What comes out of a Microsoft IOT Starter Kits – ‘Adafruit Feather M0 Kit’

You get the ‘[Adafruit Feather M0 WiFi - ATSAMD21 + ATWINC1500](https://www.adafruit.com/products/3010)’ version of Adafruit development board. Complete description of the product is available [here](https://www.adafruit.com/products/3010). For all the types that are available see [here](https://www.adafruit.com/categories/777). Effectively the development board is a light weight board having a micro-controller with WiFi built in.

Its Arduino compatible (?), built in WiFi and USB and battery charging.

The Wifi module is from Amtel, 802.11bgn-capable. Support pretty much everything. It’s got a Cortex M0+ processor (ATSAMD21G18 ARM Cortex M0 processor). 12-bit ADC (Analog to Digital conversion). 10-bit DAC (digital to analog conversion).

6 total [SERCOM](http://asf.atmel.com/docs/3.17.0/samd21/html/group__asfdoc__sam0__sercom__i2c__group.html) (Serial communications interface)

which can do SPI, I2C or UART (meaning???)

SPI – [Single Peripheral Interface](https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus). Is a asynchronous serial communication interface used for short distance communication primarily in embedded systems. Developed by Motorola, and has become and de-facto standard. Its effectively an four wire protocol, with the four logic signals working like this

* SCLK : Serial Clock (output from master).
* MOSI : Master Output, Slave Input (output from master).
* MISO : Master Input, Slave Output (output from slave).
* SS : Slave Select ([active low](https://en.wikipedia.org/wiki/Logic_level), output from master).

First three lines are shared by multiple slaves and SS line (CS in BME280) is connected to various output pins of the M0+ processor GPIO pins.

I2C is inter-integrated circuit. It standardised the data lines between various integrated circuits and their products. It reduced the number of wires to two (SDA – serial data line (pin 20) and SCL – serial clock line (pin 21)).

Pulse width modulate; (PWM). The duration on “on time” is called the pulse width. To get various analog values, you change, or modulate, that pulse width. If you repeat that pattern fast enough, it appears as if signal is steady voltage between 0 and 5v controlling the brightness of the LED.

GPIO General Purpose Input Output.

If you are using I2C with BME 280, you need to connect only the SCK and SDI lines of BME280. Rest you can leave alone.

UART is universal asynchronous receiver/transmitter. Data can be transmitted parallel and serial forms. Data is transmitted in conjunction with communication standards such as RS-232, RS-422, RS-485 etc.

For the BME 280 chip, we have following pins

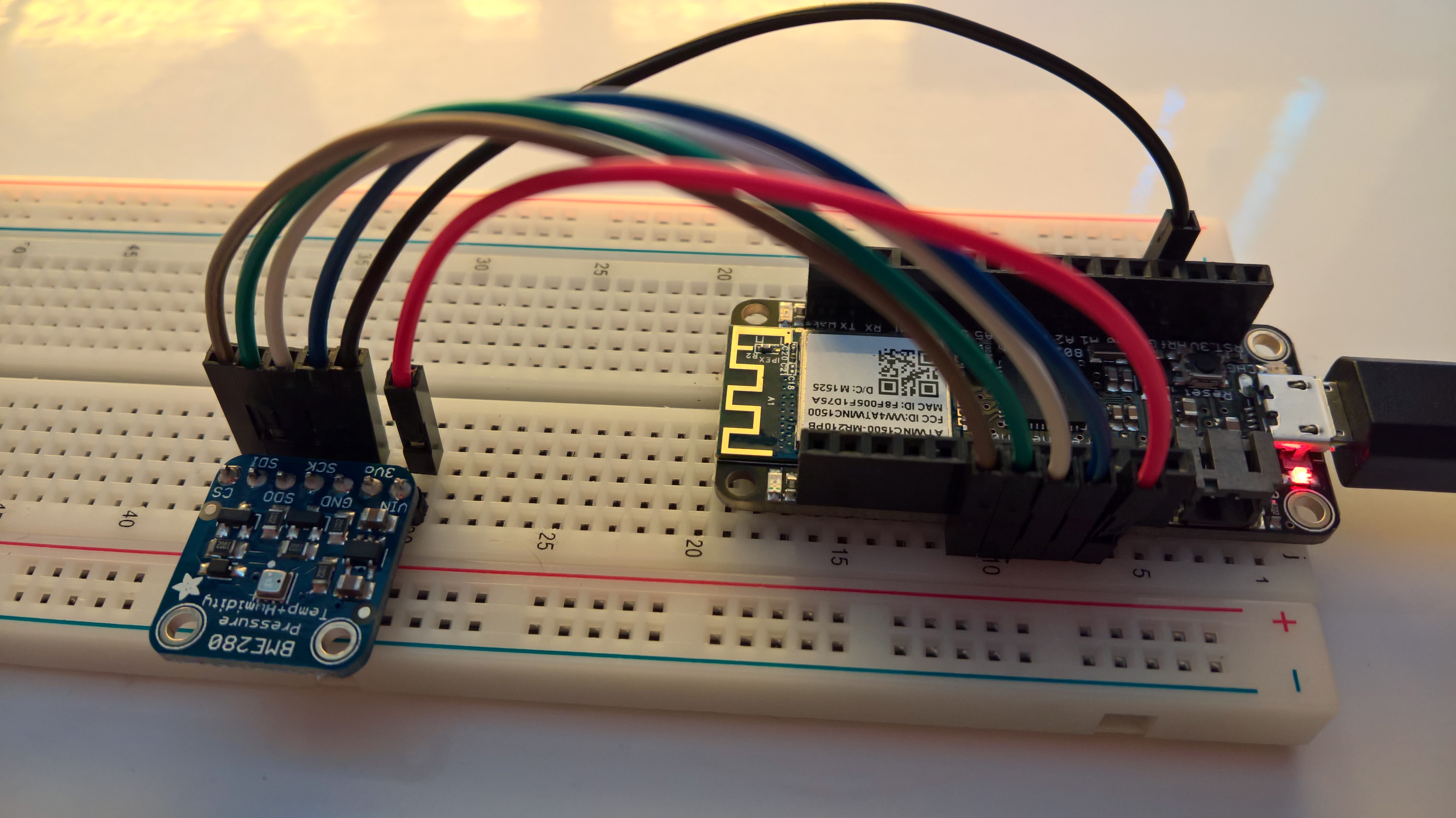
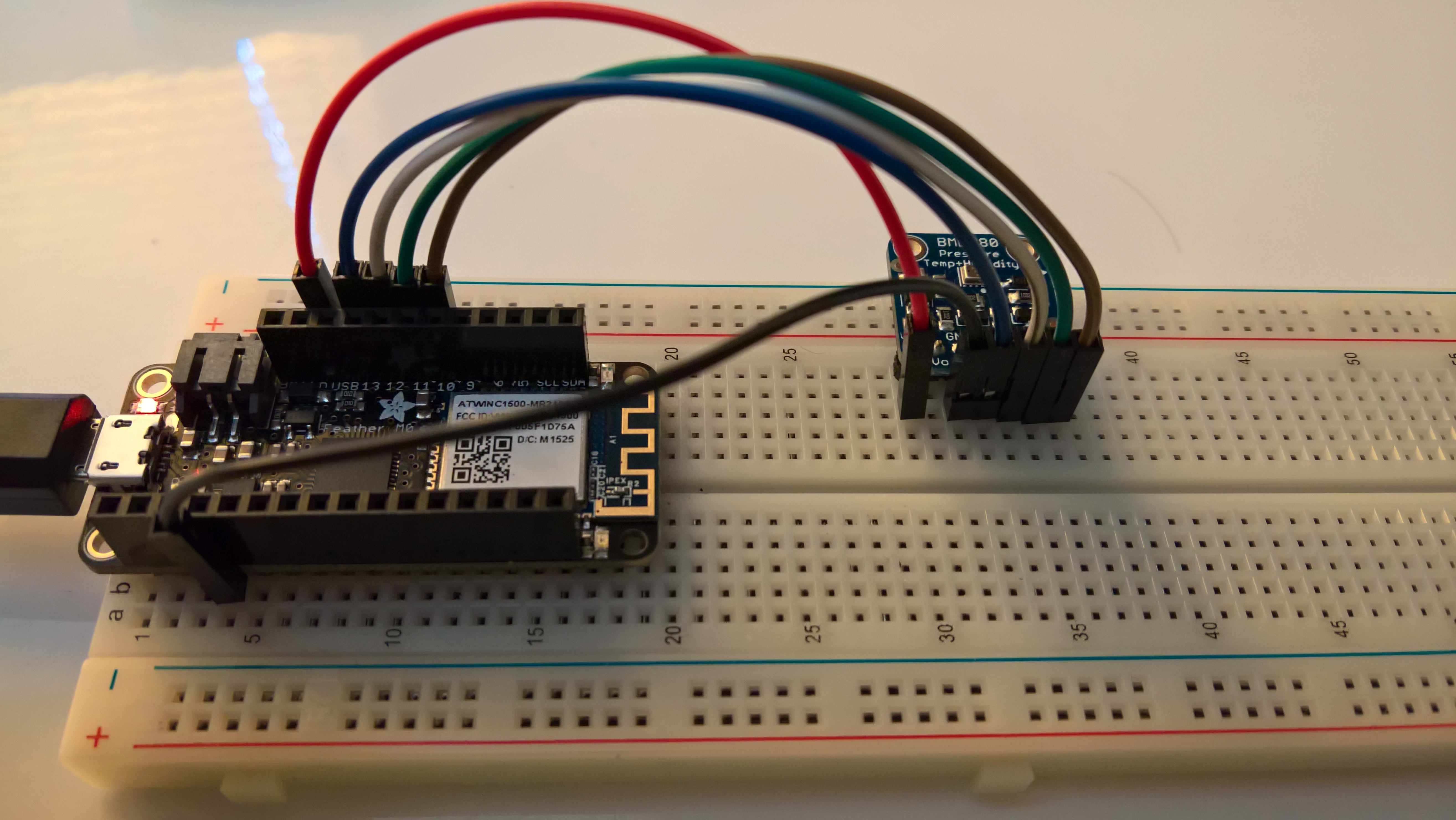
* SCK - This is the SPI Clock pin, it’s an input to the chip (SCLK)
* SDO - this is the Serial Data Out / Master In Slave Out pin, for data sent from the BMP183 to your processor (MISO)
* SDI - this is the Serial Data In / Master Out Slave In pin, for data sent from your processor to the BME280 (MOSI)
  + [SCK 🡪 GPIO in 24](https://learn.adafruit.com/adafruit-feather-m0-wifi-atwinc1500?view=all) (13???)
  + MOSI 🡪 SDI 🡪 GPIO 23 (11???)
  + MISO 🡪 SDO 🡪 GPIO 22 (12???)
* CS - this is the Chip Select pin, drop it low to start an SPI transaction. It’s an input to the chip (SS)

Q: What is I2C wiring and what is SPI Wiring? Both are different. For I2C you only need two pin connections. For SPI, one needs four pin connection.

There are three ways you can connect the BME280 with your Adafruit Feather

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. Hence I decided to use the software SPI mode.
3. Software SPI mode.

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



[Code](https://learn.adafruit.com/adafruit-bme280-humidity-barometric-pressure-temperature-sensor-breakout?view=all) to test the set-up and the changes that I made for my configuration are reproduced below

#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

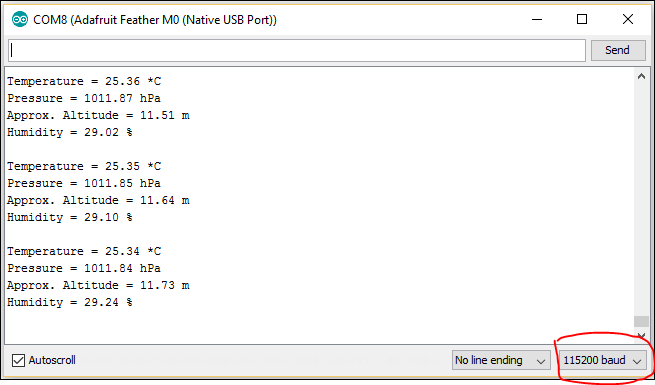
Adafruit\_BME280 bme(BME\_CS, BME\_MOSI, BME\_MISO, BME\_SCK);

void setup()

{ …

## Learnings:

* I Make sure that the baudrate specified in the ‘void setup() {…’ function - “Serial.begin(152000);” - and the serializer (CTRL + SHIFT + M, inside the Arduino IDE) baud rate match.



* + Otherwise I got a COM PORT error.
* Ensure that the library you are testing is BM**E**280 and not BM**P**280. Easy mistake to make and lost one night trying to debug this error. ☹
* In order for you to see the error messages and see the code from the very start, you can add following line at the start of the ‘void setup() {…’ function.

void setup() {

while (!Serial); // This ensures that the test actually does not start whilst we have got the serial communication going.

delay(1000);

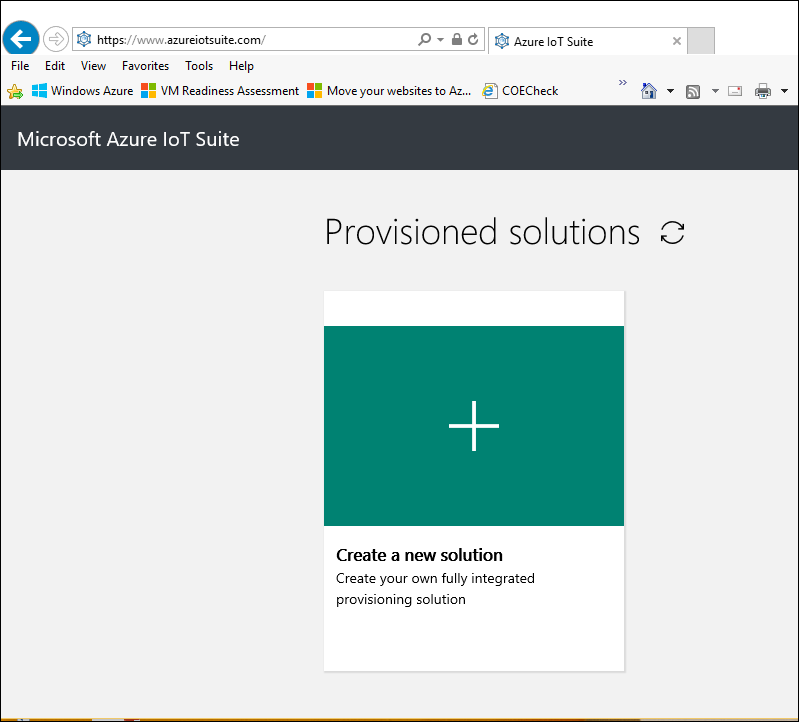
Serial.begin(115200);

…

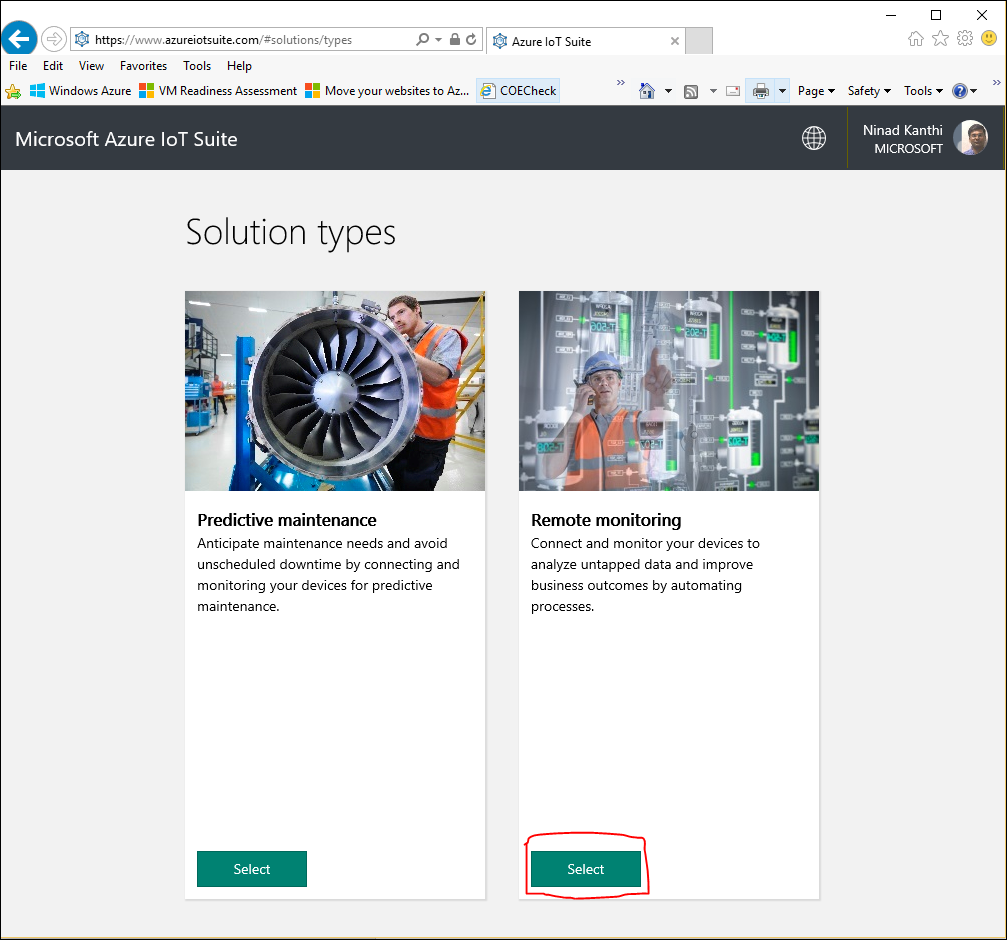
* Ensure that you close the Serializer window down before uploading a new Sketch. It gets its knickers in the twist otherwise.
* To make this work, once the program is executing.
  + Keep the Baud rate always at 152000. Never change this value
  + Always have code wait for the Serializer to be connected before doing anything else.
  + Press Reset.
  + Close Serializer if it’s still running. If the Green light on top of the board is still lit, it means that WiFi is still connected. It might cause issues.
  + Next wait for the COM port to become available under Tools menu of Arduino IDE;
  + Now Update with the latest code
  + Once loaded, make sure that the COM port is again available before plugging in and starting the Serializer.

Now that we have tested the set-up and verified that the Adafruit and BME280 are working nicely, we switch our attention to getting it working on Azure IoT suite.

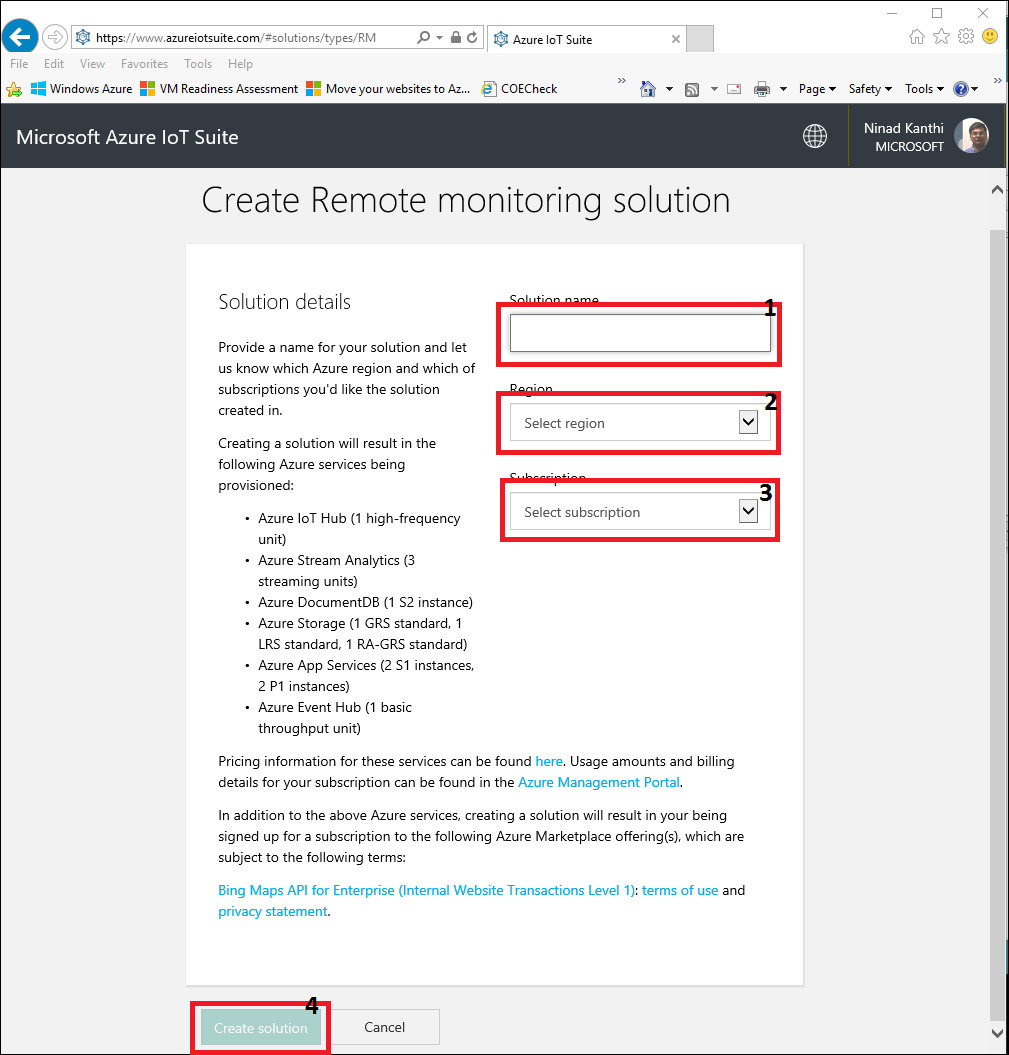
1. Navigate to AzureIoTSuite.com
2. [// if needed, authenticate yourself]
3. Click on “Create a new solution”



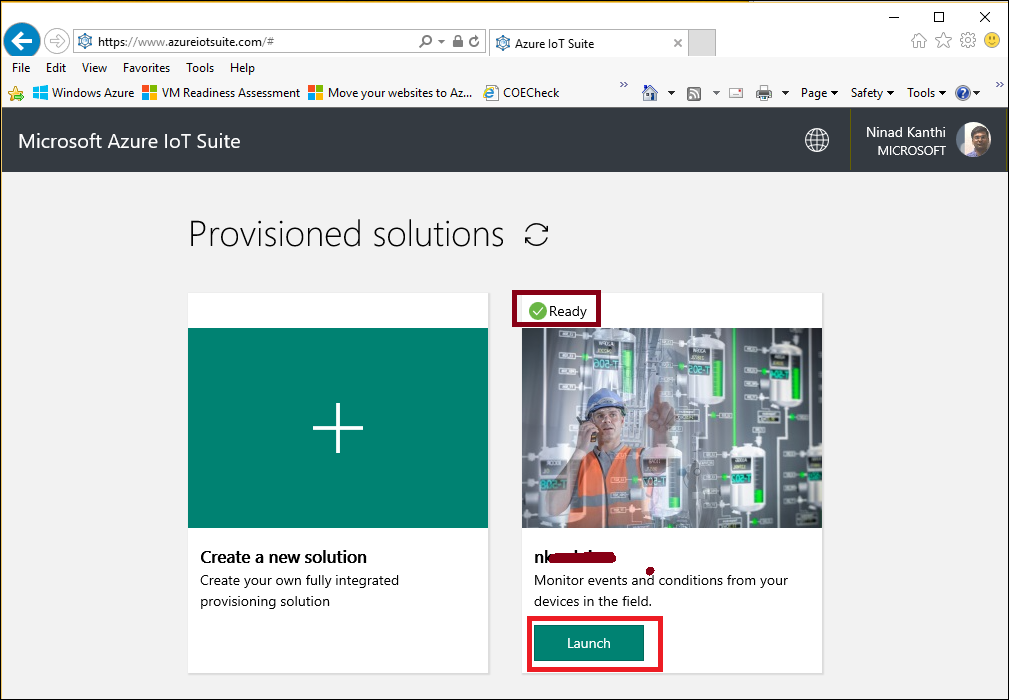
1. Select “Remote monitoring” from the solution types displayed.



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.

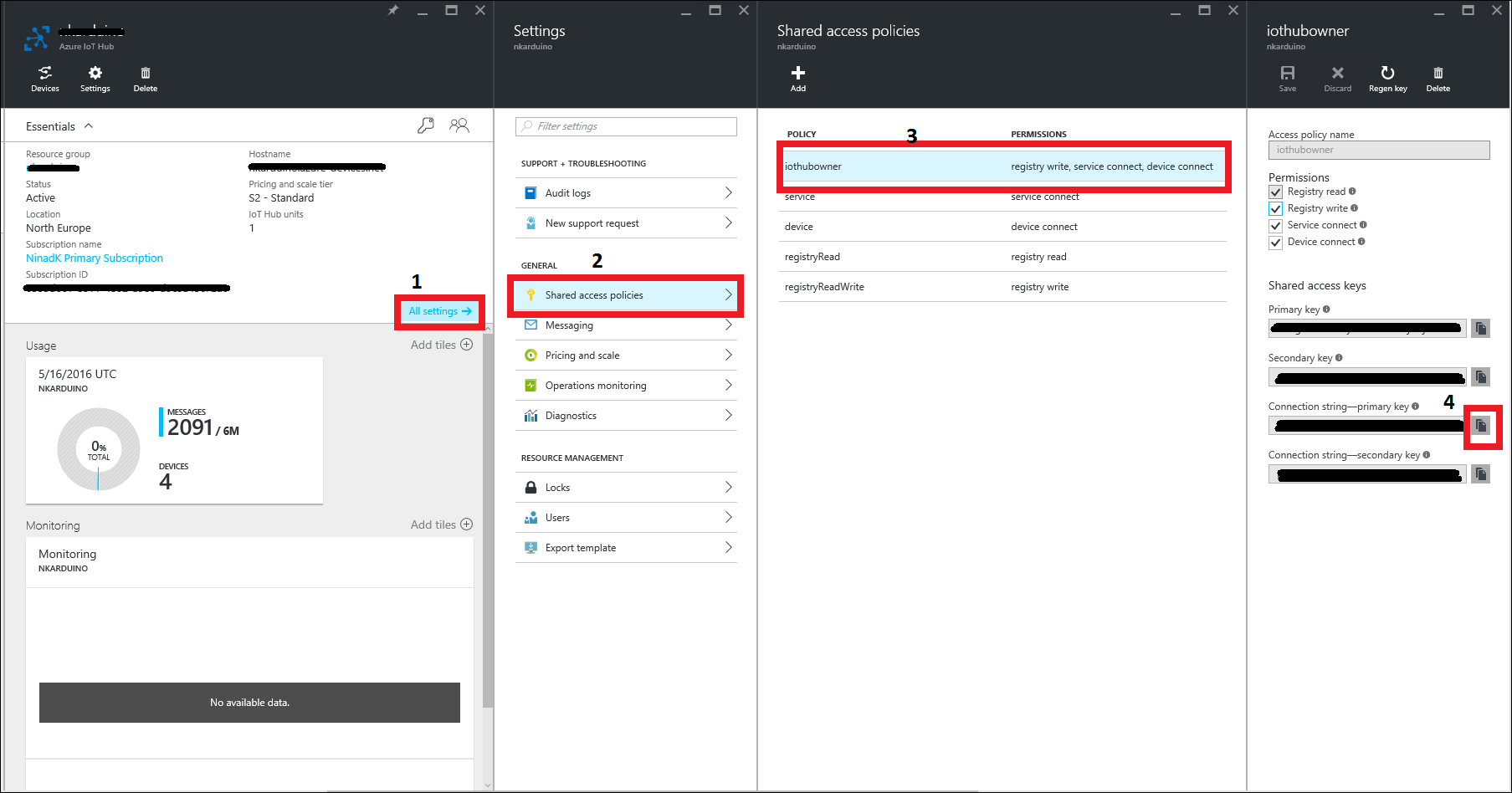


1. Finally click the “Create Solution” button. The solution should be available inside the Azure subscription in **10-15 minutes’ time**. Once the solution is ready, it will display following screen. Click on the ‘Launch’ button to see the simulated devices in action



* 1. You might need to enter your login credentials and Accept the permissions screen to continue

1. Next we want to add a new device to this solution.
2. Open the new Azure portal (portal.azure.com) and expand the blades of newly created Resource group. The resource group will have the same name that you gave to the solution in Step 5. Make a note of following items from the Azure IoT Hub that is created within the resource group.
   1. Select the Azure IoT hub within the resource group components 🡪 All Settings 🡪 Shared Access Policies 🡪 IotHubOwner 🡪 Shared access Keys 🡪 Connection String 🡪 primary key



* 1. Whilst there, also make a note of the Event-Hub compatible name of the IoT solution. All Settings 🡪 Messaging 🡪 Event Hub-compatible name. click on the Copy icon next to the text box to copy the name.

1. Download the C# solution from Github locally.
2. Open the Visual studio solution ‘**IotHub**’ inside Visual Studio 2015. Open the ‘**program.cs**’ file inside the project ‘**CreateDeviceIdentity**’. Replace the value of ‘**connectionString**” variable with the values that you had previously copied from the Azure portal.
   1. If the ‘**deviceId**’ already exists, you can change the value of the ‘**deviceId**’ as well. If this is the first time you are executing this program against the solution, it’s not necessary.

static string connectionString = "[Replace your connection string here]";

static string deviceId = "nk\_arduino\_device\_1";

1. Follow the steps [here](http://social.technet.microsoft.com/wiki/contents/articles/32975.iot-suite-remote-monitoring-adding-live-and-simulated-devices.aspx) to complete registering the device into the suite. A sample file has already been added into the Github repository.
2. Compile and execute the program. The output console window will display the device key allocated by the IoT suite. Make a note of this device key.
3. We now shouls have the three key information that we need to make our device talk with the Azure ioT solution. These are - Device Id, Device key and Event Hub-compatible name
4. Open the **remote\_monitoring.ino** solution (under …**Adafruit\Microsoft Arduino IoT Starter Kit\remote\_monitoring\** folder ) inside the Arduino IDE.
5. Execute steps 1.6 🡪 1.9 as described in the Microsoft documentation [here](https://azure.microsoft.com/en-gb/documentation/samples/iot-hub-c-m0wifi-getstartedkit/)
6. You should see the values provided by the device being displayed on your Azure web portal

Getting the thing to work.

1. It would appear that the message has been published to the buffer to be delivered, but nothing would happen.
2. Once you do have the wireless installed properly, following error you might get

Error: Time:Wed May 18 09:53:33 2016 File:C:\Users\ninadk\Documents\Arduino\libraries\AzureIoT\src\sdk\jsonencoder.c Func:JSONEncoder\_EncodeTree Line:92 (result = JSON\_ENCODER\_ERROR)

Error: Time:Wed May 18 09:53:33 2016 File:C:\Users\ninadk\Documents\Arduino\libraries\AzureIoT\src\sdk\datamarshaller.c Func:DataMarshaller\_SendData Line:193 (result = DATA\_MARSHALLER\_JSON\_ENCODER\_ERROR)

Error: Time:Wed May 18 09:53:33 2016 File:C:\Users\ninadk\Documents\Arduino\libraries\AzureIoT\src\sdk\datapublisher.c Func:DataPublisher\_EndTransaction Line:271 (result = DATA\_PUBLISHER\_MARSHALLER\_ERROR)

Error: Time:Wed May 18 09:53:33 2016 File:C:\Users\ninadk\Documents\Arduino\libraries\AzureIoT\src\sdk\agenttypesystem.c Func:Destroy\_AGENT\_DATA\_TYPE Line:1055 invalid agentData

Error: Time:Wed May 18 09:53:33 2016 File:C:\Users\ninadk\Documents\Arduino\libraries\AzureIoT\src\sdk\iotdevice.c Func:Device\_EndTransaction Line:200 (result = DEVICE\_DATA\_PUBLISHER\_FAILED)

Error: Time:Wed May 18 09:53:33 2016 File:C:\Users\ninadk\Documents\Arduino\libraries\AzureIoT\src\sdk\codefirst.c Func:CodeFirst\_SendAsync Line:889 (result = CODEFIRST\_DEVICE\_PUBLISH\_FAILED)

1. I believe one needs to install the