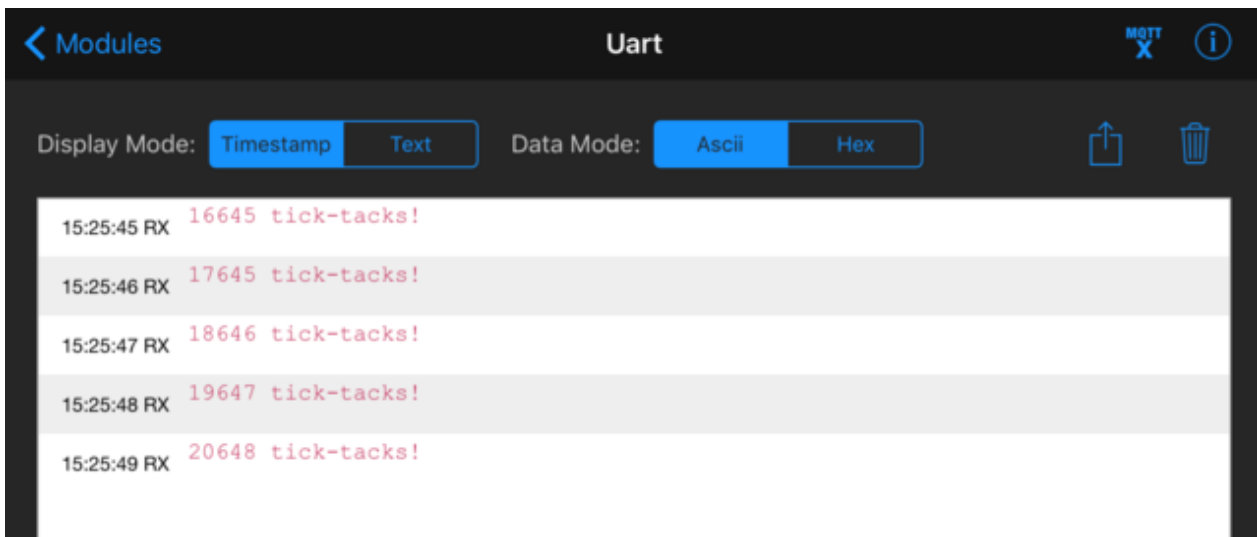


Press **Connect**





Then select **UART** from the list of Modules. Go into **Timestamp** mode and you should see messages once a second:



Go back to the sketch and change it back to:

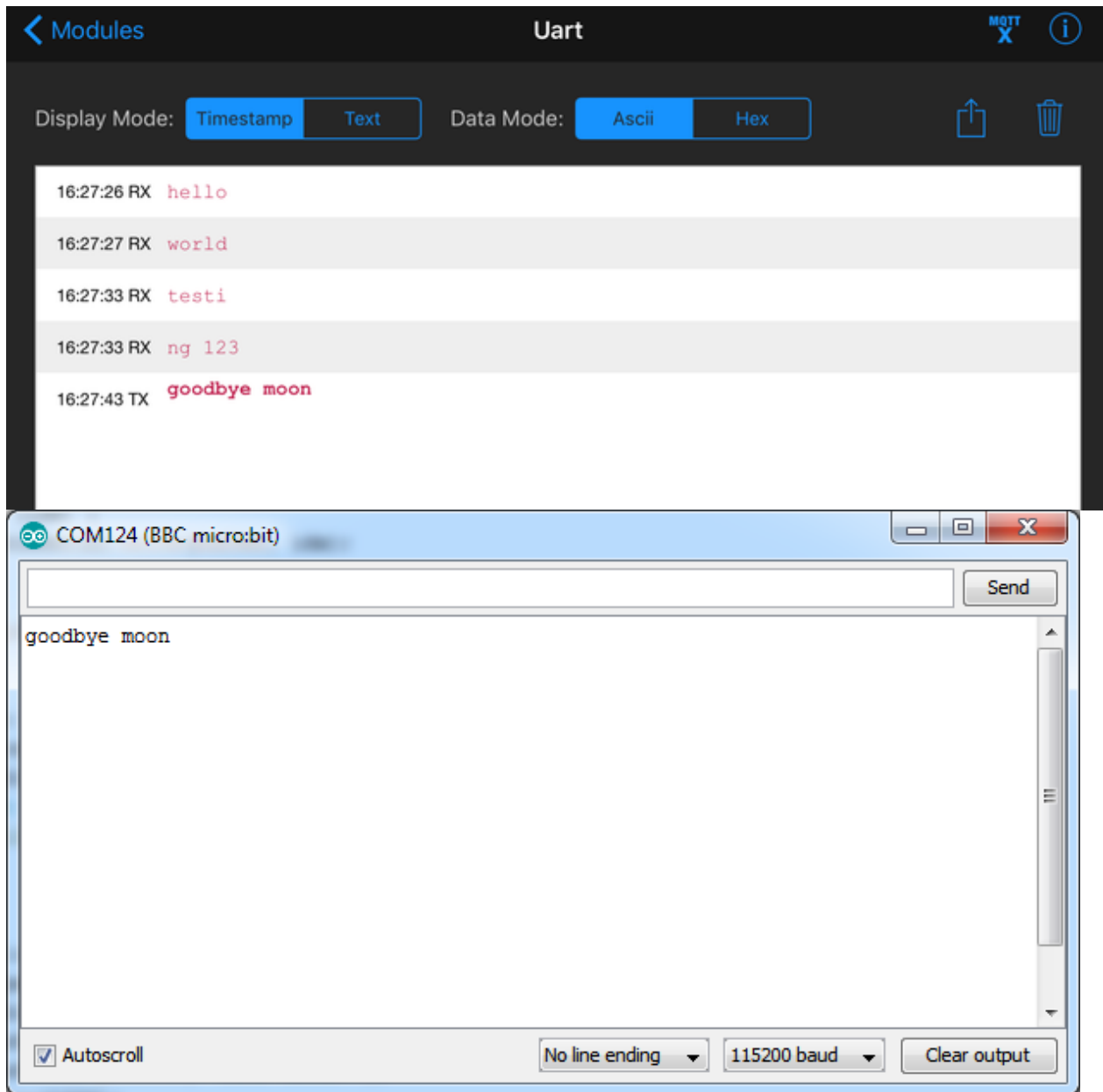
```
forward();
//loopback();
//spam();
```

Re-upload. The app will complain you disconnected, just go back and disconnect from the peripheral-list menu.

Open the serial console at **115200 baud**

Then when you go back to UART mode, you can send data from the tablet to the bit and back. Note that the microbit's UART is a little odd - don't send more than 10 or so characters 'at a

time' through the serial monitor or it may hang.



Once you've got all that working, you can try our controller sketch!

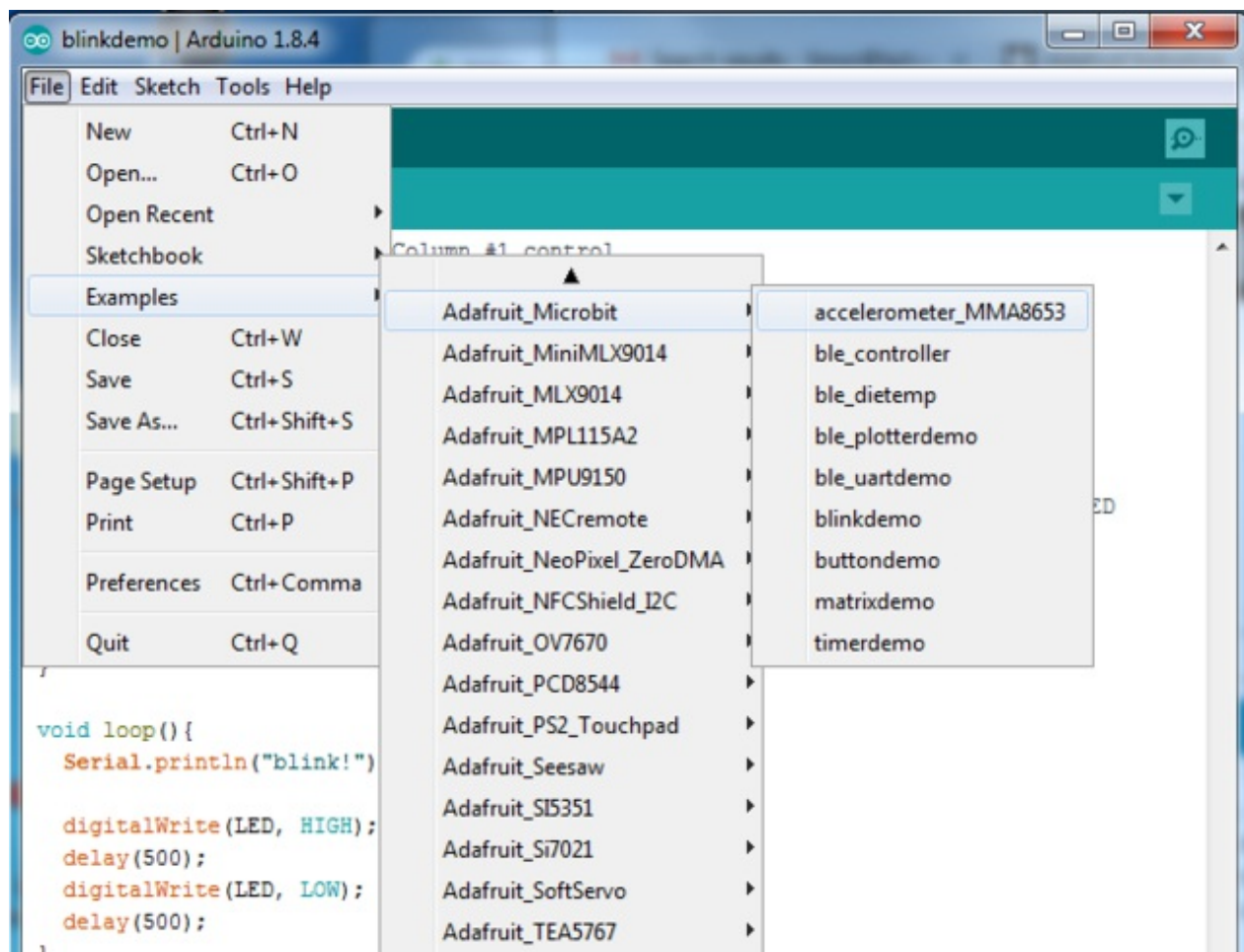
Bluetooth Plotter

The Bluefruit App has a built in plotter that will let you easily visualize data from your microbit! Be sure you got the UART examples working from earlier.

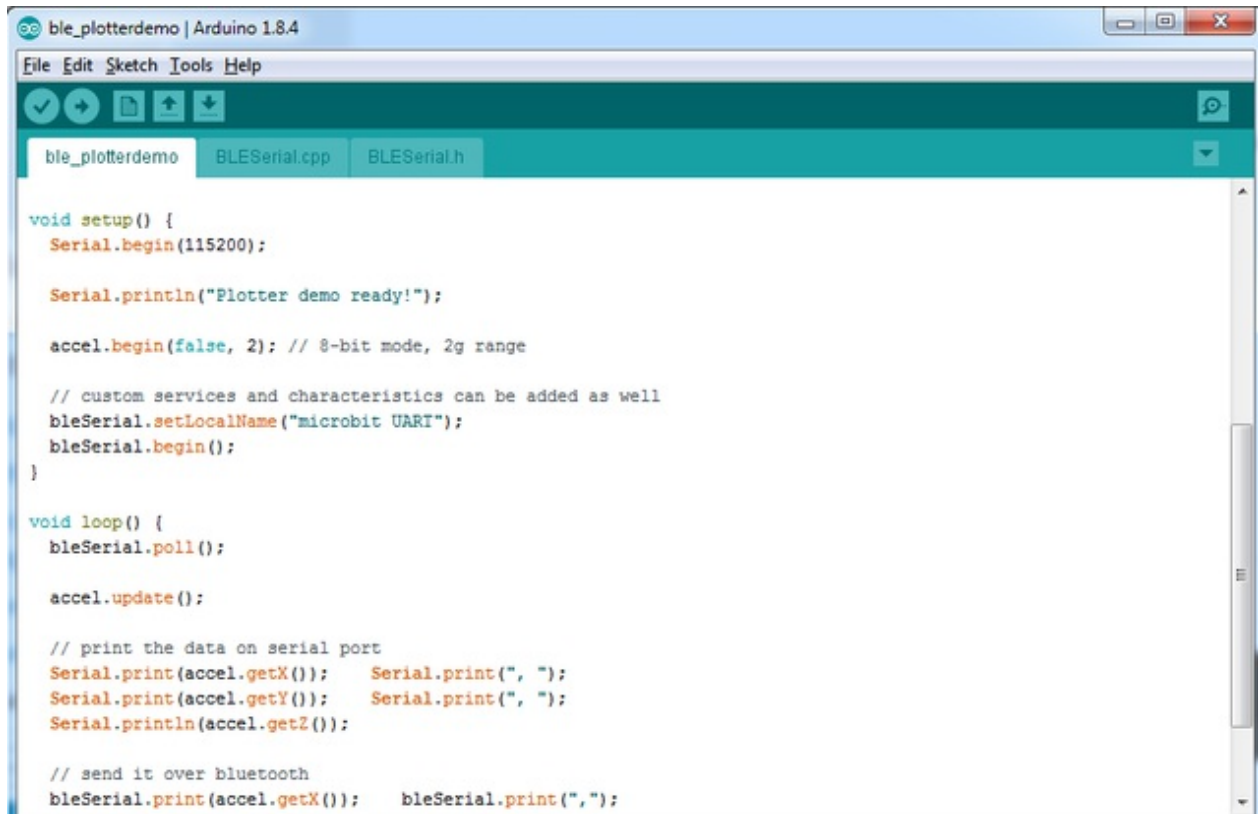
Install Library & Example Code

First up, install the Adafruit helper library (<https://adafru.it/zqF>) and friends

You can find our BLE demos in the examples menu:



Load up the BLE Plotter demo



```
ble_plotterdemo | Arduino 1.8.4
File Edit Sketch Tools Help

ble_plotterdemo BLESerial.cpp BLESerial.h

void setup() {
  Serial.begin(115200);

  Serial.println("Plotter demo ready!");

  accel.begin(false, 2); // 8-bit mode, 2g range

  // custom services and characteristics can be added as well
  bleSerial.setLocalName("microbit UART");
  bleSerial.begin();
}

void loop() {
  bleSerial.poll();

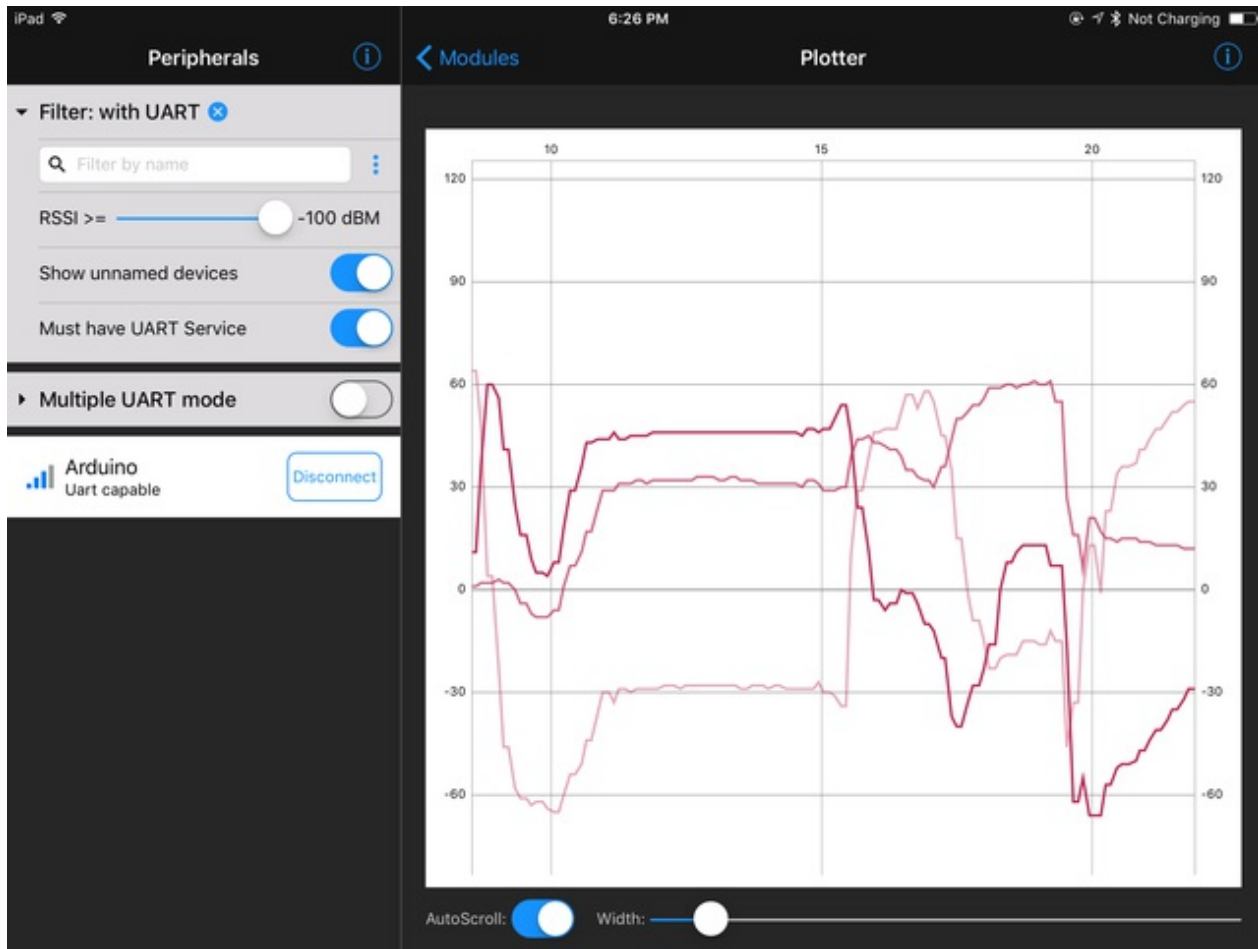
  accel.update();

  // print the data on serial port
  Serial.print(accel.getX());   Serial.print(", ");
  Serial.print(accel.getY());   Serial.print(", ");
  Serial.println(accel.getZ());

  // send it over bluetooth
  bleSerial.print(accel.getX()); bleSerial.print(",");
}
```

This time, in the App, select the **Plotter** module. You will be able to see the X, Y and Z data appear and scroll down!

You can plot anything you like, just use **bleSerial.print()** and print out your data with commas in between. At the end of a data set have a **bleSerial.println()** and it will plot each comma-separated-element as a unique graph



So if you want to just graph the total acceleration vectors $\sqrt{x^2 + y^2 + z^2}$, use this code snippet:

```
void loop() {
  bleSerial.poll();

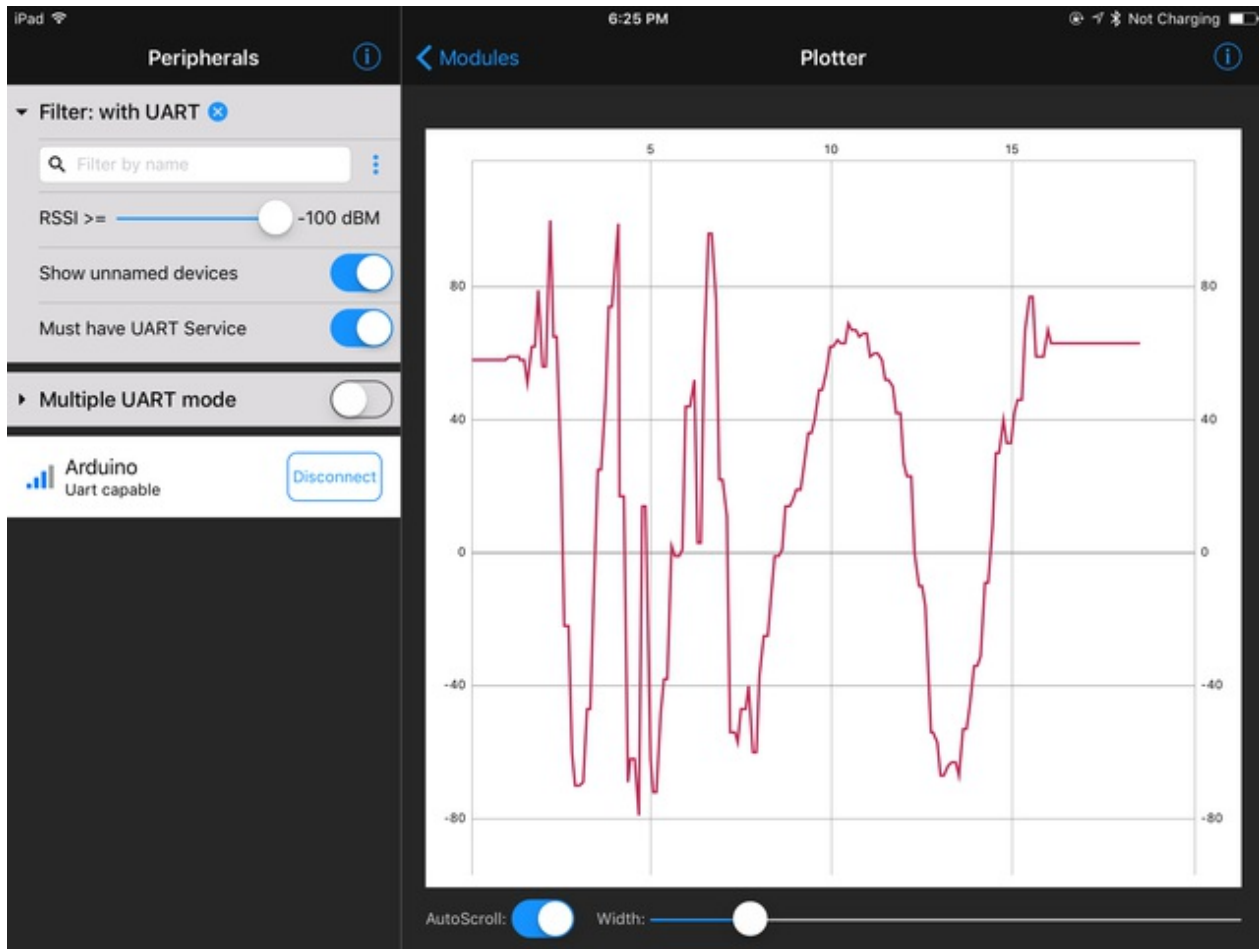
  accel.update();

  // print the data on serial port
  Serial.print(accel.getX()); Serial.print(", ");
  Serial.print(accel.getY()); Serial.print(", ");
  Serial.println(accel.getZ());

  float vector = (accel.getX() * accel.getX()) + (accel.getY() * accel.getY()) + (accel.getZ() * accel.getZ());
  vector = sqrt(vector);

  // send it over bluetooth
  bleSerial.println(vector);

  delay(100);
}
```



Bluetooth Controller

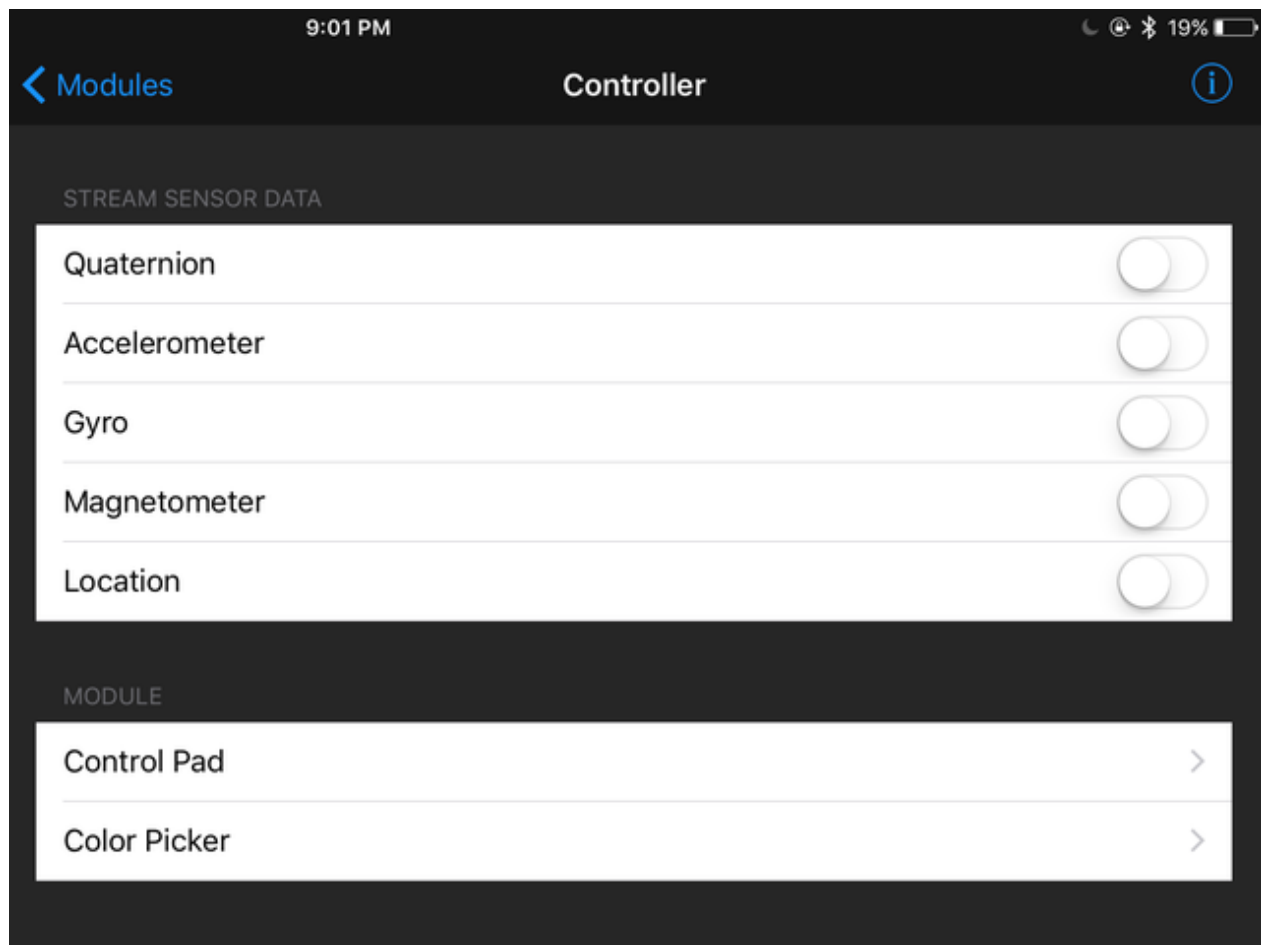
For controlling projects, you may want to use our **Controller** module. It has a bunch of useful interface features that will let you make your next LED or robotics project super easy

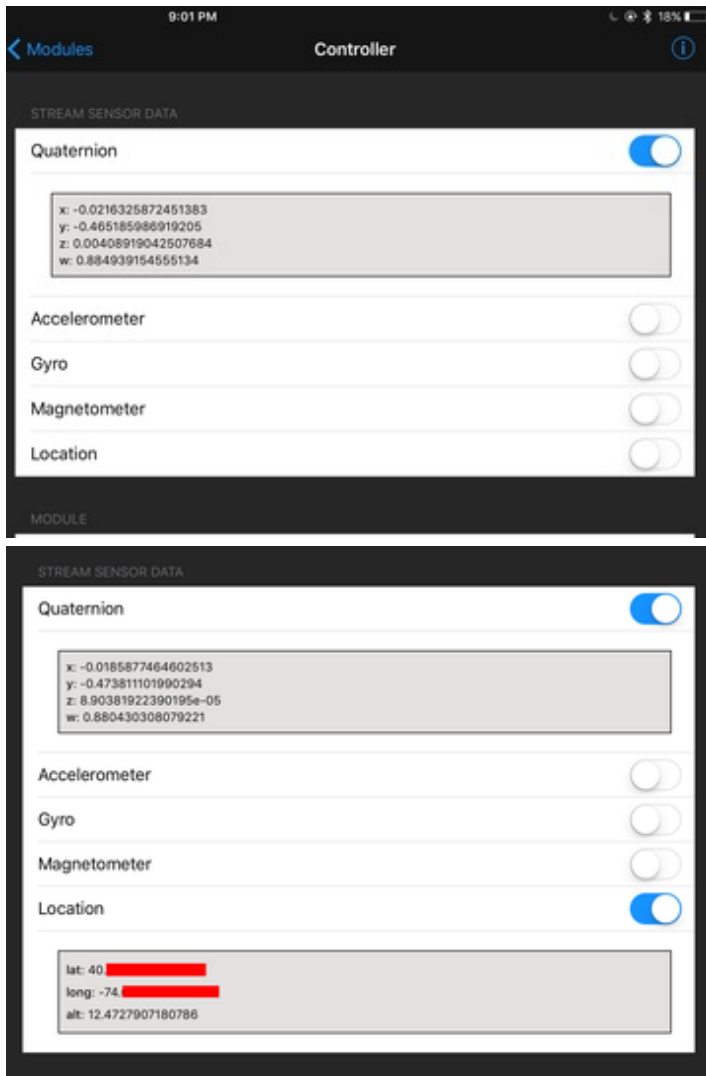
Install Library & Example Code

[Install the Adafruit helper library \(https://adafru.it/zqF\)](https://adafru.it/zqF) and friends

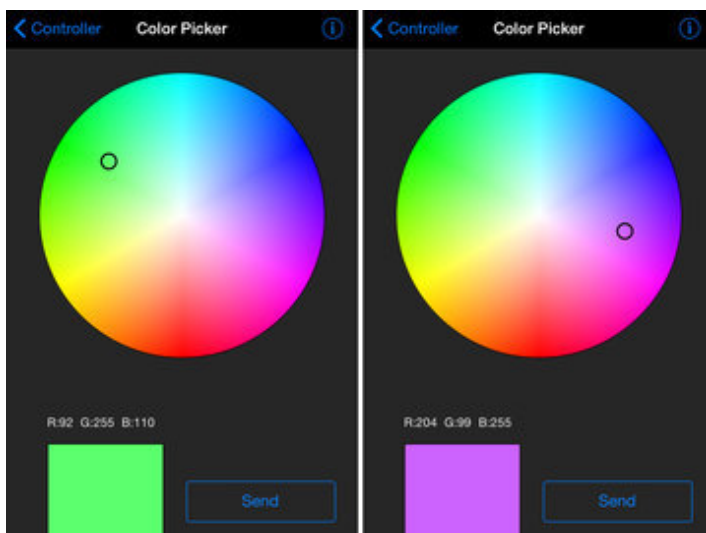
Open up the BLE Controller demo

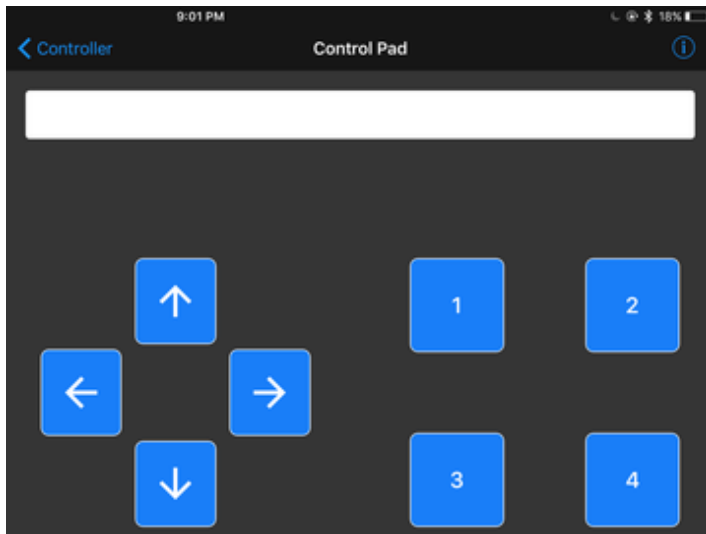
Load that into your microbit, and connect using BLE connect





The two bottom modules can be run whenever you like, click to open up the interfaces:

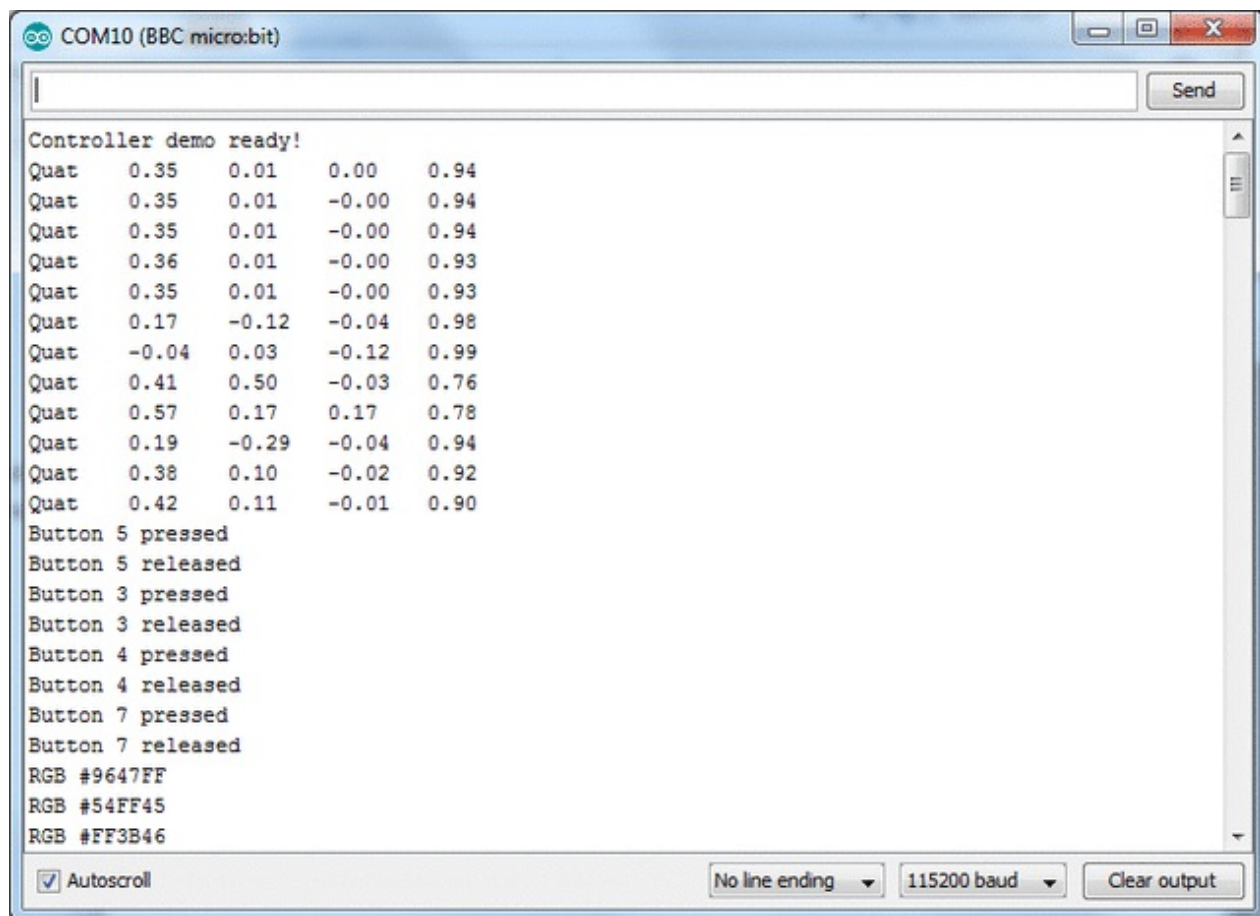




The control pad interface gives you 8 buttons that you can press - each press and release will send a signal to the microbit.

If the microbit sends any data *back* to the device, it will appear in the text bar above.

You can look at the serial monitor to see the messages as they are received.



Logging Temperature to Adafruit IO

All this Bluetooth data stuff is good if you want to plot the data or add control from your phone. But what if you want to store the data long term, or add remote control from around the world?

It's not too hard! We can use Adafruit IO to create graphs and dashboards. And, best of all, it's free just like the Bluefruit app!

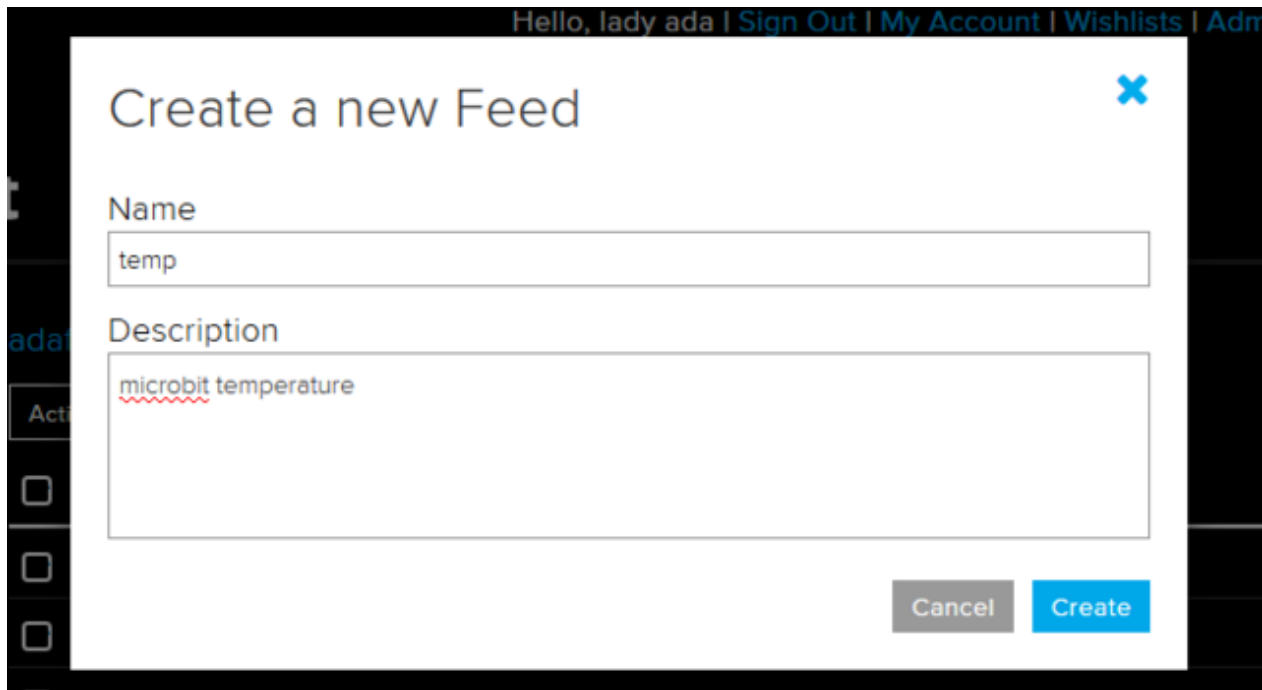
[You can read more about Adafruit IO in this guide](https://adafru.it/nEP)(<https://adafru.it/nEP>)

Before continuing, [set yourself up with an Adafruit IO account](https://adafru.it/fsU)(<https://adafru.it/fsU>)

We won't cover all the details of Adafruit IO here, so check out the guides we have already written for that good stuff!

Create a Microbit Temperature Feed

We'll want a 'place' for our temperature, so create a new feed called **temp**



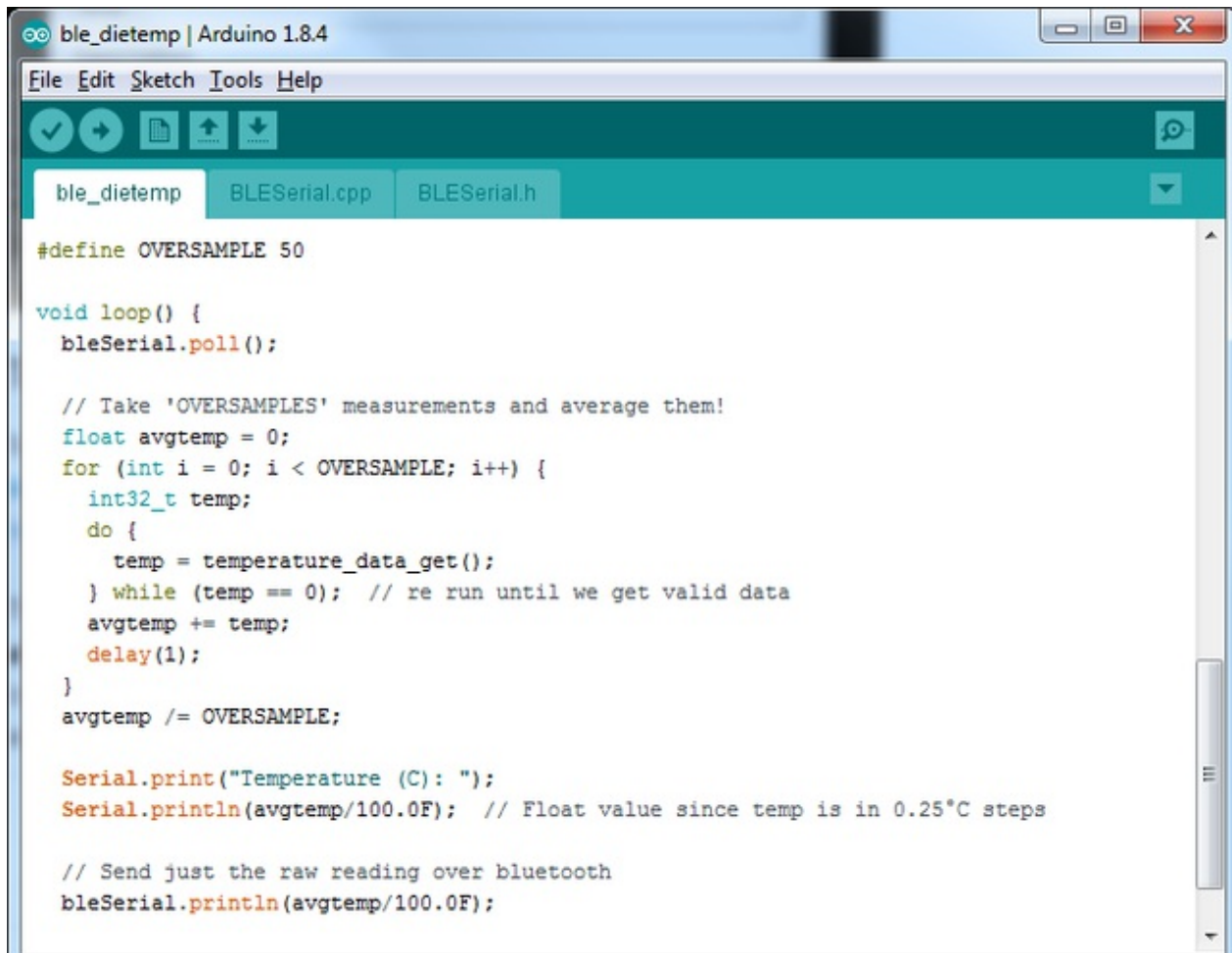
The screenshot shows a web interface for creating a new feed. At the top, there's a navigation bar with links: "Hello, lady ada", "Sign Out", "My Account", "Wishlists", and "Admin". Below this is a modal window titled "Create a new Feed" with a close button (X). Inside the modal, there are two input fields: "Name" with the text "temp" and "Description" with the text "microbit temperature". At the bottom right of the modal are two buttons: "Cancel" (grey) and "Create" (blue).

Temperature Logger Sketch

[Install the Adafruit helper library](https://adafru.it/zqF) (<https://adafru.it/zqF>) and friends

Open up the BLE die temp demo

This will read the temperature on the chip itself. **It's not precise at all** but it does go up when it gets hotter and down when it gets cooler, so its a good place to start and you don't need any additional hardware



```
#define OVERSAMPLE 50

void loop() {
  bleSerial.poll();

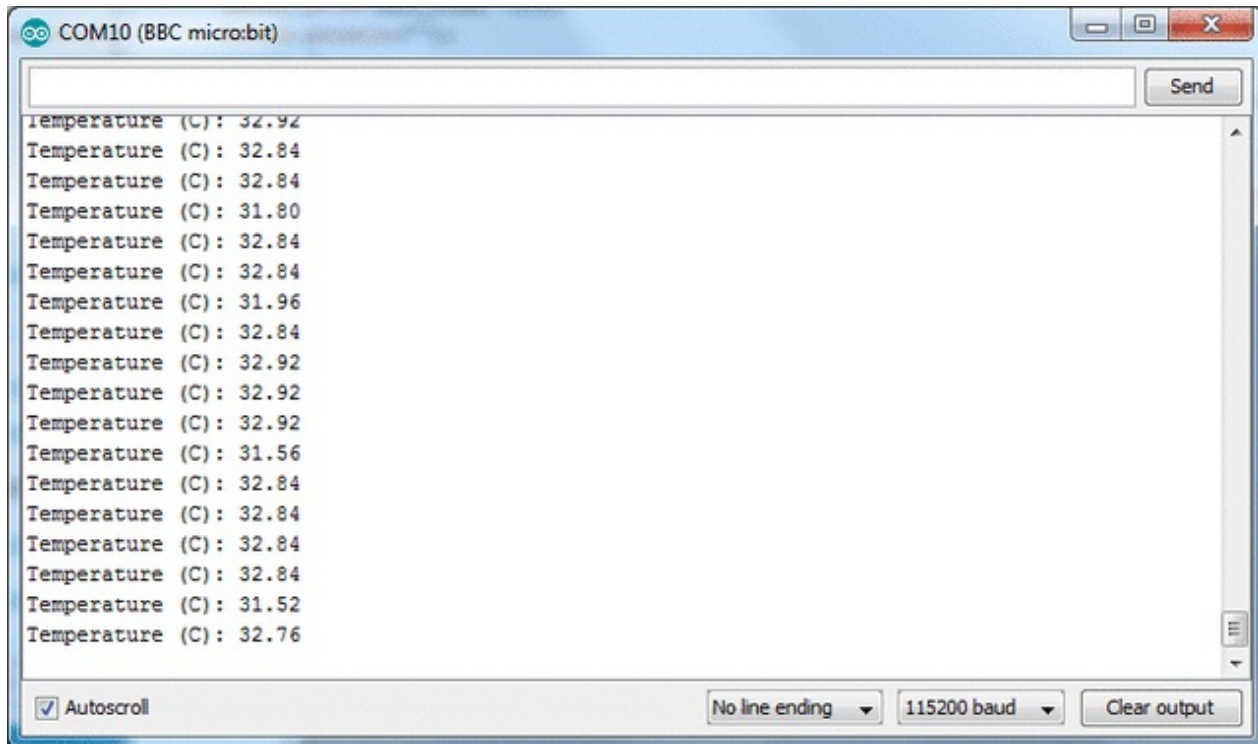
  // Take 'OVERSAMPLES' measurements and average them!
  float avgtemp = 0;
  for (int i = 0; i < OVERSAMPLE; i++) {
    int32_t temp;
    do {
      temp = temperature_data_get();
    } while (temp == 0); // re run until we get valid data
    avgtemp += temp;
    delay(1);
  }
  avgtemp /= OVERSAMPLE;

  Serial.print("Temperature (C): ");
  Serial.println(avgtemp/100.0F); // Float value since temp is in 0.25°C steps

  // Send just the raw reading over bluetooth
  bleSerial.println(avgtemp/100.0F);
}
```

Note that this sketch takes 50 readings and averages it, then waits 5000 ms (5 seconds) between data reports. That's because Adafruit IO is limited in how much data you can upload and store, so we will take it a little slowly.

Upload the sketch and open the serial monitor so you can verify the temperature data there:



Test UART Mode

Connect to the microbit over your device using Adafruit Bluefruit Connect as covered in the previous projects, and select **UART** mode. You should see data slowly coming in

Modules

Uart

MQTT X

i

Display Mode:

Timestamp

Text

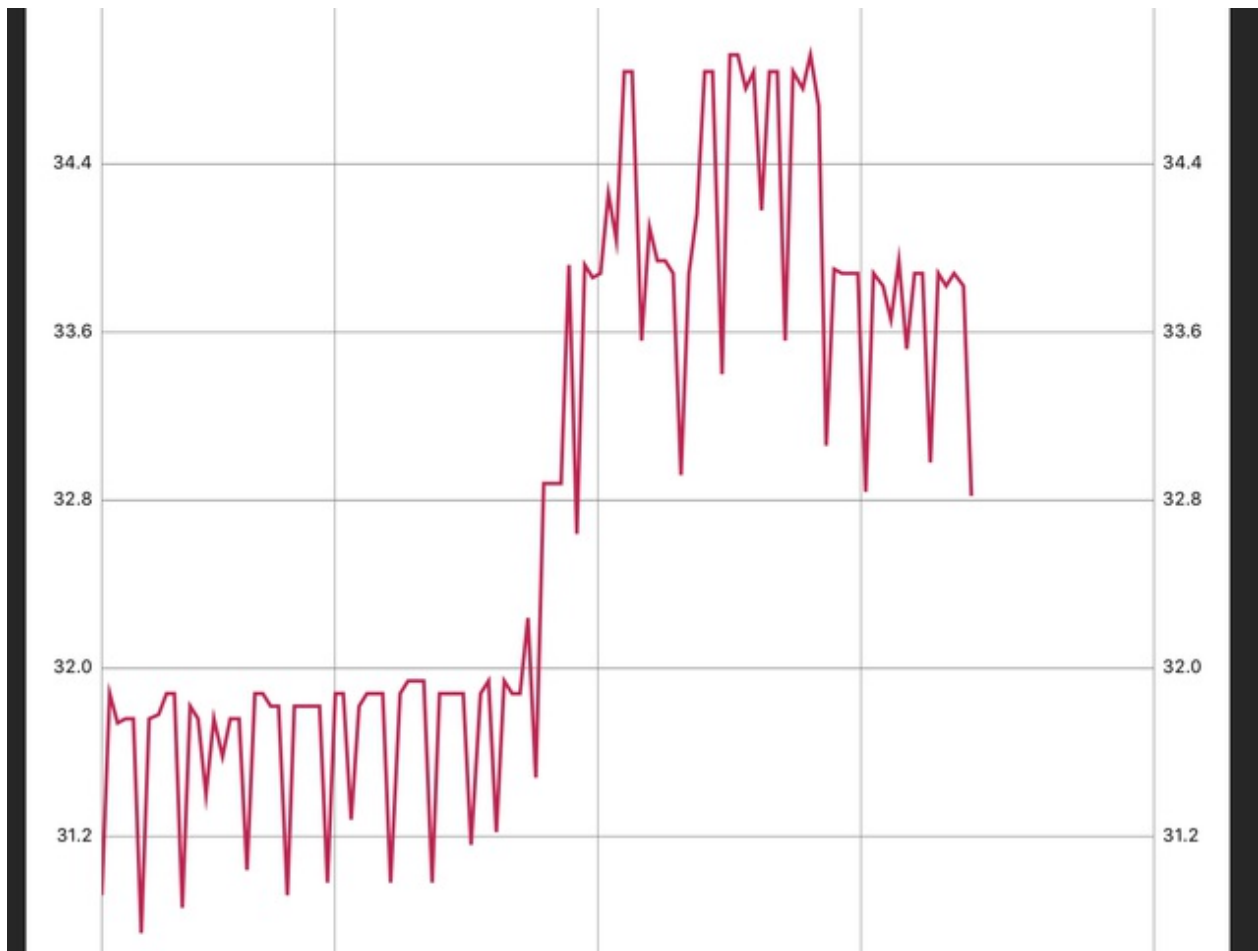
Data Mode:

Ascii

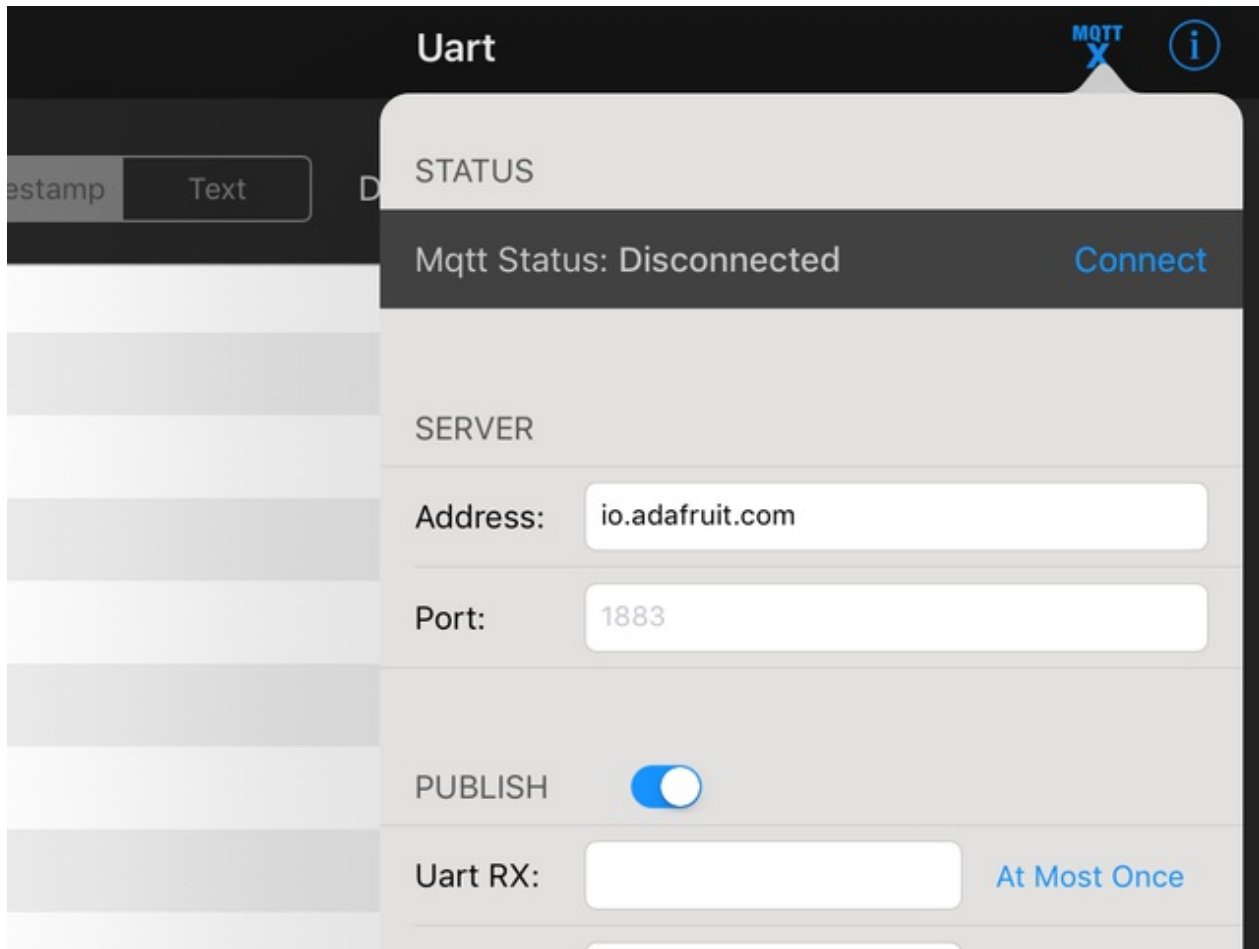
Hex

21:16:44 RX	32.00
21:16:44 RX	32.00
21:16:44 RX	32.00
21:16:44 RX	32.00
21:16:45 RX	32.00
21:16:45 RX	32.00
21:16:45 RX	29.00
21:16:45 RX	32.00
21:16:45 RX	29.00
21:16:45 RX	32.00
21:16:45 RX	32.00
21:16:45 RX	32.00

You can also plot the data. Note that the data is really not very precise or accurate. But if you heat up the nRF51 with a lamp, the temperature will slowly rise up:



Now go back to UART mode and click the **MQTT** button in the top right:



Note that the MQTT server and port will be prefilled for Adafruit IO.

Skip down and enter in your Adafruit IO **Username** and **API Key** (even though it says Password, use the long alphanumeric API key)

The image shows the Adafruit IO web interface. At the top is the Adafruit logo and navigation links: SHOP, BLOG, LEARN, FORUMS, VIDEOS. On the left is a sidebar menu with links: Profile, Feeds, Groups, Dashboards, Triggers, Settings, Guide and Tips, Bug Reports, Suggestions, Add to Forum, API Documentation, Blog/Changelog. The main content area is titled 'uniontownlabs / Settings'. It contains several sections: 'Manage AIO Keys' with a 'VIEW AIO KEY' button; 'Connect Accounts' with a 'Connect to IFTTT' button; 'Manage Account Settings' with a 'Manage Account' button; 'Manage Account Data' with a 'Download All Data' button; and a 'Destroy IO Account and All Data' button. A large pink arrow points from the 'Settings' link in the sidebar to the 'Settings' section header. Another pink arrow points from the 'VIEW AIO KEY' button to a text overlay. A third pink arrow points from the 'Destroy IO Account and All Data' button to another text overlay. A modal window titled 'YOUR AIO KEY' is open, showing instructions on how to use and regenerate the AIO key. It includes a QR code and a text field for the 'Active Key'. A pink arrow points from the 'will appear here' text to the 'Active Key' text field. A 'REGENERATE AIO KEY' button is also visible in the modal.

adafruit

SHOP BLOG LEARN FORUMS VIDEOS

Profile Feeds Groups Dashboards Triggers Settings Guide and Tips Bug Reports Suggestions Add to Forum API Documentation Blog/Changelog

uniontownlabs / Settings

Manage AIO Keys

VIEW AIO KEY

Connect Accounts

Connect to IFTTT

Manage Account Settings

Account Settings includes your time zone, name, email, password

Manage Account

Manage Account Data

Download the stored content of your Adafruit IO account

Download All Data

Destroy IO Account and All Data

YOUR AIO KEY

Your Adafruit IO key should be kept in a safe place and treated with the same care as your Adafruit username and password. Your AIO key can view all of your data, create new feeds, and manipulate your active feeds.

If you need to regenerate a new AIO key, all of your existing programs and scripts will need to be manually changed to the new key.

Active Key

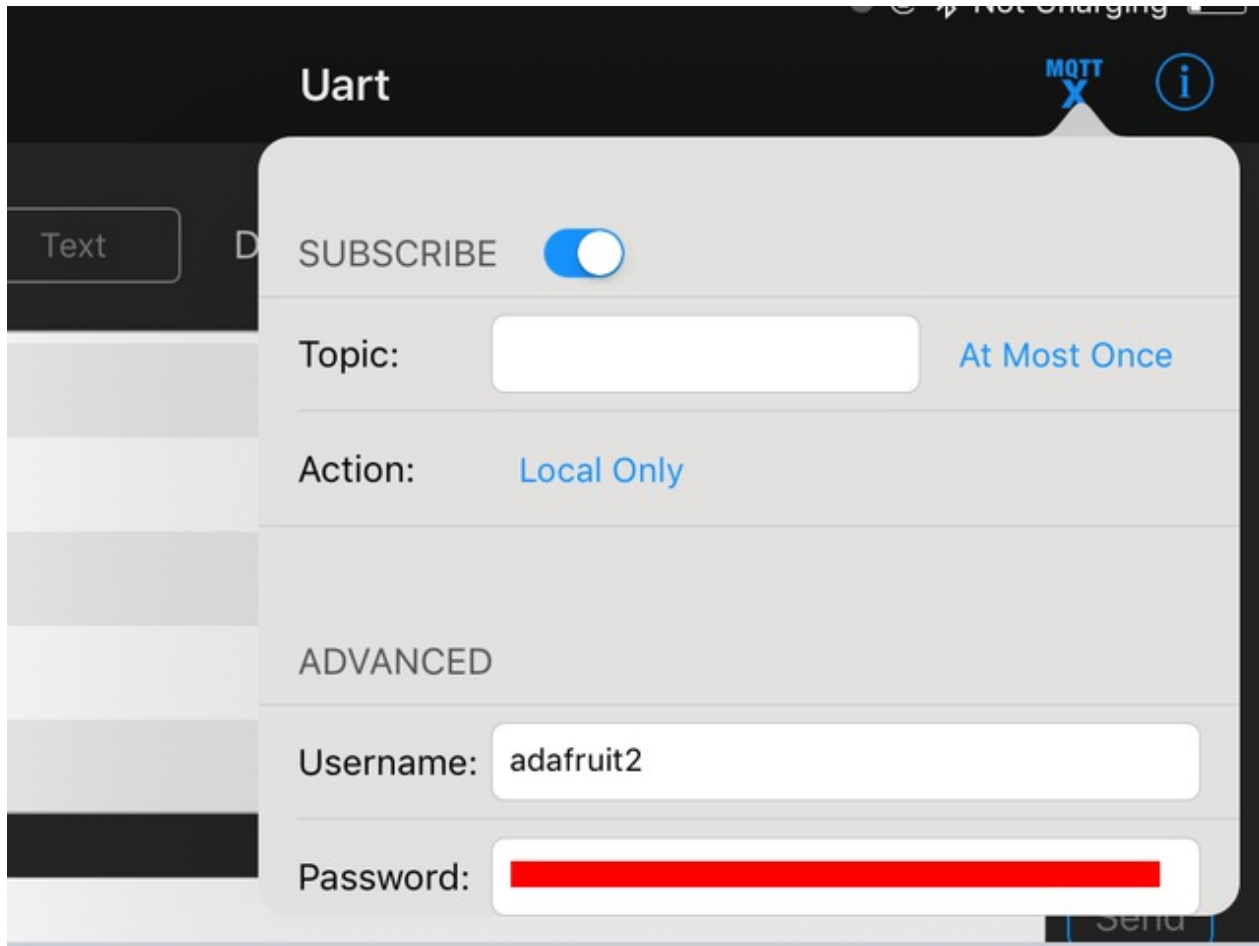
REGENERATE AIO KEY

click settings

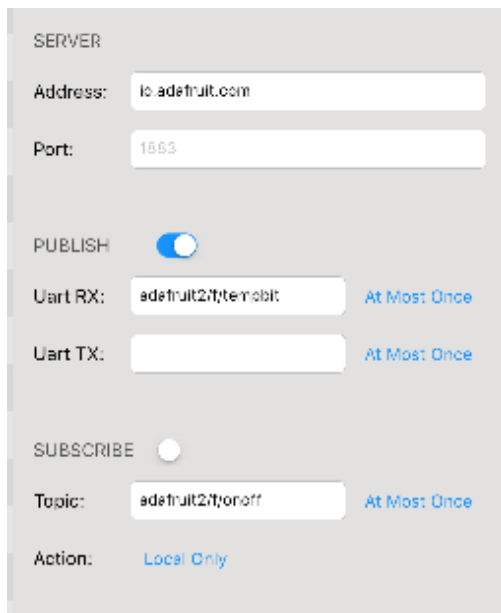
click to get AIO Key

your AIO key will appear here

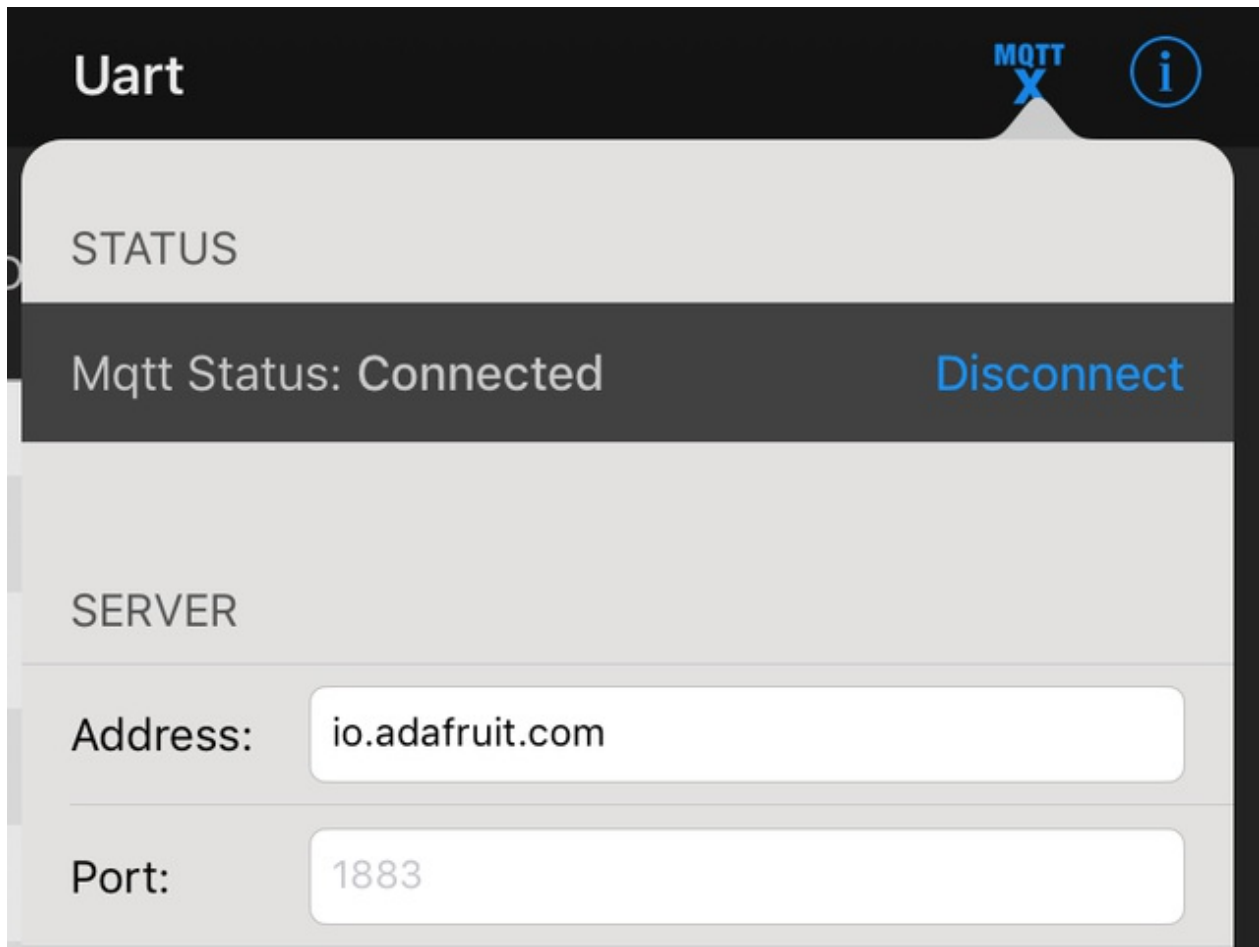
Then take that and put it here like so:



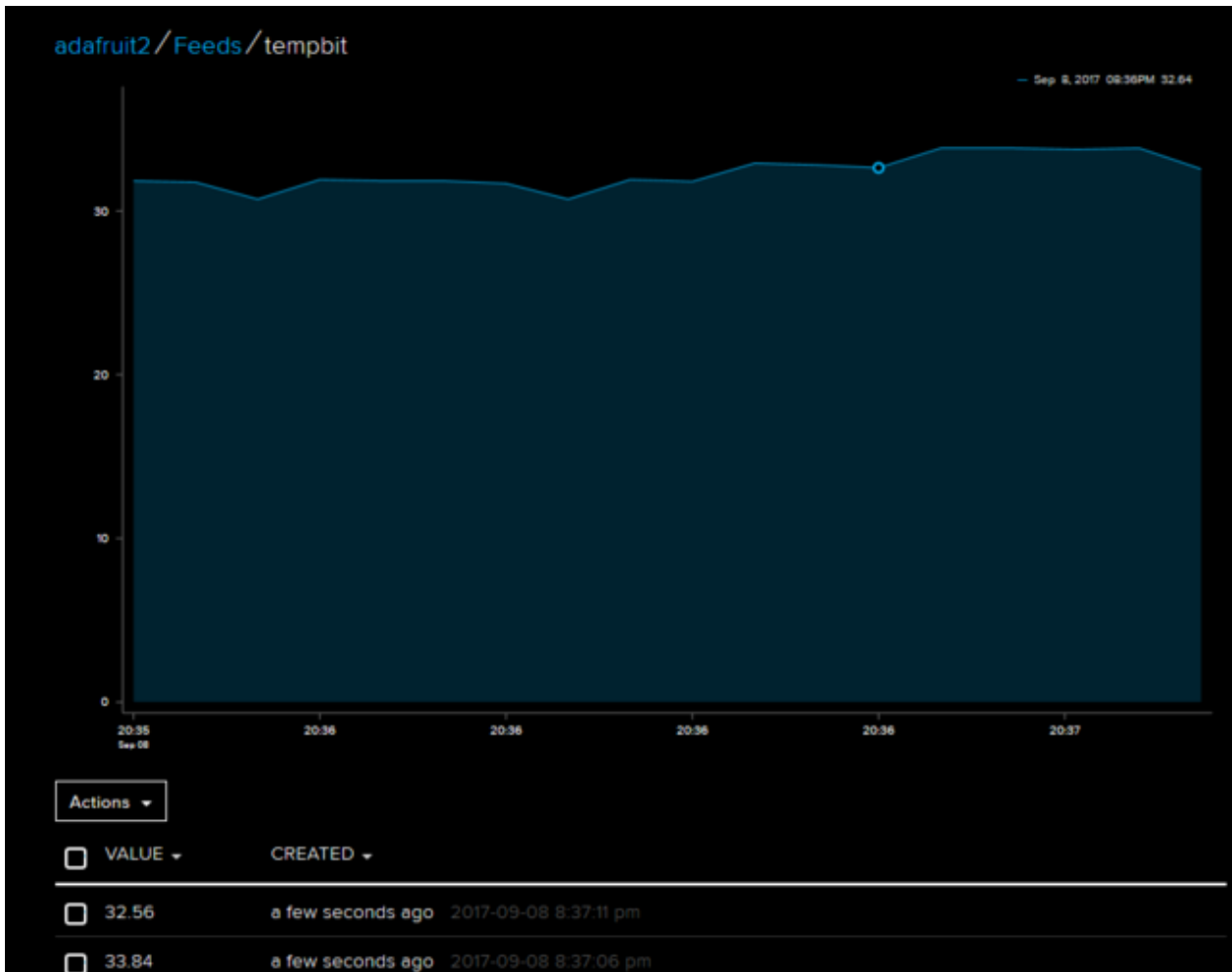
Finally enter in the feed name which is **username/f/tempbit** into the UART RX Publish entry. (There's currently a bug where you have to have something in the Subscribe so we put the same feed in there):



Then click **Connect** at the top:



Wait a few minutes and go visit your Adafruit IO Feeds page, you should see the data start to stream in!



Huzzah! [You can now create a public dashboard if you like, to share it with others \(https://adafru.it/z8e\)](https://adafru.it/z8e)