Azure IOT Suite with Adafruit Feather M0

<http://adafru.it/3031>

<http://azure.com/iotstarterkits>

Contents

[Introduction 4](#_Toc452063210)

[Prerequisites. 4](#_Toc452063211)

[Part A. Getting the Adafruit Feather Ready 4](#_Toc452063212)

[Install Arduino IDE 4](#_Toc452063213)

[Configure Arduino development environment 4](#_Toc452063214)

[Install requisite board packages 6](#_Toc452063215)

[Install Feather library. 8](#_Toc452063216)

[Initial Adafruit Feather Connectivity Test 10](#_Toc452063217)

[Test Adafruit Feather connectivity 10](#_Toc452063218)

[Check, Test and Validate Wi-Fi Connectivity 14](#_Toc452063219)

[Test Wi-Fi Firmware connectivity and version. 14](#_Toc452063220)

[Test Wi-Fi Scan 15](#_Toc452063221)

[Test Wifi Connectivity and download web page. 16](#_Toc452063222)

[Part B: Connecting with BME280 18](#_Toc452063223)

[Install requisite libraries 18](#_Toc452063224)

[Adafruit Feather and BME 280 Hardware connection 19](#_Toc452063225)

[Test BME connection and read Telemetry data 20](#_Toc452063226)

[Part C: Connecting with Microsoft Remote Monitoring IOT Suite 22](#_Toc452063227)

[Provision Azure IoT Suite 22](#_Toc452063228)

[Register Telemetry Device 25](#_Toc452063229)

[Transmit Metadata and Telemetry messages to IoT Hub 27](#_Toc452063230)

[Install required libraries 27](#_Toc452063231)

[Appendix 30](#_Toc452063232)

[Components of Microsoft Azure IoT Starter Kits 30](#_Toc452063233)

[USB Connectivity Issues with Adafruit Feather 31](#_Toc452063234)

Figures

[Figure 1:Arduino IDE Preferences 5](#_Toc452063235)

[Figure 2: Arduino Boards Manager Configuration 6](#_Toc452063236)

[Figure 3:Install Arduino Boards Package - 1 6](#_Toc452063237)

[Figure 4: Install Arduino Board Package - 2 7](#_Toc452063238)

[Figure 5:Arduino IDE USB Port Setup 8](#_Toc452063239)

[Figure 6: Installing Adafruit Board Drivers 9](#_Toc452063240)

[Figure 7: Arduino IDE USB Connection 10](#_Toc452063241)

[Figure 8: Arduino Reset from Hung State 11](#_Toc452063242)

[Figure 9: IDE Output window messages - successful compilation and install 12](#_Toc452063243)

[Figure 10: Feather USB Connection and Blink Output 13](#_Toc452063244)

[Figure 11: Arduino library folder structure 14](#_Toc452063245)

[Figure 12: Wi-Fi Firmware check Program 15](#_Toc452063246)

[Figure 13: Wifi Scan Results 16](#_Toc452063247)

[Figure 14: Wi-Fi Successful connection output 17](#_Toc452063248)

[Figure 15: Install Adafruit Unified Sensor Library 18](#_Toc452063249)

[Figure 16: BME280 Pin Connection - 1 19](#_Toc452063250)

[Figure 17: BME 280 Pin Connections - 2 20](#_Toc452063251)

[Figure 18: Successful BME280 Connectivity Test Output 21](#_Toc452063252)

[Figure 19: AzureIoTSuite.Com Create 22](#_Toc452063253)

[Figure 20: Azure IoT Suire Remote Monitoring Solution create 23](#_Toc452063254)

[Figure 21: Enter Azure IoT Suite Details 23](#_Toc452063255)

[Figure 22: Launch Solution Dashboard Portal 24](#_Toc452063256)

[Figure 23: Admin Dashboard Portal 25](#_Toc452063257)

[Figure 24: Register Device, Step 2 of 3 26](#_Toc452063258)

[Figure 25: Registered Device Credentials 26](#_Toc452063259)

[Figure 26: Registered Device Status 27](#_Toc452063260)

[Figure 27: remote\_monitoring SDK error display 28](#_Toc452063261)

[Figure 28:Successful Transmission of Telemetry Data from device 29](#_Toc452063262)

[Figure 29: Proof is in the pudding 29](#_Toc452063263)

[Figure 30: Upload Debug Preferences 31](#_Toc452063264)

[Figure 31: Port search during Upload 32](#_Toc452063265)

[Figure 32: Arduino and COM port connectivity 33](#_Toc452063266)

[Figure 33: Arduino Library Structure 33](#_Toc452063267)

# Introduction

For those who are not familiar, I would recommend that you visit <http://AzureIoTSuite.com> about information on what services the suite offers. This document assumes that the reader is aware of Microsoft Azure and understands the various services that are provided by it.

The Microsoft Azure IoT Start Kit for Adafruit has the ‘[Adafruit Feather M0 WiFi - ATSAMD21 + ATWINC1500](https://www.adafruit.com/products/3010)’ version of Adafruit development board as core component. The core component, Adafruit Feather, effectively is a light weight board having a micro-controller with Wi-Fi built in. Board has a Cortex M0+ processor (ATSAMD21G18 ARM Cortex M0 processor). 12-bit ADC (Analog to Digital conversion). 10-bit DAC (digital to analog conversion). The Wi-Fi module is from Amtel, 802.11bgn-capable.

The aim of this mini project is for the Adafruit Feather to communicate with to the Microsoft Azure IoT Hub. Adafruit Feather should be able to send telemetry messages and also respond to commands sent to it by the IoT hub.

The core of the documentation is already available [here](https://azure.microsoft.com/en-us/documentation/samples/iot-hub-c-m0wifi-getstartedkit/) but this document describes the various workarounds that were implemented for the entire solution to work. In fact, this document builds on that [information](https://azure.microsoft.com/en-us/documentation/samples/iot-hub-c-m0wifi-getstartedkit/) that is already provided by Microsoft

## Prerequisites.

1. You need to have an Azure subscription.
   1. If you don’t have Azure subscription, you can get a [free one-time free trial subscription](https://azure.microsoft.com/en-us/pricing/free-trial/) for a month which provides you with $200 worth of credit.
2. It is assumed that you have [Microsoft Azure IoT Starter Kits](https://www.adafruit.com/products/3031) which has Adafruit Feather MO Wifi and other rest of components.

In the first phase of this project we’ll ensure that the our IoT device is configured correctly, is correctly connected and can send and receive messages. In the second part of this project we’ll configure the Azure IoT Suite to send. And finally we’ll connect the IoT device to the Azure IoT Suite and see its messages being transmitted to the Azure IoT Hub.

# Part A. Getting the Adafruit Feather Ready

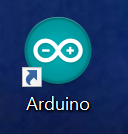
## Install Arduino IDE

In order for us to test and communicate with Adafruit Feather, we’ll first need to install the Arduino IDE on the development machine.

* You can download the installer application from [here](https://www.arduino.cc/en/Main/Software). The version used during this project was 1.6.9. The installer will install USB drivers onto the development machine. Click ‘yes’ to all the confirmation messages that come-up during the install.

## Configure Arduino development environment

1. Start the Arduino IDE (It should’ve created a short-cut at the desktop during the install)



1. File 🡪Preferences 🡪 ‘Preferences’ dialog box shown.
2. Enter the following values in the ‘**Additional Board Manager URLs:**’ field

<https://adafruit.github.io/arduino-board-index/package_adafruit_index.json>

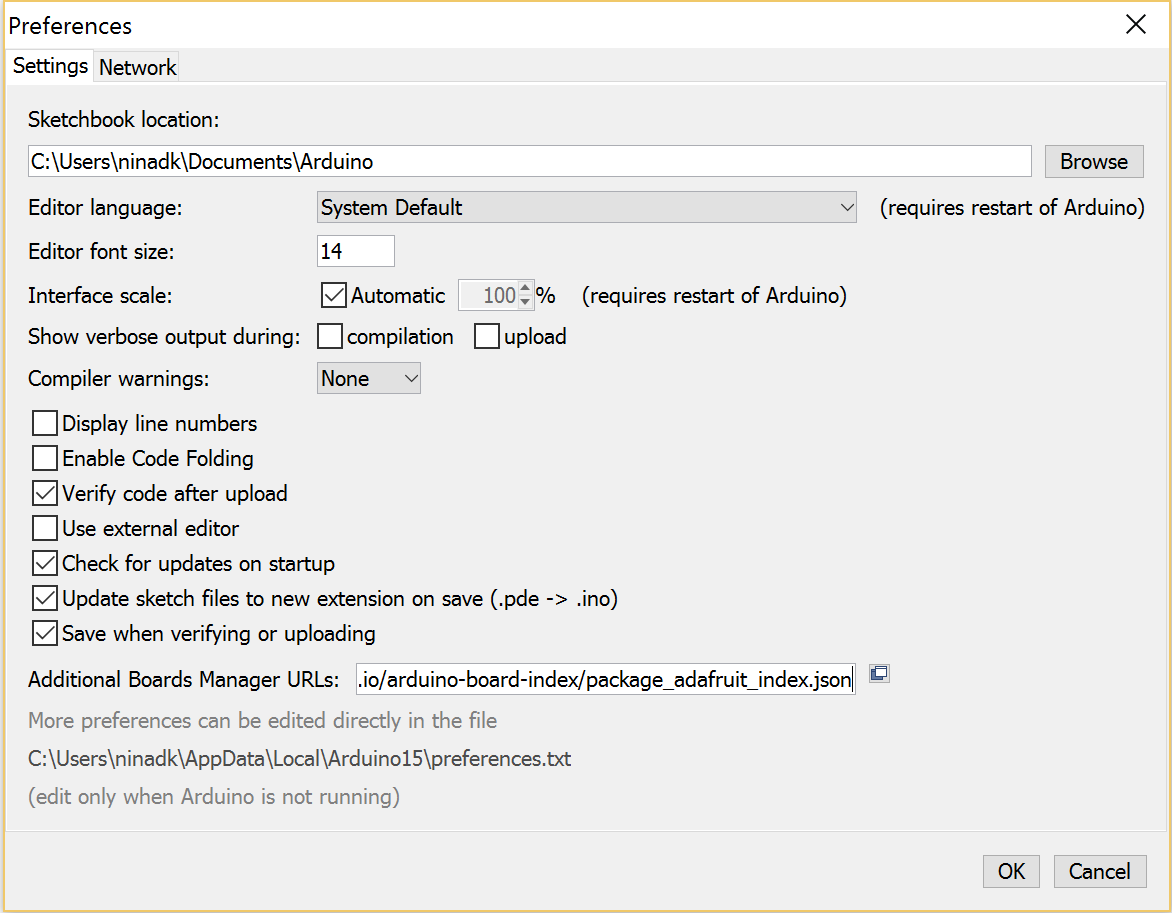


Figure :Arduino IDE Preferences

1. Click OK to save the settings
2. Ensure that correct board is now set-up within the IDE (as shown below)

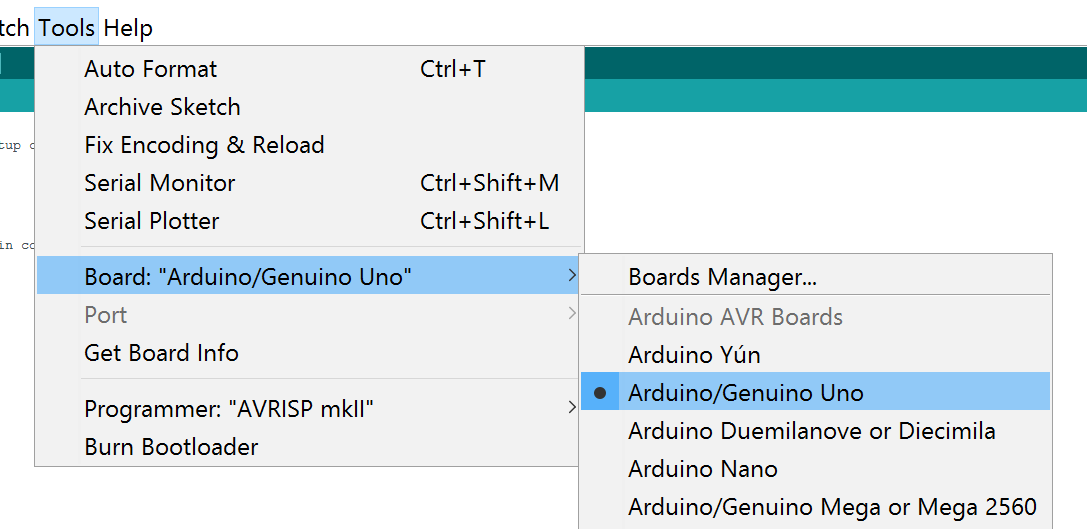


Figure : Arduino Boards Manager Configuration

1. [If the display settings are such that you are struggling to view/read IDE properly, change the ‘**Interface scale:**’ value.
2. File 🡪Preferences 🡪 ‘Preferences’ dialog box shown. Uncheck the ‘**Automatic**’ check box and set the value to 200%. You’ll need to re-start the IDE for the changes to take effect.

## Install requisite board packages

1. Open Boards Manager (Tools 🡪 Board: “Arduino/Genuine Uno” 🡪 Board Manager).

Board Manager dialog box will be shown.

1. Select and install the **Arduino SAMD Boards** version **1.6.2 or later and click Install.**

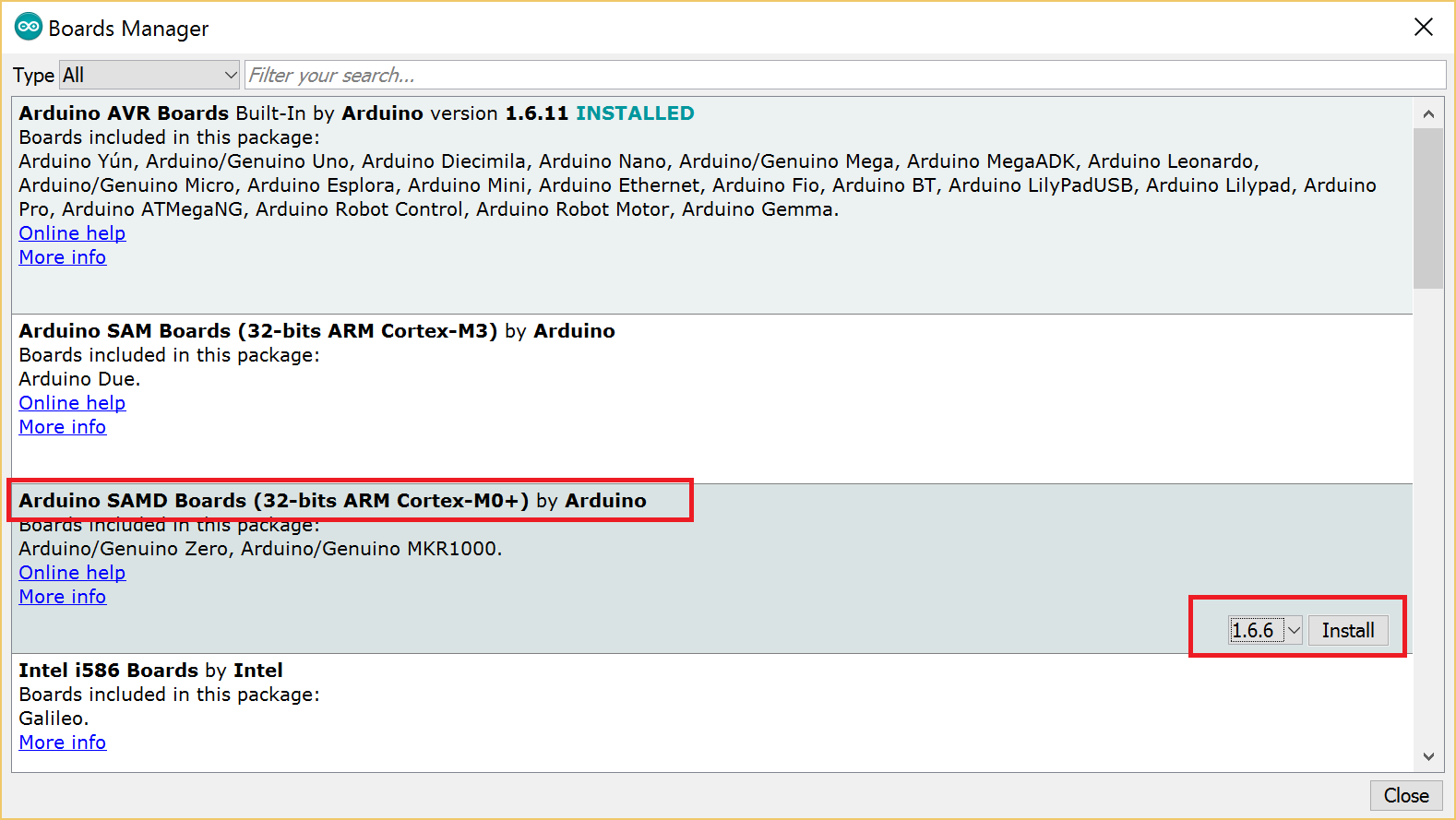
****

Figure :Install Arduino Boards Package - 1

1. Once the installation is finished, scroll down and find the package - **Adafruit SAMD Boards** package. Install the package.

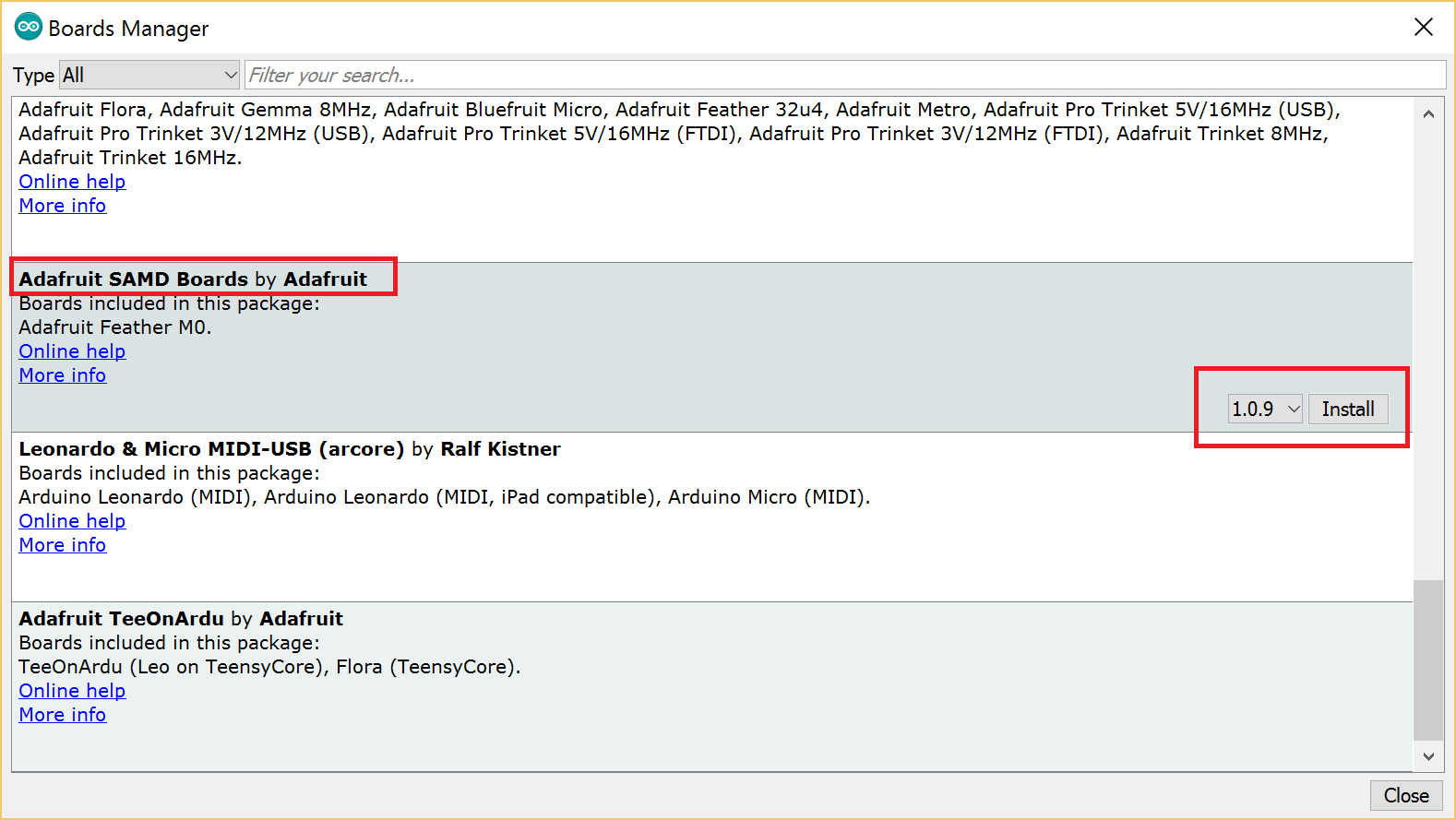


Figure : Install Arduino Board Package - 2

1. Once both the board package installation is finished, restart the IDE for the changes to take effect.
2. Open the IDE once again. You should now be able to select and upload to the new boards listed in the **Tools->Board** menu

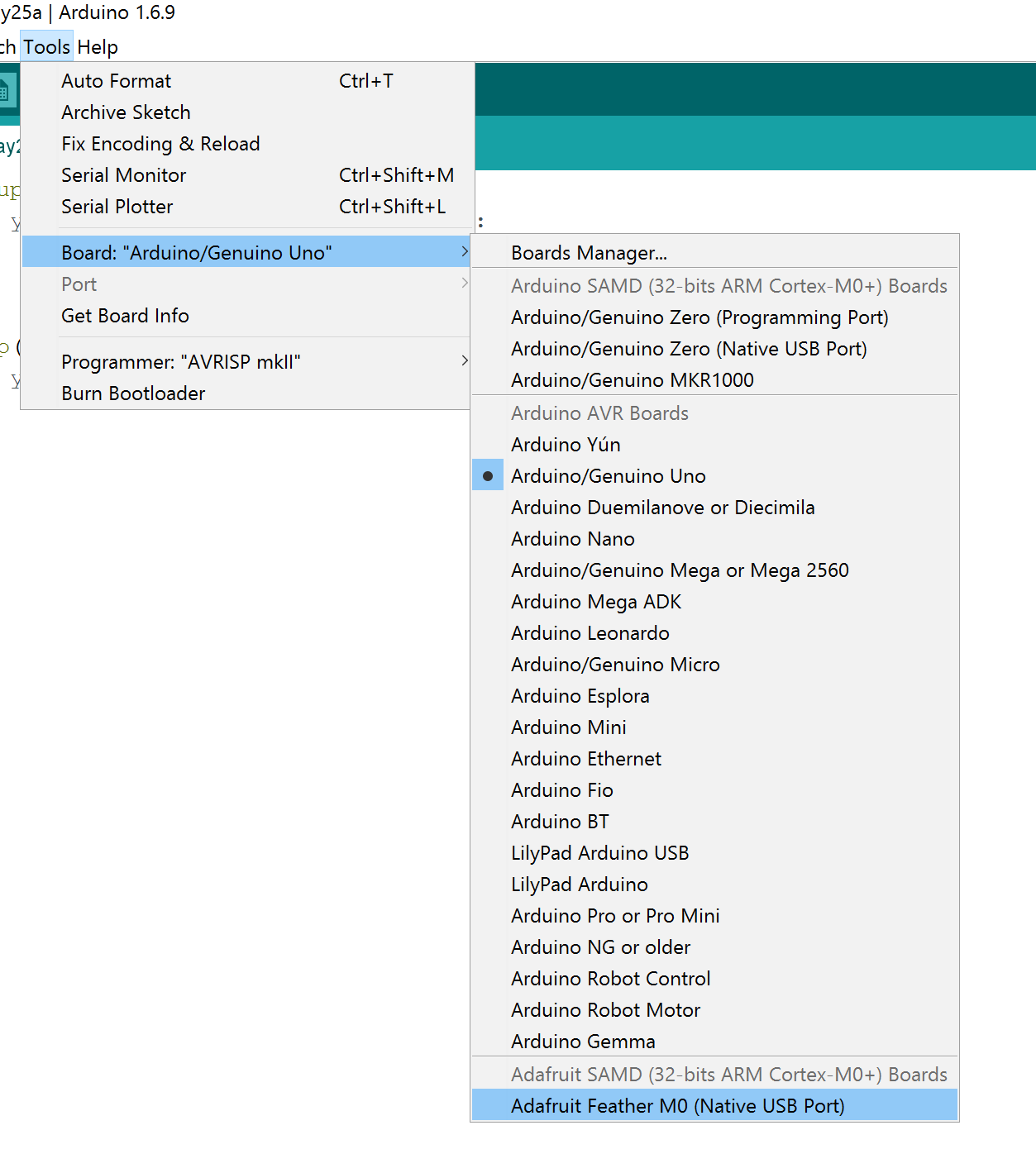


Figure :Arduino IDE USB Port Setup

## Install Feather library.

1. Download the installer from [here](https://github.com/adafruit/Adafruit_Windows_Drivers/releases/download/1.0.0.0/adafruit_drivers.exe). Run the downloaded executable.
2. Agree on the licensing terms dialog box
3. Select the libraries that you want to install as shown below

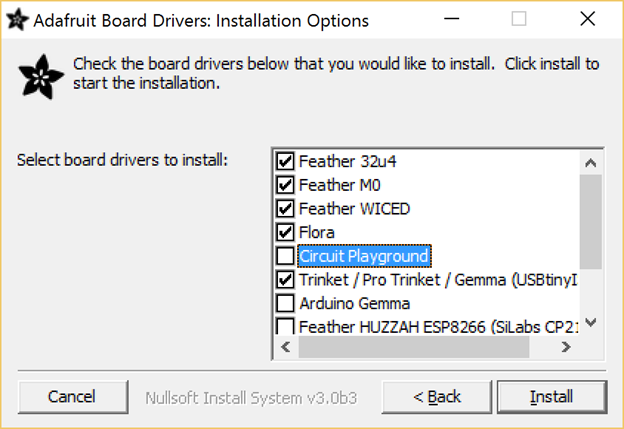


Figure : Installing Adafruit Board Drivers

1. During installation, click ‘Install’ on all of the Windows Security dialog boxes that come up.

Installations complete, it’s now time to test the Feather and its connectivity.

## Initial Adafruit Feather Connectivity Test

For this stage, we require following components required from the IoT Starter Kit

* 1x Assembled [Adafruit Feather M0 WiFi](https://www.adafruit.com/product/3010) w/ [Feather Stacking Headers](https://www.adafruit.com/product/2830)
* 1x [USB Cable - A/Micro B](https://www.adafruit.com/product/592)
* 1x [Full-sized Breadboard](https://www.adafruit.com/product/239)

### Test Adafruit Feather connectivity

1. Connect the USB Cable, one end to the USB port of development machine and other end (Micro USB connector) to the Adafruit Feather – see the connection figure here
2. If not already open, open an instance of Arduino IDE. The USB-Feather connection should now be visible under the IDE - as shown below.

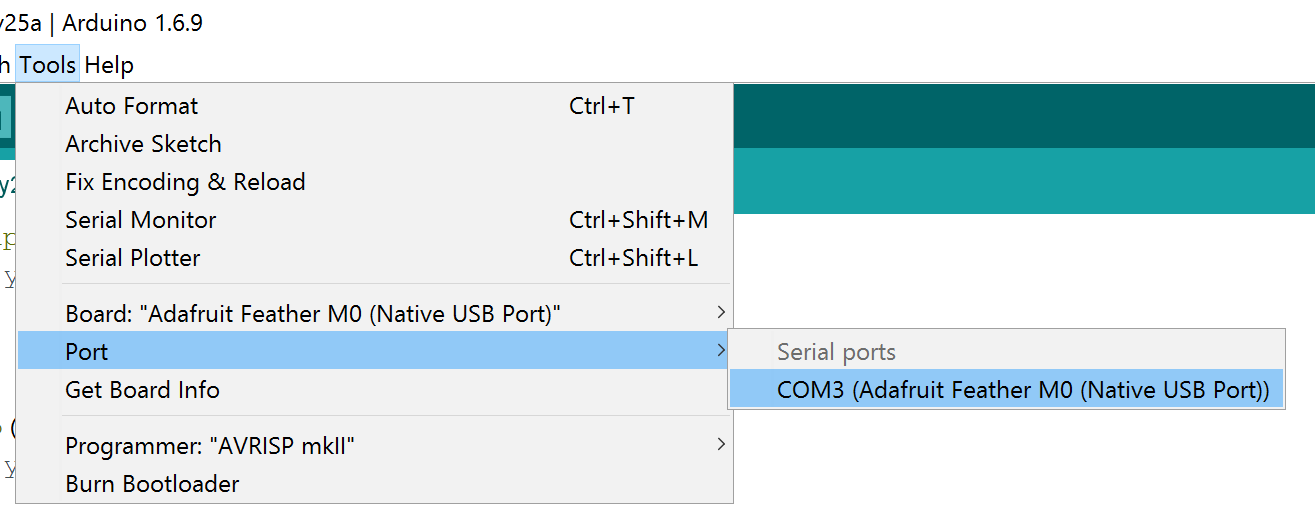


Figure : Arduino IDE USB Connection

1. Select the COM port {in above case COM3} to connect the IDE to Feather.
2. Load the following sketch[[1]](#footnote-1) inside the IDE – replacing the existing blank content that is currently shown. You can also load the follsing sketch by; File 🡪 Examples 🡪01.Basics 🡪Blink

// the setup function runs once when you press reset or power the board

void setup() {

// initialize digital pin 13 as an output.

pinMode(13, OUTPUT);

}

// the loop function runs over and over again forever

void loop() {

digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)

delay(1000); // wait for a second

digitalWrite(13, LOW); // turn the LED off by making the voltage LOW

delay(1000); // wait for a second

}

1. Upload[[2]](#footnote-2) the program. To Upload, either use the shortcut key (CTRL+U) or from the menu (sketch 🡪 Upload)
   1. IDE might want to ask you a location to save the existing sketch. Select a local folder and give ‘Blink’ as the name to the sketch. Keep in mind that the project containing single files are saved with extension ‘.ino’ and a folder of same name as the sketch name is created by default.
   2. Whilst Uploading, you might find that the Upload process is hung/stuck – indicated by “Uploading” message being continuously displayed at the bottom of the IDE. Press the Reset button twice on the Feather for it to come out of the hung state.
      1. For the exact sequence on how this should be done see the Appendix [section](#_USB_Connectivity_Issues) of this document
   3. Wait few seconds and start the Upload process again from the IDE.
      1. Double-clicking the Reset button puts the Feather into the Bootloader mode. Single click of the reset button puts the Feather into program launch mode.

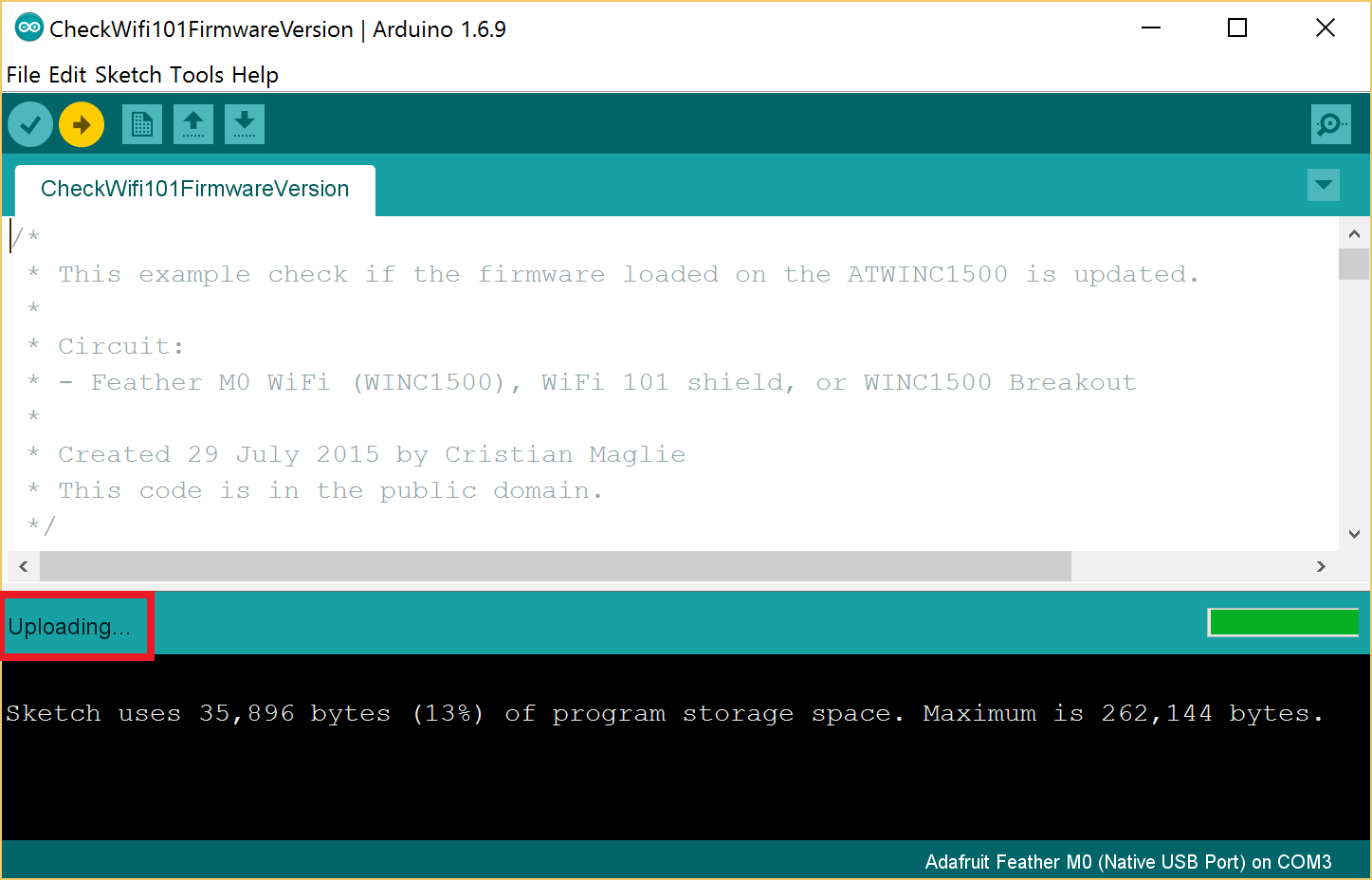


Figure : Arduino Reset from Hung State

1. Once the program has successfully been transferred, you’ll get the following confirmation messages displayed within the Output window of the Arduino IDE.

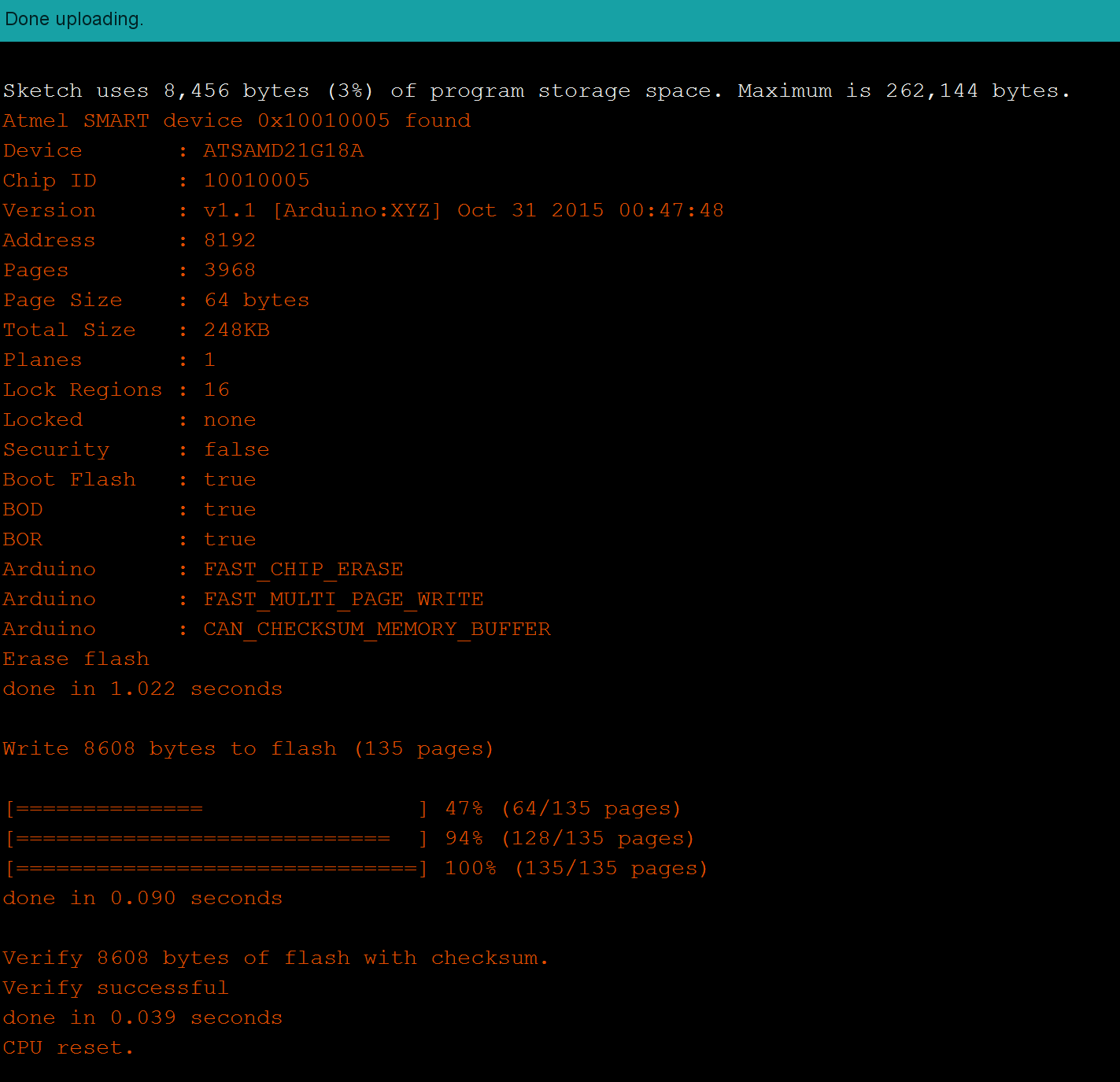


Figure : IDE Output window messages - successful compilation and install

1. The Red-LED next to the USB connector will also flash.
2. The sketch will start executing. You should not see the Red light flash on-off

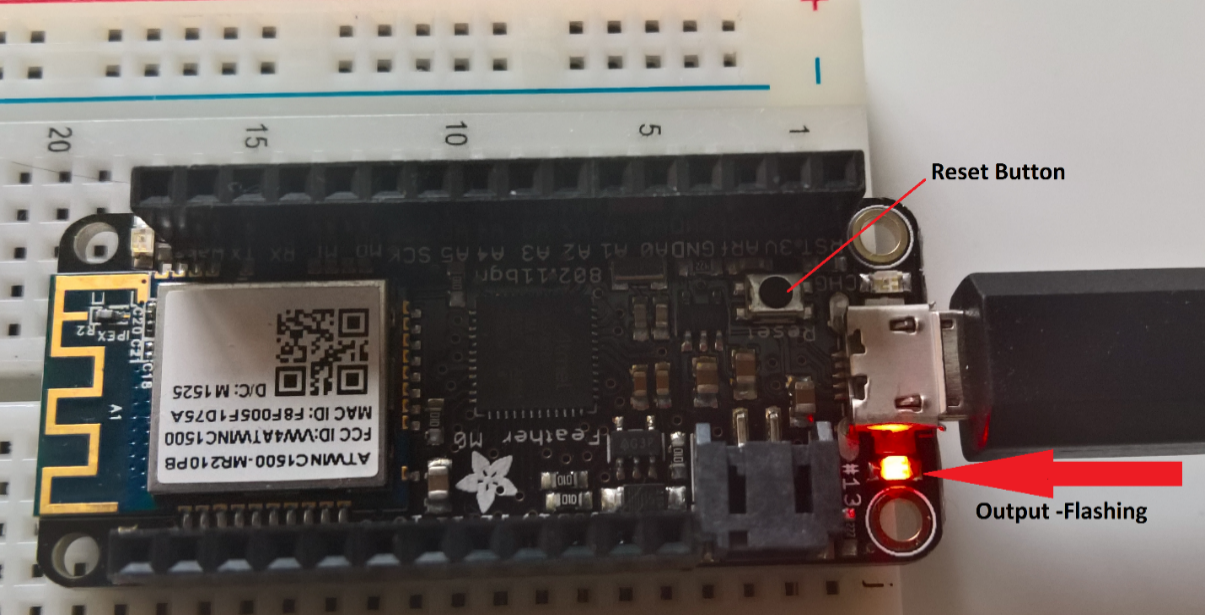


Figure : Feather USB Connection and Blink Output

1. Congratulations – You’ve successfully connected, compiled, uploaded and executed your first program on Arduino.

## Check, Test and Validate Wi-Fi Connectivity

Next, we’ll test the connectivity of the WIFI chip on the Feather

1. Download the latest Adafruit\_WINC1500 zip files from [here](https://github.com/adafruit/Adafruit_WINC1500/archive/master.zip)
2. Extract the content of the zipped files. Rename the uncompressed Zip folder as Adafruit\_WINC1500. Check that the Adafruit\_WINC1500 folder contains a folder named **src** and examples and a file named **library.properties.** 
   1. More information about Arduino library structure can be found [here](https://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use)
3. Copy the **Adafruit\_WINC1500** library folder to your **sketchbookfolder\libraries\** folder – by default it’s the **C:\Users\{username}\Documents\Arduino\libraries**. You may need to create the libraries subfolder if it’s your first library. The resulting folder structure should look like following

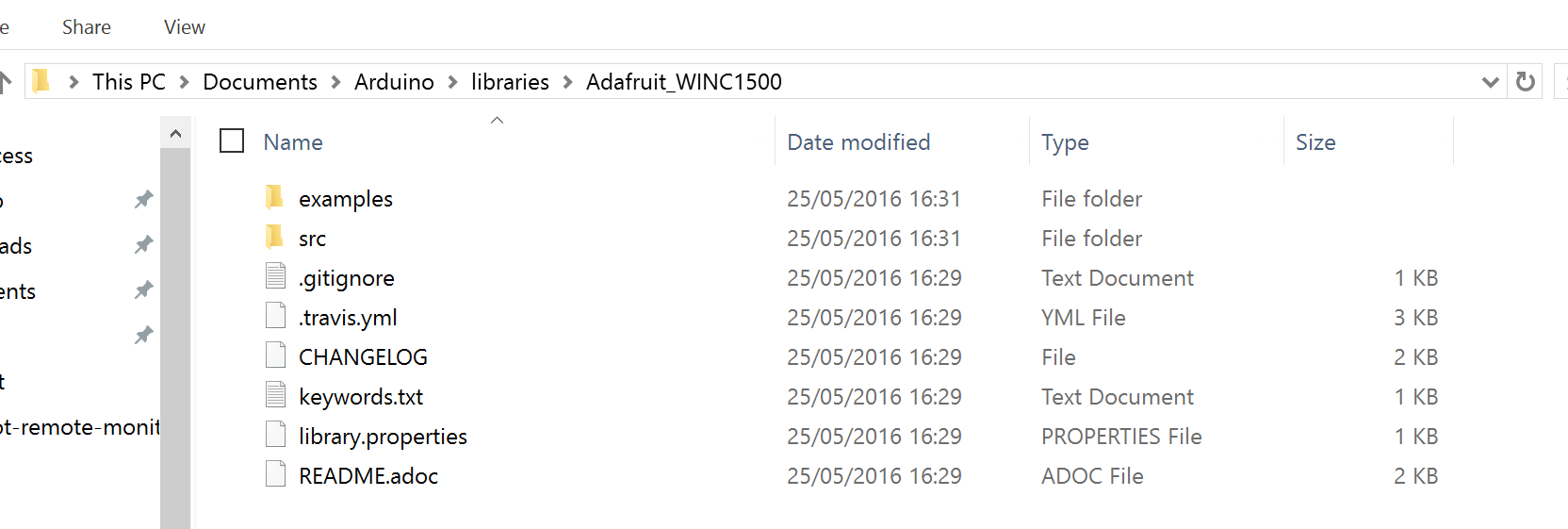


Figure : Arduino library folder structure

1. Restart the Arduino IDE.

### Test Wi-Fi Firmware connectivity and version.

1. Open the Arduino IDE.
2. Navigate to WIFI library and open the ‘**CheckWifi101Firmware’ sketch. (File 🡪 Examples 🡪 Adafruit\_WINC1500 🡪 CheckWifi101Firmware)**
3. Original IDE will open the sketch in another instance of the IDE. Switch over to the IDE where the ‘**CheckWifi101Firmware’** sketch is open
4. Ensure that COM port connectivity is established.
5. the sketch (sketch 🡪 Upload)
6. The sketch will not run because the code is waiting for the Serial Monitor[[3]](#footnote-3) window to be connected.
7. Open the Serial Monitor window. (Tools 🡪 Serial Monitor)
8. You should get the following display to indicate that the Wi-Fi Firmware and update has worked



Figure : Wi-Fi Firmware check Program

Note:

* If you have version 19.3 or less, the firmware is too old
* If you get not response, the firmware is either way to old, or something is amiss with your wiring!

### Test Wi-Fi Scan

This test will scan for the Wifi connections available and display the results available

1. Load the sketch ‘’. (**File** 🡪 **Examples** 🡪 **Adafruit\_WINC1500->ScanNetworks)**
2. **Upload the sketch to the Feather.**
3. **Start the Serial Monitor. It should display the result of scan for various Wi-Fi networks in the neighborhood.**

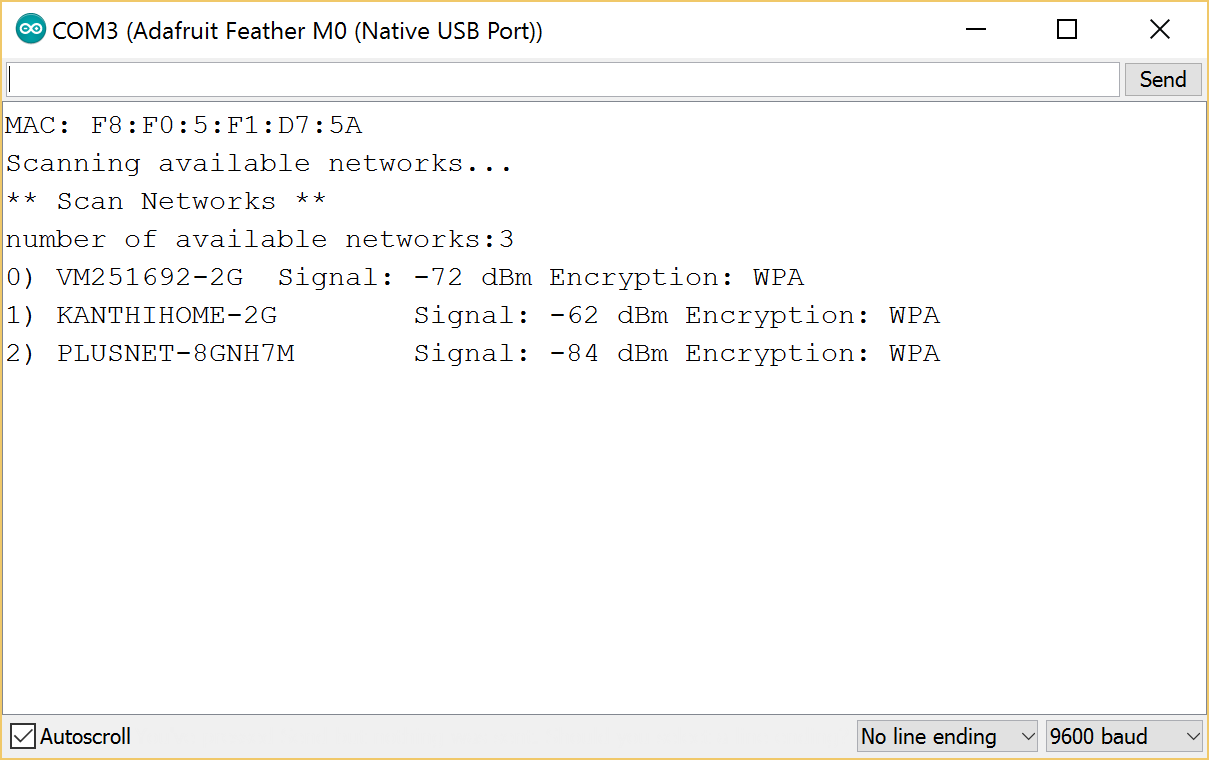


Figure : Wifi Scan Results

And now finally we’ll connect with the Wifi network using the Adafruit Feather and download a page from the website.

### Test Wifi Connectivity and download web page.

1. Open sketch (**WiFi101WebClient**) inside Arduino IDE. (**File** 🡪**Examples** 🡪 **Adafruit\_WINC1500->WiFiWebClient**)
2. Switch to the Arduino IDE which has for the ‘**WiFiWebClient**’ sketch open.
3. Edit the **ssid** and **pass** variables to contain your network and password

//Adafruit\_WINC1500 WiFi;

char ssid[] = "yournetwork"; // your network SSID (name)

char pass[] = "yourpassword"; // your network password (use for WPA, or use as key for WEP)

1. Upload the sketch to the Adafruit Feather.
2. Once the sketch has been successfully uploaded, open the Serial Monitor. You should get following output –

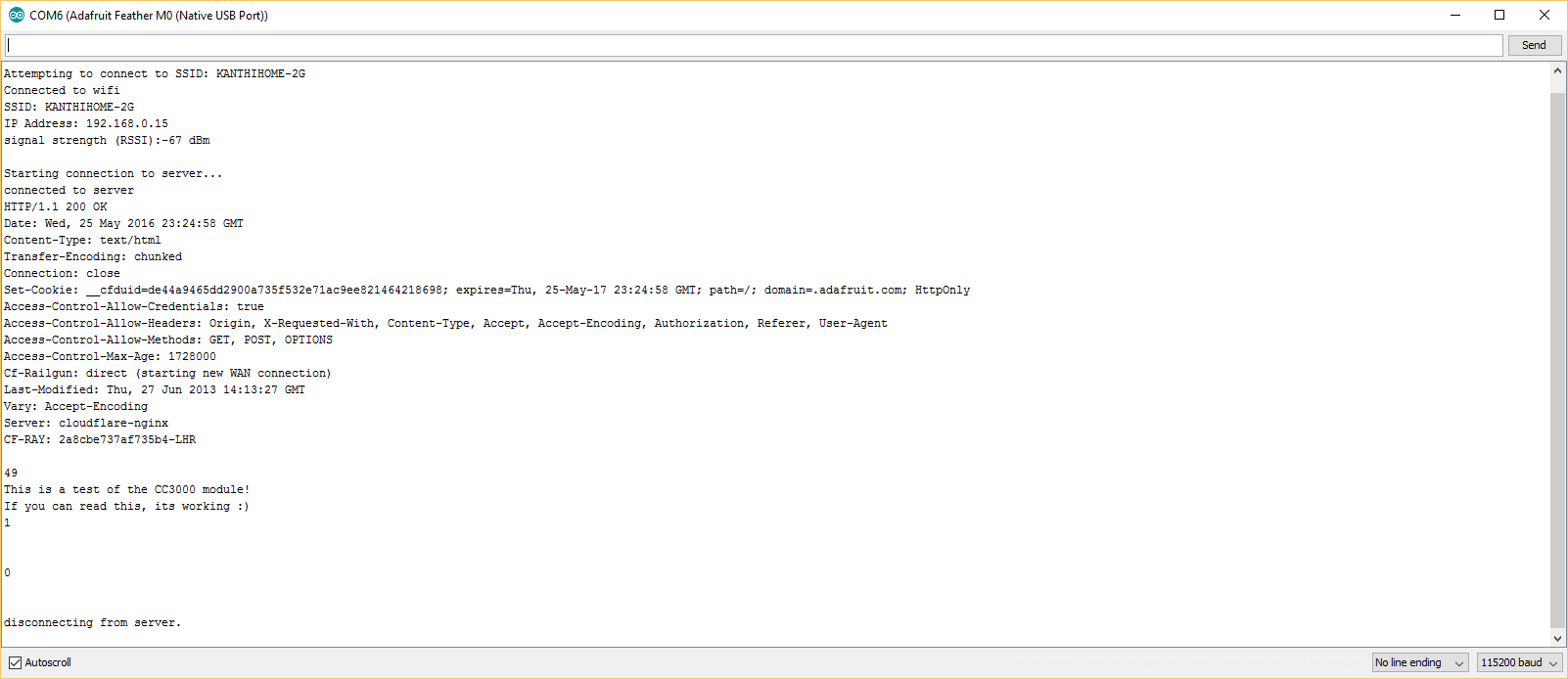


Figure : Wi-Fi Successful connection output

1. – indicating that Wi-Fi connection and download from external web site has been successful.

This concludes our connectivity tests with Wi-Fi. Next we will install the BMR280 chip on the board and run the preliminary connectivity tests with it.

# Part B: Connecting with BME280

Now that we have established the connectivity tests with Adafruit Feather in a standalone mode, next step is to plug-in the Temperature/Humidity measuring chip that comes as part of the Microsoft Azure IoT Starter Kit – the [BME 280 Sensor](https://www.adafruit.com/product/2652).

The components from that kit that you require for this phase of testing are

* 1x Assembled [BME280 I2C or SPI Temperature/Humidity/Pressure Sensor](https://www.adafruit.com/product/2652)
* 1x [Premium Male/Male Jumper Wires - 20 x 3"](https://www.adafruit.com/product/1956)

## Install requisite libraries

1. Download the Adafruit\_BME280 library from [here](https://github.com/adafruit/Adafruit_BME280_Library)
2. Extract the content of the zipped files. Rename the uncompressed Zip folder as **Adafruit\_BME280**. Check that the **Adafruit\_BME280** folder contains **Adafruit\_BME280.cpp** and **Adafruit\_BME280.h files**.
3. Copy the **Adafruit\_BME280** library folder your **sketchbookfolder\libraries\** folder – by default it’s the **C:\Users\{username}\Documents\Arduino\libraries**.
   1. An example of what the upload should look like was mentioned in the earlier [section](#Arduino_Library_Structure) of this document.
4. Install AdaFruit Sensor library (Sketch 🡪Include Library 🡪Manage Libraries).
   1. Search for ‘sensor’ in the Topics field and select ‘**Adafruit Unified Sensor**’ from the list of entries as shown below

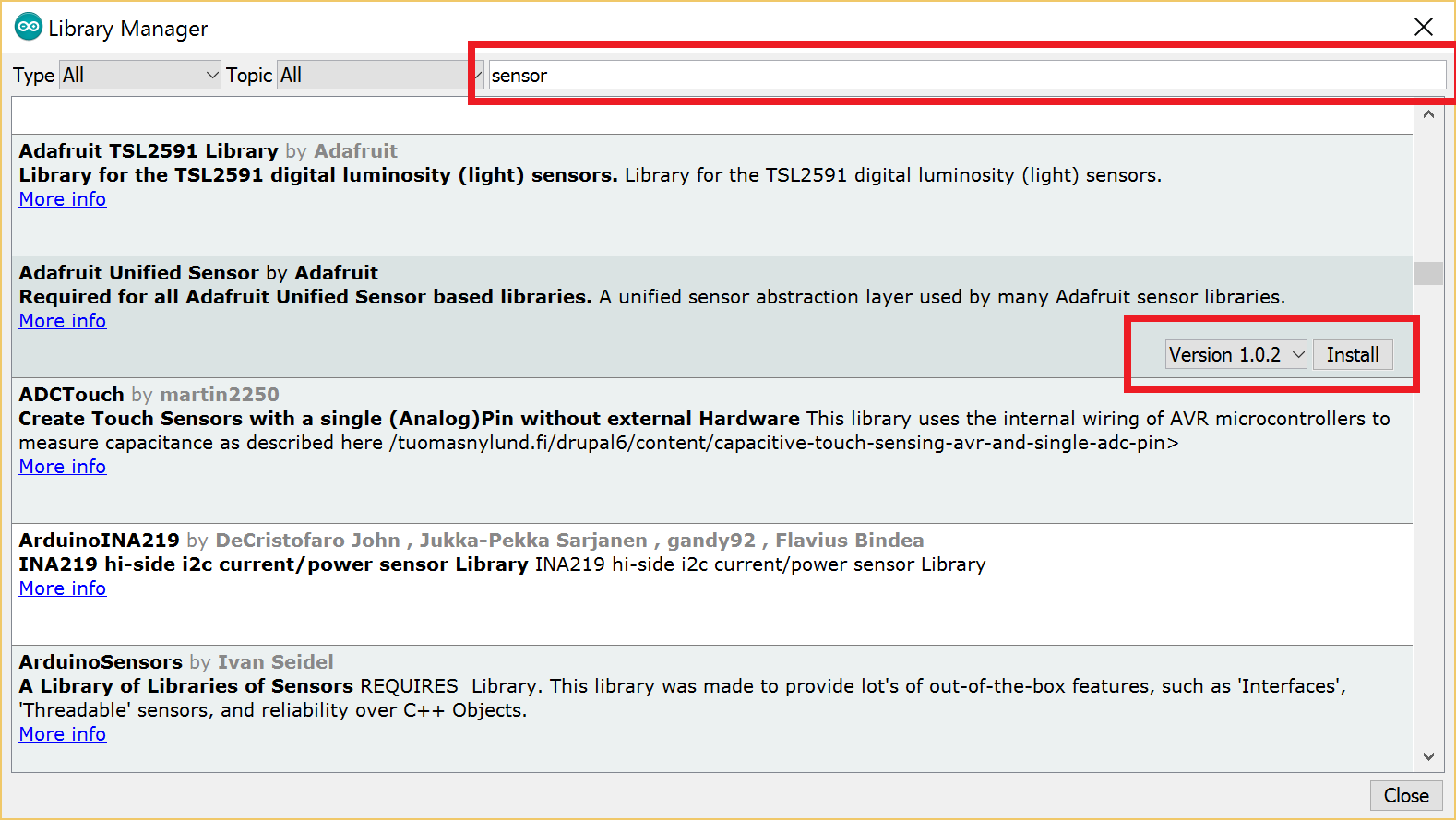


Figure : Install Adafruit Unified Sensor Library

1. Restart the Arduino IDE.

There are three ways you can connect the BME280 with your Adafruit Feather. *Detailed explanation on these three modes is beyond the scope of this document*.

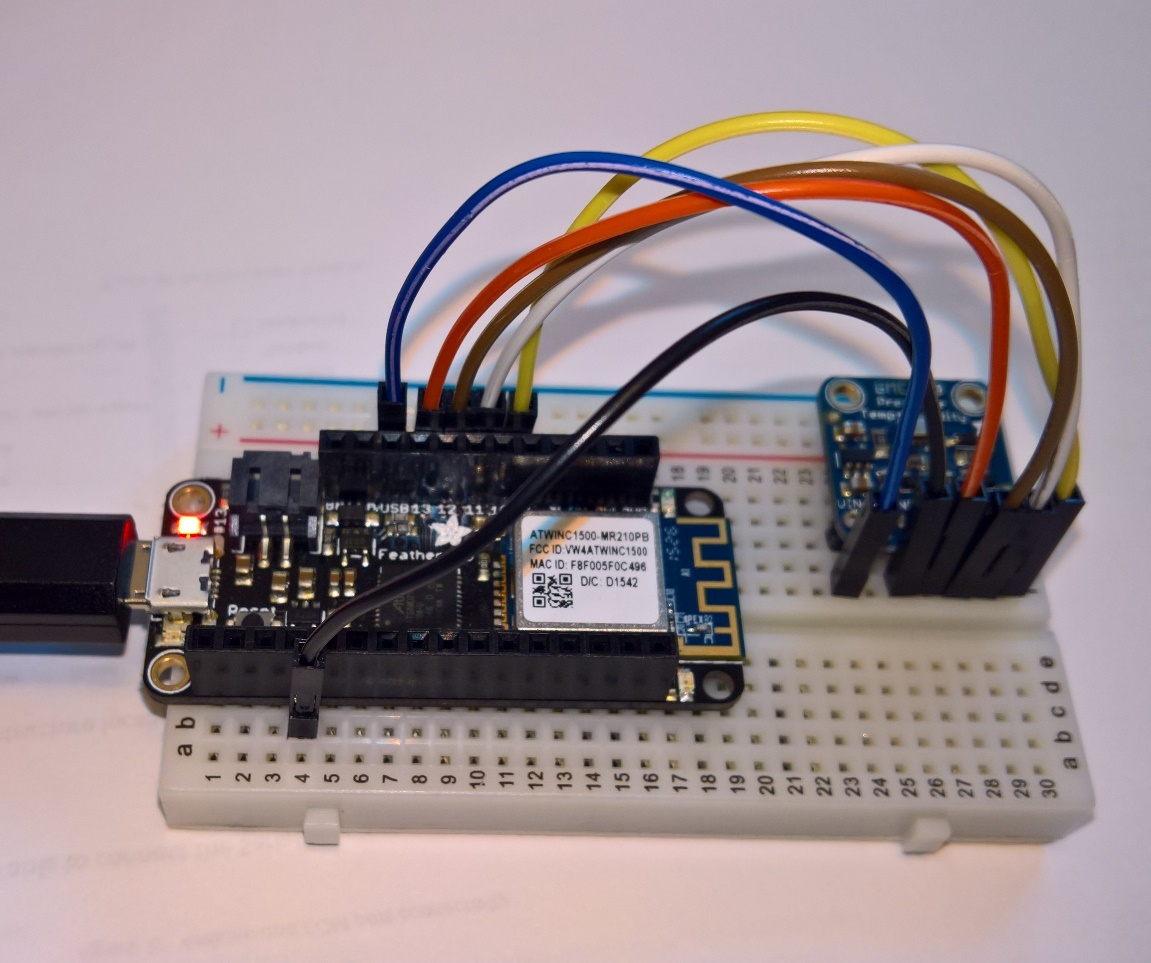
1. **I2C mode**. This is for Inter-integrated circuit mode and not using this for our examples.
2. **Hardware SPI mode**. Although the documentation in the Microsoft IOT starter kit mentions that you set-the connections up in this mode, the documentation of Adafruit sort of mentions that the MOSI/MISO/SCK pins are also used for WiFi module. Just at an off chance of avoiding conflict, I decided to use the software SPI mode.
3. **Software SPI mode**. We will be using this connectivity mode. Our connection pins and pin-out photographs are mentioned next.
   1. Basic pin-layout of the Adafruit Feather can be viewed from [here](https://cdn-learn.adafruit.com/assets/assets/000/030/918/medium800/microcomputers_2771_pinout_v1_0.png?1457305552)

## Adafruit Feather and BME 280 Hardware connection

1. For our test and connections, we’ll use following pin configurations

|  |  |  |  |
| --- | --- | --- | --- |
|  | BME 280 Pin | Adafruit Feather Pin | Adafruit Feather Port Pin |
| 1 | CS | 10 | PB6 |
| 2 | SDI | 11 | PB7 |
| 3 | SDO | 12 | PD6 |
| 4 | SCK | 13 | PD7 |
| 5 | GND | GND | GND |
| 6 | 3V3 | Not Connected |  |
| 7 | VIN | USB | USB |

Table : BME280 Adafruit Feather Pins Connection

The connection photographs and related configuration to test is shown below. 

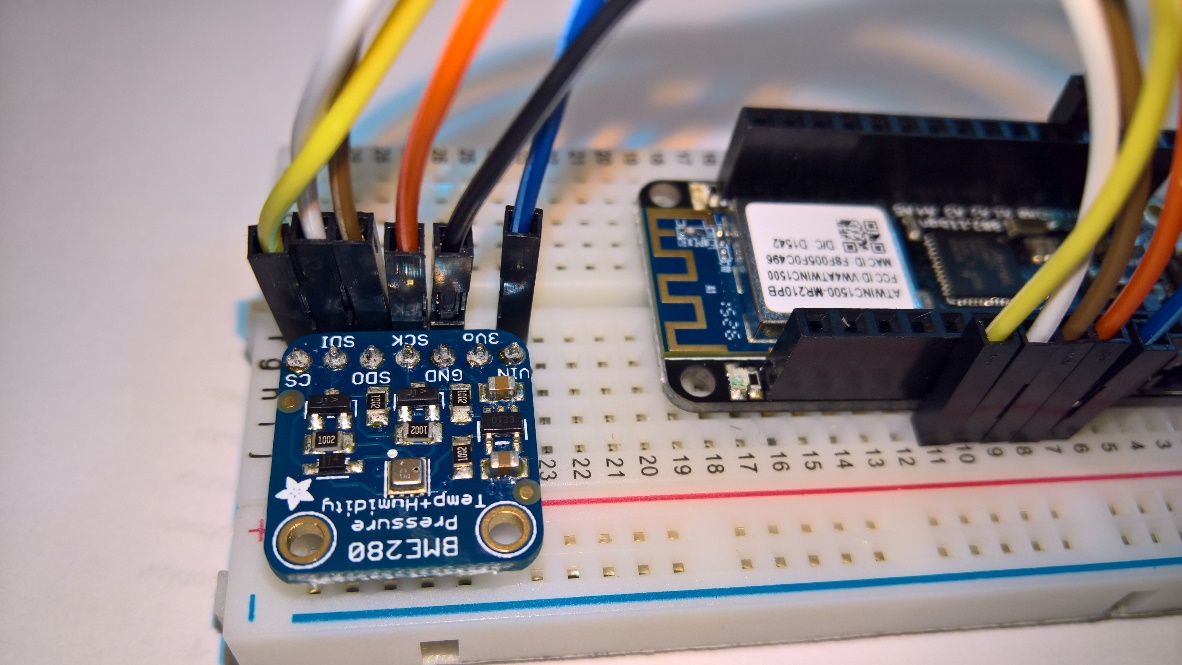


Figure : BME280 Pin Connection - 1

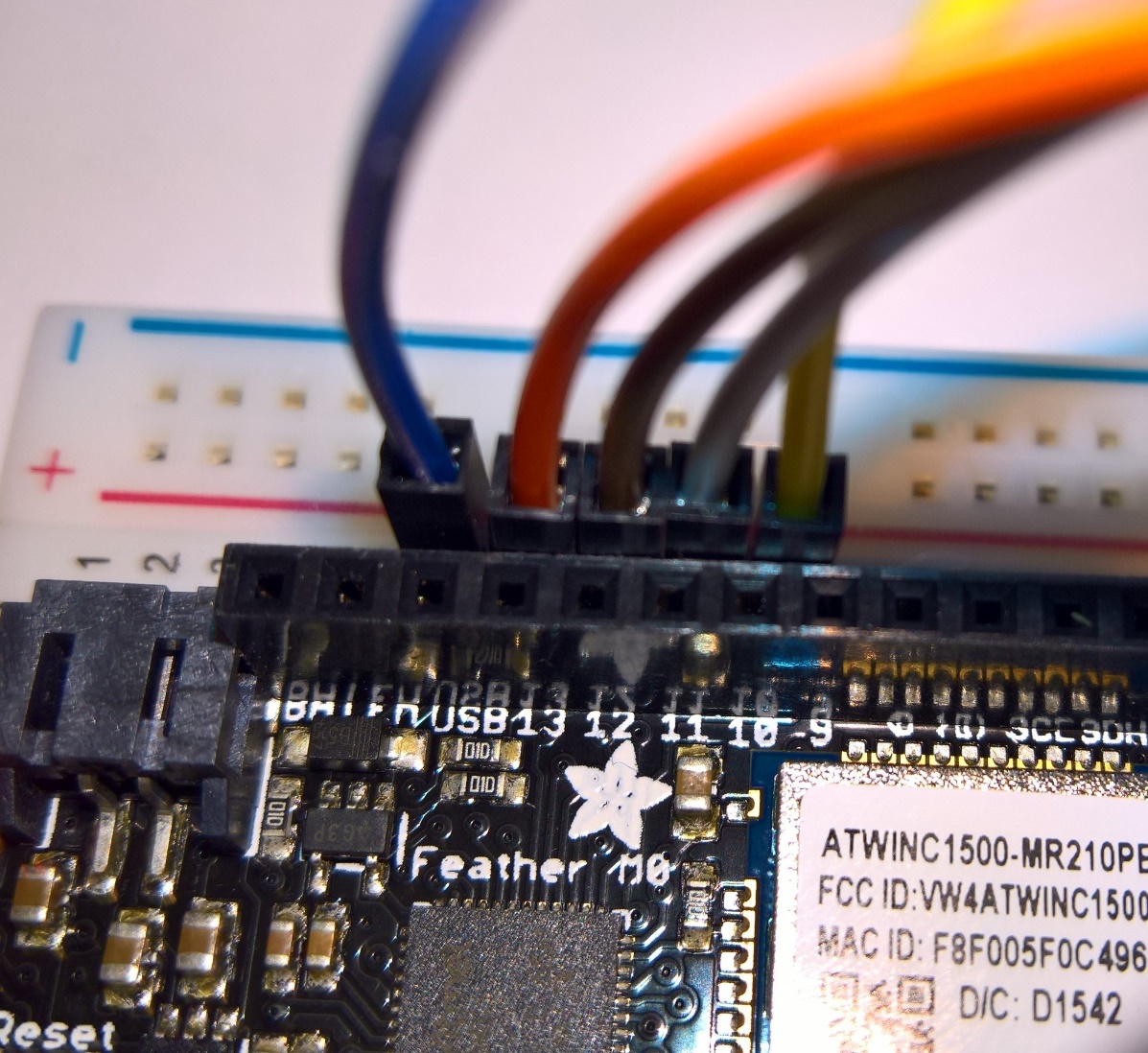


Figure : BME 280 Pin Connections - 2

## Test BME connection and read Telemetry data

1. Open sketch **bme280test** inside the Arduino IDE. (**File 🡪 Examples 🡪 Adafruit BME280 Library 🡪 bme280test**)
2. Make following changes to the sketch.

#include <Wire.h>

#include <SPI.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_BME280.h>

#define BME\_SCK 13

#define BME\_MISO 12

#define BME\_MOSI 11

#define BME\_CS 10

#define SEALEVELPRESSURE\_HPA (1013.25)

//Adafruit\_BME280 bme; // I2C

//Adafruit\_BME280 bme(BME\_CS); // hardware SPI

Adafruit\_BME280 bme(BME\_CS, BME\_MOSI, BME\_MISO, BME\_SCK);

void setup()

{ …

1. Upload the program to the Feather.
2. You should get following output –

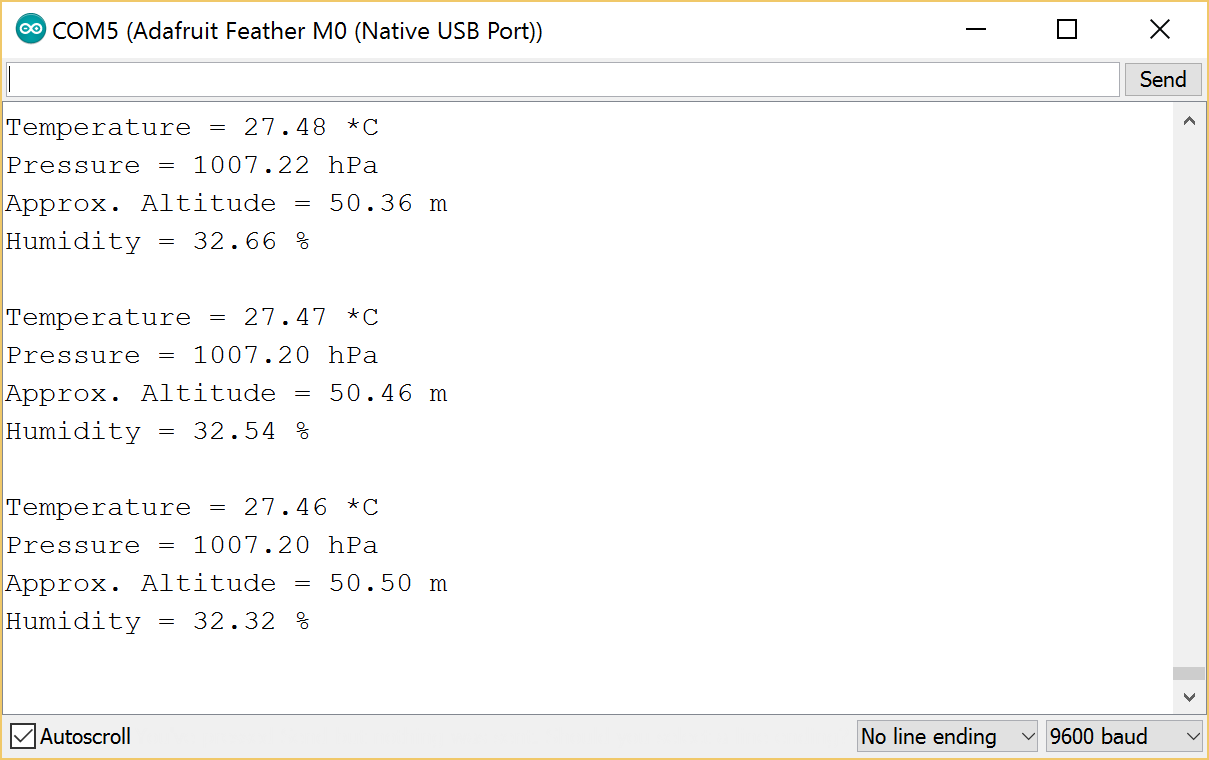


Figure : Successful BME280 Connectivity Test Output

1. – indicating successful working and connectivity between BME280 and Adafruit Feather.

# Part C: Connecting with Microsoft Remote Monitoring IOT Suite

Now let’s compete the final part of this equation by integrating the telemetry device (= Adafruit Feather + BME280) with the Azure IoT hub.

We will use the preconfigured solution <http://AzureIoTSuite.com> for this exercise. The beauty of this solution is that it creates all the Azure related services from one click.

## Provision Azure IoT Suite

1. Navigate to <http://AzureIoTSuite.com>
   1. [// if needed, authenticate yourself]
   2. Additional information about the AzureIoTSuite is beyond the scope this document.
      1. For more information and details about the IoT suite, reader is advised to refer to [here](https://azure.microsoft.com/en-us/documentation/suites/iot-suite/), [here](http://social.technet.microsoft.com/wiki/contents/articles/32941.iot-suite-under-the-hood-remote-monitoring.aspx) and [here](http://social.technet.microsoft.com/wiki/contents/articles/32975.iot-suite-remote-monitoring-adding-live-and-simulated-devices.aspx).
      2. Reference architecture of the solution is available [here](https://azure.microsoft.com/en-gb/updates/microsoft-azure-iot-reference-architecture-available/) and [here](http://download.microsoft.com/download/A/4/D/A4DAD253-BC21-41D3-B9D9-87D2AE6F0719/Microsoft_Azure_IoT_Reference_Architecture.pdf)
2. Click on “Create a new solution”

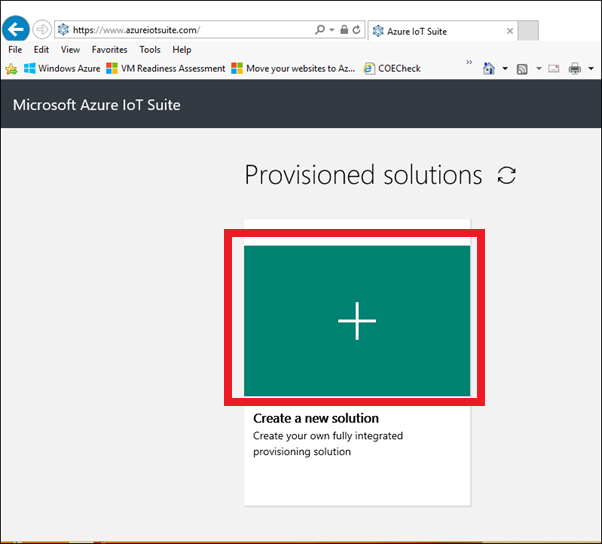


Figure : AzureIoTSuite.Com Create

1. Select “Remote monitoring” from the solution types displayed.



Figure : Azure IoT Suire Remote Monitoring Solution create

1. Enter a unique solution name, Region where the solution would be deployed and the name of the Subscription under which the solution would be deployed.

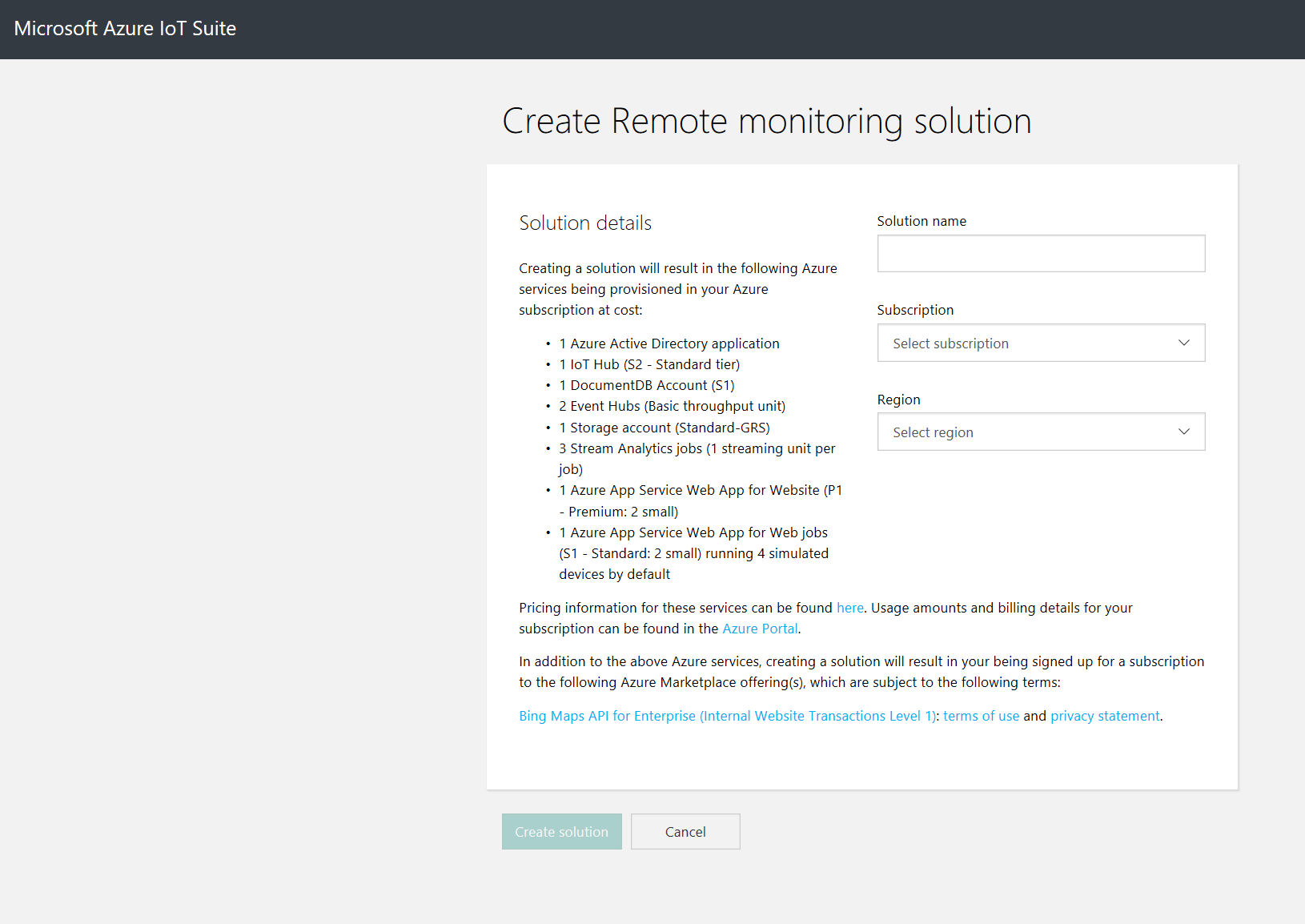


Figure : Enter Azure IoT Suite Details

1. Finally click the “Create Solution” button. The solution should be available inside the Azure subscription in **10-15 minutes’ time**. Once the solution is ready, it will display following screen. Click on the ‘Launch’ button to see the simulated devices in action

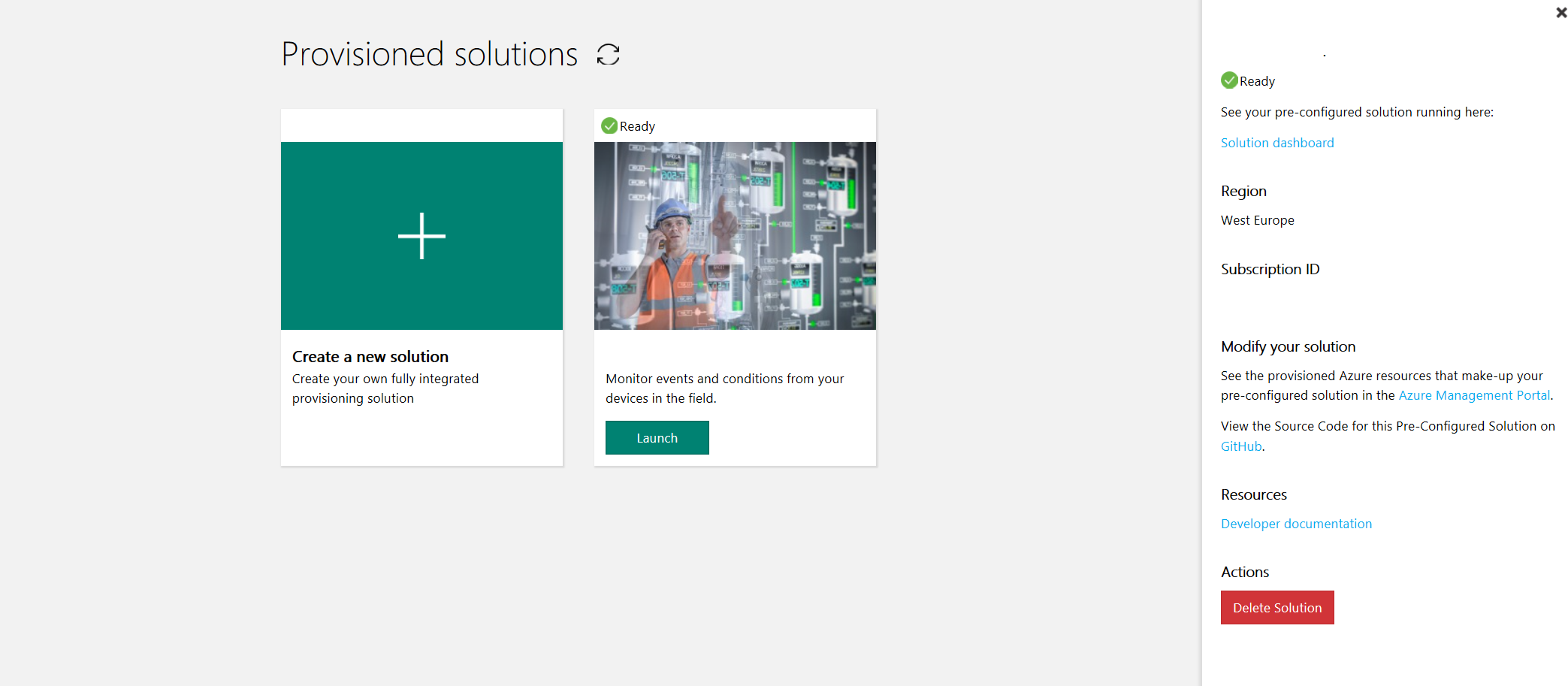


Figure : Launch Solution Dashboard Portal

* 1. You might need to enter your login credentials and Accept the permissions screen to continue
  2. If you are accessing the website for the first time, Profile read permission page also might be shown. Click ‘**Accept**’ to continue

1. You’ll be shown the Dashboard page of the portal.
   1. **[IMP]** At this point it is important to ensure that the user who is connected is logged in as the **Administrator** to the site. See the following snapshot – similar to what you should see when you navigate to your dashboard site

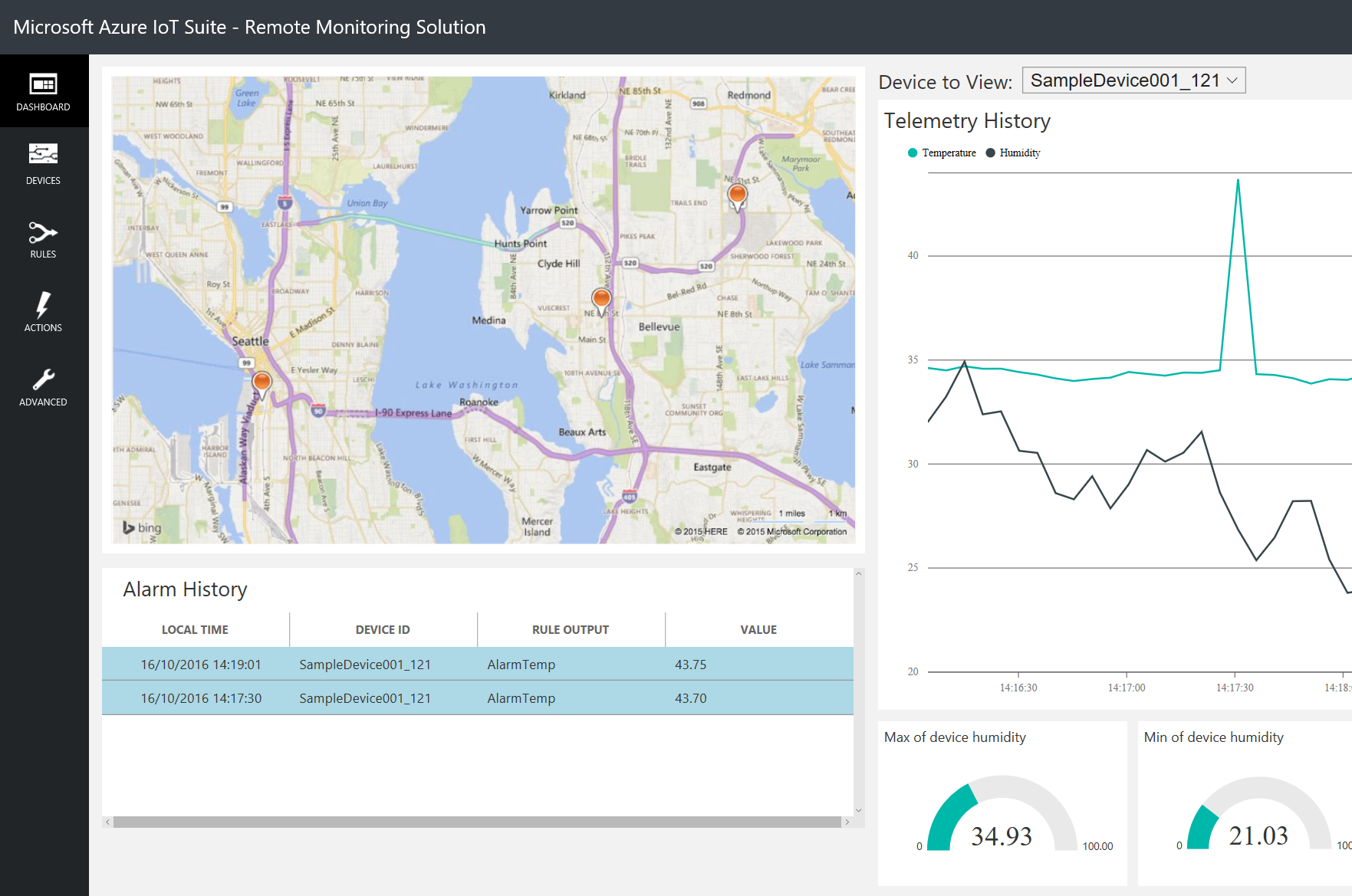


Figure : Admin Dashboard Portal

* 1. If you are logged in as a non-Admin user, functionality such as adding new Device will not be available to you via the portal web-site. I’ve seen this happen when the user I am logging on is also added as user into multiple Azure AAD tenants. The steps to undertake, if you have a non-admin access to the AzureIoTSuite.com will be added later in the document.

## Register Telemetry Device

1. We’ll now add our telemetry device to the solution by first registering it.
2. On the Admin Dashboard portal, click on the Add Device button at lower left corner of the page.
3. On the next page, select ‘**Custom Device**’ and click ‘**Add New**’ button underneath.
4. Whilst registering a new device, on Step 2 of 3, few choices are shown. Select the middle choice –‘**Let me define my own ID**’ (1), enter the device id as ‘**feather\_arduino\_device\_1**’ (2) and click ‘**Check ID**’ (3). If the device id is available, indicated by the green tick, click the **Create** button at the bottom (4). You can provide a unique name of your choice in the device id field as long as its unique to the IoT solution. We’ll re-use this same value later

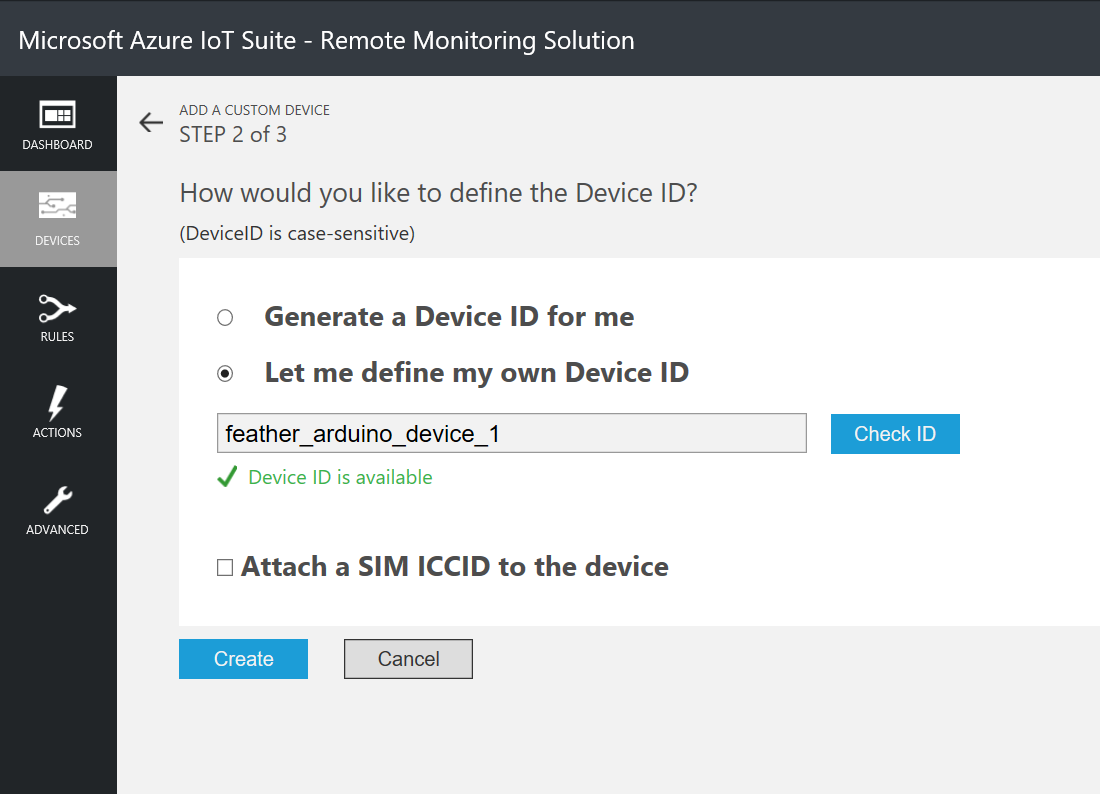


Figure : Register Device, Step 2 of 3

1. Once the device has been registered successfully, the next page will display the credentials needed for the device to connect to the Azure IoT Suite. Make sure you make a local copy of all the credentials as they’ll be used in the next step.
   1. You need to have following credentials locally stored after this step: Device ID, IoT Hub Hostname and Device Key

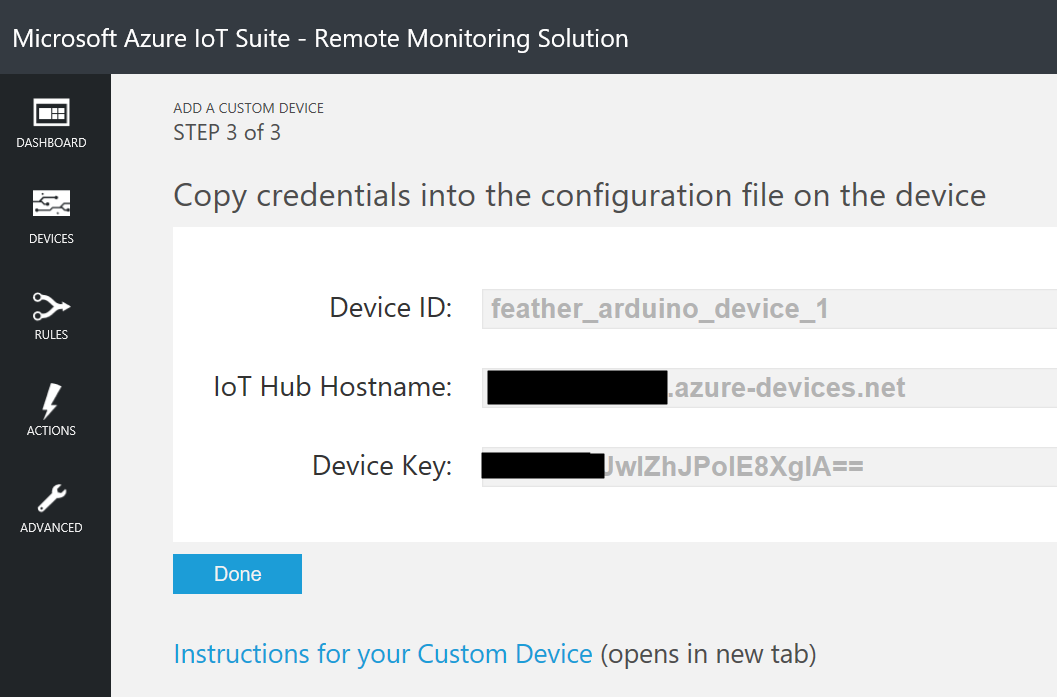


Figure : Registered Device Credentials

1. Once the device has been successfully registered, next you’ll be shown the Devices page. It’ll show that your new device – **feather\_arduino\_device\_1** - has been successfully registered and its status will be shown as ‘Pending’ meaning it’s registers but not present as no messages have been received about it. Its status is shown as pending as the IoT Hub does not have metadata information about the device like its location, commands etc. We’ll do that by sending Metadata message and command message to the IoT Suite.

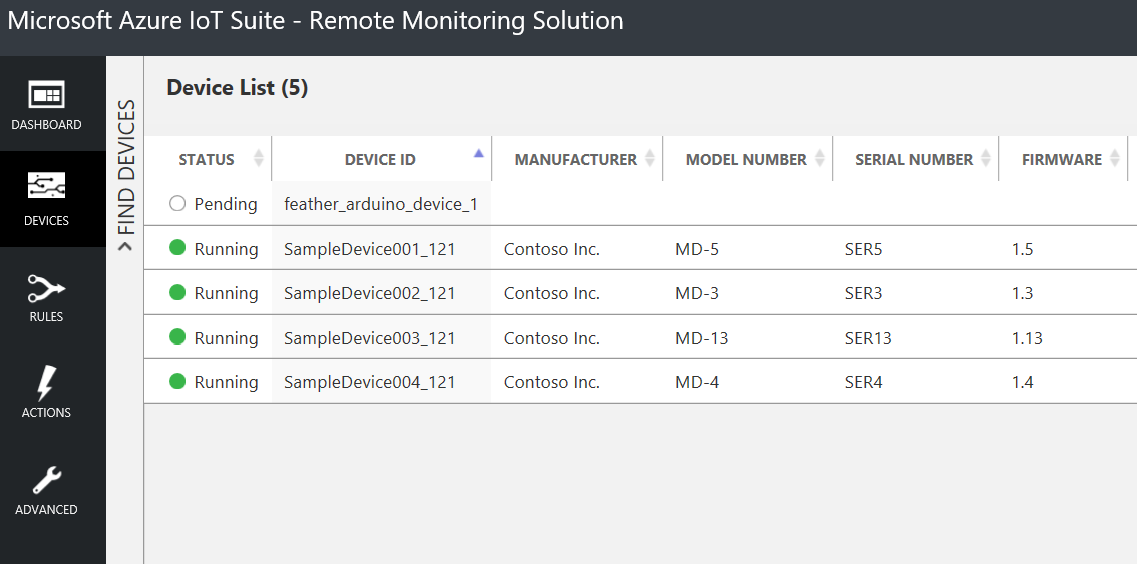


Figure : Registered Device Status

## Transmit Metadata and Telemetry messages to IoT Hub

First thing first, we need to install some libraries for our application to work.

### Install required libraries

1. Download the latest Azure IoT library from [here](https://github.com/arduino-libraries/AzureIoT/archive/master.zip)
2. Extract the contents locally and rename the extracted folder from ‘**AzureIoTHub-master**’ to **AzureIoT**.
3. Copy the ‘**AzureIoT**’ folder under the Arduino libraries folder, as described in this [section](#Arduino_Library_Structure) of the document. If any previous versions of **AzureIoT** folder exists, delete it.
4. Restart Arduino IDE. Use the Library Manager from the IDE (Sketch 🡪 Include Libraries 🡪 Manage Libraries) to install following components
   1. Adafruit Unified Sensor, v 1.0.2
   2. RTCZero by Arduino, v 1.4.3
   3. NTPClient by Fabrice Weinberg, v 3.0.0[[4]](#footnote-4)
5. Restart IDE.

. The final application is stored under the Github repository and is called **remote\_monitoring\_nk**

1. Load the sketch ‘remote\_monitoring\_nk’ into the IDE. (File 🡪 Open).
2. Make changes to two files
   1. Remote\_monitoring\_nk
      1. static const char ssid[] = ""; //update this value
      2. static const char pass[] = ""; //update this value
   2. simplesample\_http.c
      1. static const char\* deviceId = "[deviceName]"; // update this value
      2. static const char\* connectionString = "HostName=[YourHostNameHere].azure-devices.net;DeviceId=[enter deviceID];SharedAccessKey=[Your Secret Keys here]"; //update this value
   3. No changes should be required to bme280.cpp if the connections are same as described [here](#_Adafruit_Feather_and). Otherwise change the pin number
3. Compile the sketch and upload the binary onto the Adafruit Feather
4. Watch the output in the Serial Monitor. The output should display the following.

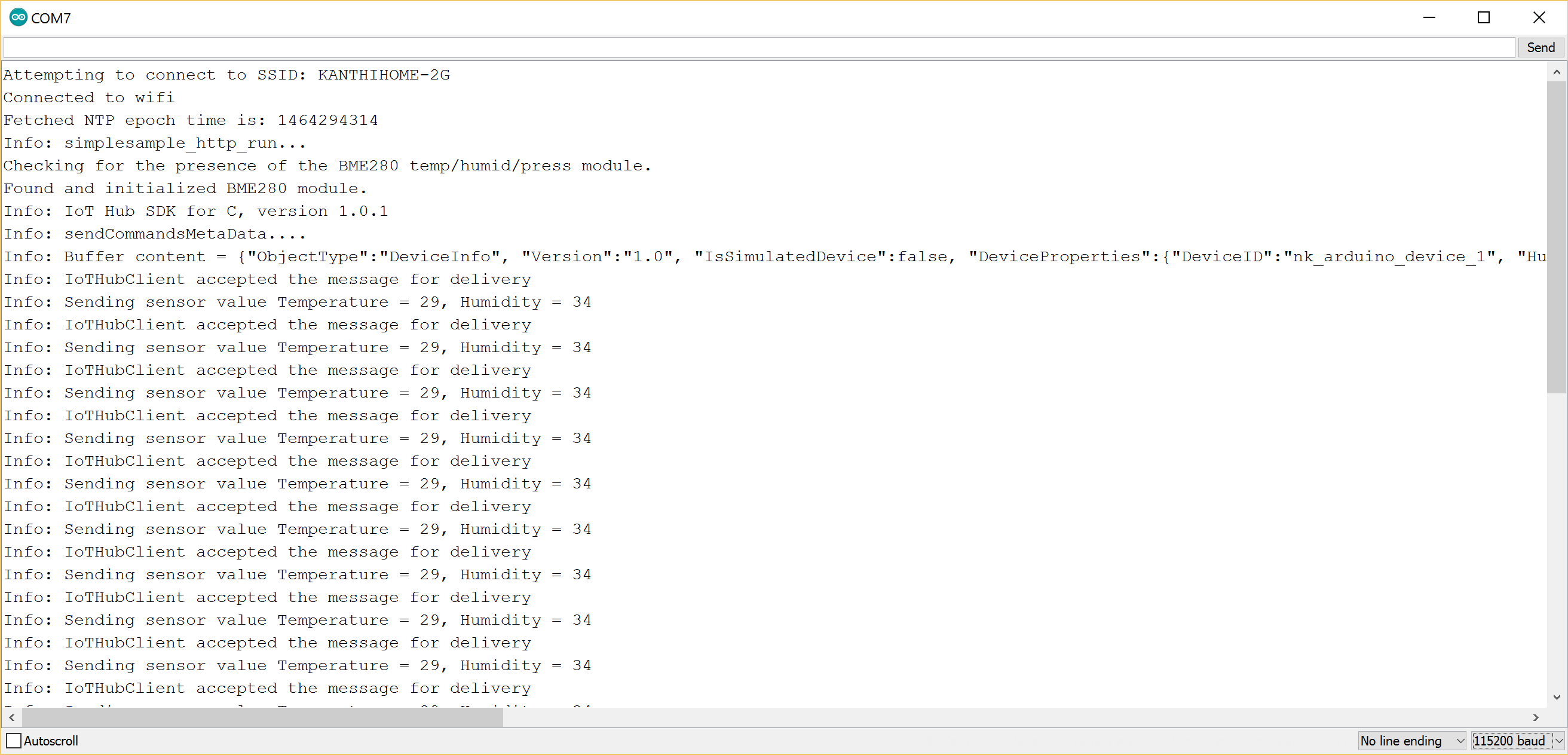


Figure :Successful Transmission of Telemetry Data from device

1. And the proof in the pudding – open your IoT suite portal. You should see the telemetry messages being received there!

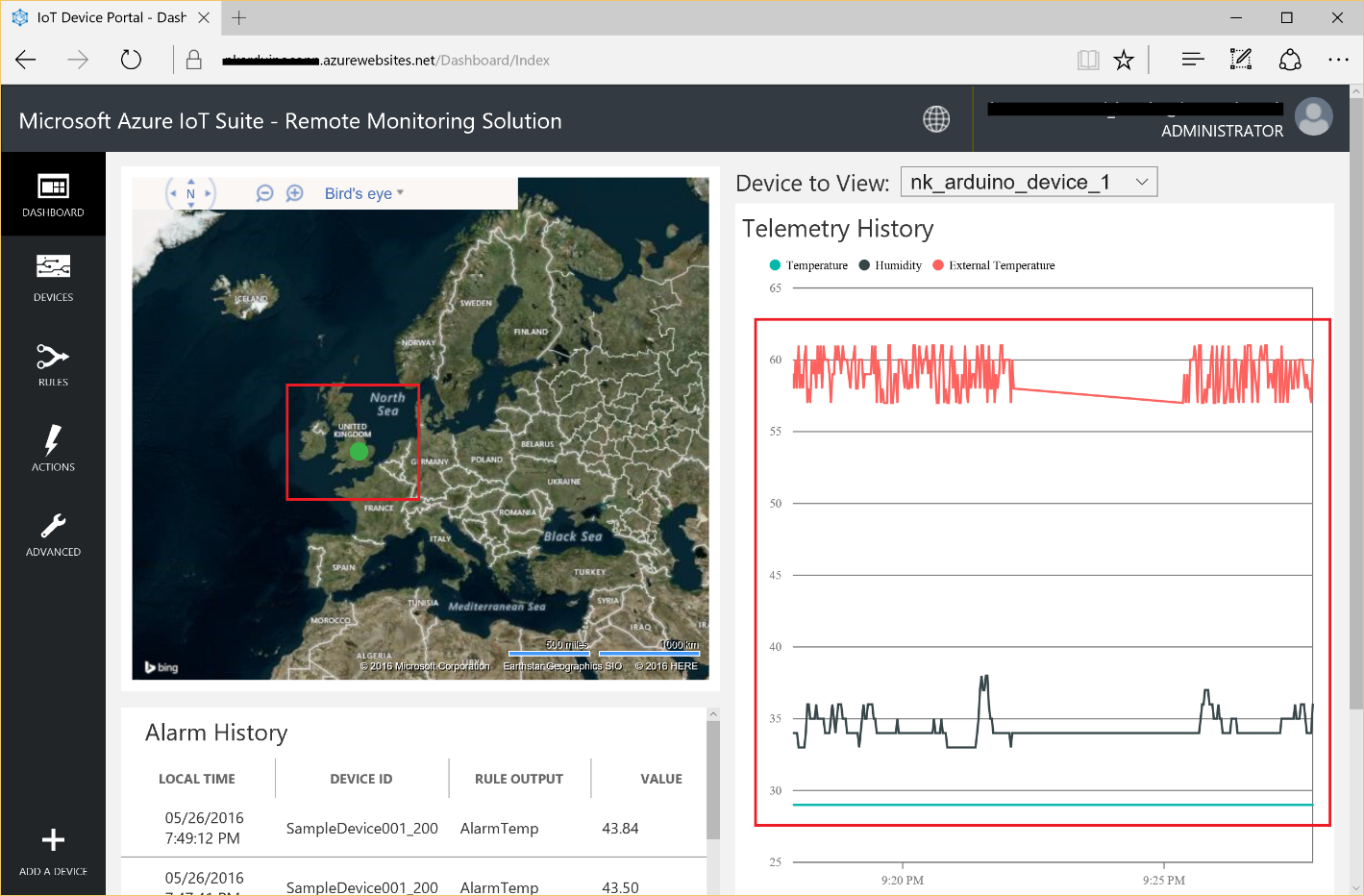


Figure : Proof is in the pudding

1. As extra steps, you can now send commands as well down to the device from the portal.
2. Enjoy!!!

# Appendix

## Components of Microsoft Azure IoT Starter Kits

* The kit includes:
* 1x Assembled [Adafruit Feather M0 WiFi](https://www.adafruit.com/product/3010) w/ [Feather Stacking Headers](https://www.adafruit.com/product/2830)
* 1x [FeatherWing OLED - 128x32 OLED Add-on](https://www.adafruit.com/product/2900)
* 1x Assembled [BME280 I2C or SPI Temperature/Humidity/Pressure Sensor](https://www.adafruit.com/product/2652)
* 1x [Micro Servo](https://www.adafruit.com/product/169)
* 1x [PIR (motion) Sensor](https://www.adafruit.com/product/189)
* 1x [Fast Vibration Switch](https://www.adafruit.com/product/1766)
* 1x [Magnetic Contact Switch (door sensor)](https://www.adafruit.com/product/375)
* 1x [Full-sized Breadboard](https://www.adafruit.com/product/239)
* 1x Assembled [TCS34725 RGB Color Sensor](https://www.adafruit.com/product/1334)
* 1x [Premium Male/Male Jumper Wires - 20 x 6"](https://www.adafruit.com/product/1957)
* 1x [Premium Male/Male Jumper Wires - 20 x 3"](https://www.adafruit.com/product/1956)
* 1x [USB Cable - A/Micro B](https://www.adafruit.com/product/592)

Component bag containing:

* 3x 12mm Tactile Switches
* 1x [Breadboard Trim Potentiometer 10K](https://www.adafruit.com/product/356)
* 1x Diffused 10mm Green LED
* 1x Diffused 10mm Red LED
* 5x 10K 5% 1/4W Resistor
* 5x 560 Ohm 5% 1/4W Resistor
* 1x [Piezo Buzzer](https://www.adafruit.com/product/160)
* 1x [Photo Cell Light Sensor](https://www.adafruit.com/product/161)
* 1x [Diffused RGB (tri-color) LED](https://www.adafruit.com/product/159)
* 1x [Breadboard-friendly SPDT Slide Switch](https://www.adafruit.com/product/805)

## USB Connectivity Issues with Adafruit Feather

*[NOTE:] There are two Arduino IDE available in the market. Arduino has split into two companies Arduino LLC (arduino.cc) that produces the "Zero" and Arduino SRL (arduino.org) that produces the "Zero Pro", later renamed "M0 Pro". Luckily the boards are very similar, but fact that the bootloader is different can cause connectivity errors if different IDE is used.*

***For this project IDE from Arduino LLC version 1.6.9 was used.***

*The latest version of Arduino IDE from Arduino SRL is 1.7.10. This was* ***NOT*** *used.*

It’s quite possible that the Adafruit Feather gets into a unstable state and does not respond to USB inputs or the USB port if not recognised. This can happen if you set a watchdog timer, or sleep mode that stops the USB, or any sketch that crashes the Feather.

To rectify that, do the following

1. Turn the Verbose upload mode on. (File 🡪 Preferences)

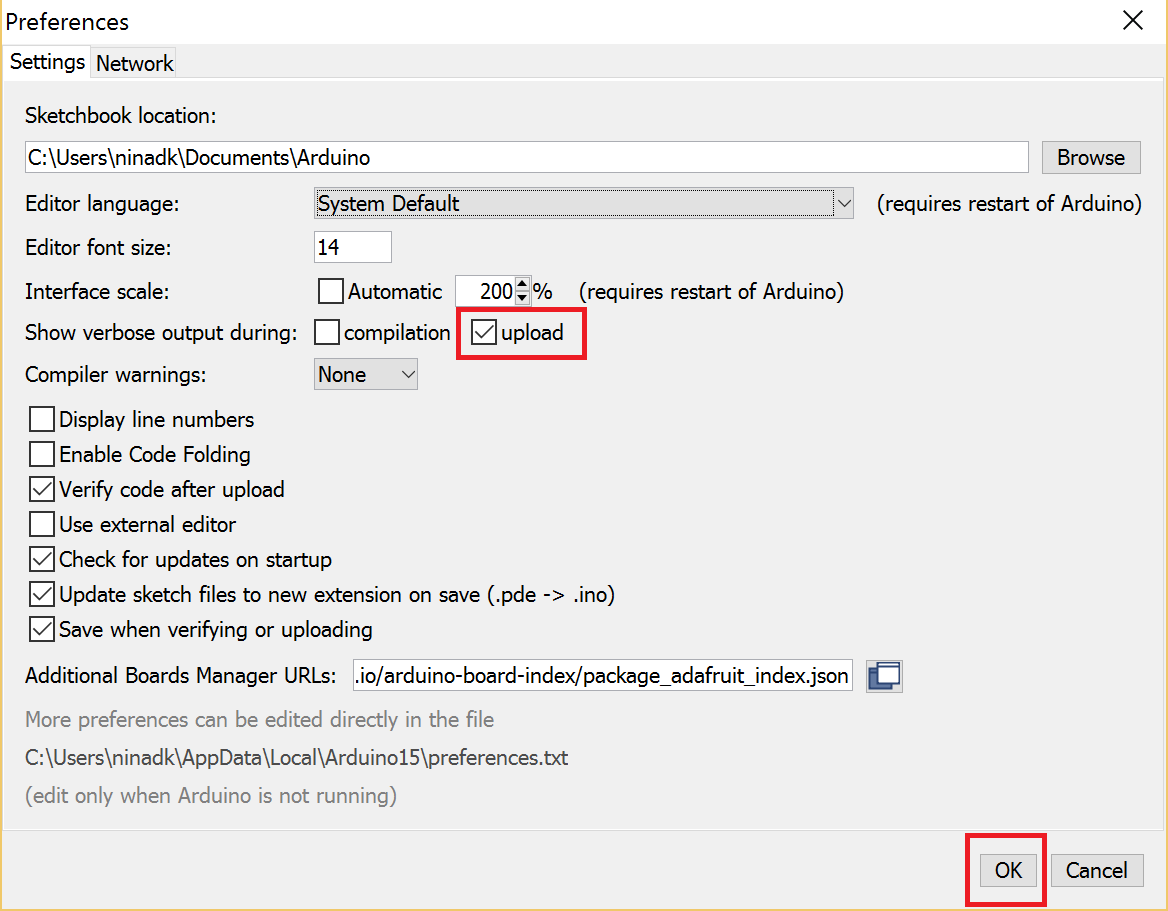


Figure : Upload Debug Preferences

1. Plug in the Adafruit Feather into the USB port using the cable that came with IoT Starter Kits. Please bear in mind that there are charge-only USB cables that are available in the market. These cables are not suitable for connectivity with Feather.
2. Open any of your sketch that compiles correctly
3. Upload the sketch. It is assumed that the sketch compiles successfully
4. The IDE will print out a bunch of COM ports as it tries to upload.



Figure : Port search during Upload

1. During this time double click on the Reset button of Feather. You’ll see the red pulsing LED that tells that the Feather is now in bootloading mode
2. The USB port should now be recognised.
3. The IDE should see the bootloader COM/Serial port and Upload properly.
4. Once the Upload is completed, the COM Port may again not be available. Wait for few seconds (~10/15 seconds) and it should become available again.
5. Once the COM Port is available, select that port
   * If COM Port option is greyed out, press the reset button on Feather and wait for few seconds for it to become available again.

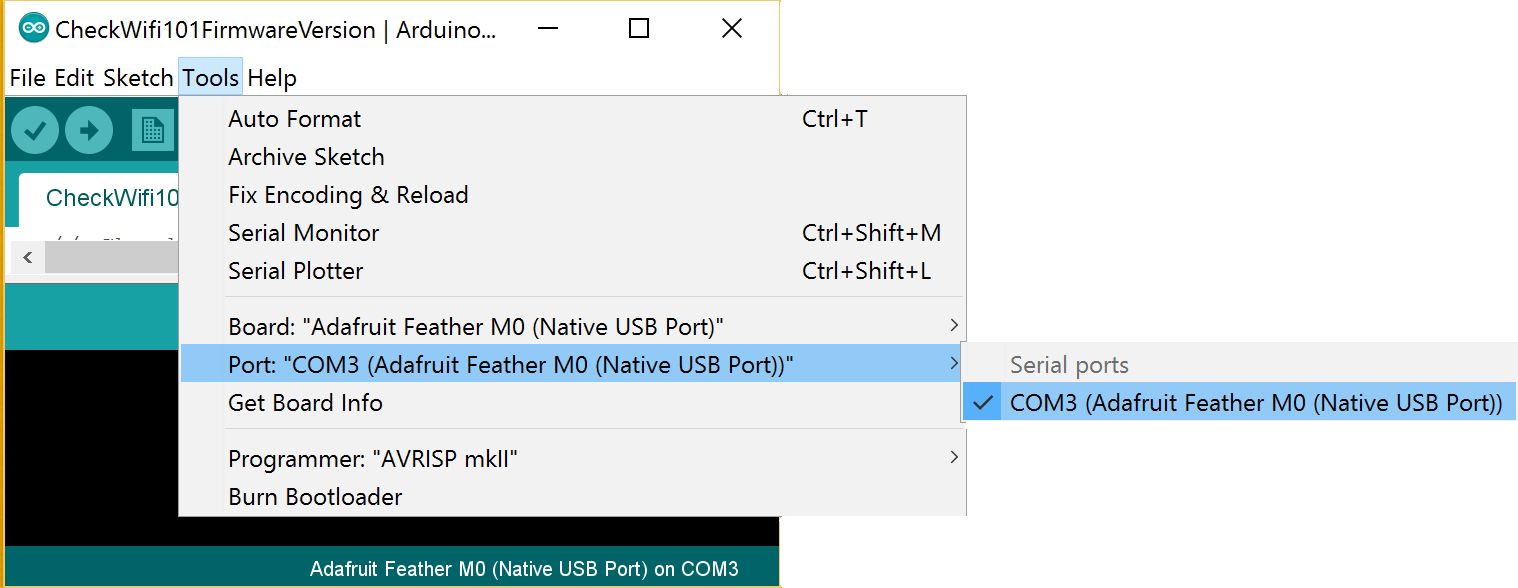


Figure : Arduino and COM port connectivity

Upload

1. Now you should be able to connect the Serial Monitor to debug and view the outputs of the sketch

The Arduino library folder structure locally looked like following

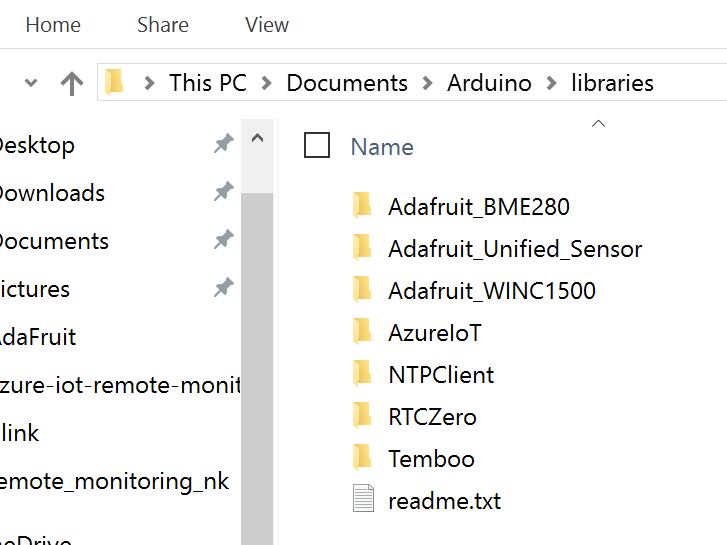


Figure : Arduino Library Structure

1. In Arduino environment, sketch is the code/program file that gets compiled [↑](#footnote-ref-1)
2. Upload is the process of compiling and burning the binary into the EEPROM of Feather. Once burned and installed, Feather will keep on continuously executing the program until a hardware interrupt/reset is invoked. [↑](#footnote-ref-2)
3. Think of Serializer window as the debug window. Code

   while (!Serial) { … ; at the start of the program will ensure that the program wait for serial port to be connected before executing the next statement. [↑](#footnote-ref-3)
4. You can take the NTPClient.Cpp and NTPClient.H files from the remote monitoring SDK and include them into this project, but this seemed like an easier and cleaner option. [↑](#footnote-ref-4)