
A “Year of IoT” with MenterMon and NorthWalesTech

IoT Sensors Workshop

The guide to learning how to build IoT Sensors as part of a workshop with MenterMon and NorthWalesTech. **#yoiot**



Ymddiriedolaeth
Elusennol Ynys Môn
*Isle of Anglesey
Charitable Trust*



This work is licensed under a

[Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).

OVERVIEW

The Workshop

In this workshop you will first register a sensor device on *TheThingsNetwork*. This is a public network for receiving wireless sensor data using LoRaWAN and sending over the Internet. You will then create the device using a LoRaWAN-enabled development board and attach various sensors to it. You will be able to see the sensor data, transmitted over LoRaWAN, and viewed on the online portal.

The Equipment

There are two different types of LoRaWAN enabled device to use in this workshop. The process and outcome is pretty much the same for both, but one has easier code to read, and the other a bit more tricky.

If you are a beginner to programming for devices such as *Arduino* or indeed a beginner to programming at all, then it is recommended to use the **TTN UNO** kit (green box) and instructions in section 2.

Otherwise, if you are proficient at programming and have some experience with, for example, Arduino boards, then opt for the **FeatherMO** kit (red box) and instructions in section 3.

We'll be working in groups. Please invite a beginner to work with you and help them get to grips with the process.

There may be the opportunity to try out both kits during this session, but there will be other sessions in the future to spend more time with either.

1. The Things Network (TTN)

Summary

TheThingsNetwork (TTN) is a public, open, community built network of gateways that receive wireless sensor data from LoRaWAN enabled devices. Anyone can register with a device, make it transmit, and retrieve the sensor data. That's exactly what we'll do now.

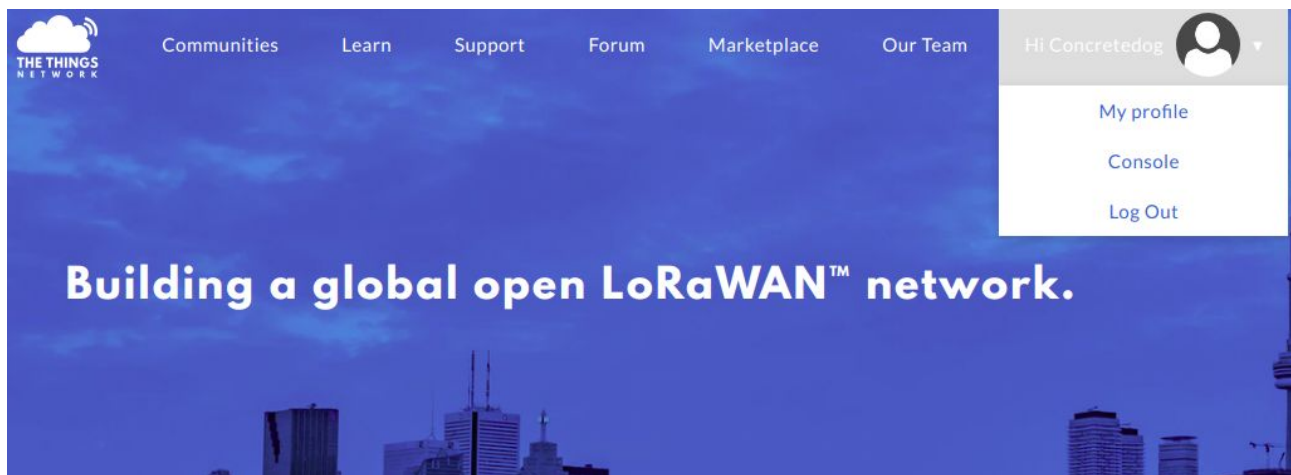
Everyone should create an account and register their own device. Please take turns to do this in your groups.

We will each:

- Register for an account
- Create a new application
- Register a device.

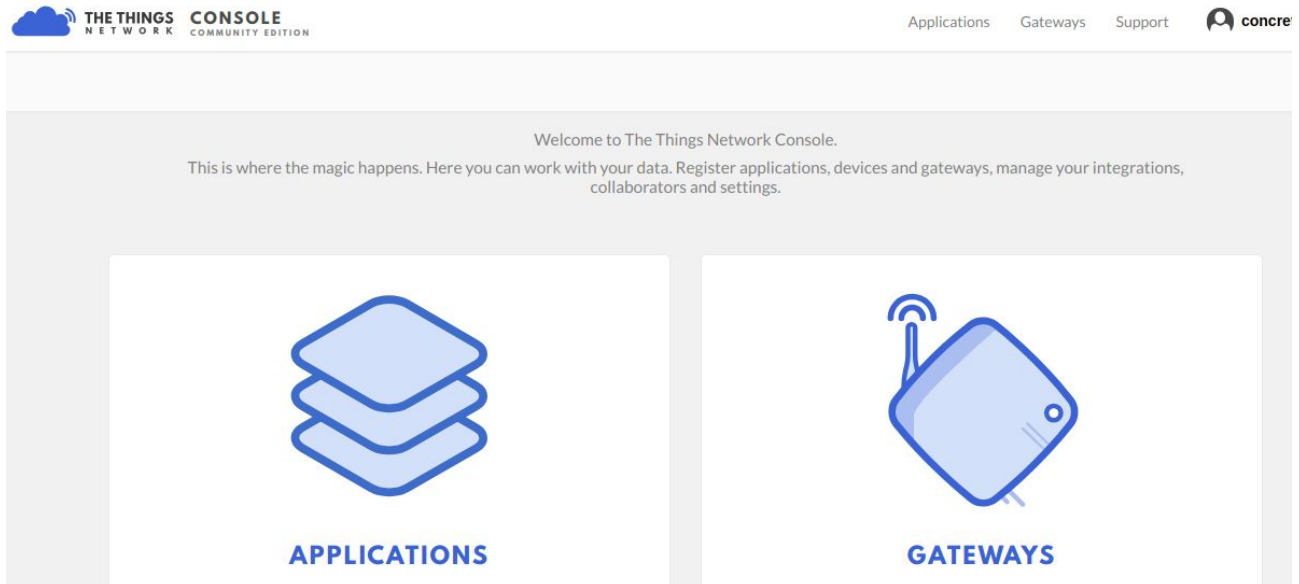
1-A Register an Account

- Go to the things network and register an account <https://www.thethingsnetwork.org>
- Once registered, on your dashboard click the “**console**” link by clicking the drop down menu next to your username on the top right hand side.

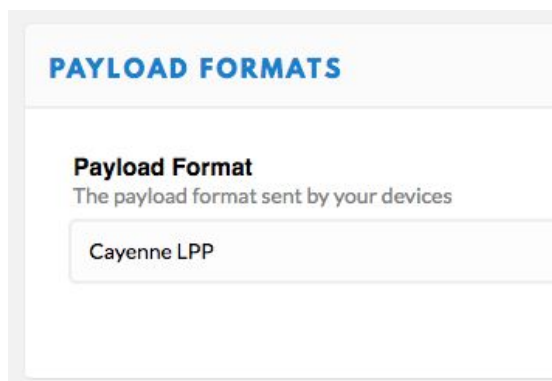


1-B Create an Application

An application is just a group of devices that will do a similar thing.



- In the console click the **applications** icon on the left
- Click the “**add application**” link and give your application an ID and a description. Note application ID’s have to be unique on the things network so if you try “test123” for example you will probably discover it is unavailable!
- Select the application to display the **Application Overview** page,
- Then choose the **Payload Formats** Tab and select **CayenneLPP** from the dropdown. You’ll learn what this means later.
- Click **Save**



1-C Register a Device

Each device must first be registered before the data can be received. The process for both device types is the same.

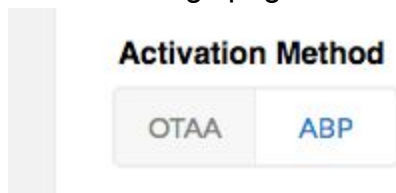
- Select the application to display the **Application Overview** page, then choose the **Devices** Tab and click the **register device** link. (or **get started by registering one**)
- Give the device a unique **Device ID**, it *doesn't matter what at this stage*.
- Click the button on the left of the Device EUI input to automatically generate this.



- Click the green register button at the bottom of the page.

In the resulting device overview you will notice that the activation method is set to **OTAA**, for these workshops it's important that we change this and use the **ABP** method.

- Click the settings tab in the upper right hand side
- On the settings page scroll down and click the **ABP** button next to the OTAA button



- Then at the bottom of the page un-tick the "**frame counter checks**" box. Note that this will bring up a yellow error message warning this is less secure and for development use only. Ignore this warning and click save.



- Click **Save**

2. Build An IoT Sensor with The Things Uno

The **TTN Uno** is an Arduino compatible development board, designed by the creators of TheThingsNetwork to make it really easy to get started with LoRaWAN and IoT. We've coupled it with sensors from the **Grove** family which makes it easy to plug and play!

Summary

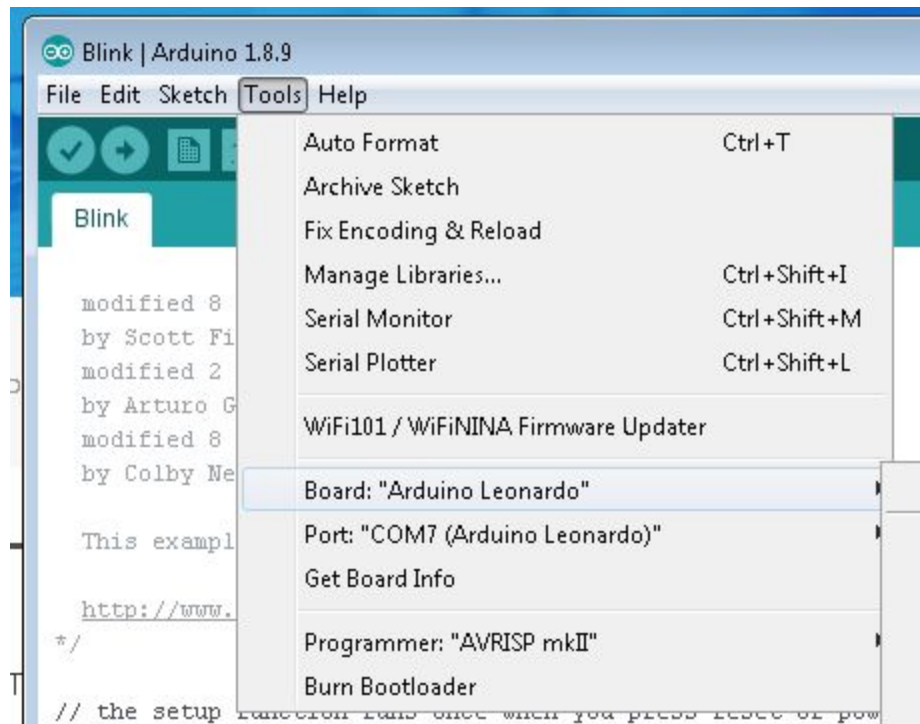
In this section you will:

- Setup the Arduino Developer Environment
- Get The Things Uno connected to your computer and make a basic program
- Connect a sensor and read some data.
- Transmit the data over the LoRaWAN TTN network and read the data.

2-A Setup the Arduino Environment and Connect The Things Uno

- Download and install the latest arduino IDE from <https://www.arduino.cc/en/Main/Software#download>
- If you are on windows, don't forget to install the device drivers included in the download.
- Plug in the device with the Micro USB cable.
- *Windows users:* If your device does not show up as a "Leonardo" when you plug it in, then please follow these steps:
 - a. Download drivers from here: <https://www.ftdichip.com/Drivers/VCP.htm>
 - b. Extract the ZIP somewhere you can find it.
 - c. Refer to the manual driver installation guide here: <https://www.arduino.cc/en/Guide/DriverInstallation>

- In the Arduino IDE click **Tools > Board** and then check it is set to **“Arduino Leonardo”**
- Click **Tools > Port** and select the one that points to a Leonardo.



- Next open a basic example as follows: **File > Examples > 01.Basics > Blink**



- Now click the compile button



- Upload this sketch/program to the Things Uno by clicking the upload button:
- You should see that a green LED labelled “13” near the usb socket on the Things Uno board is now flashing regularly once a second. Let’s change the rate at which it flashes to ensure we can edit, then upload new versions.
- Back in the blink sketch scroll down and change the delay values that are 1000 to 200

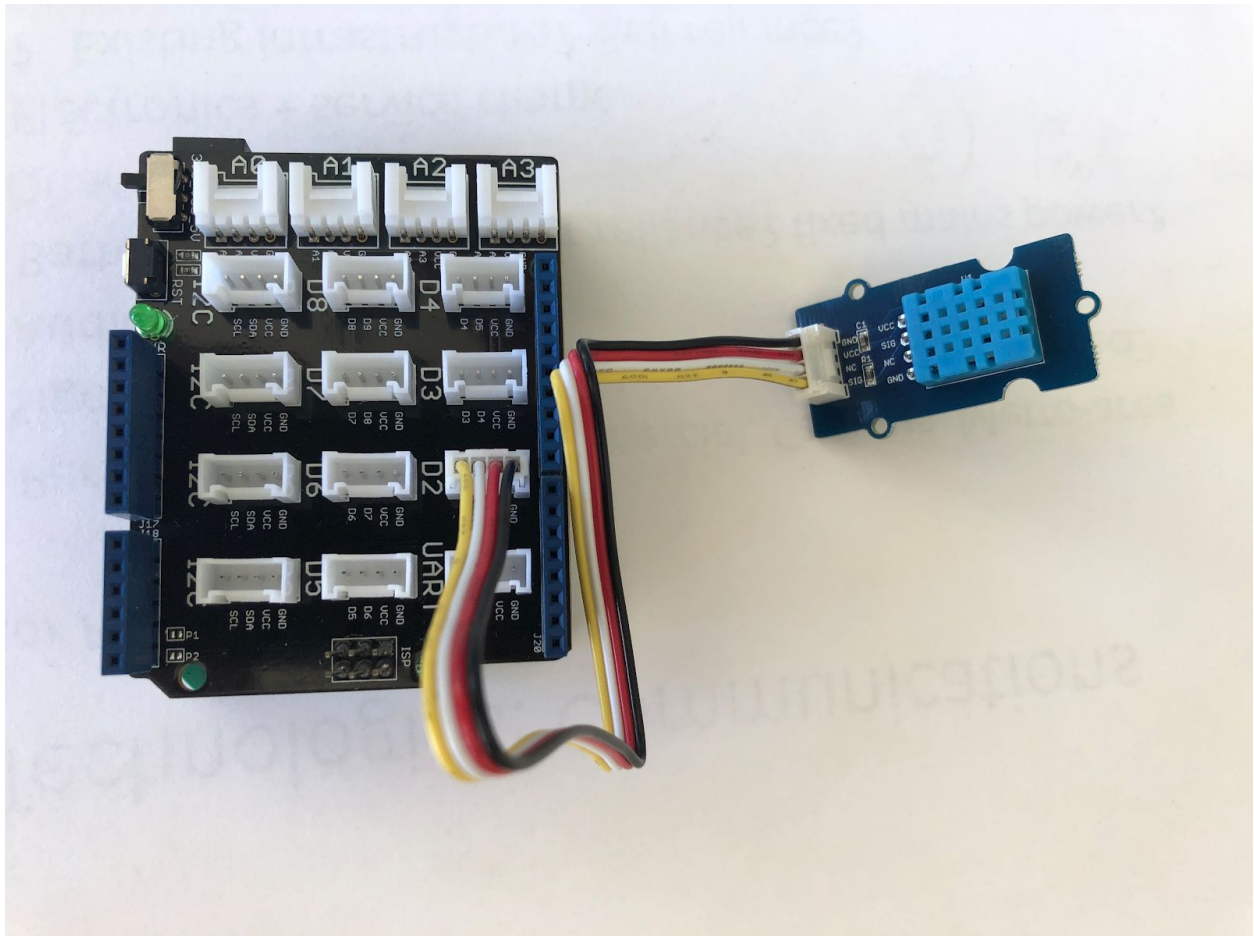
```
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(200); // wait for a second
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
  delay(200); // wait for a second
}
```

- Then repeat the **Compile** and **Upload** steps above to verify and upload the sketch. You should now see the LED is flashing much more quickly!

2-B Connecting a Sensor, Read the Environment

Connect the sensor

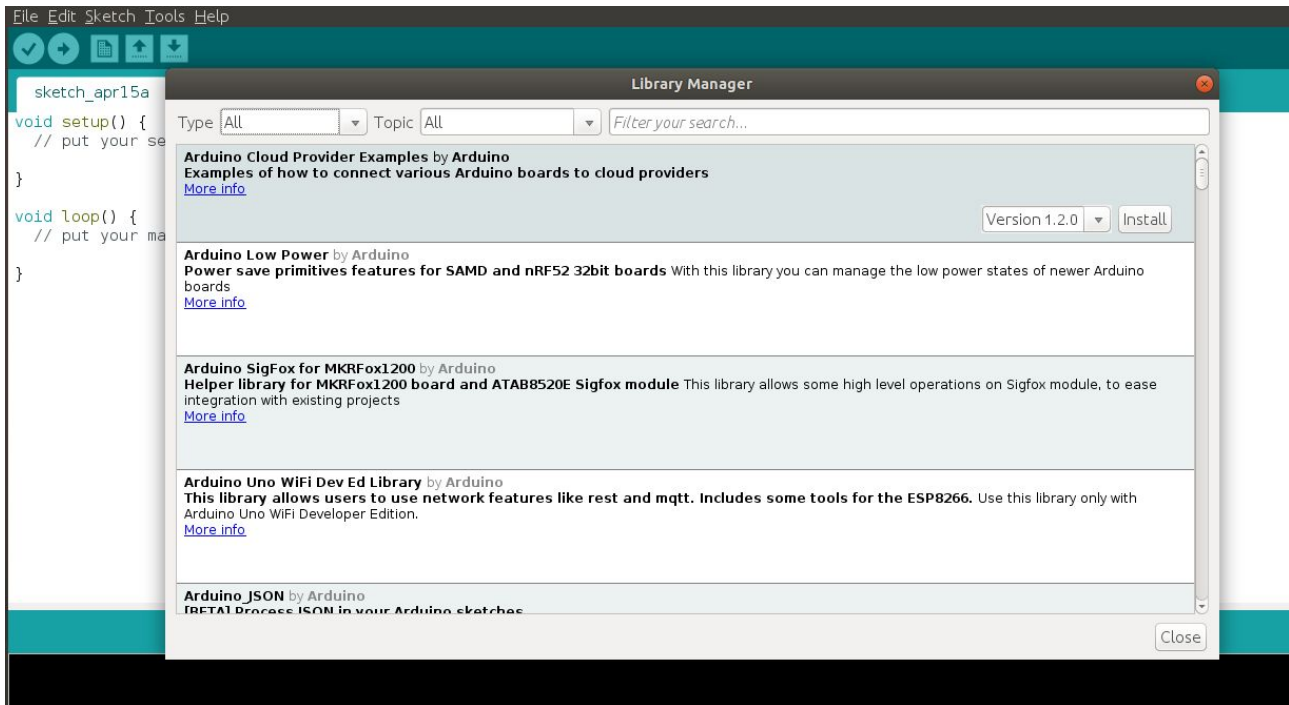
- Find the Grove Temperature and Humidity Sensor in the Kit.
- Use a connector wire and plug it into the Adapter using the port marked D2



Load the library.

Most of the sensors have a pre-built “library” of code which takes the hard work out of getting sensor data from the electronics. We need to download the library and add it to the Arduino IDE so we can use it in our code. We will use the **Arduino Library Manager**

- Open the Arduino IDE and then click **Tools > Manage Libraries**



- Using the search box, search for **Grove DHT11**





- Then click **Install** to download and install the library into the Arduino IDE
- Click **close** to close the library manager.

Load the Program

Our first program can be found here, browse here to have a look:

https://github.com/NorthWalesTech/yoiot-ThingsUno/blob/master/1_TempHum/1_TempHum.ino

- In the Arduino IDE, choose **File > New**
- Delete all the new code in the editor window *just get rid of the empty setup() and loop()*
- Copy and Paste the program code from the link above
- Save it with project name **TempHum**
- Click the Arrow button  to compile and upload the code the TTN UNO
- Click the Magnifying glass  to open the serial monitor and inspect the sensor data.
- **#winner** points awarded for the maximum temperature and/or humidity you can observe. (no blow-torches!) Change the code to record the max and min for each.

3-C Transmit the Data

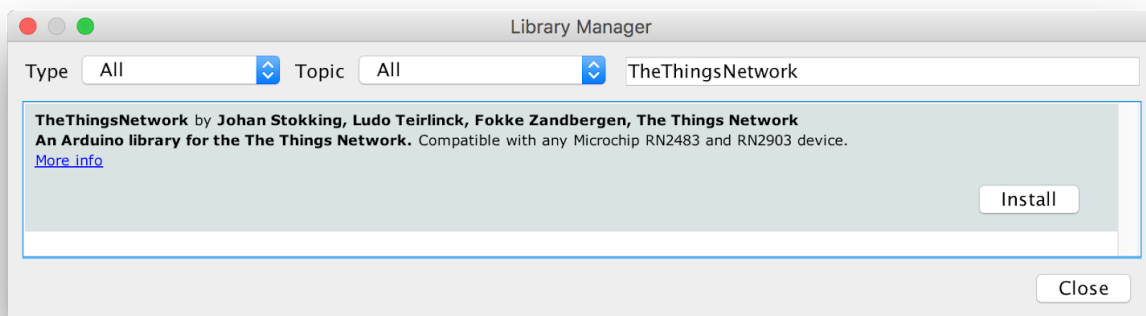
Summary of steps coming up.....

- Load more libraries
- Load the new program
- Edit it for your TTN device keys
- Upload and run it!

Load more libraries

We are now going to add another library to enable our code to use the LoRaWAN transmitter on the ThingsUno device. Again using the **Arduino Library Manager**.

- Open the Arduino IDE and then click **Tools > Manage Libraries**
- Using the search box, search for **TheThingsNetwork**



- Then click **Install** to download and install the library into the Arduino IDE

Load the Program

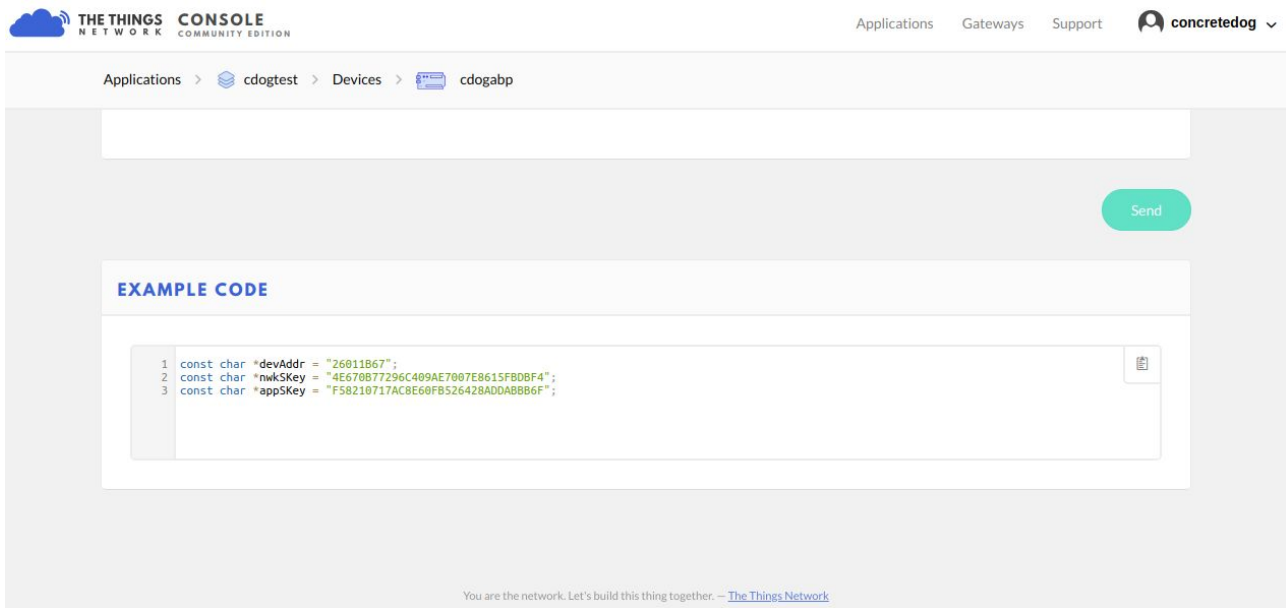
Our transmitter program can be found here, browse here to have a look:

https://github.com/NorthWalesTech/yoiot-ThingsUno/blob/master/2_TempHum_TTN_ABP/2_TempHum_TTN_ABP.ino

- In the Arduino IDE, choose File > New
- Delete all the new code in the editor window *just get rid of the empty setup() and loop()*
- Copy and Paste the program code from the link above
- Save it with project name **TempHumLoRaWAN**


We now need to copy some settings from the TTN console to this code.

- Head to the TTN console and find the device settings page.
- Scroll to the bottom and locate the **Example Code** section.



- Copy these three lines
- Back in the Arduino Code editor, replace the the corresponding lines near the top of the file to replace the ones with all '0's with your own specific settings.

So replace these;



```
File Edit Sketch Tools Help

SendABP

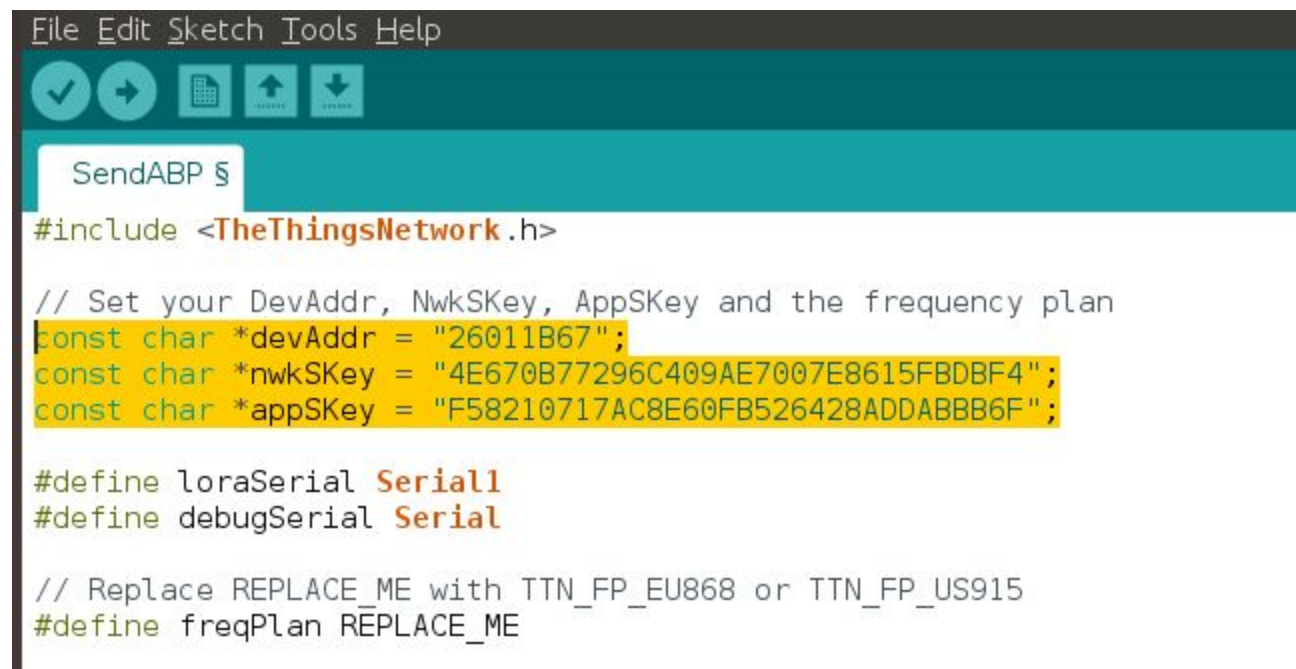
#include <TheThingsNetwork.h>

// Set your DevAddr, NwkSKey, AppSKey and the frequency plan
const char *devAddr = "00000000";
const char *nwkSKey = "00000000000000000000000000000000";
const char *appSKey = "00000000000000000000000000000000";

#define loraSerial Serial1
#define debugSerial Serial

// Replace REPLACE ME with TTN FP EU868 or TTN FP US915
```

with the ones you copied



```
File Edit Sketch Tools Help

SendABP §



#include <TheThingsNetwork.h>

// Set your DevAddr, NwkSKey, AppSKey and the frequency plan
const char *devAddr = "26011B67";
const char *nwkSKey = "4E670B77296C409AE7007E8615FBDBF4";
const char *appSKey = "F58210717AC8E60FB526428ADDABBB6F";

#define loraSerial Serial1
#define debugSerial Serial

// Replace REPLACE_ME with TTN_FP_EU868 or TTN_FP_US915
#define freqPlan REPLACE_ME
```

Compile and Run FTW!

- Click the Arrow button  to compile and upload the code the Feather
- Click the Magnifying glass  to open the serial monitor and inspect the sensor data!!
- Take a look at the **Data** tab on TheThingsNetwork console for your device. You should see transmissions being received. You should also see the data fields behind decoded `temperature_1: 20.0` etc.

References and Links

1. <https://github.com/NorthWalesTech/yoiot-ThingsUno>
2. <https://www.thethingsnetwork.org/docs/devices/uno/quick-start.html>
3. <https://www.arduino.cc/en/Guide/ArduinoLeonardoMicro#toc2>
4. <https://www.arduino.cc/en/Guide/Troubleshooting#toc16>

3. Build An IoT Sensor with the Feather M0

The Feather M0 is an Arduino compatible mini-computer [1].
We're using the model with a LoRa radio onboard.

Summary

In this section you will:

1. Setup the Arduino Developer Environment
2. Get the Feather M0 connected to your computer and make a basic program
3. Connect a sensor and read some data.
4. Transmit the data over the LoRaWAN TTN network and read the data.

3-A Setup the Arduino Environment and Connect the Feather

First we will need to install the Arduino developer toolkit, called the *IDE*.

- Download and install the latest arduino IDE from <https://www.arduino.cc/en/Main/Software#download>
- If you are on windows, don't forget to install the device drivers included in the download.

Next we need to enhance the Arduino IDE to talk to the Adafruit Feather M0 boards which are slightly different to the more common Arduino devices.

*The next few pages are extracted from the official guide. We've just pulled out the relevant pages here, but the complete info about the Feather M0 can be found [in the links section](#), if you are interested, but you **don't** need to read this now.....*

Follow the section of the guide included here to configure the Arduino IDE to allow us to program the Feather M0 devices.

In this guide you'll also get "*Blink*" running - the most basic program to flash the light on the board. **For extra #winner points, change the blink to quickly flash 3-times a second.**

Once you've done this, our document continues to do some more fun tasks.

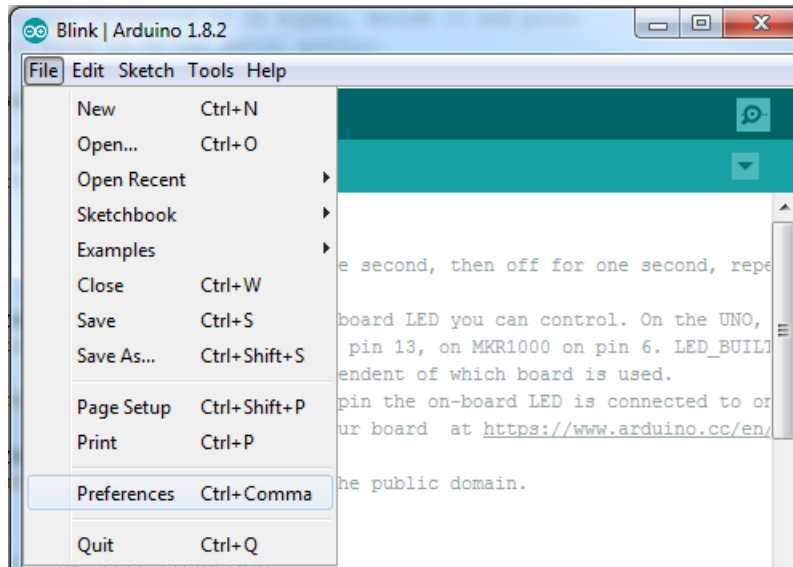
Arduino IDE Setup

The first thing you will need to do is to download the latest release of the Arduino IDE. You will need to be using **version 1.8** or higher for this guide

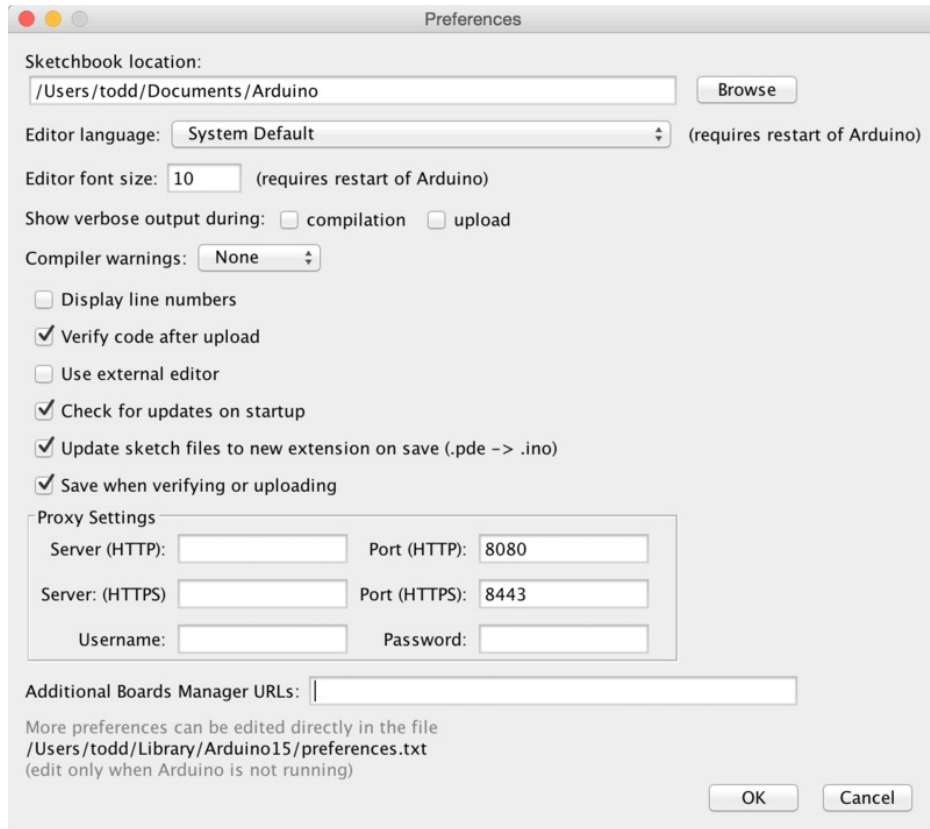
<https://adafru.it/f1P>

<https://adafru.it/f1P>

After you have downloaded and installed the **latest version of Arduino IDE**, you will need to start the IDE and navigate to the **Preferences** menu. You can access it from the **File** menu in *Windows* or *Linux*, or the **Arduino** menu on *OS X*.



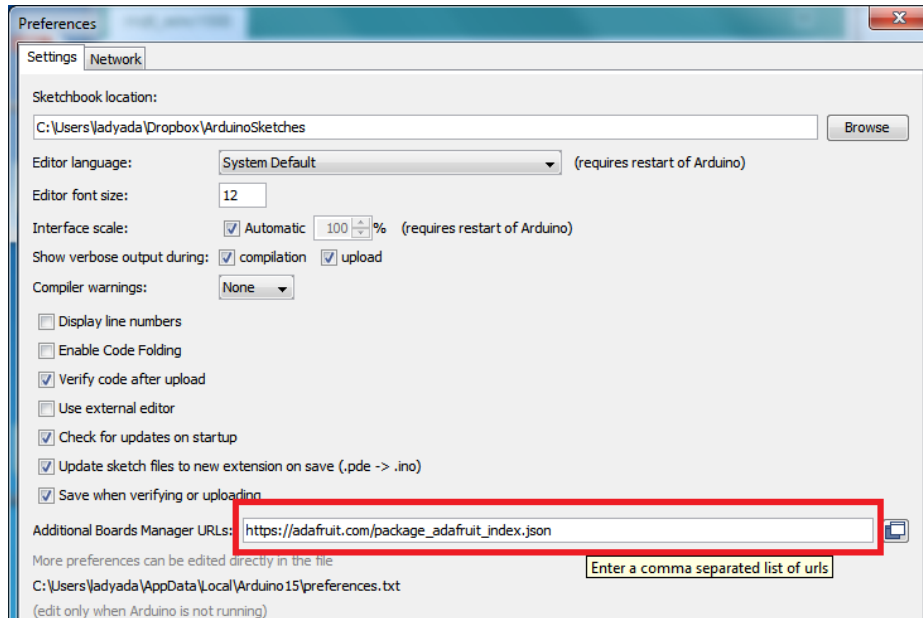
A dialog will pop up just like the one shown below.



We will be adding a URL to the new **Additional Boards Manager URLs** option. The list of URLs is comma separated, and *you will only have to add each URL once*. New Adafruit boards and updates to existing boards will automatically be picked up by the Board Manager each time it is opened. The URLs point to index files that the Board Manager uses to build the list of available & installed boards.

To find the most up to date list of URLs you can add, you can visit the list of [third party board URLs on the Arduino IDE wiki \(https://adafruit.github.io/arduino-board-index/package_adafruit_index.json\)](https://adafruit.github.io/arduino-board-index/package_adafruit_index.json). We will only need to add one URL to the IDE in this example, but *you can add multiple URLs by separating them with commas*. Copy and paste the link below into the **Additional Boards Manager URLs** option in the Arduino IDE preferences.

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



Here's a short description of each of the Adafruit supplied packages that will be available in the Board Manager when you add the URL:

- **Adafruit AVR Boards** - Includes support for Flora, Gemma, Feather 32u4, Trinket, & Trinket Pro.
- **Adafruit SAMD Boards** - Includes support for Feather M0 and M4, Metro M0 and M4, ItsyBitsy M0 and M4, Circuit Playground Express, Gemma M0 and Trinket M0
- **Arduino Leonardo & Micro MIDI-USB** - This adds MIDI over USB support for the Flora, Feather 32u4, Micro and Leonardo using the [arcore project \(https://adafru.it/eSI\)](https://adafru.it/eSI).

If you have multiple boards you want to support, say ESP8266 and Adafruit, have both URLs in the text box separated by a comma (,)

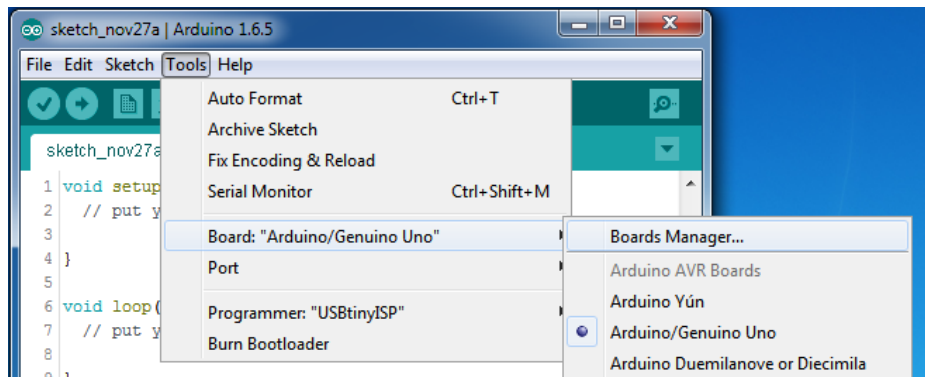
Once done click **OK** to save the new preference settings. Next we will look at installing boards with the Board Manager.

Now continue to the next step to actually install the board support package!

Using with Arduino IDE

The Feather/Metro/Gemma/Trinket M0 and M4 use an ATSAM21 or ATSAM51 chip, and you can pretty easily get it working with the Arduino IDE. Most libraries (including the popular ones like NeoPixels and display) will work with the M0 and M4, especially devices & sensors that use I2C or SPI.

Now that you have added the appropriate URLs to the Arduino IDE preferences in the previous page, you can open the **Boards Manager** by navigating to the **Tools->Board** menu.



Once the Board Manager opens, click on the category drop down menu on the top left hand side of the window and select **All**. You will then be able to select and install the boards supplied by the URLs added to the preferences.

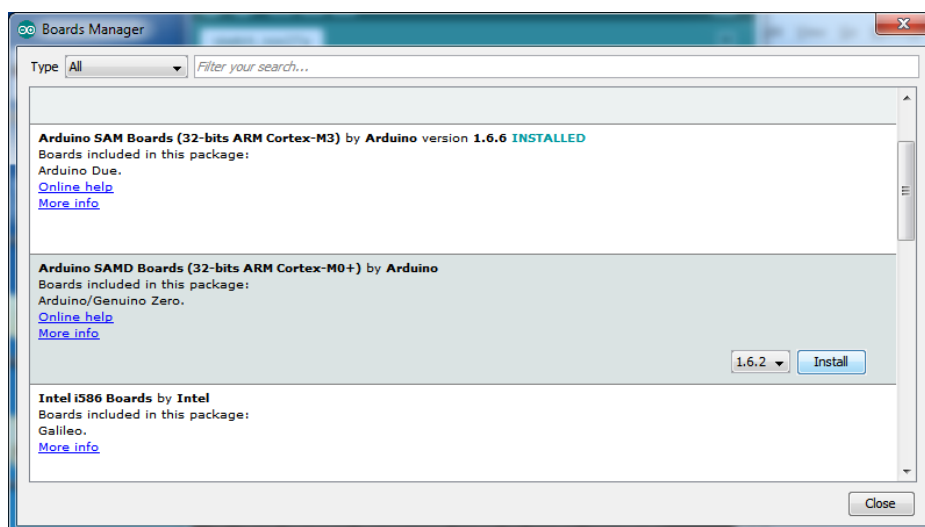


Remember you need **SETUP** the Arduino IDE to support our board packages - see the previous page on how to add adafruit's URL to the preferences

Install SAMD Support

First up, install the latest **Arduino SAMD Boards** (version **1.6.11** or later)

You can type **Arduino SAMD** in the top search bar, then when you see the entry, click **Install**

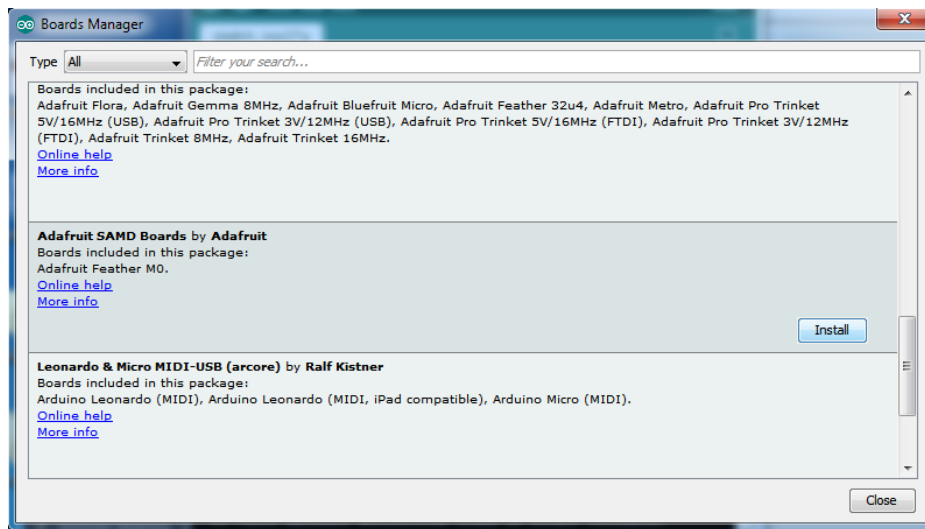


Install Adafruit SAMD

Next you can install the Adafruit SAMD package to add the board file definitions

Make sure you have **Type All** selected to the left of the *Filter your search...* box

You can type **Adafruit SAMD** in the top search bar, then when you see the entry, click **Install**

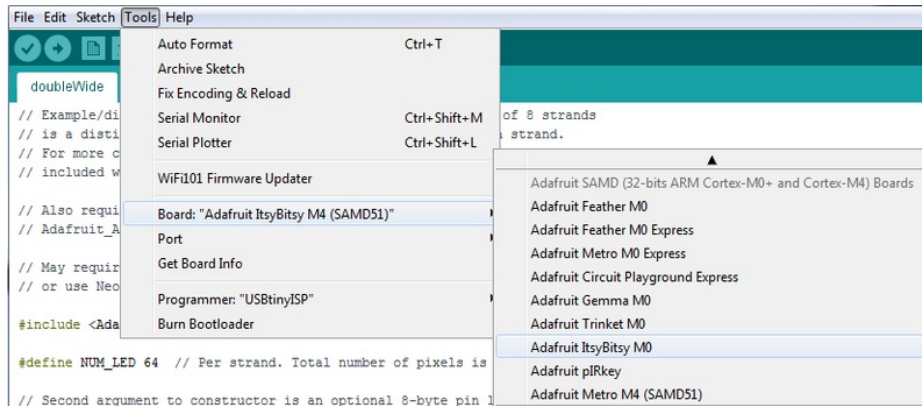


Even though in theory you don't need to - I recommend rebooting the IDE

Quit and reopen the Arduino IDE to ensure that all of the boards are properly installed. You should now be able to select and upload to the new boards listed in the **Tools->Board** menu.

Select the matching board, the current options are:

- **Feather M0** (for use with any Feather M0 other than the Express)
- **Feather M0 Express**
- **Metro M0 Express**
- **Circuit Playground Express**
- **Gemma M0**
- **Trinket M0**
- **ItsyBitsy M0**
- **Hallowing M0**
- **Crickit M0** (this is for direct programming of the Crickit, which is probably not what you want! For advanced hacking only)
- **Metro M4 Express**
- **ItsyBitsy M4 Express**
- **Feather M4 Express**
- **Trellis M4 Express**
- **Grand Central M4 Express**



Install Drivers (Windows 7 & 8 Only)

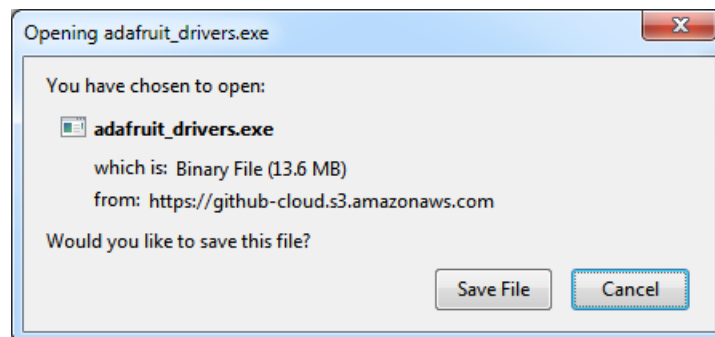
When you plug in the board, you'll need to possibly install a driver

Click below to download our Driver Installer

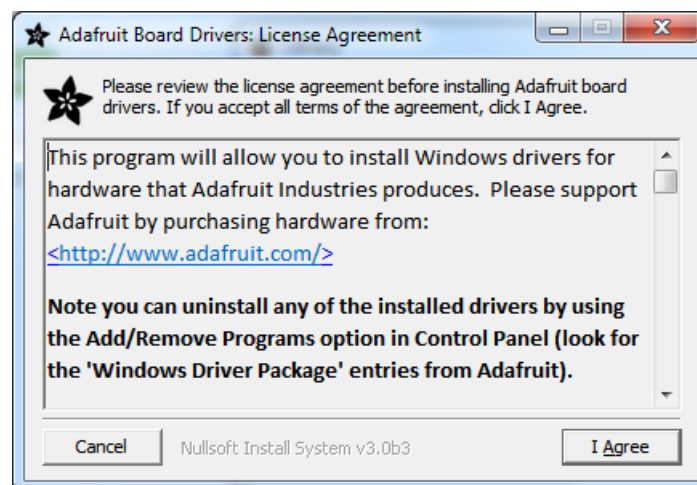
<https://adafru.it/EC4>

<https://adafru.it/EC4>

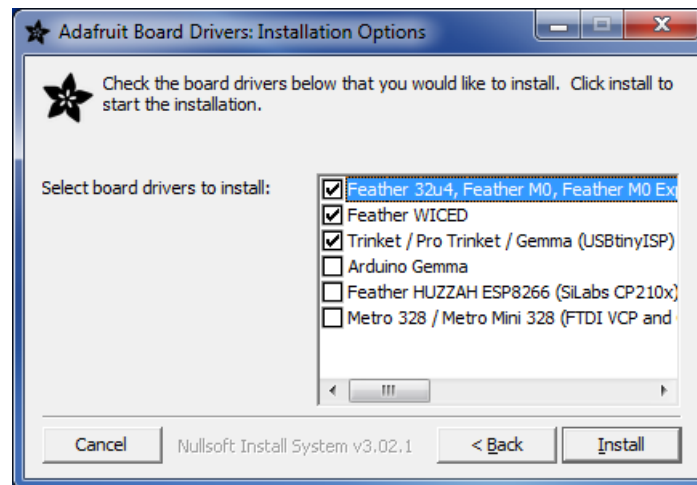
Download and run the installer



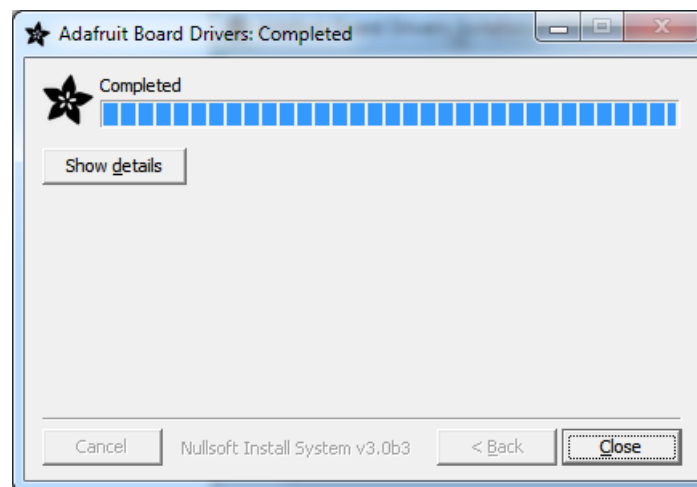
Run the installer! Since we bundle the SiLabs and FTDI drivers as well, you'll need to click through the license



Select which drivers you want to install, the defaults will set you up with just about every Adafruit board!



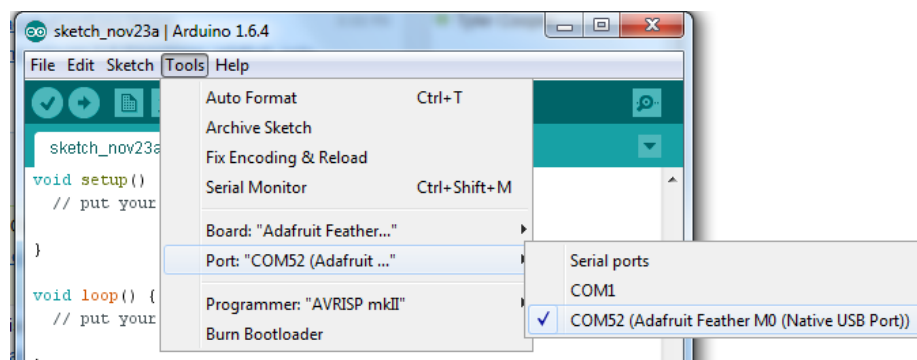
Click **Install** to do the installin'



Blink

Now you can upload your first blink sketch!

Plug in the M0 or M4 board, and wait for it to be recognized by the OS (just takes a few seconds). It will create a serial/COM port, you can now select it from the drop-down, it'll even be 'indicated' as Trinket/Gemma/Metro/Feather/ItsyBitsy/Trellis!



Now load up the Blink example

```
// 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
}
```

And click upload! That's it, you will be able to see the LED blink rate change as you adapt the **delay()** calls.

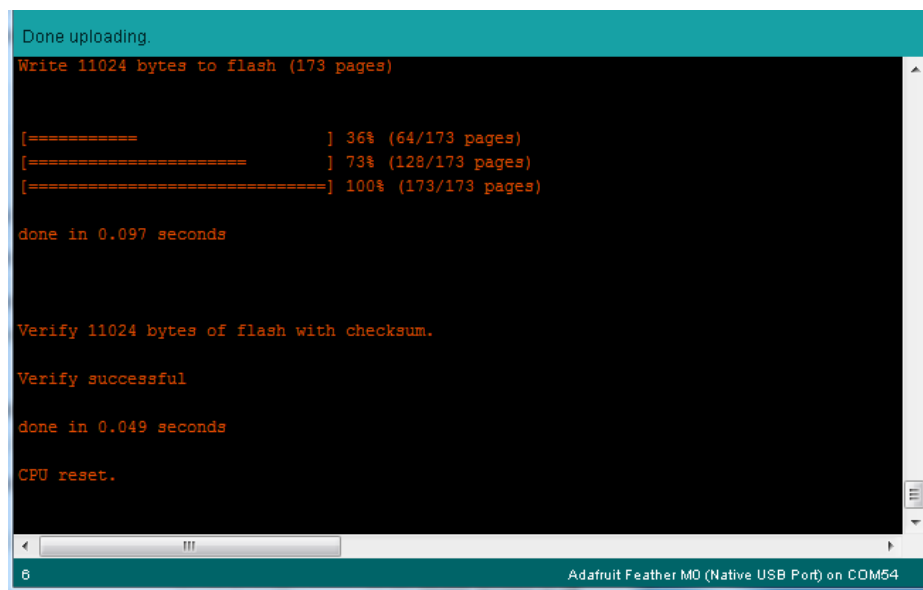
If you're using **Trellis M4 Express**, you can go to the next page cause there's no pin 13 LED - so you won't see it blink. Still this is a good thing to test compile and upload!



If you are having issues, make sure you selected the matching Board in the menu that matches the hardware you have in your hand.

Successful Upload

If you have a successful upload, you'll get a bunch of red text that tells you that the device was found and it was programmed, verified & reset

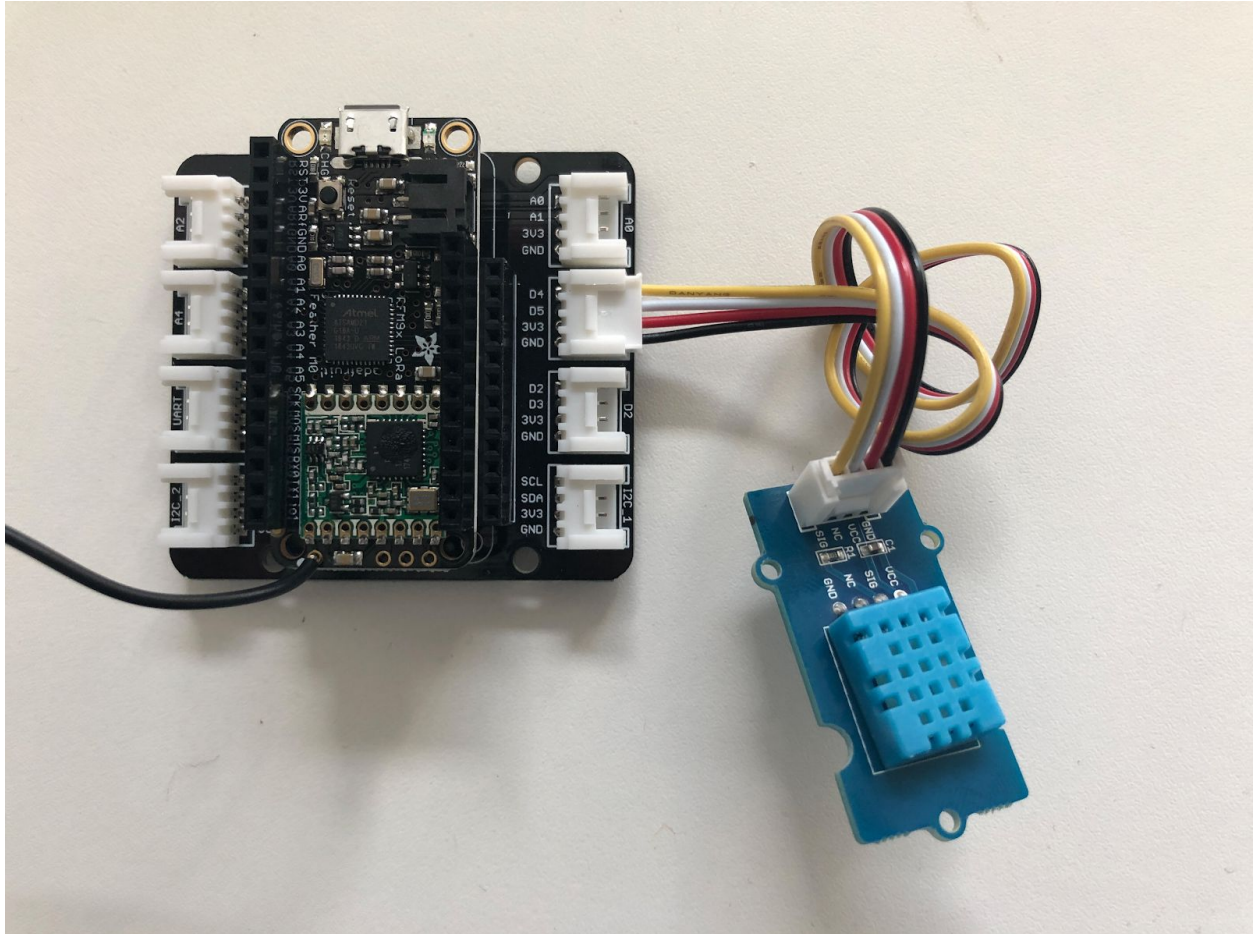


After uploading, you may see a message saying "Disk Not Ejected Properly" about the ...BOOT drive. You can ignore that message: it's an artifact of how the bootloader and uploading work.

3-B Connecting a Sensor, Read the Environment

Connect the sensor

- Find the Grove Temperature and Humidity Sensor in the Kit.
- Use a connector wire and plug it into the Adapter using the port marked D4



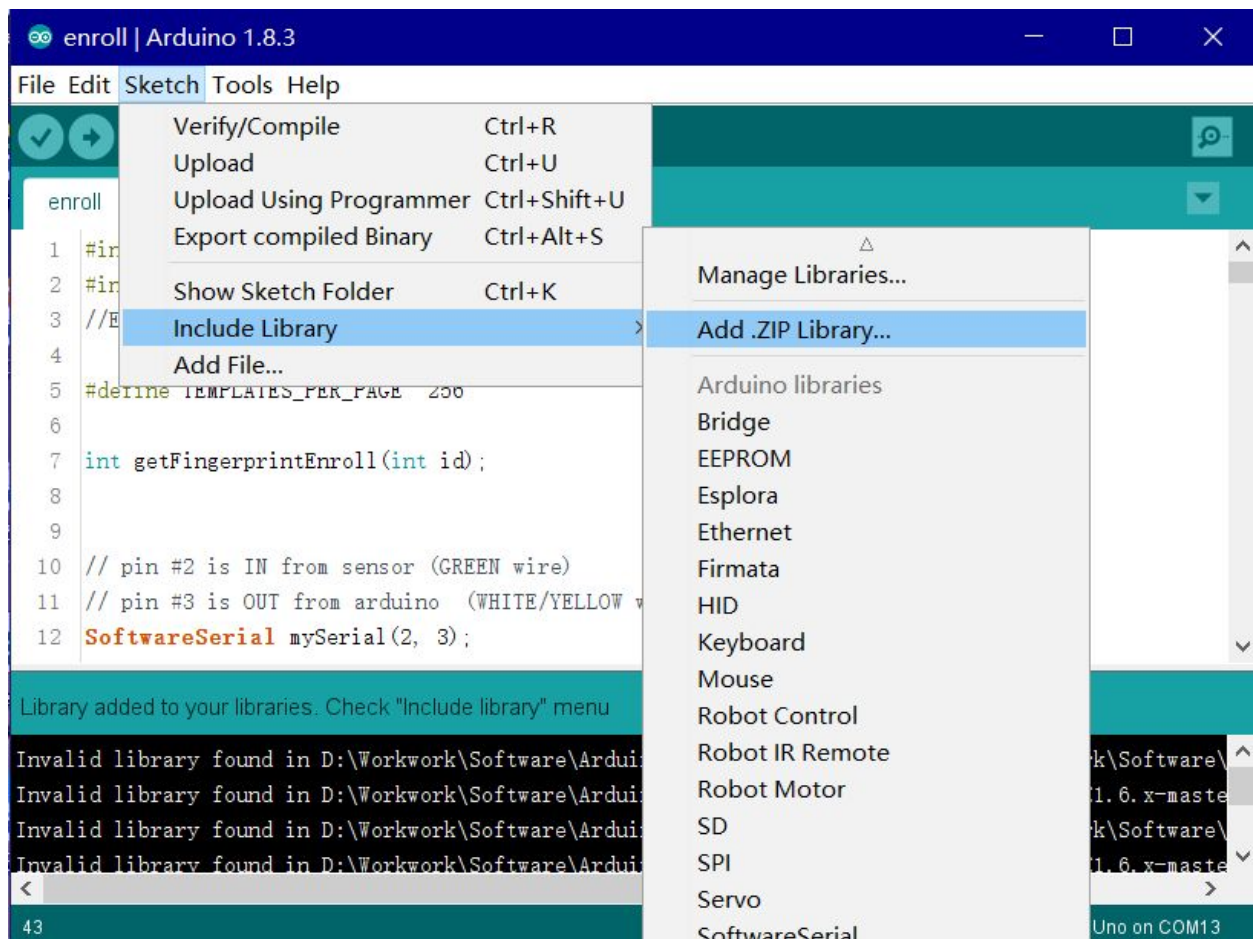
Load the library.

Most of the sensors have a pre-built “library” of code which takes the hard work out of getting sensor data from the electronics. We need to download the library and add it to the Arduino IDE so we can use it in our code

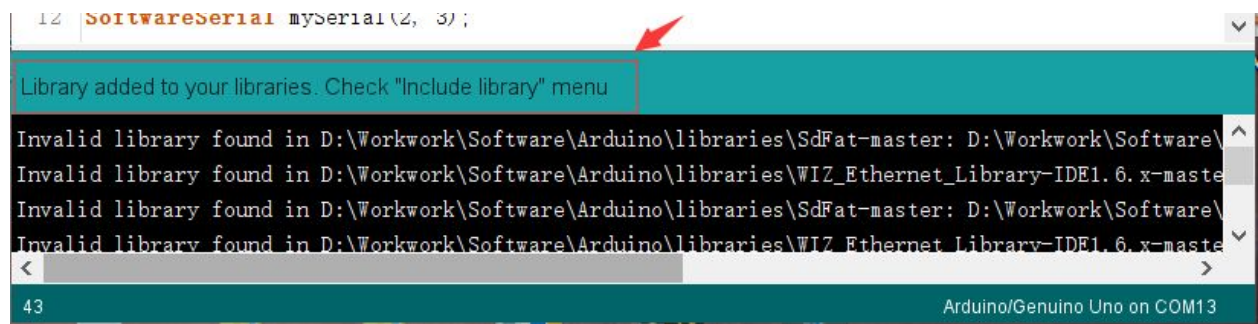
The library can be found here:

https://github.com/NorthWalesTech/voiot-featherM0LoRa/raw/master/Grove_Temperature_And_Humidity_Sensor-master.zip

- Download it to your computer..... *somewhere you can find it shortly.*
- Open the Arduino IDE, click on **Sketch > Include Library > Add .ZIP Library.**





- Select the zip file you just downloaded, and if the library installs correct, you will see Library added to your libraries in the notice window.



Load the Program

Our first program can be found here, browse here to have a look:

https://github.com/NorthWalesTech/yoiot-featherM0LoRa/blob/master/1_TempHum/1_TempHum.ino

- In the Arduino IDE, choose File > New
- Delete all the new code in the editor window just get rid of the empty setup() and loop()
- Copy and Paste the program code from the link above
- Save it with project name TempHum
- Click the Arrow button  to compile and upload the code the Feather
- Click the Magnifying glass  to open the serial monitor and inspect the sensor data.
- #winner points awarded for the maximum temperature and/or humidity you can observe. (no blow-torches!) Change the code to record the max and min for each.

3-C Transmit the Data

Summary of steps coming up.....

- Load more libraries
- Load the new program
- Edit it for your TTN device keys
- Upload and run it!

Load more libraries.

We'll now load two more arduino code libraries.

The LMIC library handles LoRaWAN protocol and CayenneLPP allows us to transmit data in a format that the TTN network can understand, so we can read it on-screen later.

- Find the ZIP bundles of these libraries at the links below.
- As before, load them using the menu: **Sketch > Include Library > Add .ZIP Library**

LMIC:

<https://github.com/NorthWalesTech/voiot-featherM0LoRa/raw/master/arduino-lmic-master-EU868.zip>

CayenneLPP:

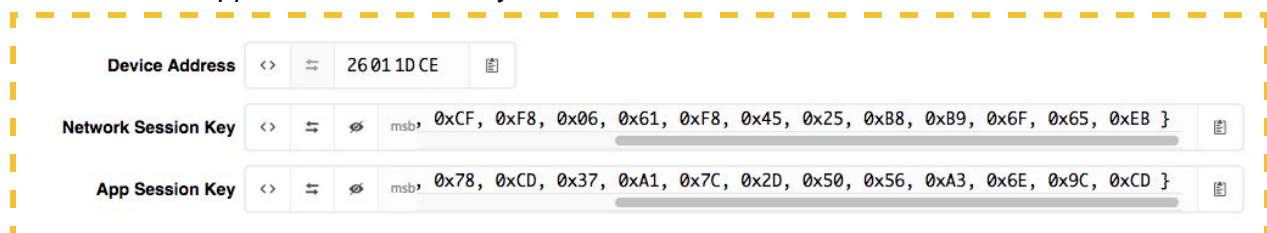
<https://github.com/NorthWalesTech/voiot-featherM0LoRa/raw/master/CayenneLPP-master.zip>

Load the Program

Our transmitter program can be found here, browse here to have a look:



https://github.com/NorthWalesTech/yoiot-featherMOLoRa/blob/master/2_TempHum_LoRa_TTN_ABP/2_TempHum_LoRa_TTN_ABP.ino

- In the Arduino IDE, choose File > New
- Delete all the new code in the editor window *just get rid of the empty setup() and loop()*
- Copy and Paste the program code from the link above
- Save it with project name **TempHumLoRaWAN**
- Find your new device settings on the TTN console - Copy and paste the following parameters:
 - ◆ Device Address
 - ◆ Network Session Key
 - ◆ Application Session Key





The screenshot shows the TTN console interface for a device. It has three input fields, each with a dropdown menu for format (hex, decimal, binary) and a copy icon. The 'Device Address' field is set to 'hex' and contains '26 01 1D CE'. The 'Network Session Key' field is set to 'hex' and contains a 16-byte array: '0xCF, 0xF8, 0x06, 0x61, 0xF8, 0x45, 0x25, 0x88, 0xB9, 0x6F, 0x65, 0xEB'. The 'App Session Key' field is set to 'hex' and contains a 16-byte array: '0x78, 0xCD, 0x37, 0xA1, 0x7C, 0x2D, 0x50, 0x56, 0xA3, 0x6E, 0x9C, 0xCD'.

Take care to copy the values in the right format, as shown above.

Click the  button to change the format and the  icon to copy the value.

Here's what it should look like:

```
44 // LoRaWAN NwkSKey, network session key
45 static const PROGMEM u1_t NWKSKEY[16] = { 0x6E, 0x25, 0x85, 0x31, 0xCF, 0xF8, 0x06, 0x61, 0xF8, 0x45, 0x25, 0x88, 0xB9, 0x6F, 0x65, 0xEB };
46
47 // LoRaWAN AppSKey, application session key
48 static const u1_t PROGMEM APPSKEY[16] = { 0xD4, 0xEE, 0x26, 0x5F, 0x78, 0xCD, 0x37, 0xA1, 0x7C, 0x2D, 0x50, 0x56, 0xA3, 0x6E, 0x9C, 0xCD };
49
50 // LoRaWAN end-device address (DevAddr)
51 static const u4_t DEVADDR = 0x26011DCE; // copy from TTN device page, don't forget the leading '0x'...
```

-
- Click the Arrow button  to compile and upload the code the Feather
 - Click the Magnifying glass  to open the serial monitor and inspect the sensor data!!
 - Take a look at the **Data** tab on TheThingsNetwork console for your device. You should see transmissions being received. You should also see the data fields behind decoded `temperature_1: 20.0` etc.

Extra Tasks.

1. Plug in the battery, go for a walk!
(you can hot-swap between battery and USB, It'll charge the battery from USB)
2. Combine some extra sensor inputs. You'll find some more examples in the code repository for this workshop. <https://github.com/NorthWalesTech/yoiot-featherM0LoRa>
3. By clicking on a row in the Data section of the ThingsNetwork console, you can inspect some values. Have a look at "Estimate Airtime".
 - Change the data rate from DR_SF7 to other values (DR_SF5, DR_SF9, DR_SF10, DR_SF12) and observe the effect on airtime.

```
LMIC_setDrTxpow(DR_SF7,14);
```

References:

1. <https://www.adafruit.com/product/3178>
2. <https://learn.adafruit.com/adafruit-feather-m0-radio-with-lora-radio-module/setup>
3. http://wiki.seeedstudio.com/Grove-TemperatureAndHumidity_Sensor/
4. <https://www.arduino.cc/en/Guide/Libraries>





4. Libraries and Sensors

The Kits come with various sensors to experiment with. Here are the links to the information about how to use them, and how to find any libraries that are needed, using the Arduino Library Manager

Sensors Examples

Most of the Grove Libraries comes with some examples for reading data. These can be found in the **File > Examples > ...your sensor...** menu.

Sensors

<p>Grove Sensor - Magnetic Switch http://wiki.seeedstudio.com/Grove-Magnetic_Switch/</p> <pre>int switched digitalRead(...);</pre>	
<p>Grove Button http://wiki.seeedstudio.com/Grove-Button/</p> <pre>int pressed digitalRead(...);</pre>	
<p>Grove Sensor - Temp and Humidity http://wiki.seeedstudio.com/Grove-TemperatureAndHumidity_Sensor/</p> <div><p>Grove Temperature And Humidity Sensor by Seeed Studio Version 1.0.0 INSTALLED Arduino library to control Grove Temperature And Humidity Sensor, it contains chip DHT11 AM2302. This temperature & humidity sensor provides a pre-calibrated digital output. A unique capacitive sensor element measures relative humidity and the temperature is measured by a negative temperature coefficient (NTC) thermistor. It has excellent reliability and long term stability. More info</p></div>	
<p>Grove Buzzer http://wiki.seeedstudio.com/Grove-Buzzer/</p> <pre>pinMode(PIN, OUTPUT); digitalWrite(PIN, HIGH/LOW); // buzz on/off</pre>	

<p>Grove Rotary Angle Sensor</p> <p>http://wiki.seeedstudio.com/Grove-Rotary_Angle_Sensor/</p> <pre>int angle_val analogRead(...);</pre>	
<p>Grove Light Sensor</p> <p>http://wiki.seeedstudio.com/Grove-Light_Sensor/</p> <pre>int light_val analogRead(...);</pre>	
<p>Grove Water Sensor</p> <p>http://wiki.seeedstudio.com/Grove-Water_Sensor/</p> <pre>int water_present digitalRead(...);</pre>	
<p>Grove Ultrasonic Distance Sensor</p> <p>http://wiki.seeedstudio.com/Grove-Ultrasonic_Ranger/</p> <div> <p>Grove Ultrasonic Ranger by Seeed Studio Arduino library for controlling Grove Ultrasonic Ranger, using gennal I/O communication. Arduino library for controlling Grove Ultrasonic Ranger, using gennal I/O communication. More info</p> <p>Version 1.0.1 <input type="button" value="Install"/></p> </div>	
<p>Grove 4-digit Display</p> <p>http://wiki.seeedstudio.com/Grove-4-Digit_Display/</p> <div> <p>Grove 4-Digit Display by Seeed Studio Version 1.0.0 INSTALLED Arduino library to control Grove_4Digital_Display TM1637. 4 digit display module is usually a 12 pin module. In this Grove gadget, we utilize a TM1637 to scale down the controlling pins into 2 Grove pins. It only takes 2 digital pins of Arduino or Seedeuino to control the content, even the luminance of this display. For projects that require of alpha-numeric display, this can be a nice choice. More info</p> </div>	

Feather M0 RFM To Particle Mesh Grove Adapter Mapping

Feather M0 Pinout: <https://learn.adafruit.com/assets/46254>

Diff No.1 - D4 on adapter goes to D9 on feather (with opt D5 to D10)

Diff No.2 - D2 on adapter goes to D5 on feather (with opt D6 to D10)

