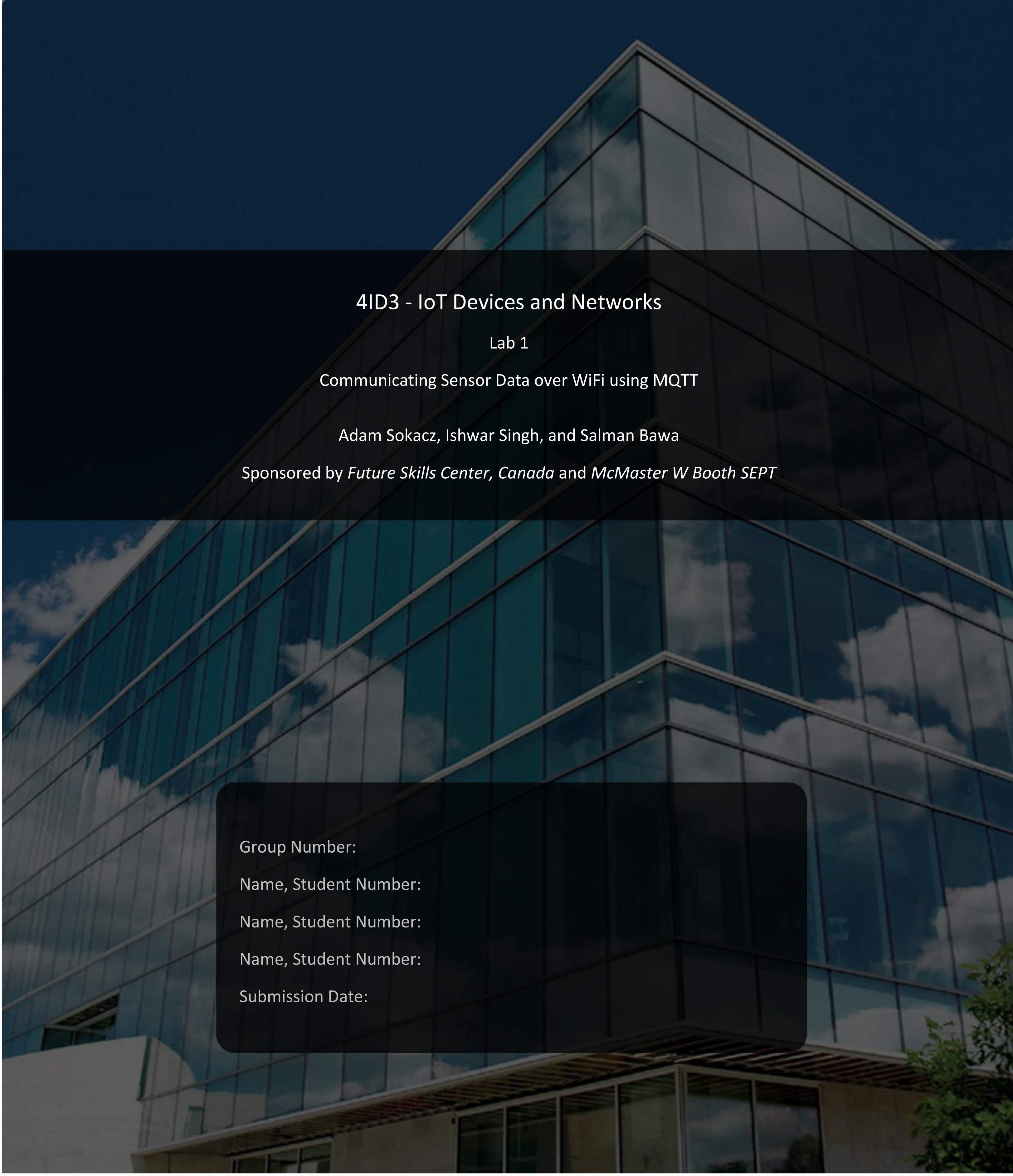


4ID3 - IoT Devices and Networks

Lab 1

Communicating Sensor Data over WiFi using MQTT

Adam Sokacz, Ishwar Singh, and Salman Bawa

Sponsored by *Future Skills Center, Canada* and *McMaster W Booth SEPT*

Group Number:

Name, Student Number:

Name, Student Number:

Name, Student Number:

Submission Date:

W BOOTH SCHOOL OF ENGINEERING PRACTICE AND TECHNOLOGY

McMaster-Mohawk Bachelor of Technology Partnership

McMaster University ENGINEERING

MOHAWK

Objective

In this lab, we will be constructing a basic IoT network with a single node that collects data from connected sensors and communicates that data to a remote MQTT broker using a local WiFi connection.

Contents

Feedback

Q1 - What would you rate the difficulty of this lab?

(1 = easy, 5 = difficult)

1 **2** **3** **4** **5**

Comments about the difficulty of the lab:

Q2 - Did you have enough time to complete the lab within the designated lab time?

YES **NO**

Q3 - How easy were the lab instructions to understand?

(1 = easy, 5 = unclear)

1 **2** **3** **4** **5**

List any unclear steps:

Q4 - Could you see yourself using the skills learned in this lab to tackle future engineering challenges?

(1 = no, 5 = yes)

1 **2** **3** **4** **5**

Additional Resources

Arduino Programming Refresher (https://youtu.be/CbJHL_P5RJ8)

Mosquitto MQTT Broker Tutorial (https://youtu.be/DH-VSAACtBk)

NodeRED Fundamentals Tutorial (https://youtu.be/3AR432bguOY)

ESP8266 Overview (https://youtu.be/dGrJi-ebZgI)

Pre-Lab Questions

Q1 - In your own words, describe the **publish-subscribe messaging pattern**. What role does the broker play? What role do the clients play?

(Suggested: 2 sentences)

Q2 - In your own words, what does **QoS** mean for MQTT transmissions? What **levels** of QoS exist?

(Suggested: 3 sentences)

Q3 – What is the role of each of the **fields below**, when connecting to an MQTT broker?

Host IP Address	
Topic Name	
Username	
Password	
Protocol (tcp/ws/wss/tls)	

Post-Lab Questions

Q1 – Using software like **PowerPoint** or **Draw.IO**, create a diagram to represent the nodes in this IoT network and **label the data** being exchanged between each node.

(Suggested: Diagram, 5 points)

Q2 - Explain your **observations** in Node-Red user interface as you interact with the sensors. Is the change instantaneous?

(Suggested: 2 sentences, 2 points)

Q3 – If the microcontroller loses battery, will the **NodeRED dashboard** still be connected to the MQTT server? Explain your answer.

(Suggested: 3 sentences, 2 points)

Q3 – Draw a diagram or in a detailed paragraph, explain the differences in connecting an **analogue, IIC,** and **SPI** sensor to an ESP-based development board. The submission must include **pin connections** on the development board and where they correspond to on each sensor.

(Suggested: Sketch, 5 points)

Q4 – Using the official Paho-MQTT examples, write a short **Python script** that subscribes to the same topic and MQTT broker used in this lab, and **prints** the published payload to the terminal window.

https://github.com/eclipse/paho.mqtt.python/tree/master/examples

(Suggested: Python script, 5 points)

Q5 – Using a remote MQTT broker on the cloud allows client Node-RED dashboards to visualize sensor data without needing to be in the same **geographical location** as the network. Explain **2 security** vulnerabilities with the network established in this lab and **2 ways** these issues could be **mitigated**.

(Suggested: 4 sentences, 2 points)

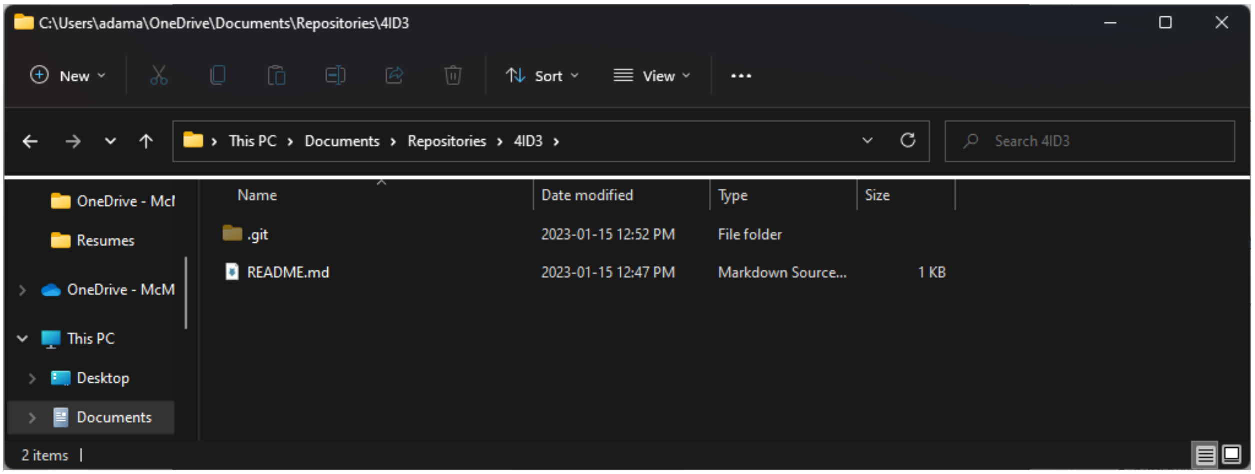
Q6 - Below, write a **LinkedIn post** about **4 key learning takeaways** from this lab.

(Suggested: Paragraph or screenshot, 2 points)

Setting up the Workspace

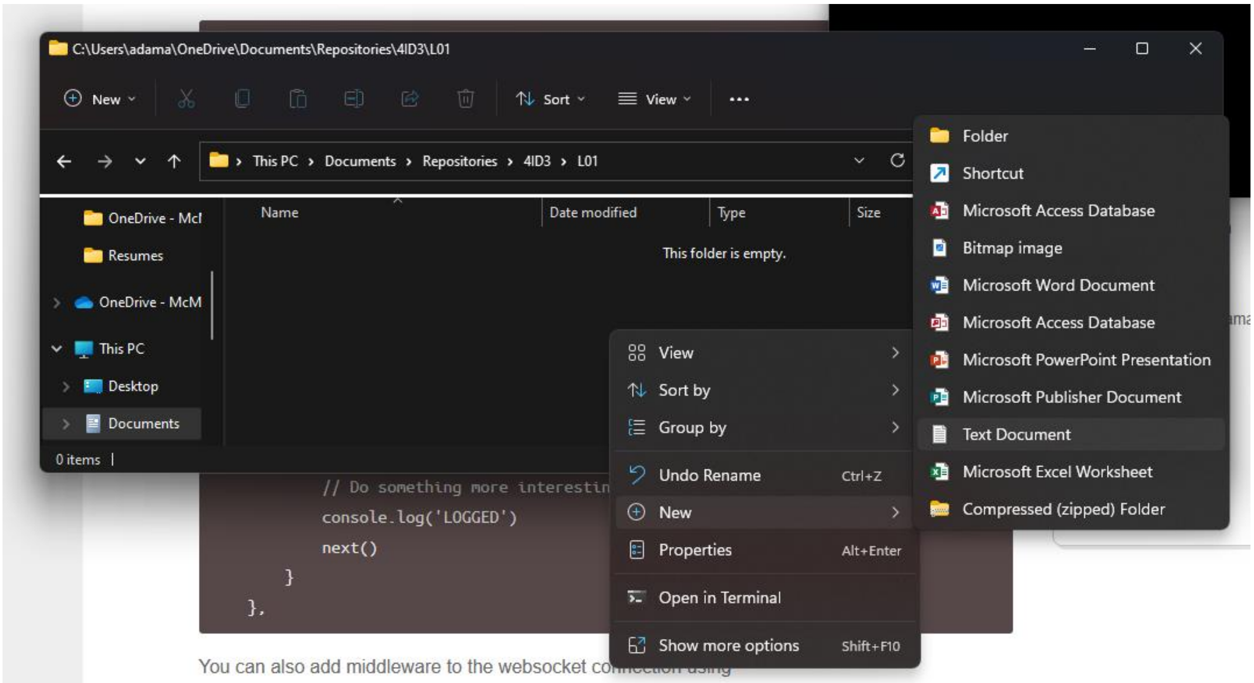
Each lab, we will be creating a new folder in the local git repository that was created in the provided pre-lab to store and document technologies that you have worked on.

Navigate to your local git repository for this course.

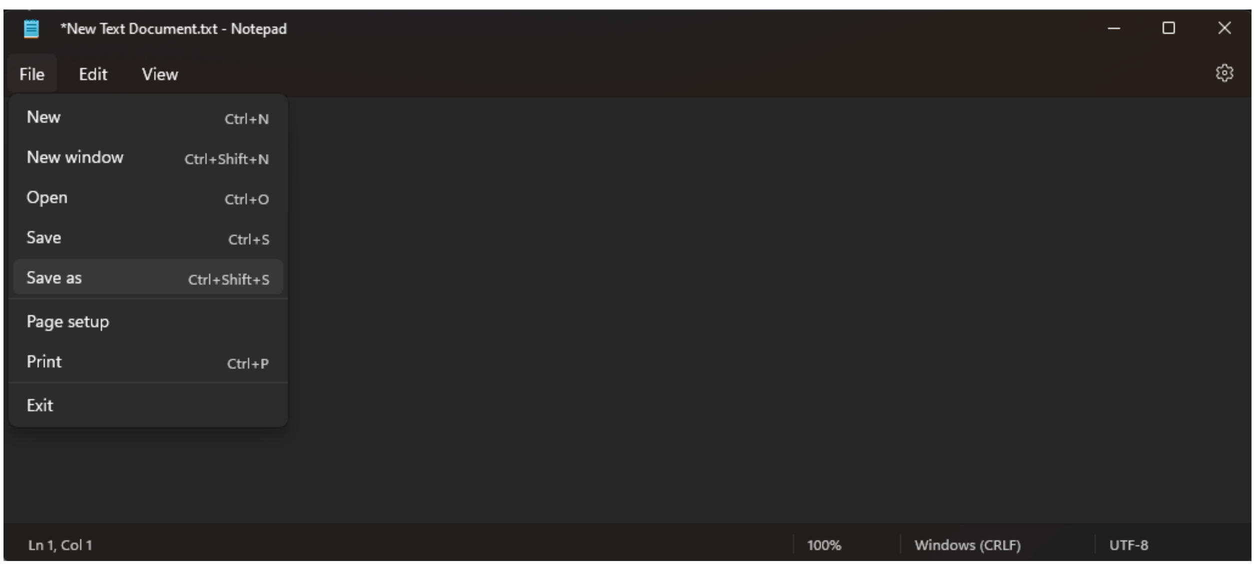


Create a new folder named **L01**. Navigate inside this folder.

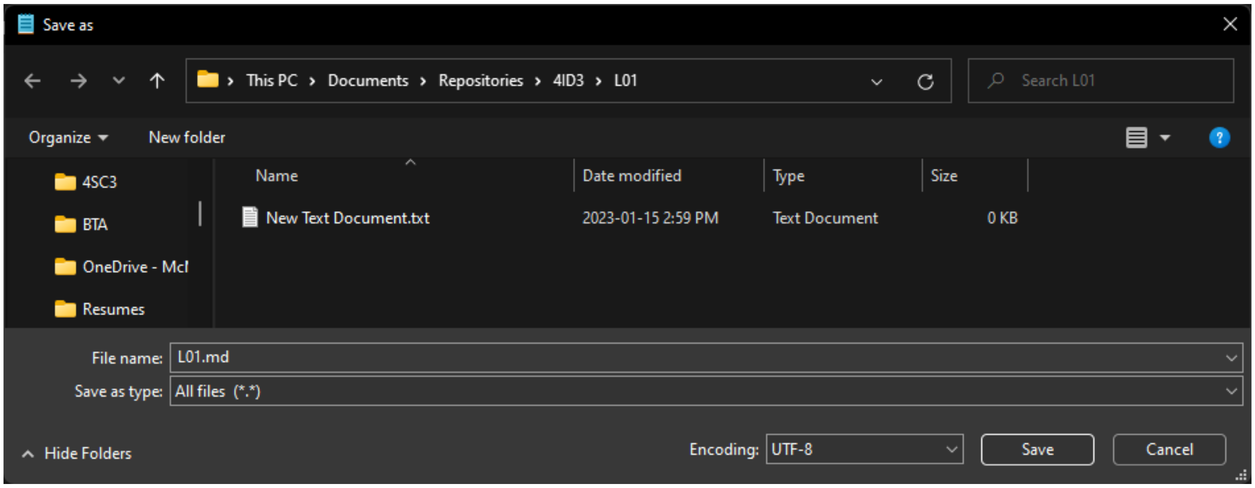
Create a new text file in the folder.



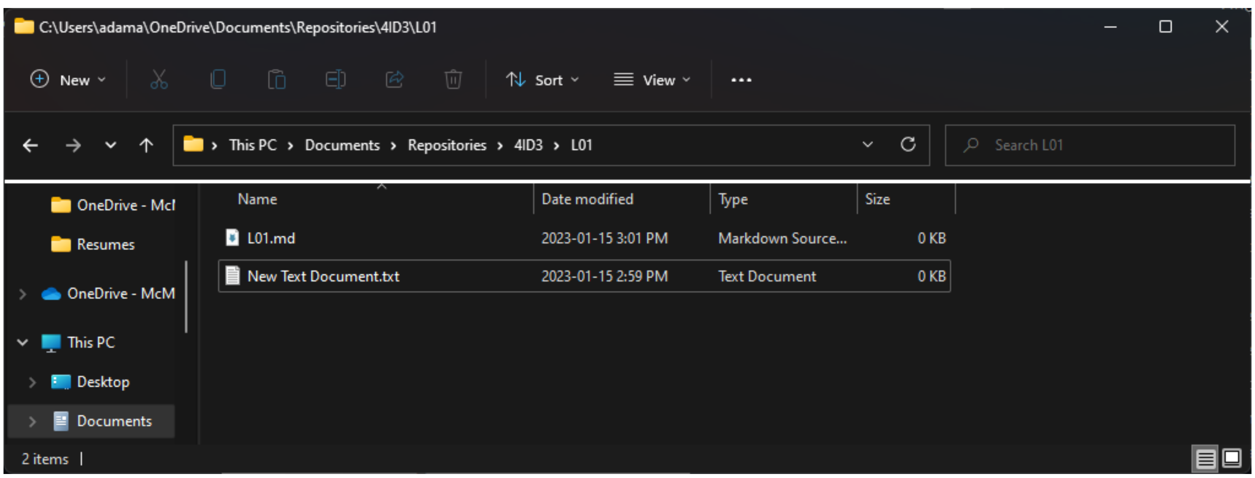
Press **File > Save as.**



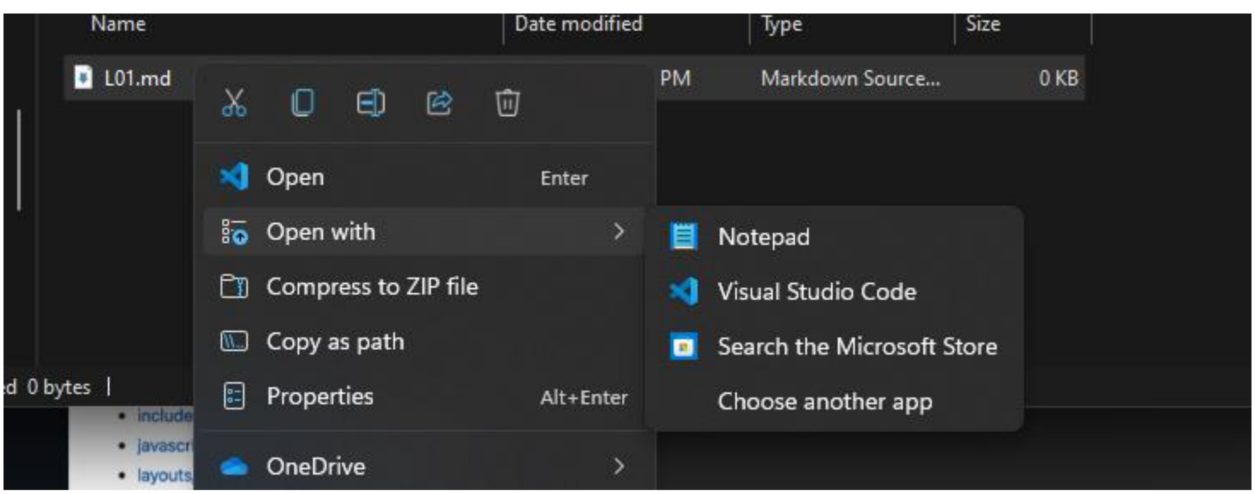
Save it as **L01.md**. Ensure that the **Save as type** is set to **All files (*.*)**.



Now, you should have two files, a **text file** and a **markdown file**. Delete the text file.



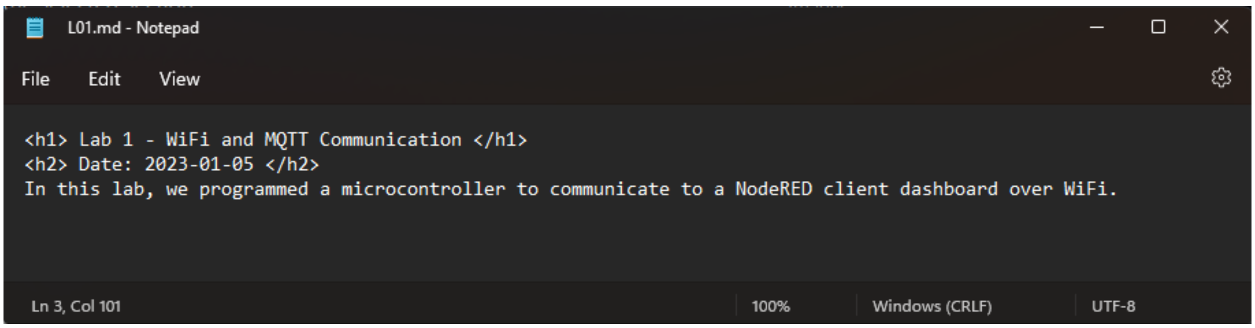
To open the markdown file, **right-click** and select **Open with**. Choose **Notepad**.



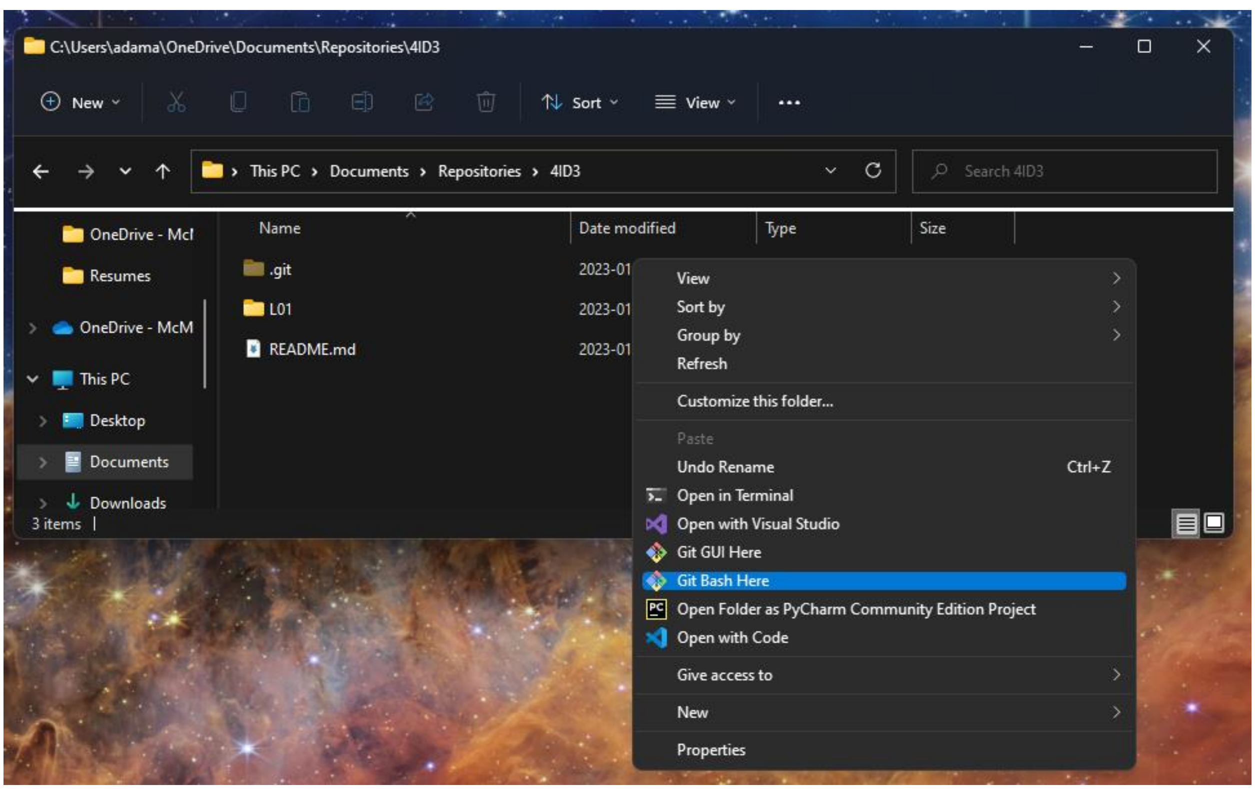
Writing markdown documents to explain your code is very similar to HTML. A reference guide can be found here:

https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax

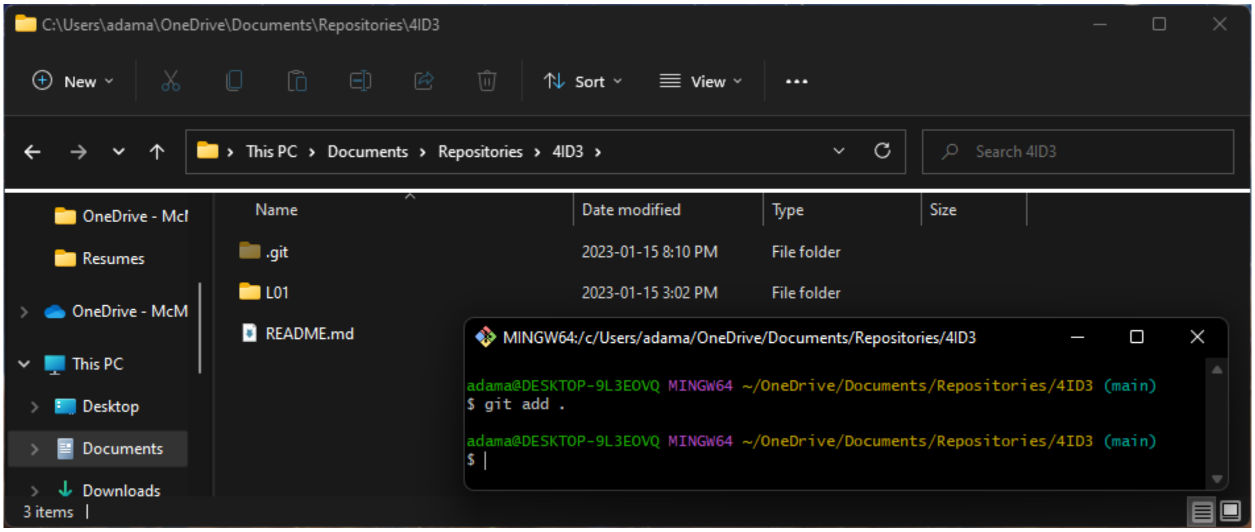
Write the following text in the markdown file and save it.



Right-click in the root of your local repository and launch **git bash**.



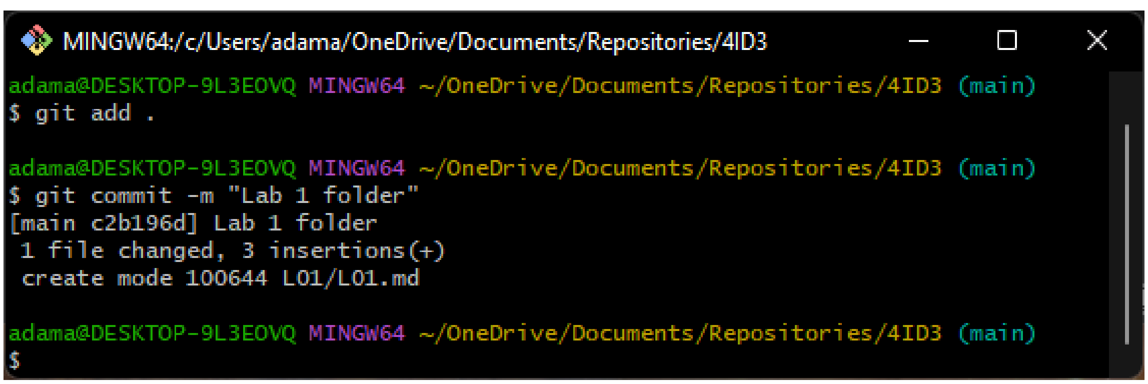
First, we need to add all the changes to the index that will be synced with GitHub. This will be done with the git add command.



git add .

The period '.' Is used as a shorthand for selecting all changes.

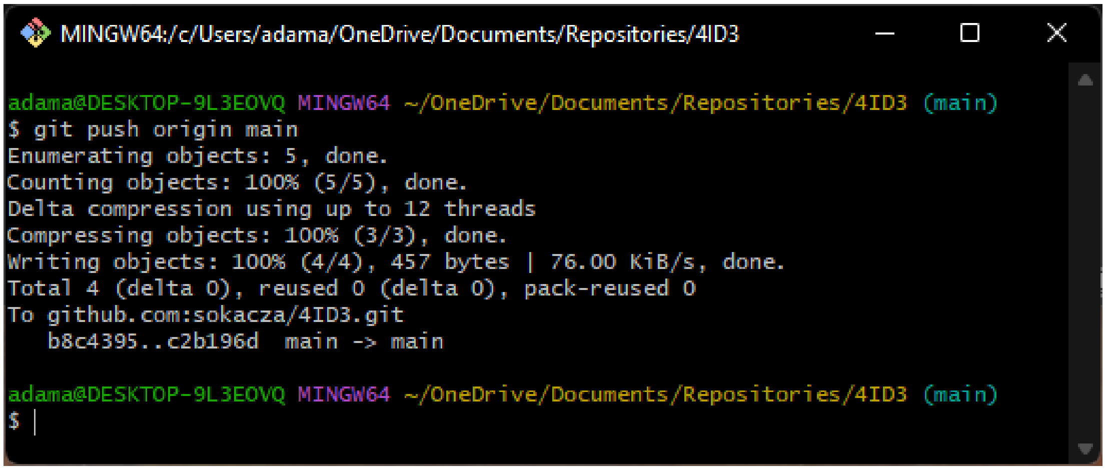
Next, when we are happy with the changes we chose to upload, we can use the commit command to package them to be synced.



git commit -m "Lab 1 folder"

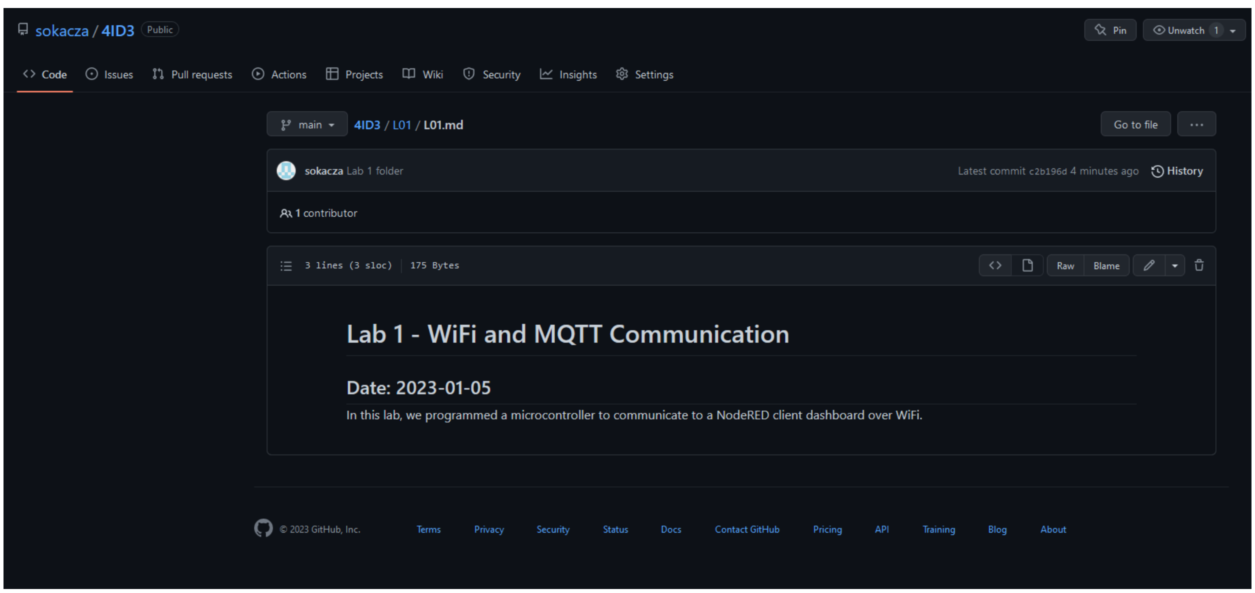
The '-m' flag stands for message, and it adds a message that explains what changes were made.

Lastly, to sync your local git repository with GitHub, use the git push command.



git push origin main

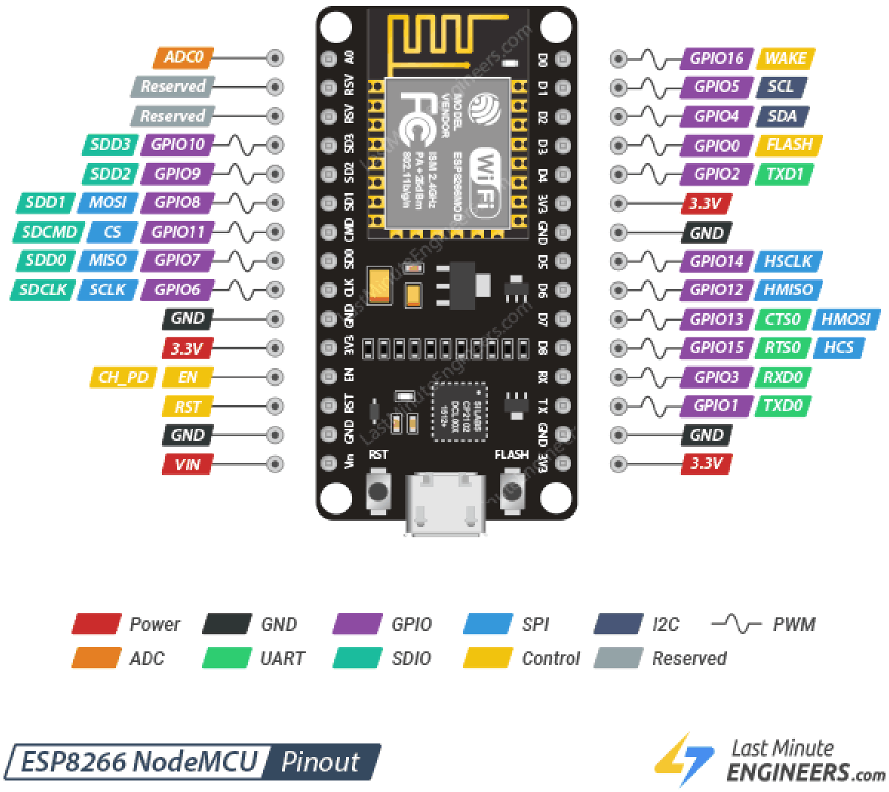
Now, log into GitHub and verify that the changes have been made.



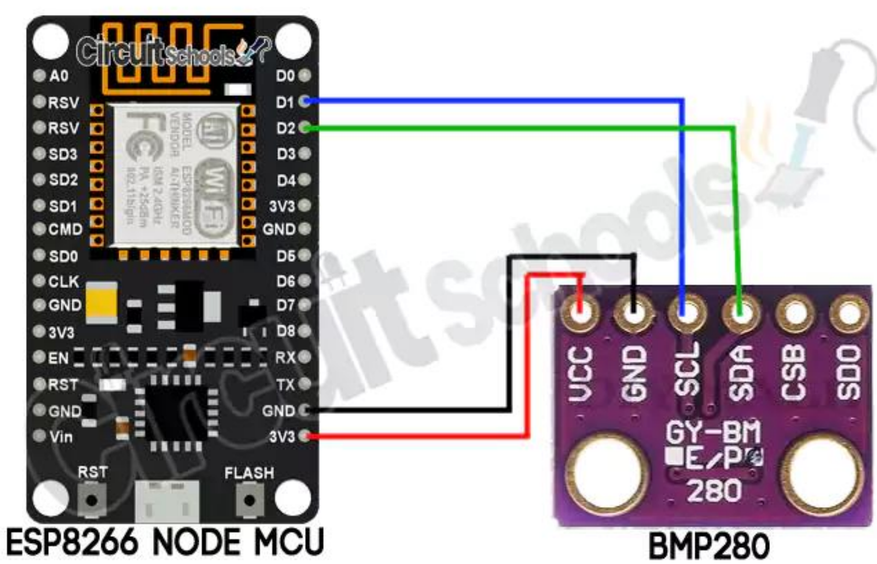
Now, if you are collaborating and wish to sync your local git repo with the remote GitHub repo, use the git pull command. In this case, we see that our local git repo is already up-to-date.

Wiring Diagram

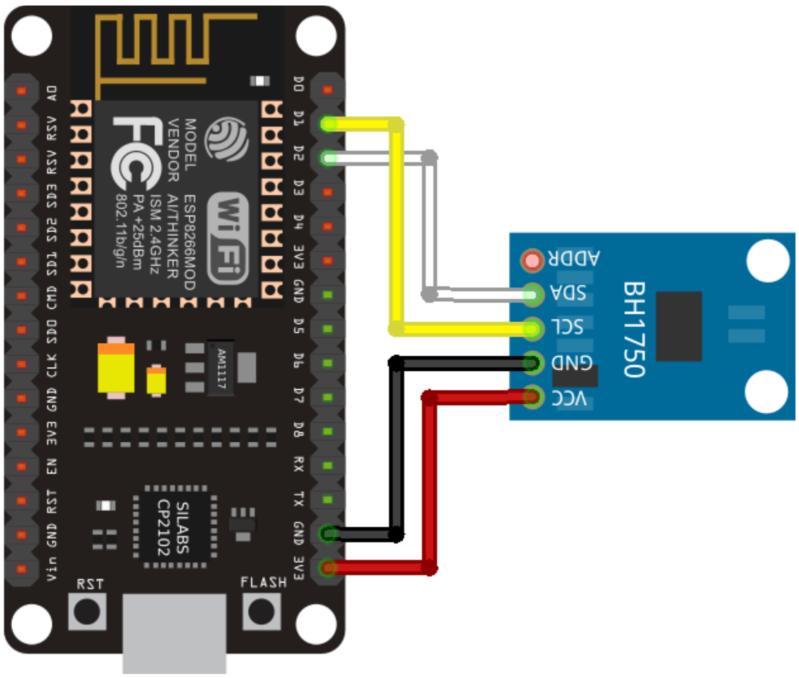
ESP8266 Development Board Pinout:



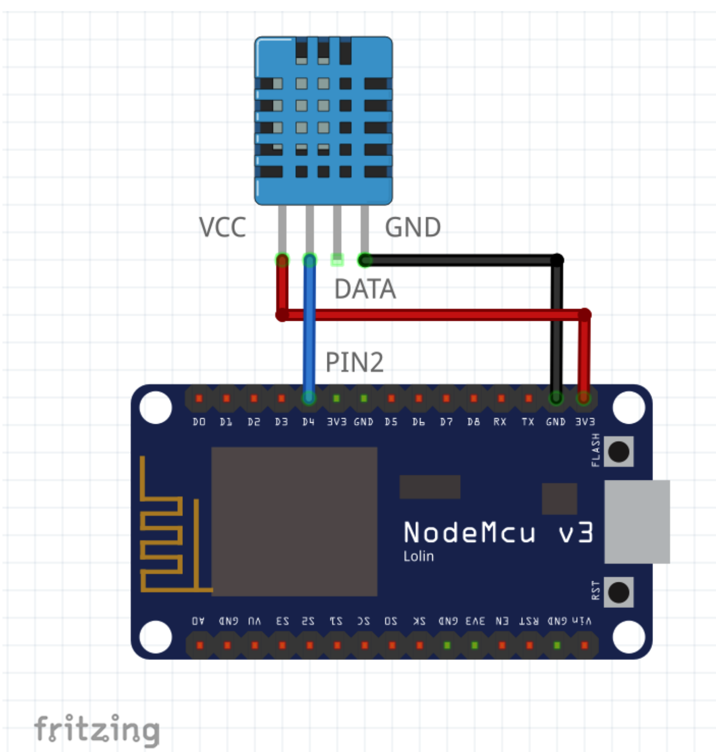
BMP180 Connection:



BH1750 Connection:



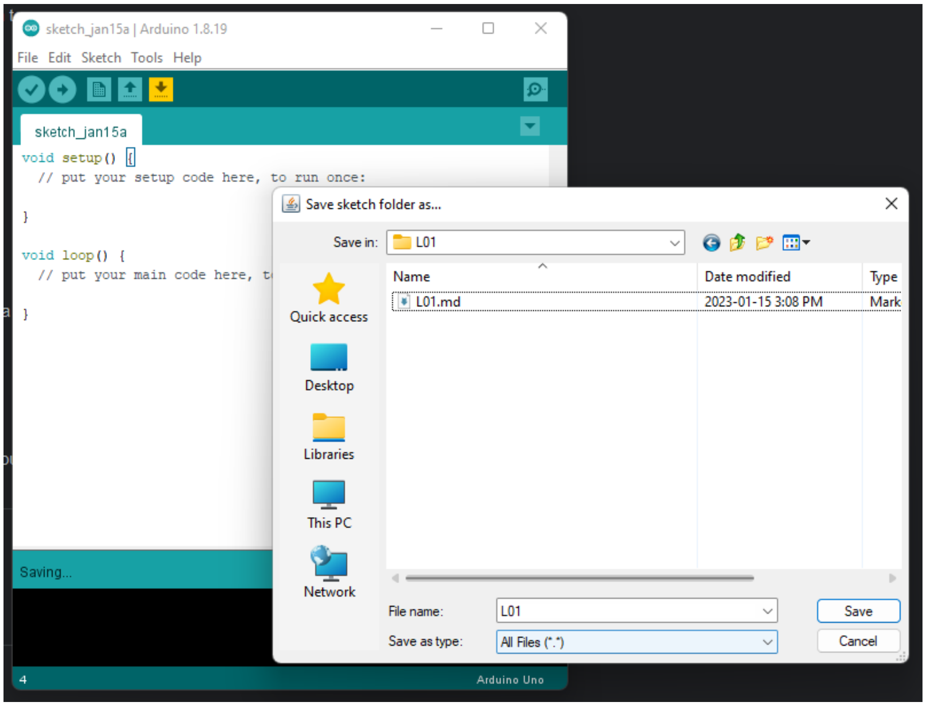
DHT11 Connection *(Except connect DHT11 signal pin to D5)* :

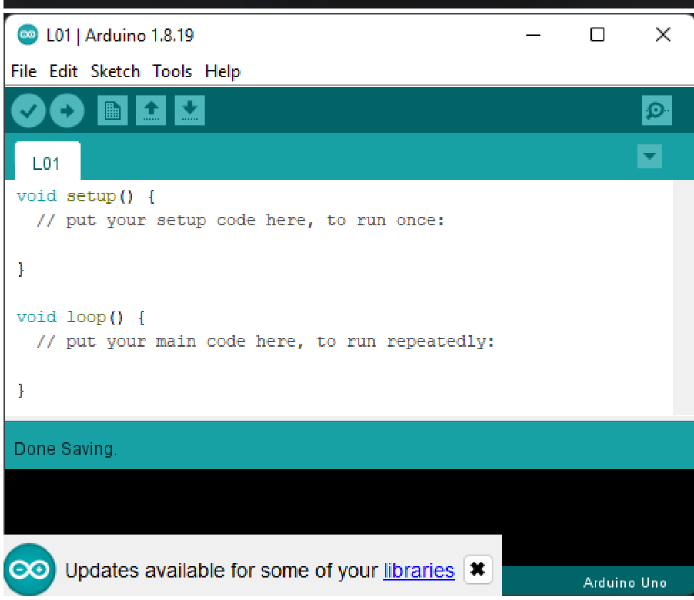


Reading Sensor Data

The goal of this section of the lab is to read data from 3 sensors and print them to the serial monitor, before we communicate to a server.

Launch the Arduino IDE and **Save as** into your lab 1 folder.



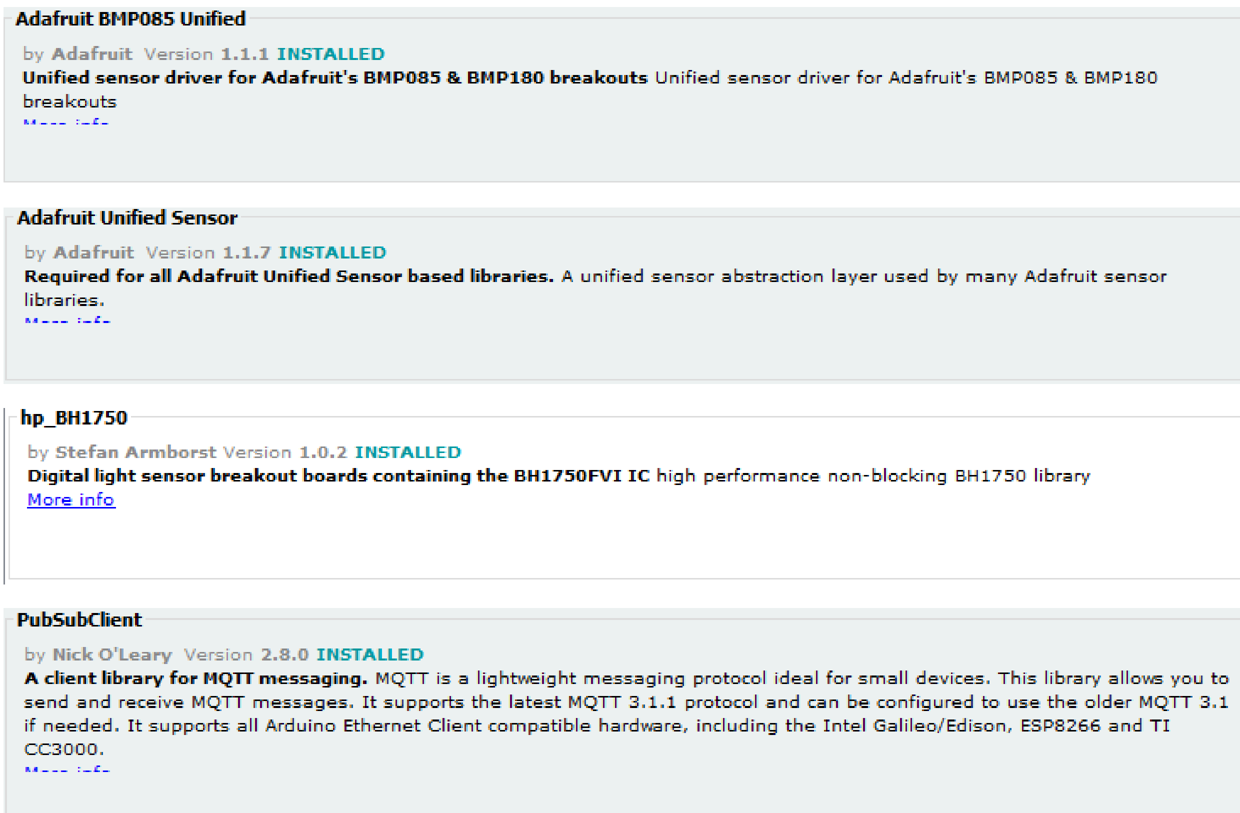


The 3 sensors we will be using are as follows:

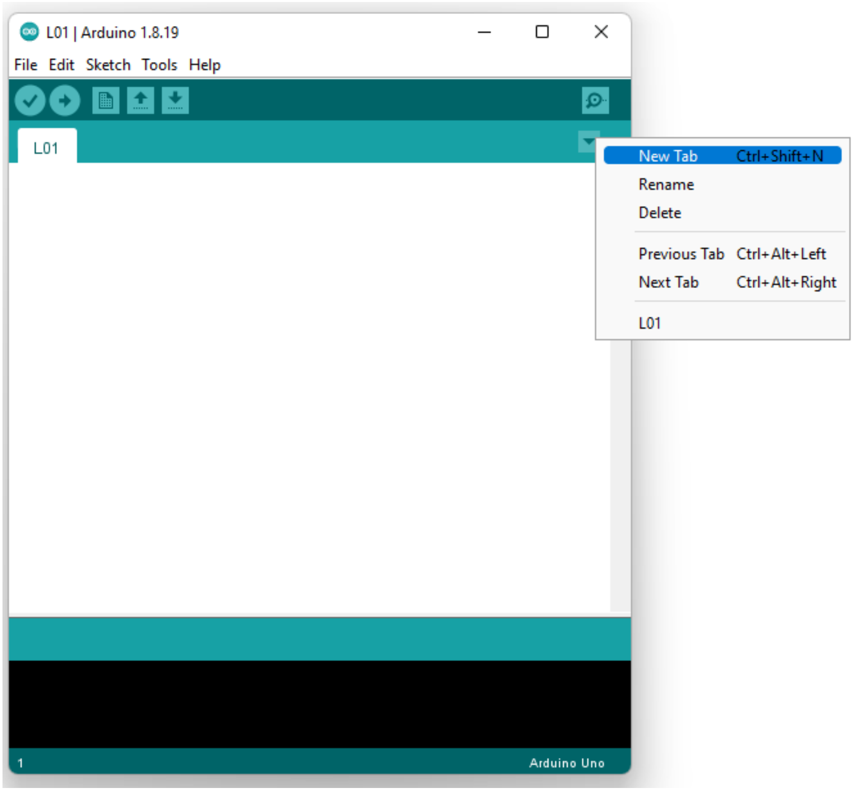
- DHT11 for Temperature and Humidity
- BMP180 for Pressure
- BH1750 for Light Intensity

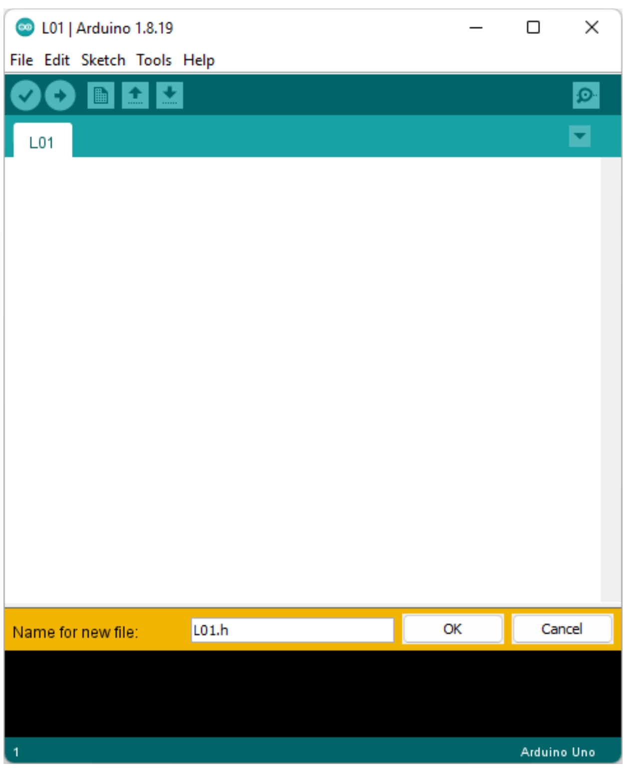
Ensure that the following libraries are installed from the **Arduino Library Manager**:

- Adafruit Unified Sensor by Adafruit
- Adafruit BMP085 Unified by Adafruit
- Hp_BH1750 by Stefan Armborst
- PubSubClient by Nick O'Leary
- The ESP8266 driver as described in the pre-lab



Create a new header file. Name this file **L01.h**.





In this file, we will keep our dependencies, preprocessor macros, global variables, and global objects.

Include the libraries for each sensor. Don't worry about including the same one twice, the #ifndef directives will prevent the compiler from looking at the same library twice.

L01 L01.h

```
//DHT11 Libraries
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

//BMP180 Libraries
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP085_U.h>

//HP_BH1750 Libraries
#include <hp_BH1750.h>

```

Include any macros that you need. Macros act as a find-and-replace. The C++ preprocessor will copy '14' everywhere in the code where 'DHTPIN' is found at compile time.

```

//Macros
#define DHTPIN 14
#define DHTTYPE DHT11
#define DELAY_BETWEEN_SAMPLES_MS 5000

```

Lastly, lets instantiate some global objects for classes that we will be using throughout the code.

```

//Instantiate Sensor Objects
DHT_Unified dht(DHTPIN, DHTTYPE);
Adafruit_BMP085_Unified bmp = Adafruit_BMP085_Unified(10085);
hp_BH1750 BH1750;

```

This concludes the header file. Let's move back to the implementation file.

Firstly, include our header file.

L01 | L01.h

```

#include "L01.h"

void setup(){

}


void loop(){

}


```

Next, set up the void setup() function.

```

#include "L01.h"

void setup(){

  //Start the serial monitor at 115200 baud
  Serial.begin(115200);

  //Create a sensor object that is passed into the getSensor method of the dht class
  //Only the dht sensor requires this
  sensor_t sensor;
  dht.temperature().getSensor(&sensor);
  dht.humidity().getSensor(&sensor);

  //Run the begin()method on each sensor to start communication
  dht.begin();
  bmp.begin();
  BH1750.begin(BH1750_TO_GROUND);

}

```

Moving onto the void loop() function, it should be noticed that both the DHT sensor and BMP sensor are part of the same Adafruit unified library, so they will be polled differently than the BH sensor.

```
void loop(){

  //Polling the DHT and BMP sensor using events
  sensors_event_t dhtTempEvent, dhtHumEvent, bmpEvent;
  dht.temperature().getEvent(&dhtTempEvent);
  dht.humidity().getEvent(&dhtHumEvent);
  bmp.getEvent(&bmpEvent);

  //Polling the BH sensor
  BH1750.start();
  float lux=BH1750.getLux();

```

Next, we want to print the sensor readings to the serial monitor.

```
//Printing sensor readings to serial monitor
 Serial.println("\n-");
(!isnan(dhtTempEvent.temperature)) ? Serial.println("Temperature: " + String(dhtTempEvent.temperature) + " degC") : Serial.println("Temperature Sensor Disconnected");

(!isnan(dhtHumEvent.relative_humidity)) ? Serial.println("Humidity: " + String(dhtHumEvent.relative_humidity) + " %") : Serial.println("Humidity Sensor Disconnected");

(!isnan(bmpEvent.pressure)) ? Serial.println("Pressure: " + String(bmpEvent.pressure) + " hPa") : Serial.println("Pressure Sensor Disconnected");

(!isnan(lux)) ? Serial.println("Light Intensity: " + String(lux) + " lux") : Serial.println("Lux Sensor Disconnected");

```

This is a shorthand for a one line if-else statement. It is called a ternary statement.

Firstly, we are checking if there is a value stored in each event attribute or if the sensor failed to retrieve a value.

If **isnan()** is **True** then there is **no value** stored. Because we want the opposite condition, we will place a **not operator** '!' in front of the condition.

The **?** operator separates the comparison from the true condition.

The **:** operator separates the true statement from the false statement.

(condition) ? If true : If false

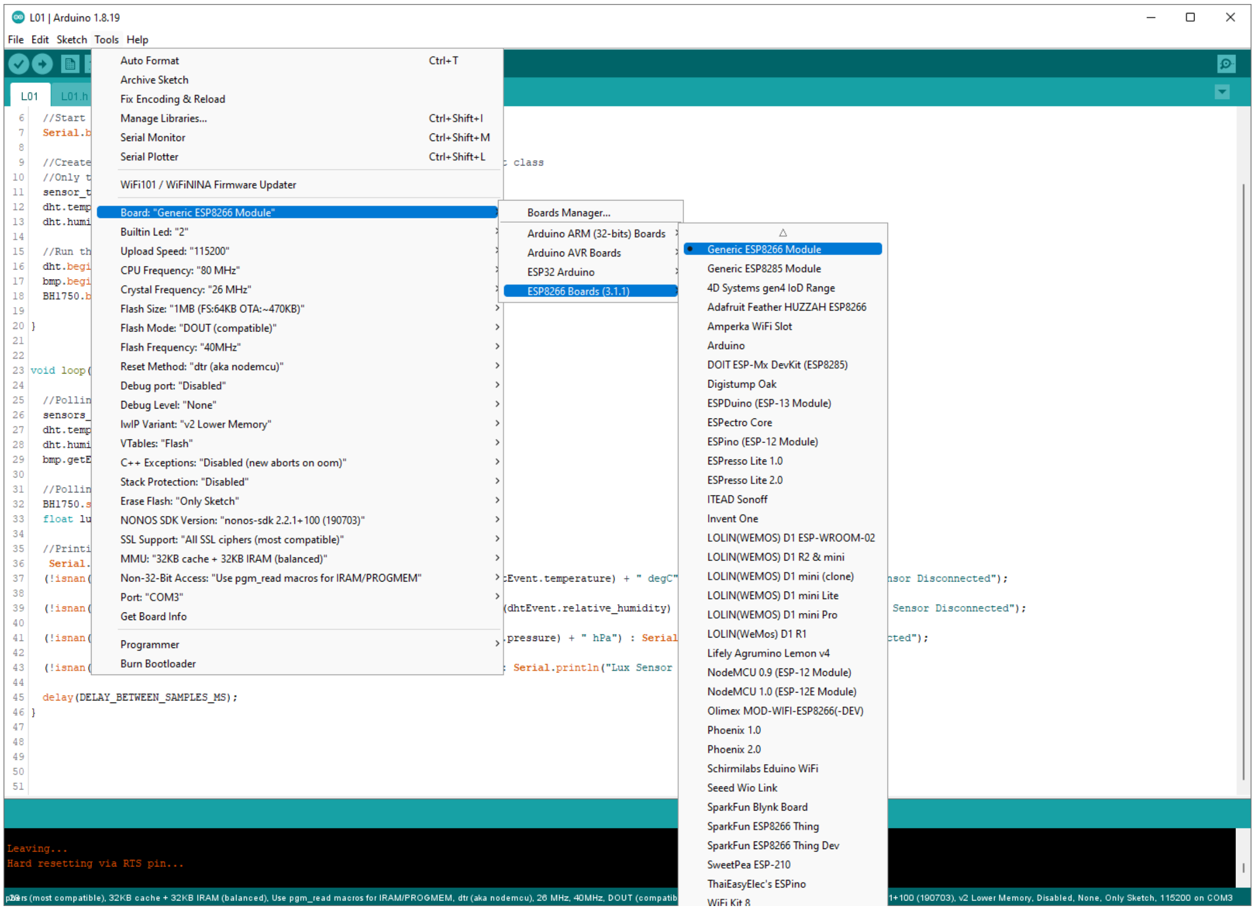
(If we have a temperature value) ? Print the temperature value : Print that the sensor is disconnected

Secondly, to concatenate different datatypes into a print statement, we must **cast them to strings** and **concatenate them** together. An easy way to do this is to use **String**(float value) to cast and **+** operator to concatenate two strings together.

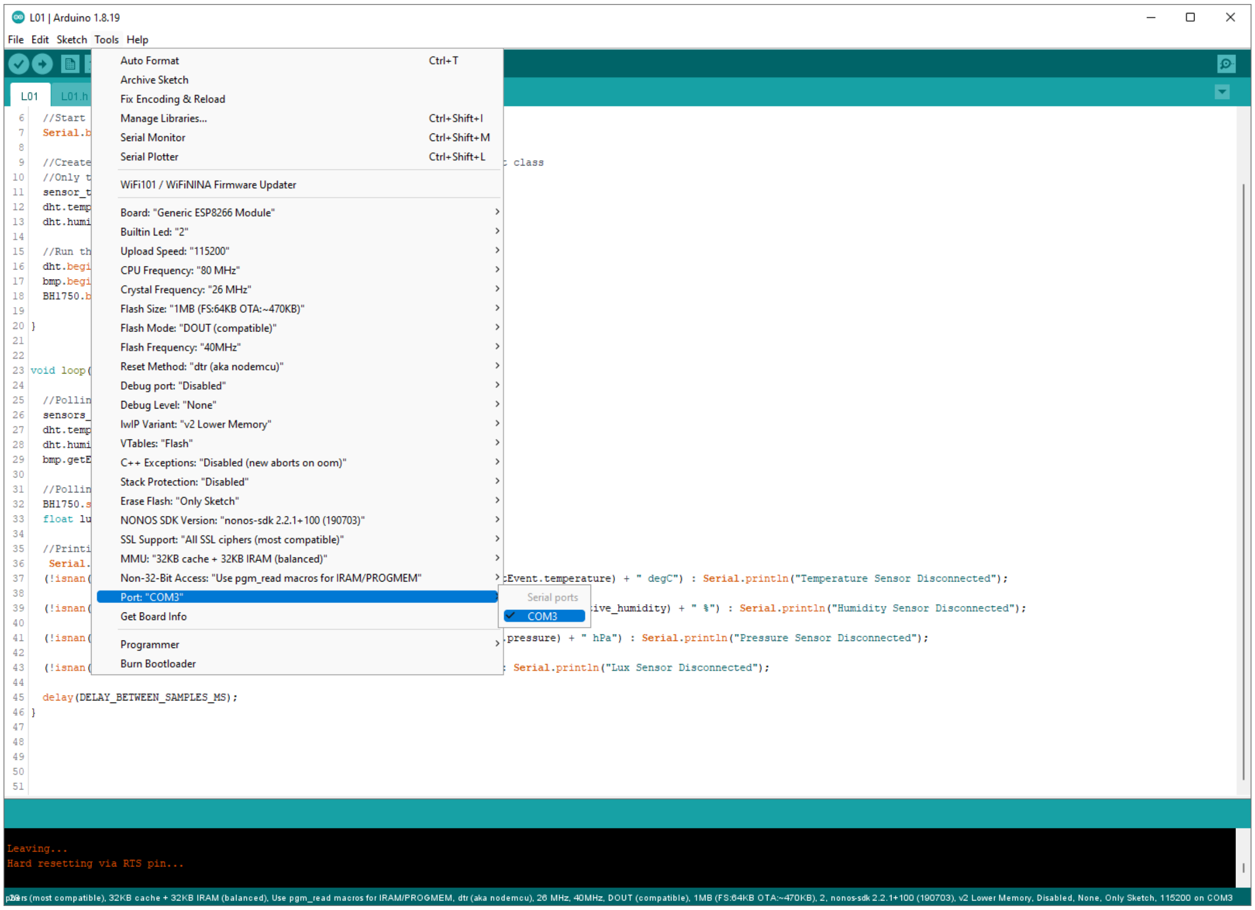
Thirdly, when printing to the serial monitor, the **print()** command does not include a newline character. In order to print on a new line, we could add it in **print(string + '\n')** or use the **println()** function.

Lastly, we want to let time pass between polls. This can be done simply by adding a delay in the loop.

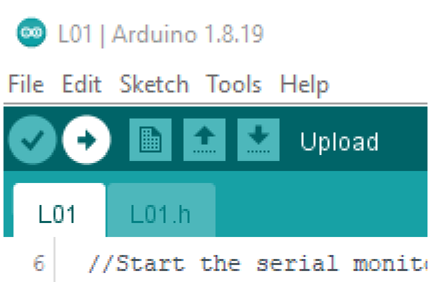
To upload to the board, change the board to Generic ESP8266 Module.



Leave the default communication settings the same, with the exception of the COM port. Select the COM port that your microcontroller is connected to.



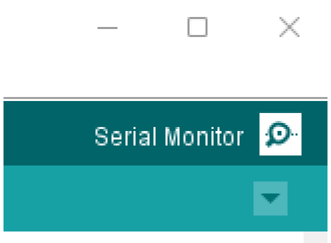
Press the **upload** button to compile and upload the code. Keep an eye on the terminal window for any errors that arise.

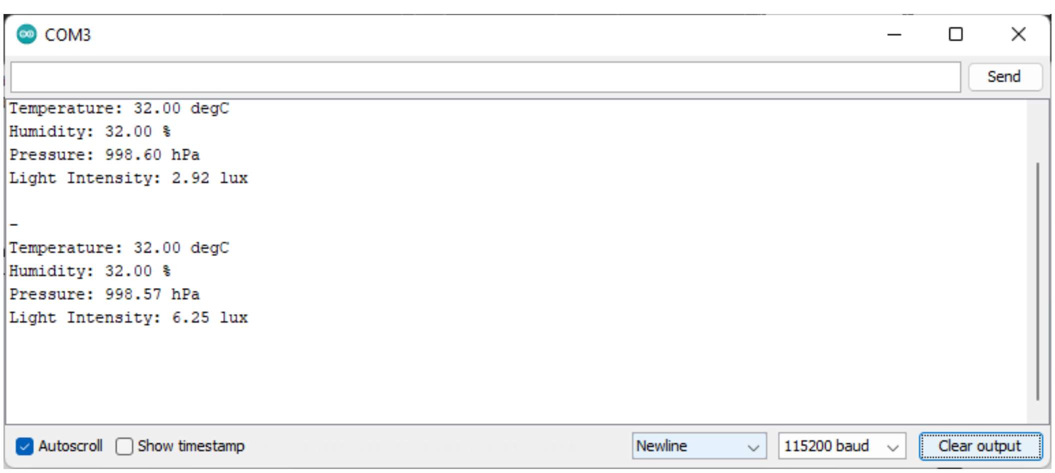


A successful upload should look like this:

```
Crystal is 26MHz
MAC: 48:3f:da:aa:57:09
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Flash params set to 0x0340
Compressed 280256 bytes to 205447...
Writing at 0x00000000... (7 %)
Writing at 0x00004000... (15 %)
Writing at 0x00008000... (23 %)
Writing at 0x0000c000... (30 %)
Writing at 0x00010000... (38 %)
Writing at 0x00014000... (46 %)
Writing at 0x00018000... (53 %)
Writing at 0x0001c000... (61 %)
Writing at 0x00020000... (69 %)
Writing at 0x00024000... (76 %)
Writing at 0x00028000... (84 %)
Writing at 0x0002c000... (92 %)
Writing at 0x00030000... (100 %)
Wrote 280256 bytes (205447 compressed) at 0x00000000 in 18.2 seconds (effective 122.9 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

After the upload is complete (NOT DURING), launch the **Serial monitor** to view the microcontroller output.

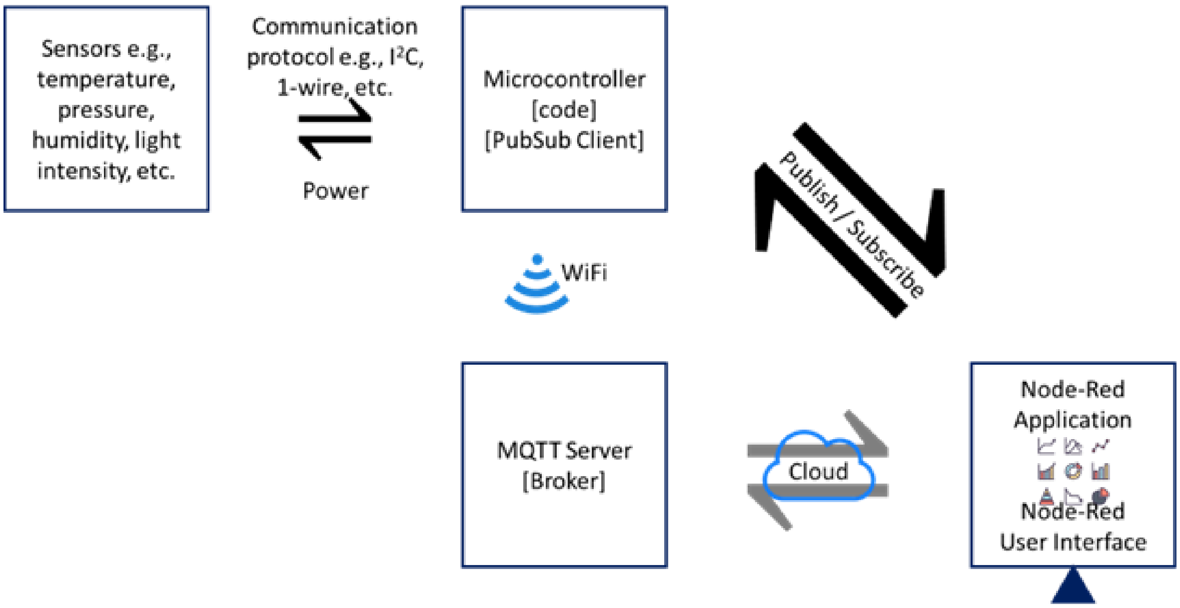




Publishing to an MQTT Broker

The ESP8266 has a built-in WiFi transceiver, which allows it to connect to the internet easily. We will be using the PubSubClient library to:

a) Connect to a local WiFi network
b) Connect to a public (or private) MQTT broker
c) Connect to a NodeRED interface



Modify your header file to include some **global variables** for the WiFi driver, the **MQTT libraries**, and instantiate **the MQTT Client** object. Also change the delay to 20 seconds (20,000 ms).

When adding the broker ip address, you have two options:

a) Use an online public broker
b) Use a local broker on the same network

Public Broker

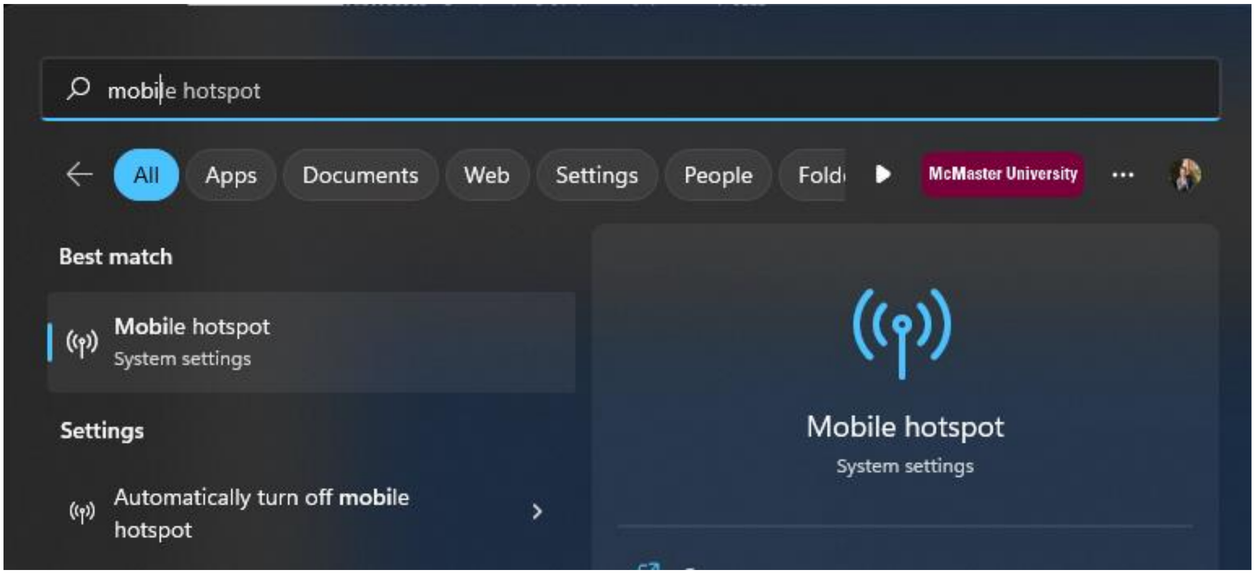
Use the IP address of a known public broker such as Mosquitto's test broker:

test.mosquitto.org

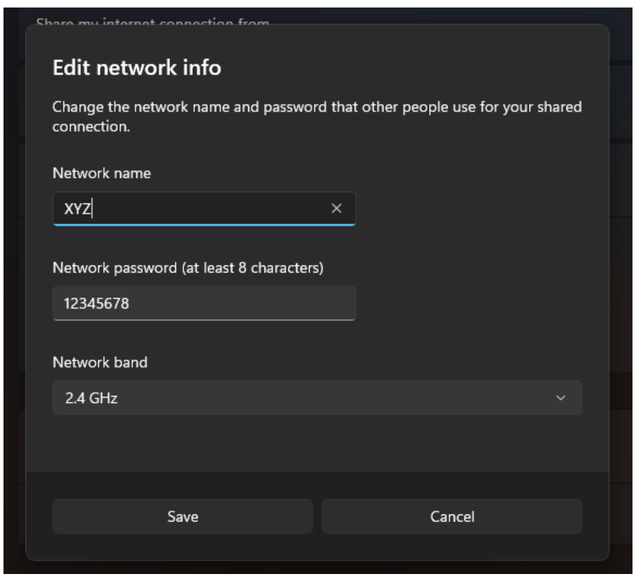
Port is 1883

Hotspotting your Microcontroller

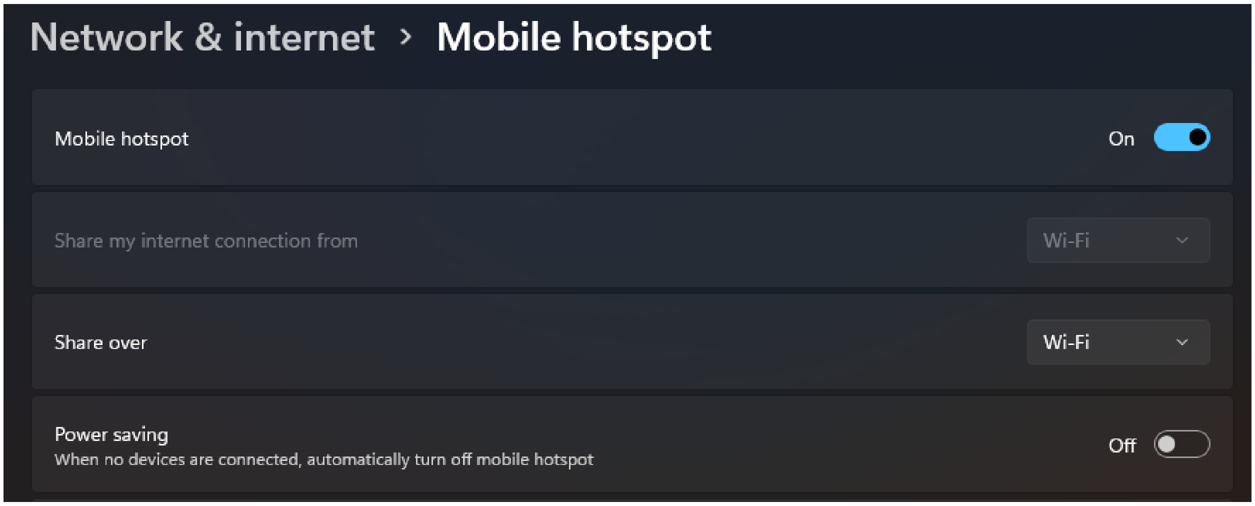
Open **Windows Mobile Hotspot**.



Set your login credentials.



Toggle it **on** and **disable power saving**.



Network & internet > Mobile hotspot
Mobile hotspot
On
Share my internet connection from
Wi-Fi
Share over
Wi-Fi
Power saving
When no devices are connected, automatically turn off mobile hotspot
Off

Code Changes

L01 | L01.h §

```
//DHT11 Libraries
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

//BMP180 Libraries
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP085_U.h>

//HP_BH1750 Libraries
#include <hp_BH1750.h>

//MQTT Libraries
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

//Macros
#define DHTPIN 14
#define DHTTYPE DHT11
#define DELAY_BETWEEN_SAMPLES_MS 5000

//Global Variables
char* ssid = "XYZ";
char* pass = "12345678";
const char* brokerAddress = "192.168.2.13";

//Instantiate Sensor Objects
DHT_Unified dht(DHTPIN, DHTTYPE);
Adafruit_BMP085_Unified bmp = Adafruit_BMP085_Unified(10085);
hp_BH1750 BH1750;

//Instantiate MQTT Client
WiFiClient espClient;
PubSubClient client(espClient);

```

In the **setup()** function, we need to initialize the WiFi driver and MQTT client.

```
void setup(){

  //Start the serial monitor at 115200 baud
  Serial.begin(115200);

  //Create a sensor object that is passed into the getSensor method of the dht class
  //Only the dht sensor requires this
  sensor_t sensor;
  dht.temperature().getSensor(&sensor);
  dht.humidity().getSensor(&sensor);

  //Run the begin()method on each sensor to start communication
  dht.begin();
  bmp.begin();
  BH1750.begin(BH1750_TO_GROUND);

  //Start the WiFi driver and tell it to connect to your local network
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, pass);

  //While it is connecting, print a '.' to the serial monitor every 500 ms
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  //Once connected, print the local IP address
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  //Set the MQTT client to connect to the desired broker
  client.setServer(brokerAddress, 1883);

}
```

If the microcontroller loses connection to the MQTT server, we want to have a callback function that would attempt to reconnect.

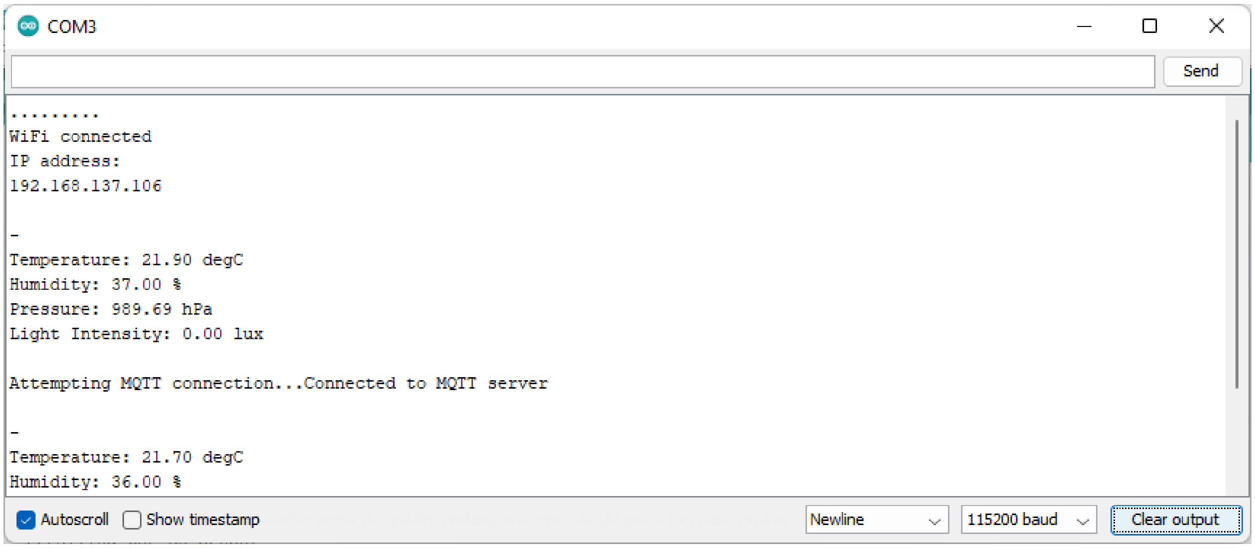
Below the setup() function, create a reconnect function. This function will be called from void loop().

```

void reconnect() {

  //While the client remains unconnected from the MQTT broker, attempt to reconnect every 2 seconds
  //Also, print diagnostic information
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");

    if (client.connect("ESP8266Client")) {
      Serial.println("Connected to MQTT server");
      client.subscribe("testTopic");
    } else {
      Serial.print("Failed to connect to MQTT server, rc = ");
      Serial.print(client.state());
      delay(2000);
    }
  }
}

```

In void loop(), if connection is lost then call the reconnect function. Once you are connected, publish each sensor data to its associated topic.

```

void loop(){

  //Polling the DHT and BMP sensor using events
  sensors_event_t dhtTempEvent, dhtHumEvent, bmpEvent;
  dht.temperature().getEvent(&dhtTempEvent);
  dht.humidity().getEvent(&dhtHumEvent);
  bmp.getEvent(&bmpEvent);

  //Polling the BH sensor
  BH1750.start();
  float lux=BH1750.getLux();

  //Printing sensor readings to serial monitor
   Serial.println("\n-");
  (!isnan(dhtTempEvent.temperature)) ? Serial.println("Temperature: " + String(dhtTempEvent.temperature) + " degC") : Serial.println("Temperature Sensor Disconnected");

  (!isnan(dhtHumEvent.relative_humidity)) ? Serial.println("Humidity: " + String(dhtHumEvent.relative_humidity) + " %") : Serial.println("Humidity Sensor Disconnected");

  (!isnan(bmpEvent.pressure)) ? Serial.println("Pressure: " + String(bmpEvent.pressure) + " hPa") : Serial.println("Pressure Sensor Disconnected");

  (!isnan(lux)) ? Serial.println("Light Intensity: " + String(lux) + " lux") : Serial.println("Lux Sensor Disconnected");

  //If the client disconnects from the MQTT broker, attempt to reconnect
  if (!client.connected()) {
    reconnect();
  }
  if(!client.loop())
    client.connect("ESP8266Client");

  //Publish the sensor data to the associated topics
  client.publish("4ID3_G7/temperature", String(dhtTempEvent.temperature).c_str());
  delay(100);
  client.publish("4ID3_G7/humidity", String(dhtHumEvent.relative_humidity).c_str());
  delay(100);
  client.publish("4ID3_G7/pressure", String(bmpEvent.pressure).c_str());
  delay(100);
  client.publish("4ID3_G7/light", String(lux).c_str());

  delay(DELAY_BETWEEN_SAMPLES_MS);
}
```

Ensure the code compiles and reupload the modified code to the microcontroller.


COM3
Send
....
WiFi connected
IP address:
192.168.137.106
-
Temperature: 21.90 degC
Humidity: 37.00 %
Pressure: 989.69 hPa
Light Intensity: 0.00 lux
Attempting MQTT connection...Connected to MQTT server
-
Temperature: 21.70 degC
Humidity: 36.00 %
Autoscroll
Show timestamp
Newline
115200 baud
Clear output

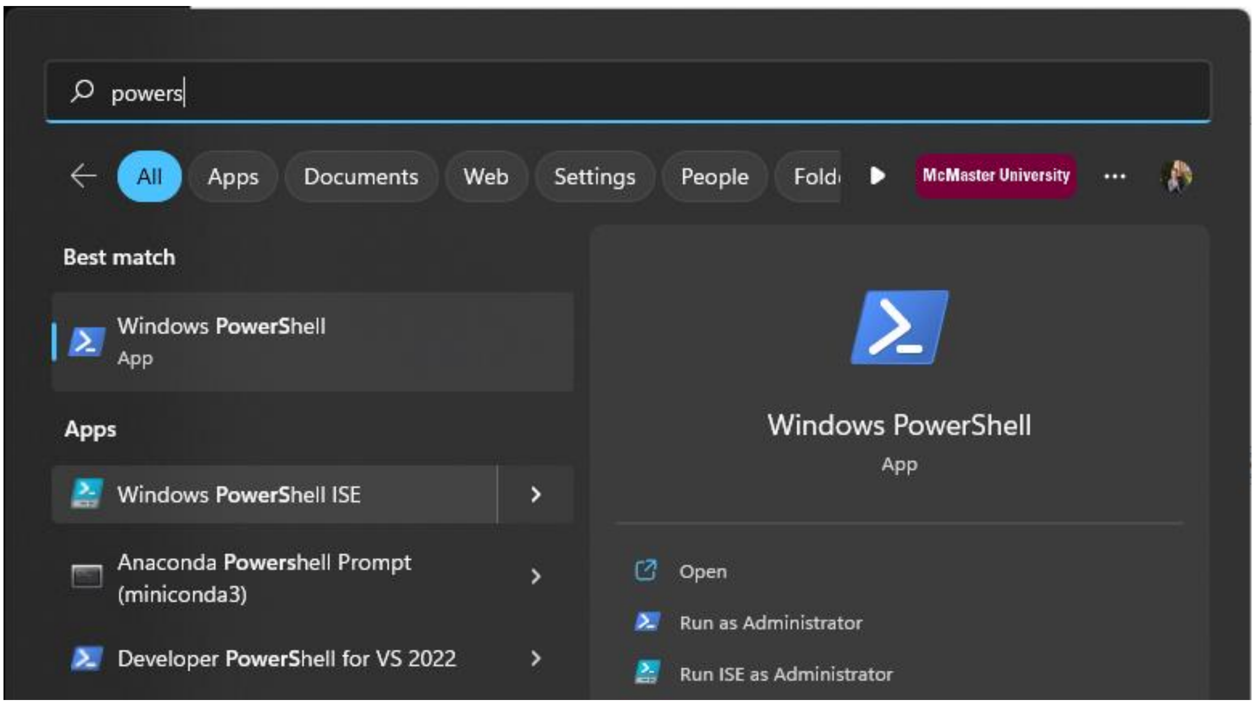
Verifying Connection

We will be using 2 different devices and applications to verify that the data is being published to the corresponding topics.

a) Using the Mosquitto terminal application
b) Using a mobile application

Mosquitto Terminal Application

Open Powershell and navigate to the install directory of the Mosquitto MQTT broker, that was installed in the pre-lab.



```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\adama> cd 'C:\Program Files\mosquitto\'
PS C:\Program Files\mosquitto> ls


    Directory: C:\Program Files\mosquitto


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        2023-01-15   7:23 PM                devel
-a----        2022-08-16   9:34 AM            230 aclfile.example
-a----        2022-08-16   9:34 AM         135368 ChangeLog.txt
-a----        2022-08-16   9:34 AM           1568 edl-v10
-a----        2022-08-16   9:34 AM          14197 epl-v20
-a----        2022-07-05   5:43 PM        3415552 libcrypto-1_1-x64.dll
-a----        2022-07-05   5:43 PM         685056 libssl-1_1-x64.dll
-a----        2022-08-16   9:34 AM          40449 mosquitto.conf
-a----        2022-08-16   9:35 AM          87040 mosquitto.dll
-a----        2022-08-16   9:41 AM         382464 mosquitto.exe
-a----        2022-08-16   9:35 AM          18432 mosquittopp.dll
-a----        2022-08-16   9:35 AM          76288 mosquitto_ctrl.exe
-a----        2022-08-16   9:37 AM         122880 mosquitto_dynamic_security.dll
-a----        2022-08-16   9:34 AM          22528 mosquitto_passwd.exe
-a----        2022-08-16   9:35 AM          51712 mosquitto_pub.exe
-a----        2022-08-16   9:35 AM          79872 mosquitto_rr.exe
-a----        2022-08-16   9:35 AM          81920 mosquitto_sub.exe
-a----        2022-08-16   9:34 AM           1886 NOTICE.md
-a----        2022-08-16   9:34 AM            355 pwfile.example
-a----        2022-08-16   9:34 AM            939 README-letsencrypt.md
-a----        2022-08-16   9:34 AM           2453 README-windows.txt
-a----        2022-08-16   9:34 AM           3768 README.md
-a----        2023-01-15   7:23 PM          66085 Uninstall.exe


PS C:\Program Files\mosquitto>
```

Launch the subscribe applications with the following flags:

-h - MQTT broker IP address

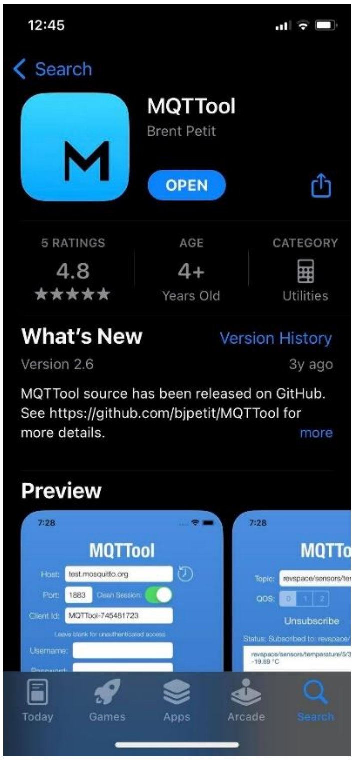
-t - the topic that your wish to connect to

```
Windows PowerShell
PS C:\Program Files\mosquitto> .\mosquitto_sub.exe -h test.mosquitto.org -t "4ID3_G7/humidity"
37.00
37.00
87.00
60.00
51.00
46.00
43.00
39.00
39.00
63.00
51.00
45.00
41.00
37.00
38.00
37.00
38.00
```

After waiting a period of time, you should see the values being printed to the terminal window every 20 seconds.

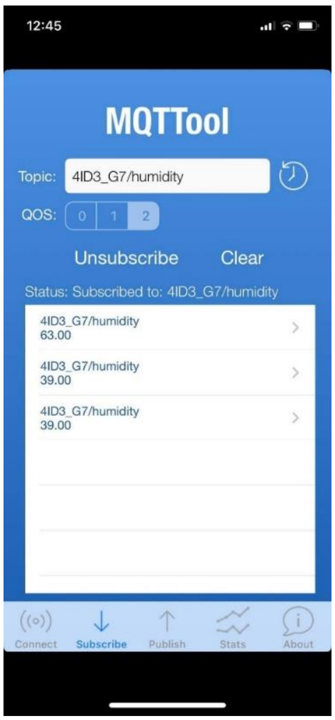
Using a Mobile Application

The phone application will likely be delayed from when the terminal application receives the message. If you are using iOS, install the following app:



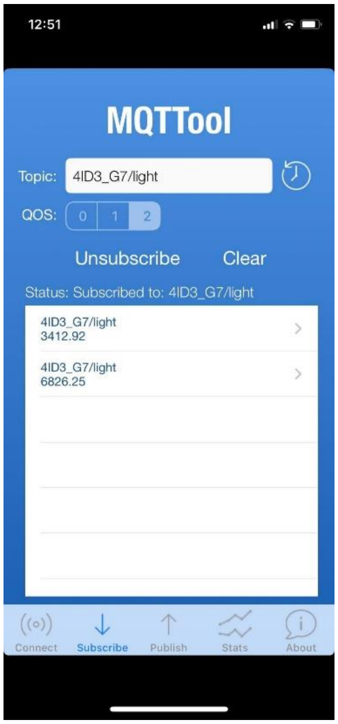
Next, connect to the public mosquito broker:

Next, subscribe to the topic of choice.



Lastly, wait for values to populate.

You can also try it with other sensor readings to ensure all of them are working correctly.

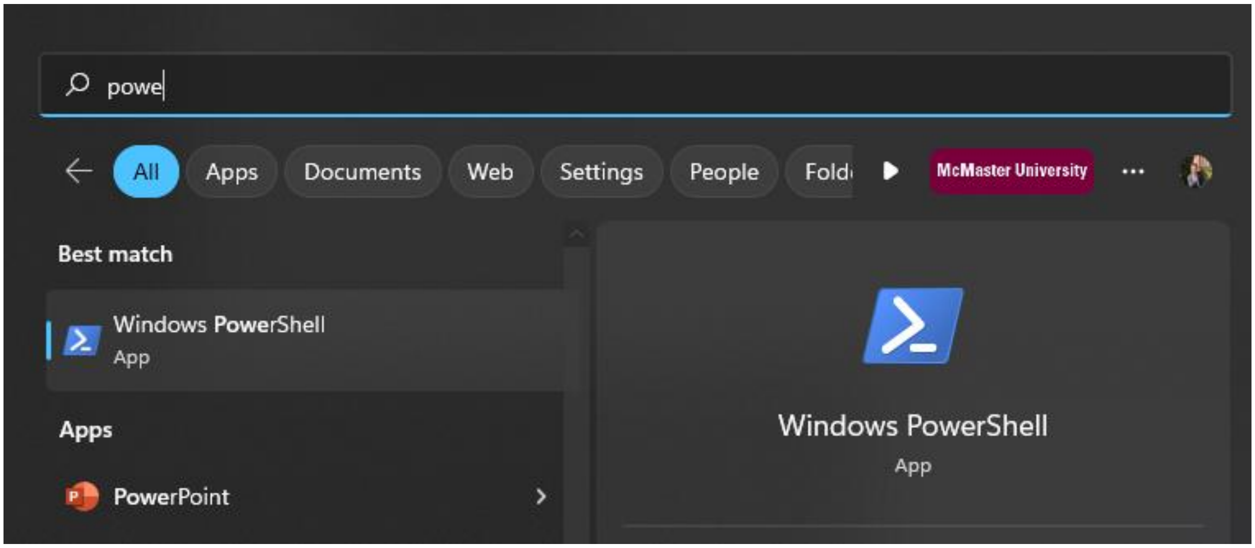


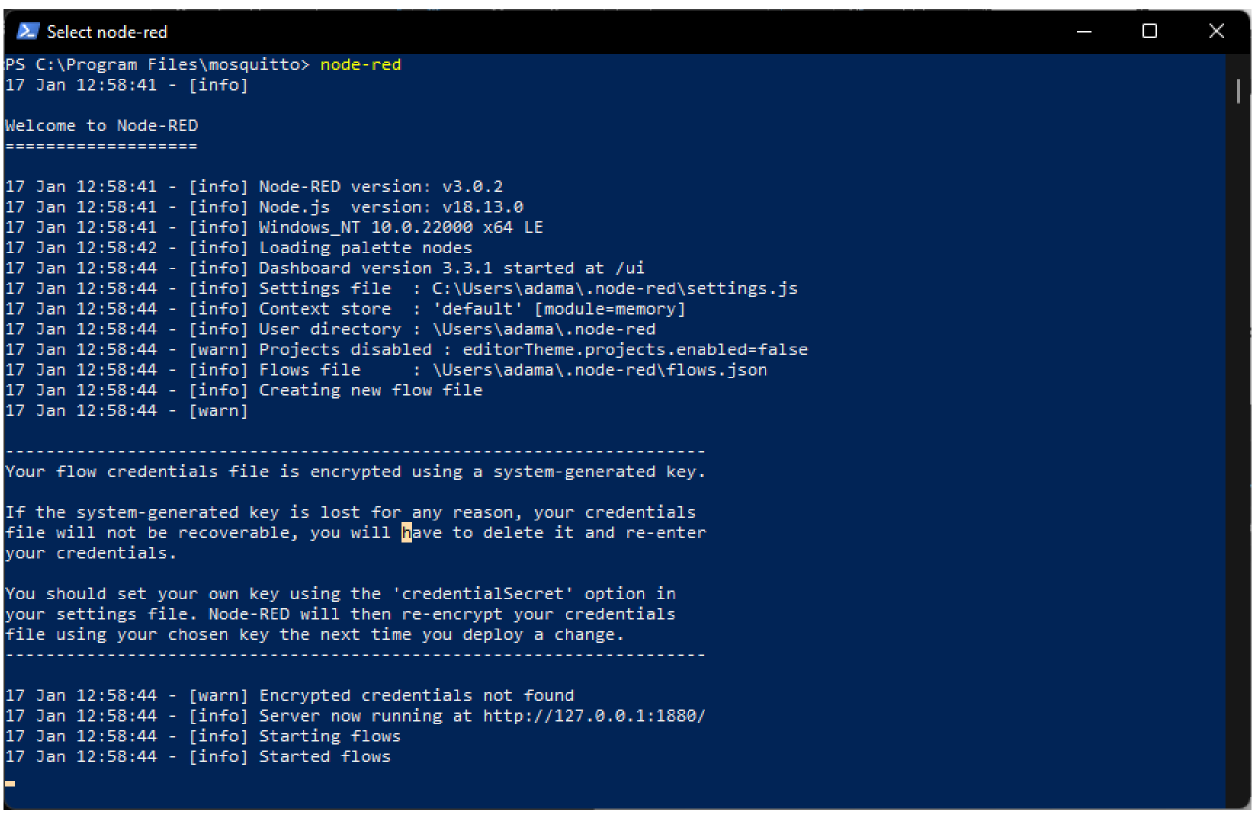
NodeRED Dashboard

Please follow the pre-lab instructions to install NodeRED.

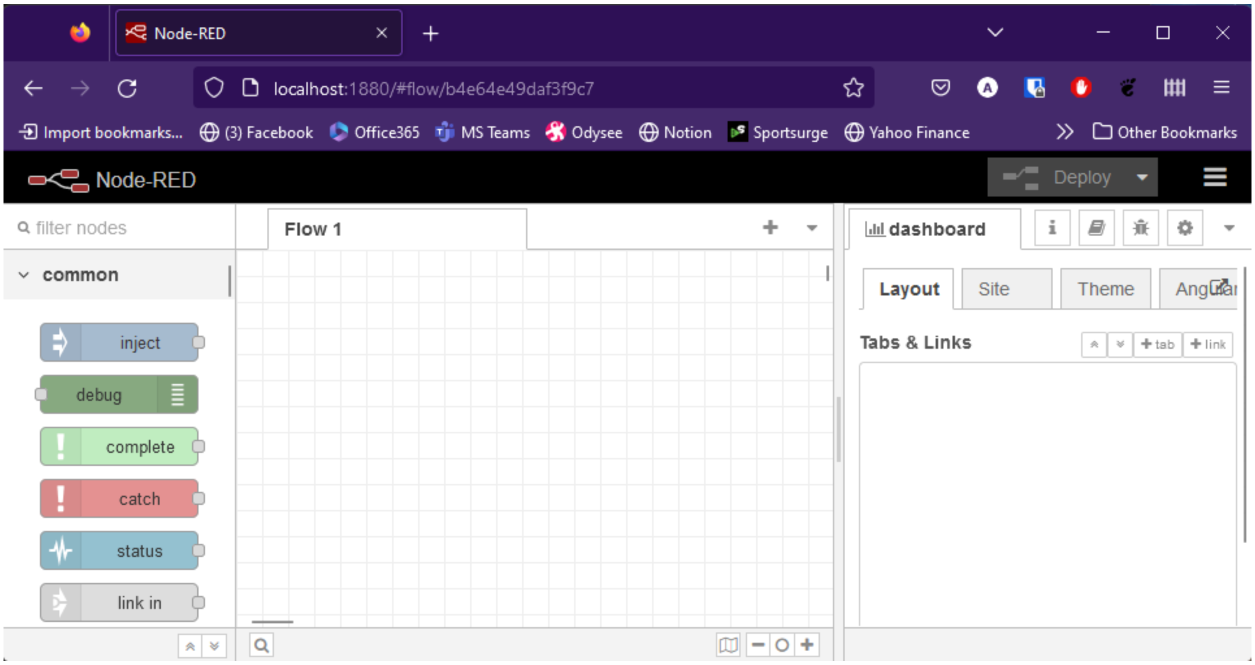
To start NodeRED, open Powershell and type the command:

node-red

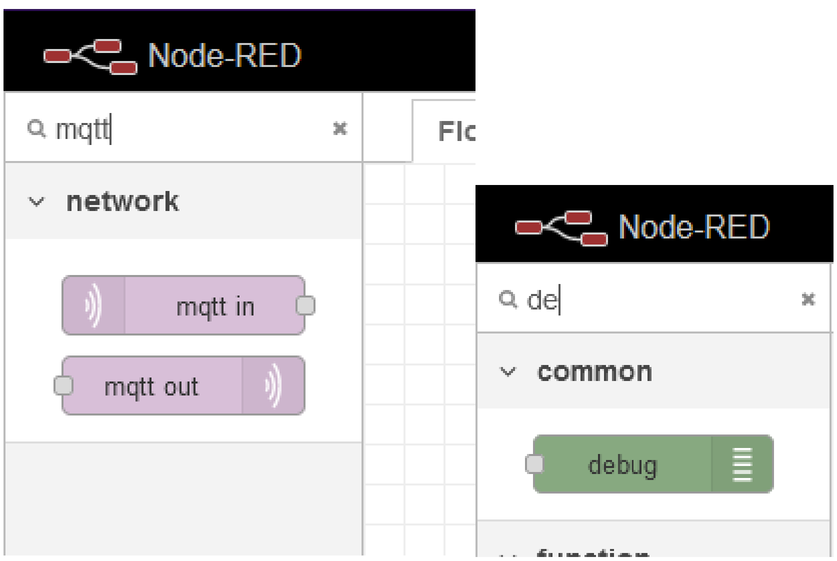




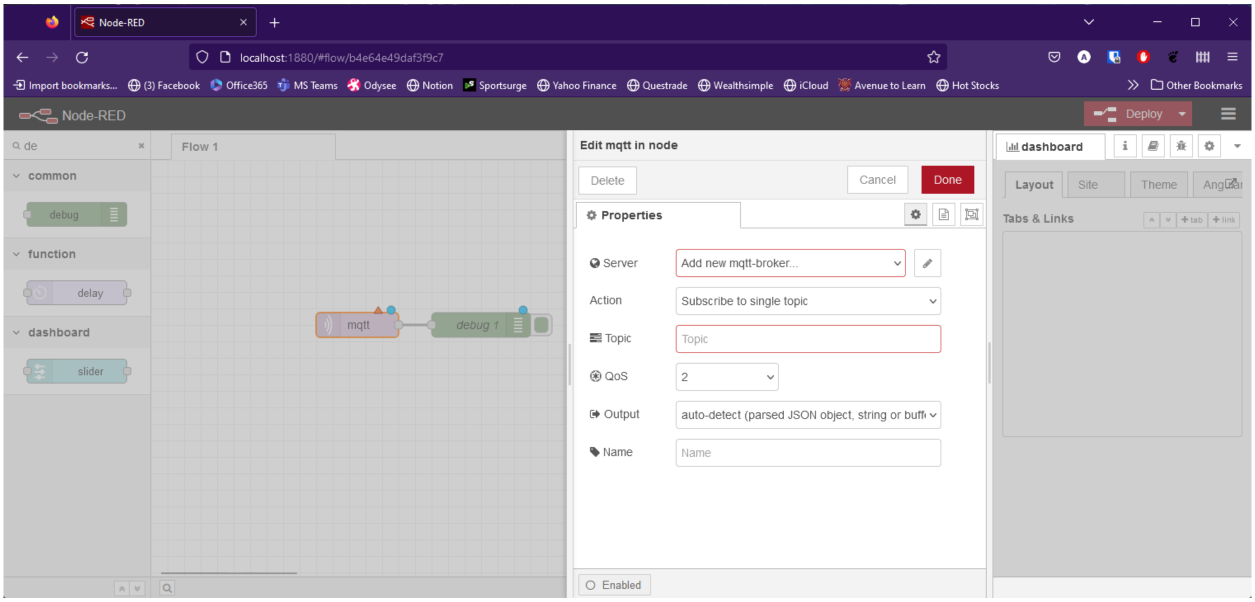
Navigate to the URL presented, in a web browser.



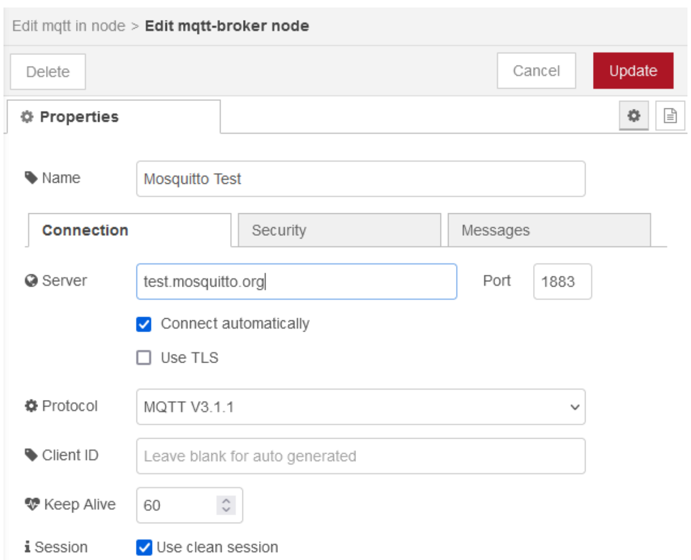
Filter nodes to find the **mqtt in** node and the **debug** node. Drag them into your flow diagram.



Click on the **mqtt in** node to begin editing its configuration.

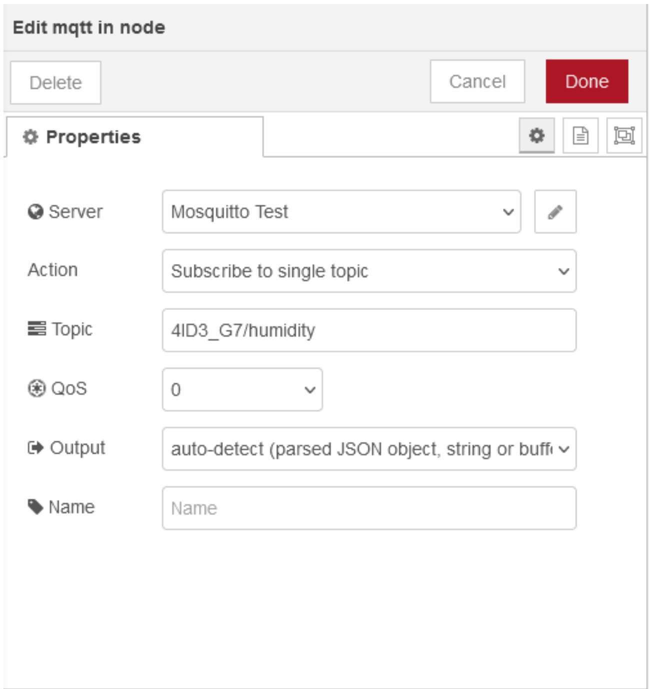


Fill in the information for the public broker.

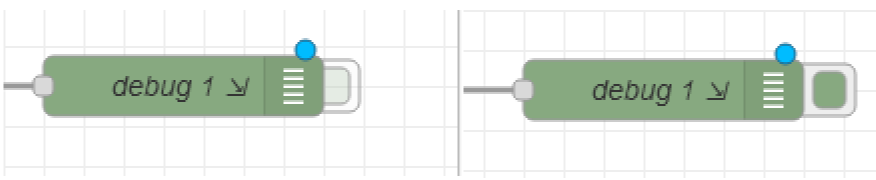


Press **Update**.

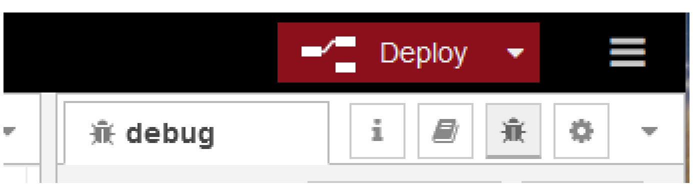
Under **Properties** fill in the **Topic** field.

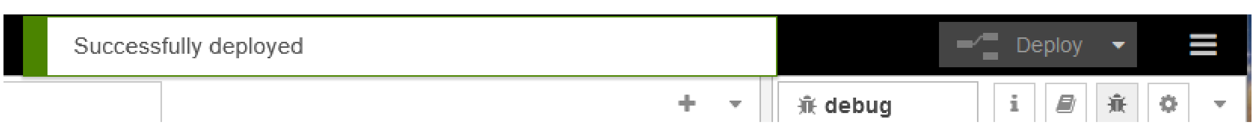


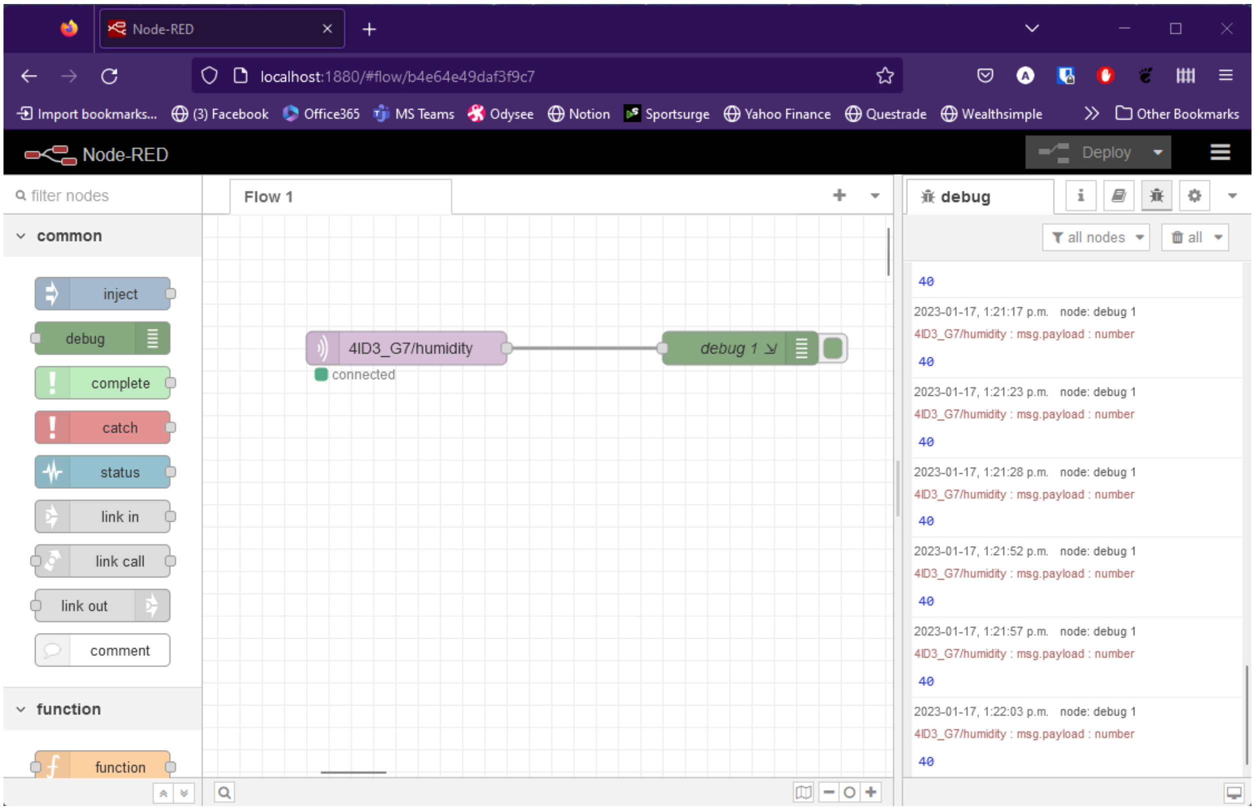
Enable the **debug** node by pressing the **green box**.



Lastly, press **Deploy** and watch the **Debug Panel** populate with sensor values.



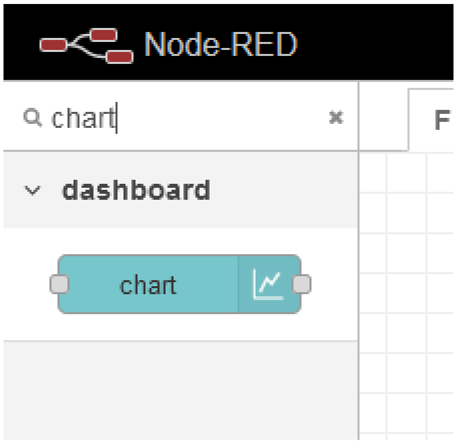



Node-RED
localhost:1880/#flow/b4e64e49daf3f9c7
Deploy
filter nodes
common
inject
debug
complete
catch
status
link in
link call
link out
comment
function
function
Flow 1
4ID3_G7/humidity
connected
debug 1
debug
all nodes
all
40
2023-01-17, 1:21:17 p.m. node: debug 1
4ID3_G7/humidity : msg.payload : number
40
2023-01-17, 1:21:23 p.m. node: debug 1
4ID3_G7/humidity : msg.payload : number
40
2023-01-17, 1:21:28 p.m. node: debug 1
4ID3_G7/humidity : msg.payload : number
40
2023-01-17, 1:21:52 p.m. node: debug 1
4ID3_G7/humidity : msg.payload : number
40
2023-01-17, 1:21:57 p.m. node: debug 1
4ID3_G7/humidity : msg.payload : number
40
2023-01-17, 1:22:03 p.m. node: debug 1
4ID3_G7/humidity : msg.payload : number
40

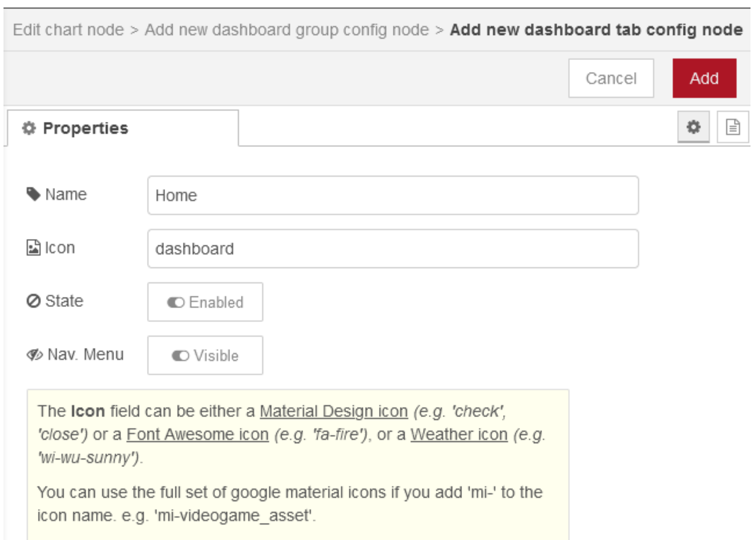
Visualizing NodeRED Data

In the pre-lab, the node-red-dashboard add-on was installed which enables us to create a dashboard of graphs, charts, and toggles to visualize and control data.

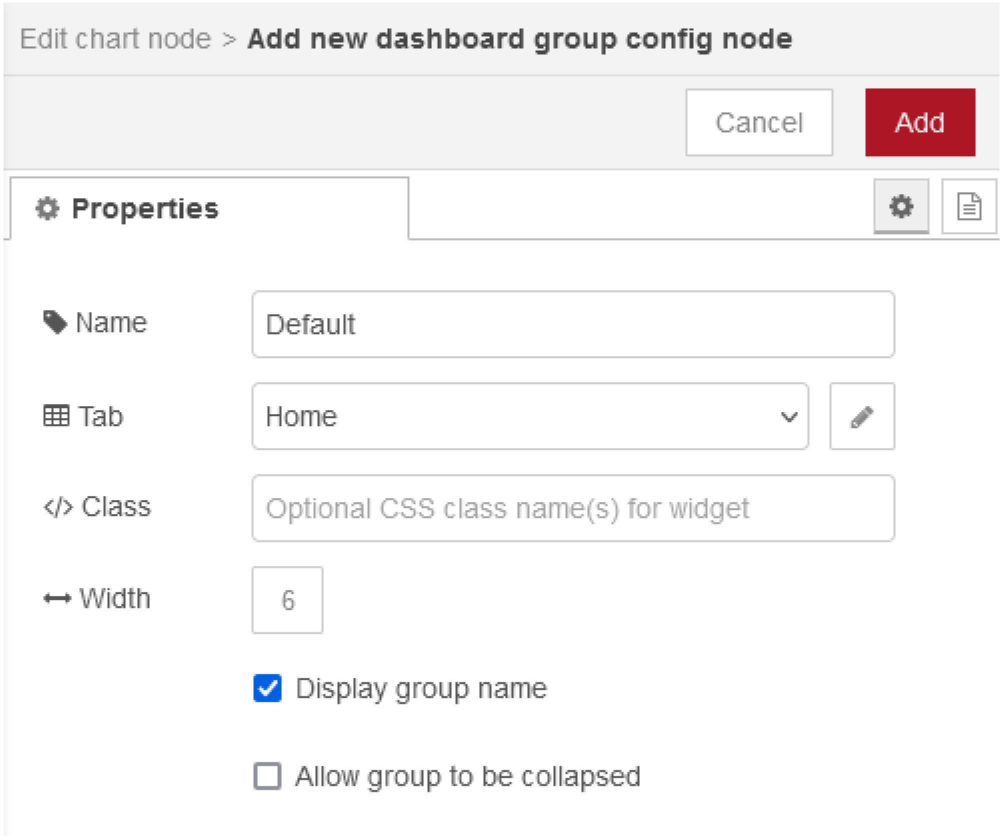
In the **filter nodes** field, search for **chart** and drag it into your flow diagram.



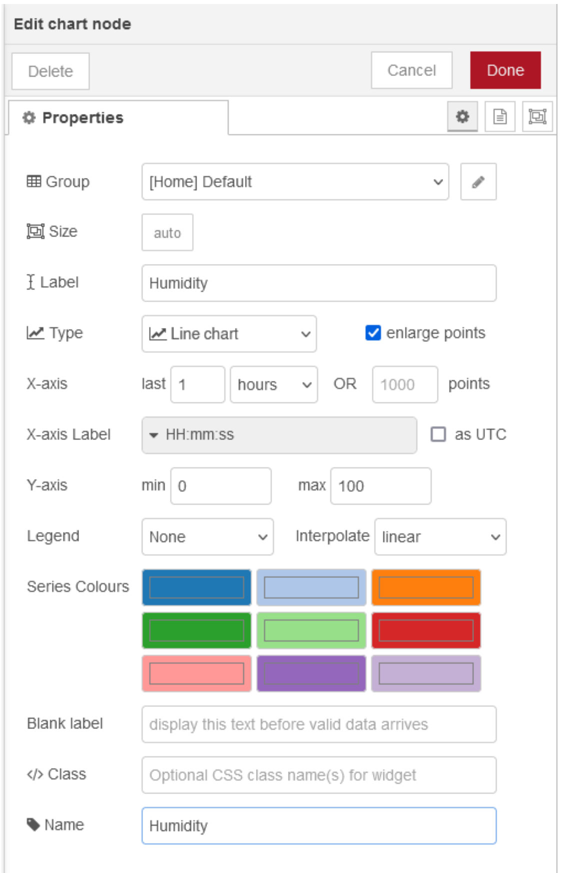
When you click on the node, you will need to create a new **Dashboard Tab**. Use the default name and press **Add**.



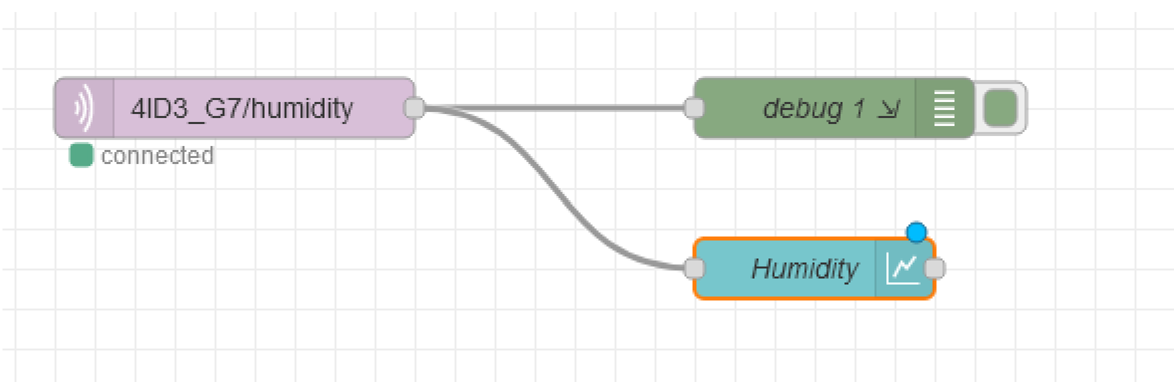
Add the dashboard group to that new dashboard by pressing **Add**.



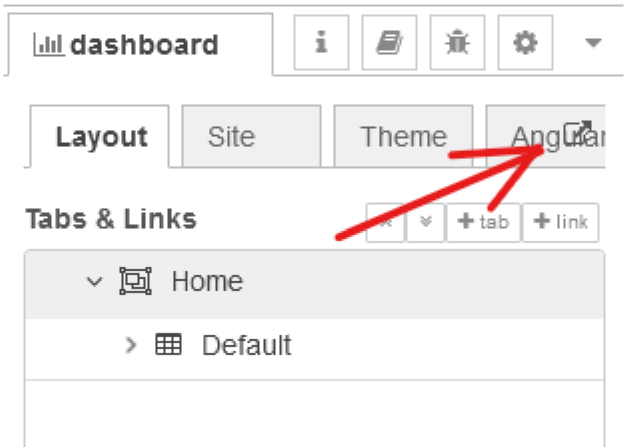
Lastly, edit the chart node to visualize your data nicely. Press **Done** when complete.



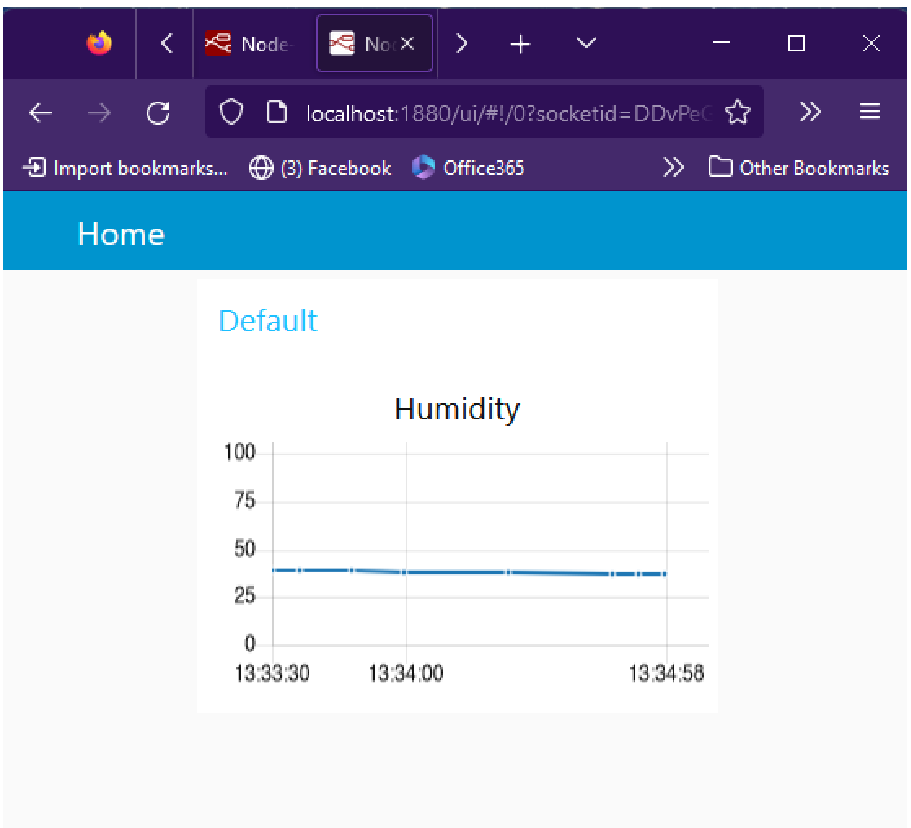
Connect your **mqtt in** node to the input of your **chart** node. Press **Deploy** to save changes.



To view the dashboard, either append **ui/** to your url (http://localhost:1880/ui) or press the **open dashboard** icon in the top right corner of the dashboard panel.

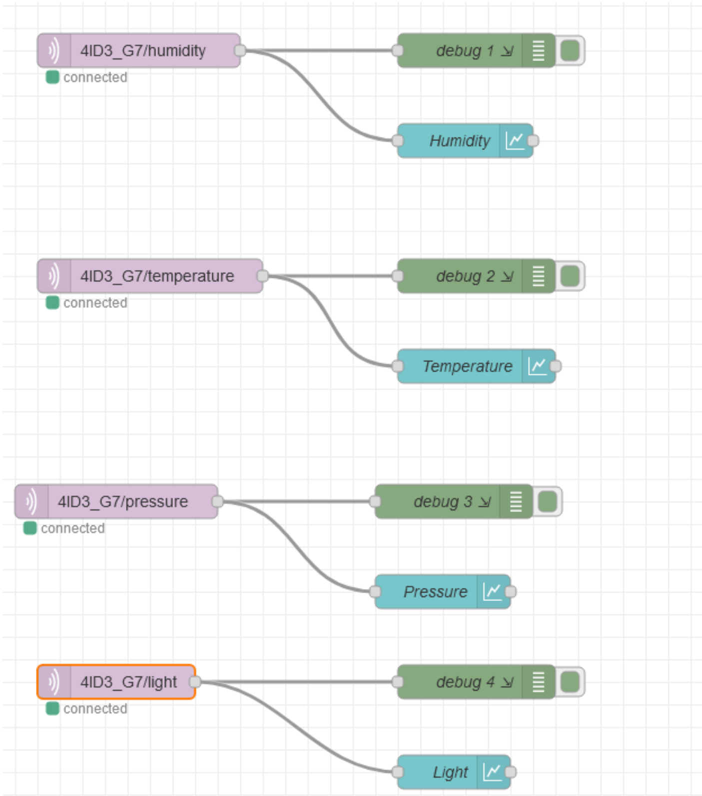


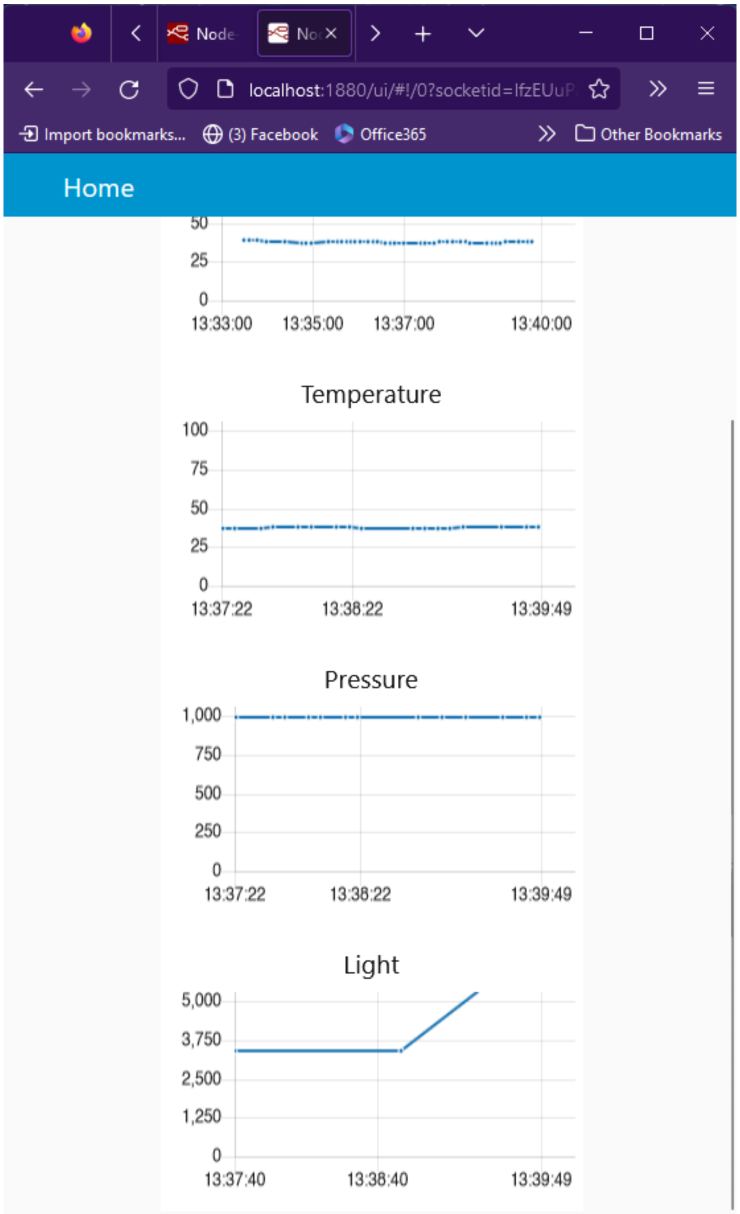
Wait for data to populate.



Now that you have seen how to visualize one topic, attempt to visualize the rest.

The resultant flow should look like this:


4ID3_G7/humidity
connected
debug 1
Humidity
4ID3_G7/temperature
connected
debug 2
Temperature
4ID3_G7/pressure
connected
debug 3
Pressure
4ID3_G7/light
connected
debug 4
Light



If values are not showing up, ensure that your y-axis scale is correct.

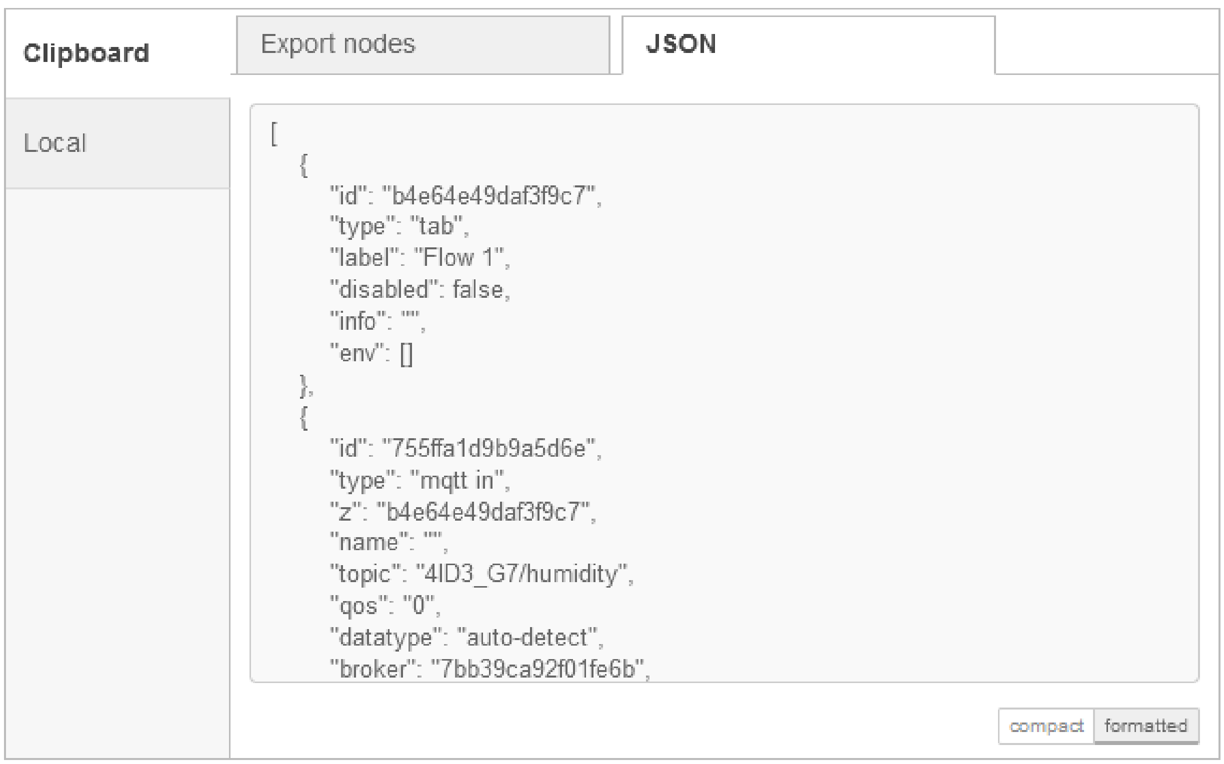
Saving and Pushing Your Project

Exporting a NodeRED Flow as JSON

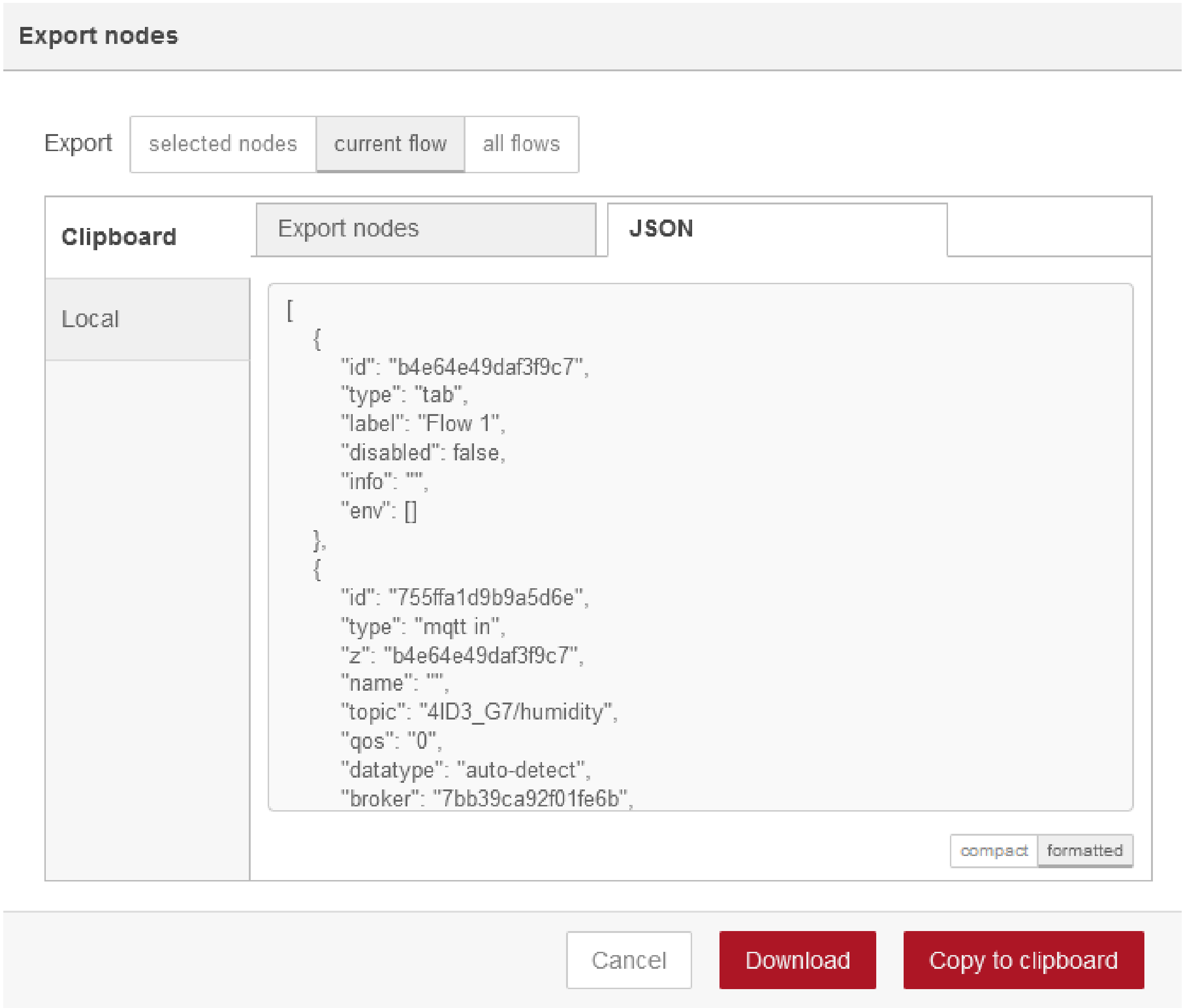
Click on the **hamburger menu** and select **export**.

Select **current flow**.

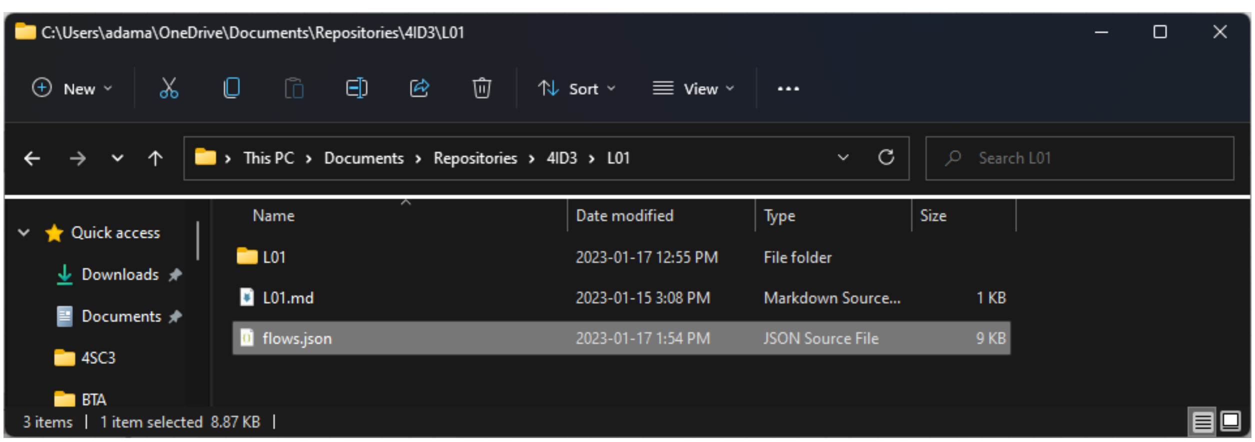
Select **JSON**.



Press **Download**.

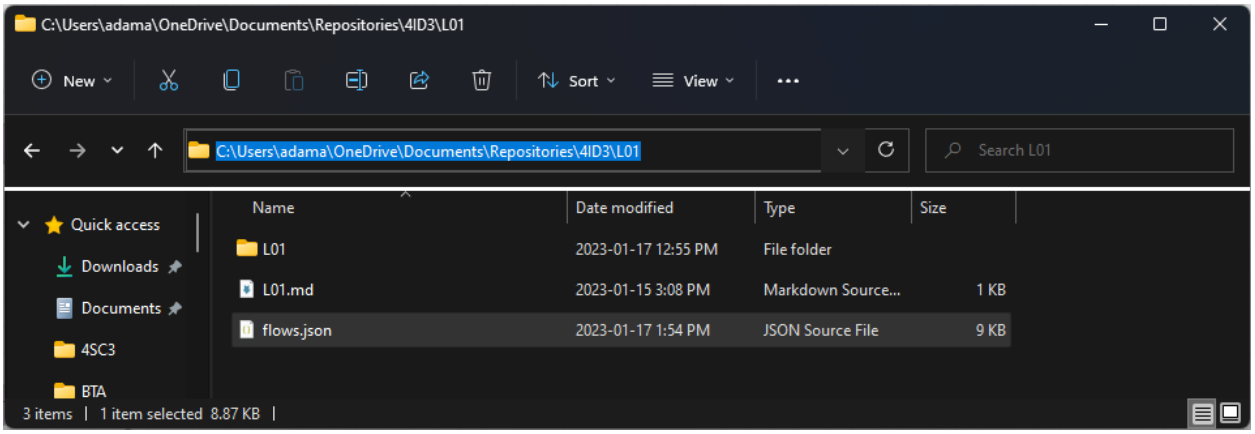


Cut and paste the **flows.json** file from your **Downloads/** folder to the **L01** folder in the **local repo.**



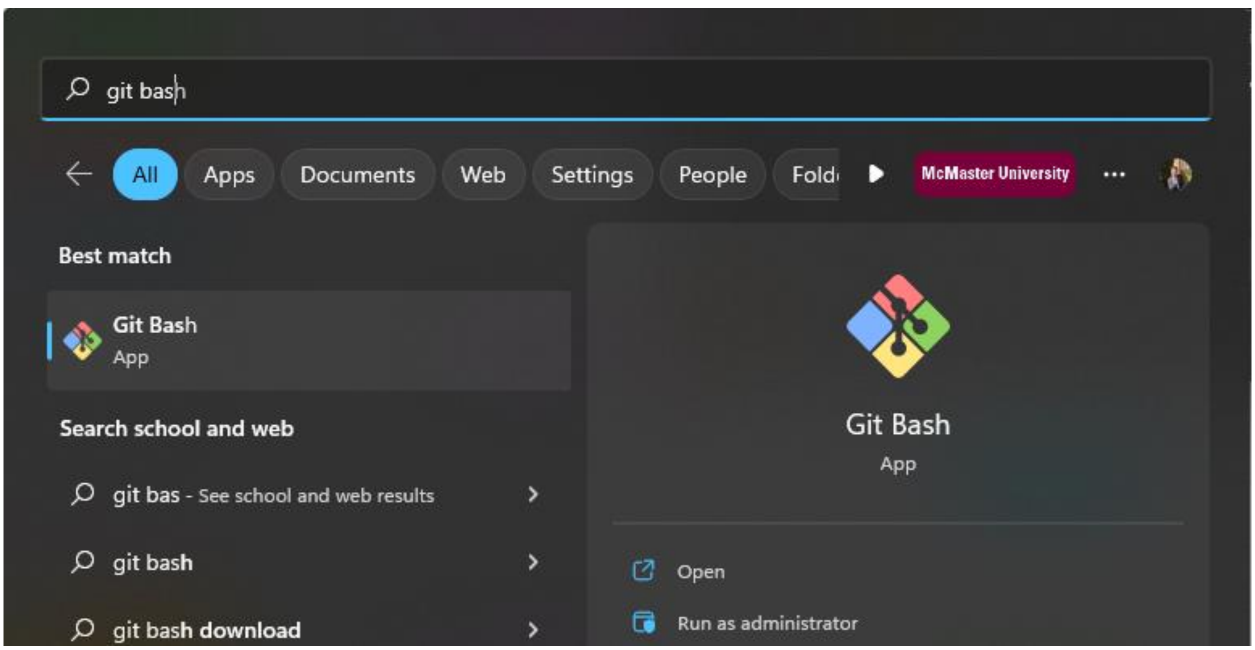
Committing and Pushing Changes to GitHub

Click on the little **folder icon** in the **navigation bar** to view to **path**.



Copy this path to clipboard.

Open **Git Bash**.



Use the **cd** command to change directories into the L01 folder.

```
MINGW64:/c/Users/adama/OneDrive/Documents/Repositories/4ID3/L01

adama@DESKTOP-9L3EOVQ MINGW64 ~
$ cd "C:\Users\adama\OneDrive\Documents\Repositories\4ID3\L01"

adama@DESKTOP-9L3EOVQ MINGW64 ~/OneDrive/Documents/Repositories/4ID3/L01 (main)
$ ls -la
total 13
drwxr-xr-x 1 adama 197609    0 Jan 17 13:55 ./
drwxr-xr-x 1 adama 197609    0 Jan 15 14:58 ../
drwxr-xr-x 1 adama 197609    0 Jan 17 12:55 L01/
-rw-r--r-- 1 adama 197609  177 Jan 15 15:08 L01.md
-rw-r--r-- 1 adama 197609 9092 Jan 17 13:54 flows.json

adama@DESKTOP-9L3EOVQ MINGW64 ~/OneDrive/Documents/Repositories/4ID3/L01 (main)
$
```

Go back one directory using the **cd ..** command. You should be in the **4ID3 folder** now.

```
MINGW64:/c/Users/adama/OneDrive/Documents/Repositories/4ID3
drwxr-xr-x 1 adama 197609    0 Jan 17 12:55 L01/
-rw-r--r-- 1 adama 197609  177 Jan 15 15:08 L01.md
-rw-r--r-- 1 adama 197609 9092 Jan 17 13:54 flows.json

adama@DESKTOP-9L3EOVQ MINGW64 ~/OneDrive/Documents/Repositories/4ID3/L01 (main)
$ cd ..

adama@DESKTOP-9L3EOVQ MINGW64 ~/OneDrive/Documents/Repositories/4ID3 (main)
$ ls -la
total 5
drwxr-xr-x 1 adama 197609  0 Jan 15 14:58 ./
drwxr-xr-x 1 adama 197609  0 Jan 15 14:56 ../
drwxr-xr-x 1 adama 197609  0 Jan 15 20:17 .git/
drwxr-xr-x 1 adama 197609  0 Jan 17 13:55 L01/
-rw-r--r-- 1 adama 197609 14 Jan 15 12:47 README.md

adama@DESKTOP-9L3EOVQ MINGW64 ~/OneDrive/Documents/Repositories/4ID3 (main)
$ |
```

Add all the new changes using the **git add .** command.

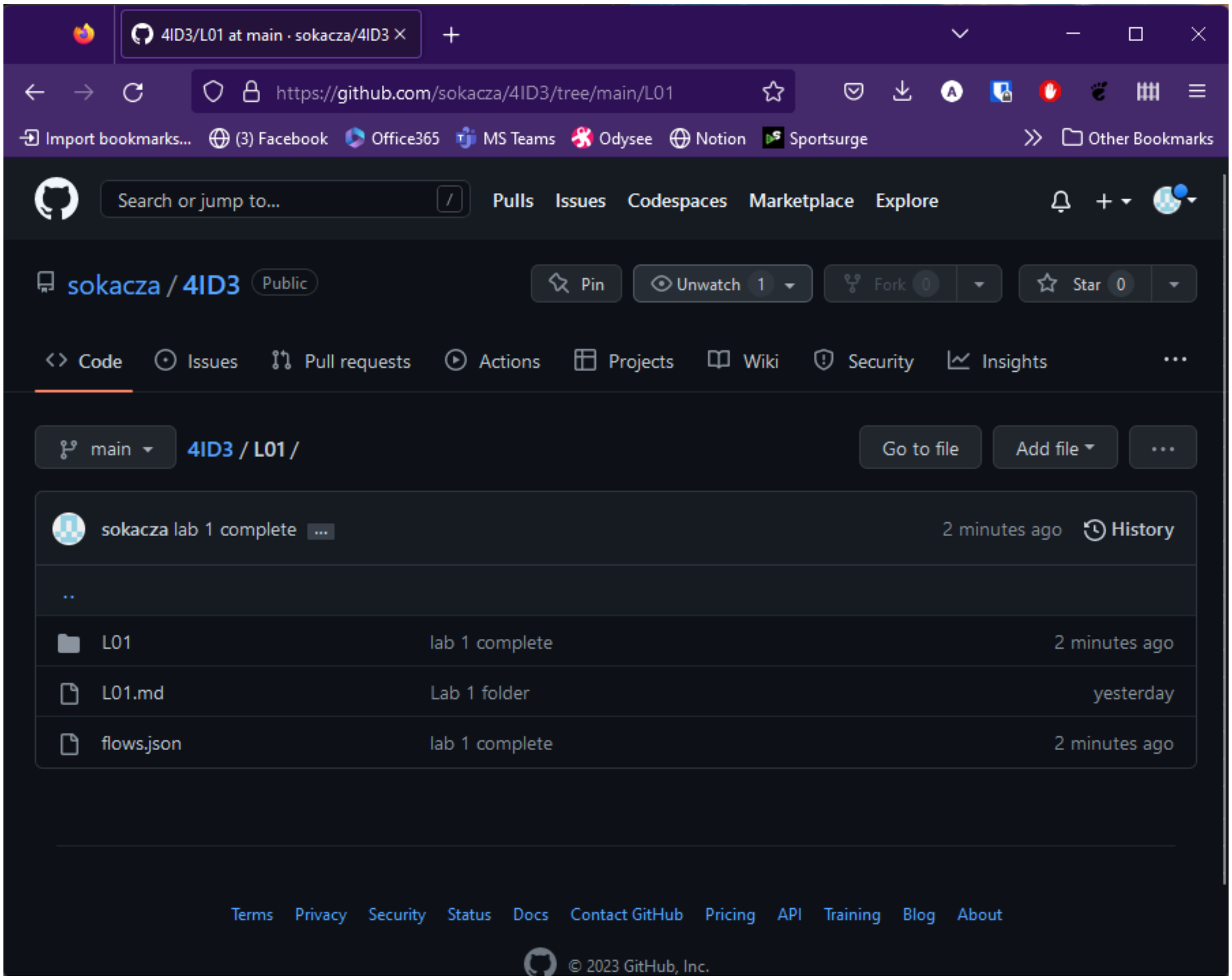
Commit the changes using the **git commit -m ""** command. Use a message that represents what has been changed since the previous commit.

```
MINGW64:/c/Users/adama/OneDrive/Documents/Repositories/4ID3

adama@DESKTOP-9L3EOVQ MINGW64 ~/OneDrive/Documents/Repositories/4ID3 (main)
$ git add .
warning: in the working copy of 'L01/flows.json', LF will be replaced by CRLF the nex
t time Git touches it

adama@DESKTOP-9L3EOVQ MINGW64 ~/OneDrive/Documents/Repositories/4ID3 (main)
$ git commit -m "lab 1 complete"
[main 0aa1d9c] lab 1 complete
 3 files changed, 522 insertions(+)
 create mode 100644 L01/L01/L01.h
 create mode 100644 L01/L01/L01.ino
 create mode 100644 L01/flows.json

adama@DESKTOP-9L3EOVQ MINGW64 ~/OneDrive/Documents/Repositories/4ID3 (main)
$ |
```

Push the changes in the **main branch** of your **local repo** to the remote **GitHub repo** named **origin**.

```
MINGW64:/c/Users/adama/OneDrive/Documents/Repositories/4ID3

adama@DESKTOP-9L3EOVQ MINGW64 ~/OneDrive/Documents/Repositories/4ID3 (main)
$ git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 3.15 KiB | 3.15 MiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:sokacza/4ID3.git
   c2b196d..0aa1d9c  main -> main

adama@DESKTOP-9L3EOVQ MINGW64 ~/OneDrive/Documents/Repositories/4ID3 (main)
$ |
```

Log into your GitHub account to verify that all changes have been uploaded.


4ID3/L01 at main · sokacza/4ID3
https://github.com/sokacza/4ID3/tree/main/L01
Import bookmarks... (3) Facebook Office365 MS Teams Odysee Notion Sportsurge Other Bookmarks
Search or jump to...
Pulls Issues Codespaces Marketplace Explore
sokacza / 4ID3 Public
Pin Unwatch 1 Fork 0 Star 0
Code Issues Pull requests Actions Projects Wiki Security Insights
main 4ID3 / L01 /
Go to file Add file
sokacza lab 1 complete
2 minutes ago History
L01 lab 1 complete 2 minutes ago
L01.md Lab 1 folder yesterday
flows.json lab 1 complete 2 minutes ago
Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About
© 2023 GitHub, Inc.

END