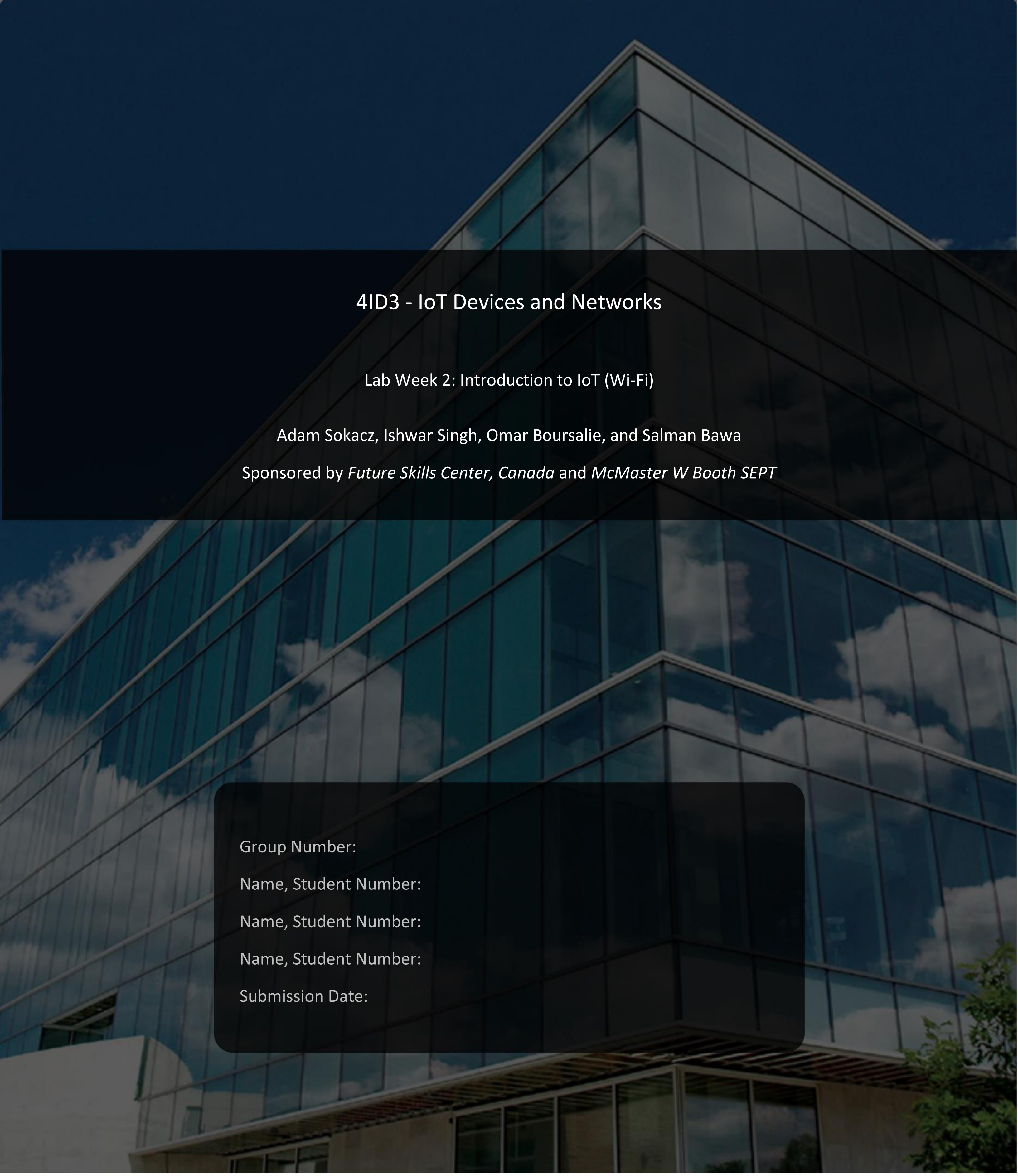


4ID3 - IoT Devices and Networks

Lab Week 2: Introduction to IoT (Wi-Fi)

Adam Sokacz, Ishwar Singh, Omar Boursalie, 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

A practice-based understanding of the “Internet” and “Things” of the Internet of Things, including temperature, pressure, humidity, and light intensity sensors, connected to a microcontroller with Publication-Subscription client, communicating over 802.11ah Wi-Fi to an MQTT server on the cloud, corresponding with a Node-Red application, showcasing outcome in a Node-Red user interface.

Learning Outcomes

By the end of this lab, students will:

1. Learn appreciation for Node-Red, Mosquitto, MQTT, various standard sensors, ESP8266 microcontroller, Arduino IDE and its specific use cases for a typical IoT setup
2. Learn to install programs on their computers and smartphones
3. Learn about Wi-Fi as an Access Technology and appreciate the criteria for choosing Wi-Fi for a given application

Contents

1. Lab Submission Formats/Requirements

Cover Page

Each submission must have a cover page clearly indicating the lab, title, instructors and TAs names, date, avenue group number, student name(s), student MACID(s), and student number(s). In addition, the cover page must have the following statement for each student:

> As a future professional member, the student is responsible for honestly performing the required work without plagiarism and cheating. Submitting this work with my name and student number is a statement of understanding that this work is our own and adheres to the Academic Integrity Policy of McMaster University. [**Full name, MACID, student number**]

Team Submission

A team must remain consistent for all the deliverables of a given lab. This means you cannot submit a prelab for a given lab with one partner but submit the lab report with a different partner. However, if one of the team members does not participate in part of the work, the other team member may submit that part individually. In addition, teams are to remain consistent throughout the entire term. Students are cautioned against the divide-and-conquer approach because both team members may be examined on lab submission/milestone content. You cannot be awarded the marks if you cannot answer the question(s). You also require full knowledge of the labs to complete the final project.

- Only one member of the team shall submit the work. All members of the team must have their copy of the work for future reference
- If you cannot answer questions from your instructor/TA or your information is not on the cover page, then you will not receive credit for the work
- All submitted work needs to be clear and concise. Work that is not legible, unclear, or too verbose will be penalized accordingly.

Milestone Demonstrations

As you complete the milestones during your scheduled lab section, you must request a TA or instructor to evaluate and record your work. You may attempt the same milestone multiple times but remember to complete your lab within three hours.

Please remember that your instructor/TA must visit multiple groups. It is essential to have your questions and/or demonstrations ready.

Please note that milestone grades are final after the lab is complete.

Remember to upload your final work to Avenue by the deadline. Check that you have uploaded the correct file(s). We cannot accept late files.

2. Pre-Lab Questions [10 marks]

In addition to the cover page, include the text of the original question followed by your clear and concise answer. Answers to design or calculation questions will receive a zero if they are not accompanied by supporting work.

Pre-labs are due at 2:45 on the day of your lab.

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, 3 points)

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

(Suggested: 3 sentences, 2 points)

Q3 – In your own words, what is the role of each field below when connecting to an MQTT broker?

Host IP Address	
Topic Name	
Protocol (TCP/WS/WSS/TLS)	

(Suggested: Paragraph or screenshot, 3 points)

Q4 – View the status of your local Git repository to verify that there aren't any uncommitted changes. Pull changes from GitHub to ensure that your local repository is up to date. Include a screenshot that demonstrates you viewed the status of your repository.

Note: Team member names and student numbers *must* appear embedded in the screenshot, not added to the post-image capture. For example, you can include student names and numbers in a notepad alongside the terminal output. -5 marks if not shown.

(Suggested: Screenshot, 2 points)

3. Experiment

3.1. Integrated Circuits & Components

The following integrated circuits are to be used in this laboratory

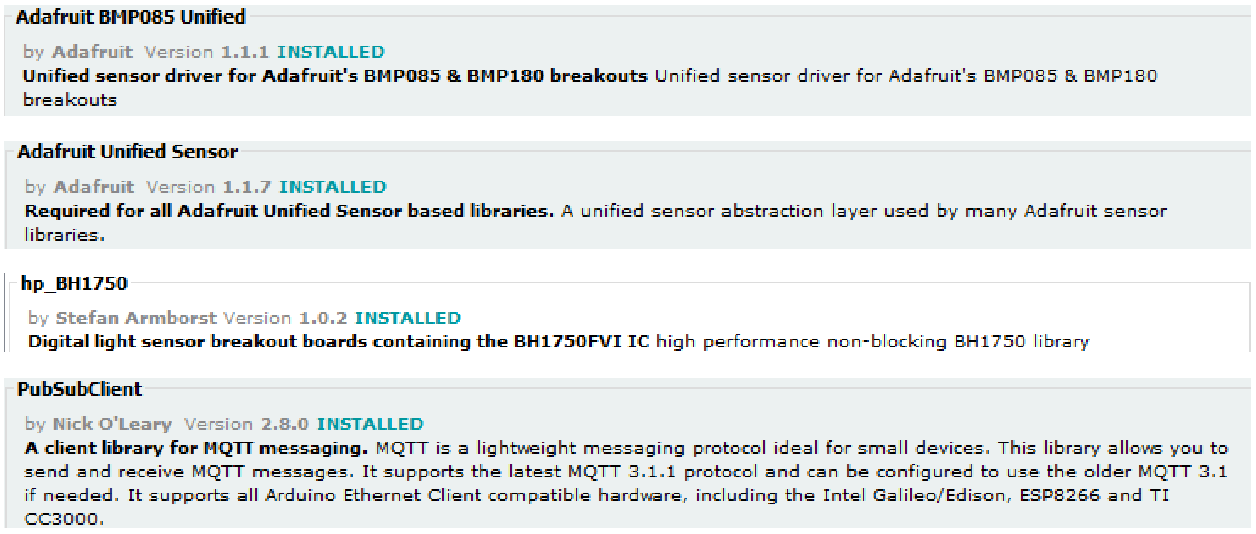
Identification	**Description**
NodeMCU v3 ESP8266	Development Board
GY-30 BH1750FVI	Light Intensity Sensor
GY-68 BMP180	Barometric Pressure Sensor
DHT11 Module	Temperature and Humidity Sensor

Obtain the data sheets for each of the above devices. Please familiarize yourself with their logical and Electrical characteristics: Bring a copy to your lab session.

3.2. Software to Install

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



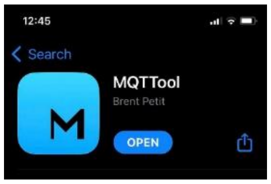
Adafruit BMP085 Unified
by **Adafruit** Version **1.1.1** **INSTALLED**
Unified sensor driver for Adafruit's BMP085 & BMP180 breakouts Unified sensor driver for Adafruit's BMP085 & BMP180 breakouts

Adafruit Unified Sensor
by **Adafruit** Version **1.1.7** **INSTALLED**
Required for all Adafruit Unified Sensor based libraries. A unified sensor abstraction layer used by many Adafruit sensor libraries.

hp_BH1750
by **Stefan Armborst** Version **1.0.2** **INSTALLED**
Digital light sensor breakout boards containing the BH1750FVI IC high performance non-blocking BH1750 library

PubSubClient
by **Nick O'Leary** Version **2.8.0** **INSTALLED**
A client library for MQTT messaging. MQTT is a lightweight messaging protocol ideal for small devices. This library allows you to send and receive MQTT messages. It supports the latest MQTT 3.1.1 protocol and can be configured to use the older MQTT 3.1 if needed. It supports all Arduino Ethernet Client compatible hardware, including the Intel Galileo/Edison, ESP8266 and TI CC3000.

Install *MQTTool* on an iOS device (only one team member needs the app). Available at:
https://apps.apple.com/ca/app/MQTTool/id1085976398



3.3. Additional Resources

- ESP8266 Overview: https://youtu.be/dGrJi-ebZgI
- Mosquitto MQTT Broker Tutorial: https://youtu.be/DH-VSAACtBk
- NodeRED Fundamentals Tutorial: https://youtu.be/3AR432bguOY

3.4. Hello World! Microcontroller to Serial Monitor [25 marks]

This milestone aims to interface sensors with a microcontroller, read data from three sensors (BH1750, BMP 180, & DHT11), and print it to the serial monitor.

3.4.1. ESP8266 Program

1. Launch the Arduino IDE and File → New Sketch.
2. File → Save As → Go to your Wi-Fi folder in Repositories and save as wifi_a (this will create an Arduino folder (called a sketchbook) called wifi_a and an Arduino file inside wifi_a also called wifi_a.ino
3. Close any other Arduino sketch windows that may be open
4. Add all team members' names and student numbers (one line per student) as comments (%)
5. After team members' names and student numbers, add the following line : #include "wifi_a.h"
6. Set up the void setup() function.

File: wifi_a.ino

```
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);

}
```

7. Start the void loop() function. Note that the DHT and BMP sensors are part of the same Adafruit unified library so that they will be polled differently than the BH sensor.

File: wifi_a.ino

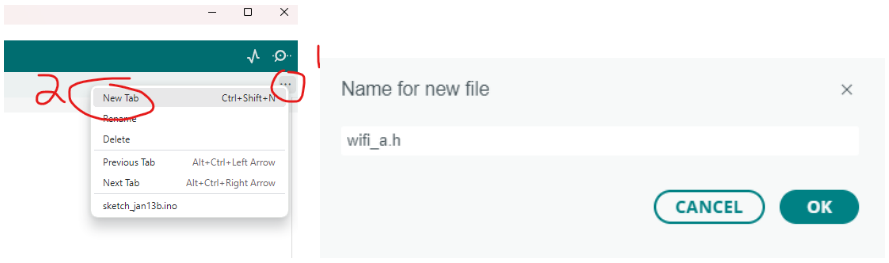
```
void loop() {

   //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();
}
```

8. Add the following code to the loop() to print the sensor readings to the serial monitor. We also added a delay in the loop to let time pass between samples. Note: 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. The print() command does not include a newline character when printing to the serial monitor. To print on a new line, we could add it in **print(string + '\n')** or use the **println()** function.

File: wifi_a.ino

```
void loop() {

   […]

   //Printing sensor readings to serial monitor
   Serial.println("\n-");

   if(!isnan(dhtTempEvent.temperature)){
       Serial.println("Temperature: " + String(dhtTempEvent.temperature) + " degC");
   }
   else{
       Serial.println("Temperature Sensor Disconnected");
   }
   if(!isnan(dhtHumEvent.relative_humidity)){
       Serial.println("Humidity: " + String(dhtHumEvent.relative_humidity) + " %");
   }
   else{
       Serial.println("Humidity Sensor Disconnected");
   }
   if(!isnan(bmpEvent.pressure)){
       Serial.println("Pressure: " + String(bmpEvent.pressure) + " hPa");
   }
   else{
       Serial.println("Pressure Sensor Disconnected");
   }
   if(!isnan(lux)){
       Serial.println("Light Intensity: " + String(lux) + " lux");
   }
   else{
       Serial.println("Lux Sensor Disconnected");
   }

   delay(DELAY_BETWEEN_SAMPLES_MS);

}
```

9. Create a new header file by clicking the three dots and selecting the new tab in the dropdown menu. Name this file **wifi_a.h**. It will be saved to the same folder as your Arduino .ino file. The header file keeps our dependencies, preprocessor macros, global variables, and global objects.



10. Include the libraries for each sensor

File: wifi_a.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>
```

11. 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.

File: wifi_a.h

```
//HP_BH1750 Libraries
#include <hp_BH1750.h>

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

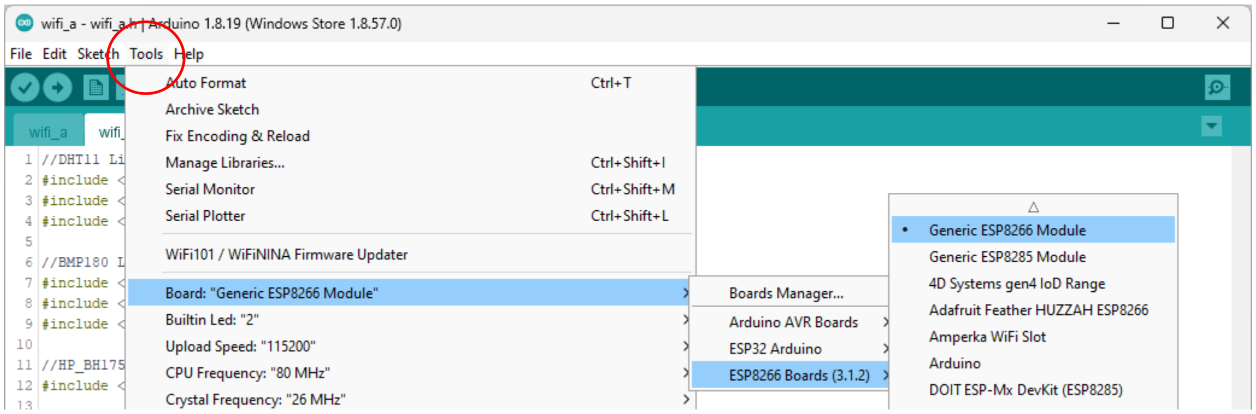
12. Instantiate the global objects for classes we use throughout the code.

File: wifi_a.h

```
#define DELAY_BETWEEN_SAMPLES_MS 5000

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

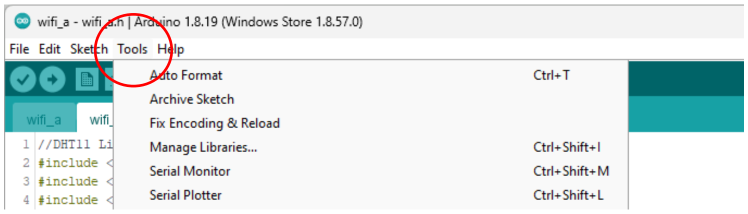
3.4.2. Upload Code to ESP8266

1. Change the board to Generic ESP8266 Module by going to *Tools > Board > ESP8266 Boards > Generic ESP8266 Module*

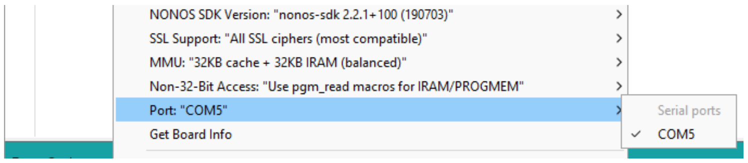


Board: "Generic ESP8266 Module"
Builtin Led: "2"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Crystal Frequency: "26 MHz"

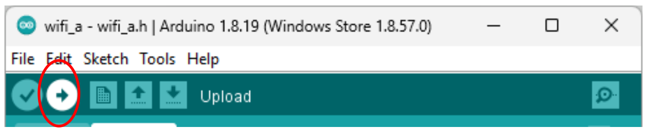
2. Plug in the ESP8266 to your computer and install drivers if prompted
3. Leave the default communication settings the same, except for the COM port. Select the COM port that your microcontroller is connected to by going to *Tools > Port > COM<#>*


wifi_a - wifi_a.h | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
wifi_a wifi_
//DHT11 Li
#include <
#include <
#include <
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L

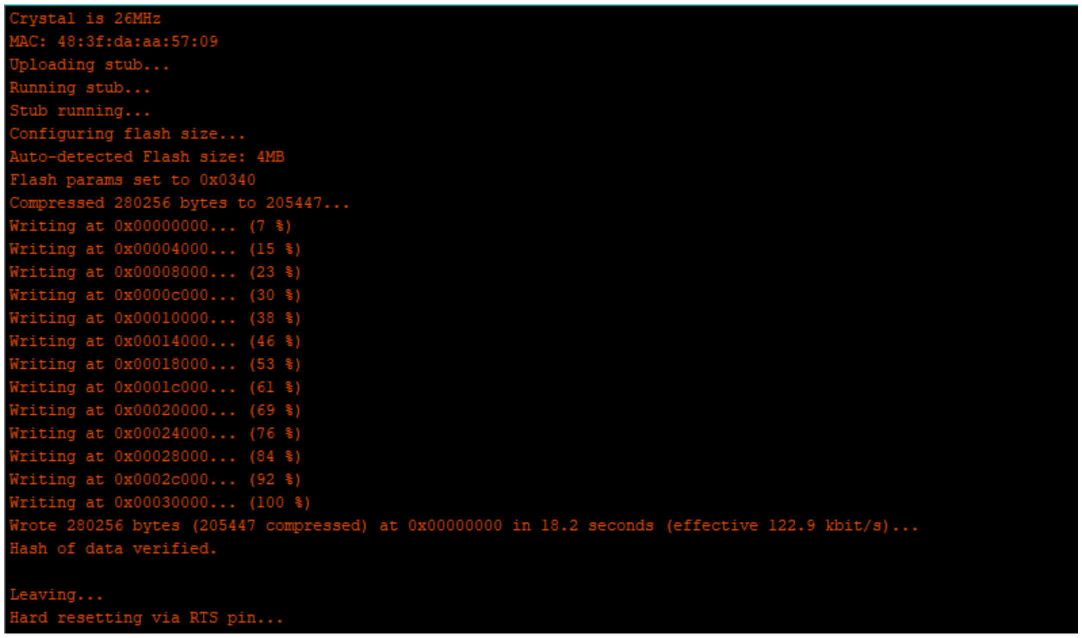
...


NONOS SDK Version: "nonos-sdk 2.2.1+100 (190703)"
SSL Support: "All SSL ciphers (most compatible)"
MMU: "32KB cache + 32KB IRAM (balanced)"
Non-32-Bit Access: "Use pgm_read macros for IRAM/PROGMEM"
Port: "COM5"

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


wifi_a - wifi_a.h | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
Upload

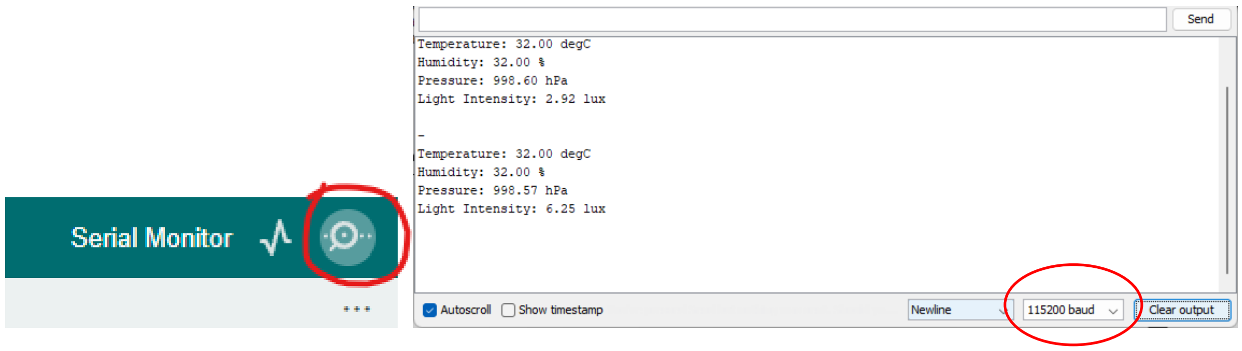
A successful upload should look like this in the Arduino IDE output (bottom left of IDE):



```
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...
```

5. After the upload is complete (NOT DURING), launch the Serial Monitor to view the microcontroller output. Top right corner (magnify glass) or *Tools > Serial Monitor*

   Note: The corresponding COM port is used while the Serial Monitor runs. You must close the Serial Monitor to release the COM port. Not closing the Serial Monitor may lock you out of this COM port for future use (you need to restart your system to reopen the COM port).



```
Temperature: 32.00 degC
Humidity: 32.00 %
Pressure: 998.60 hPa
Light Intensity: 2.92 lux

-
Temperature: 32.00 degC
Humidity: 32.00 %
Pressure: 998.57 hPa
Light Intensity: 6.25 lux
```

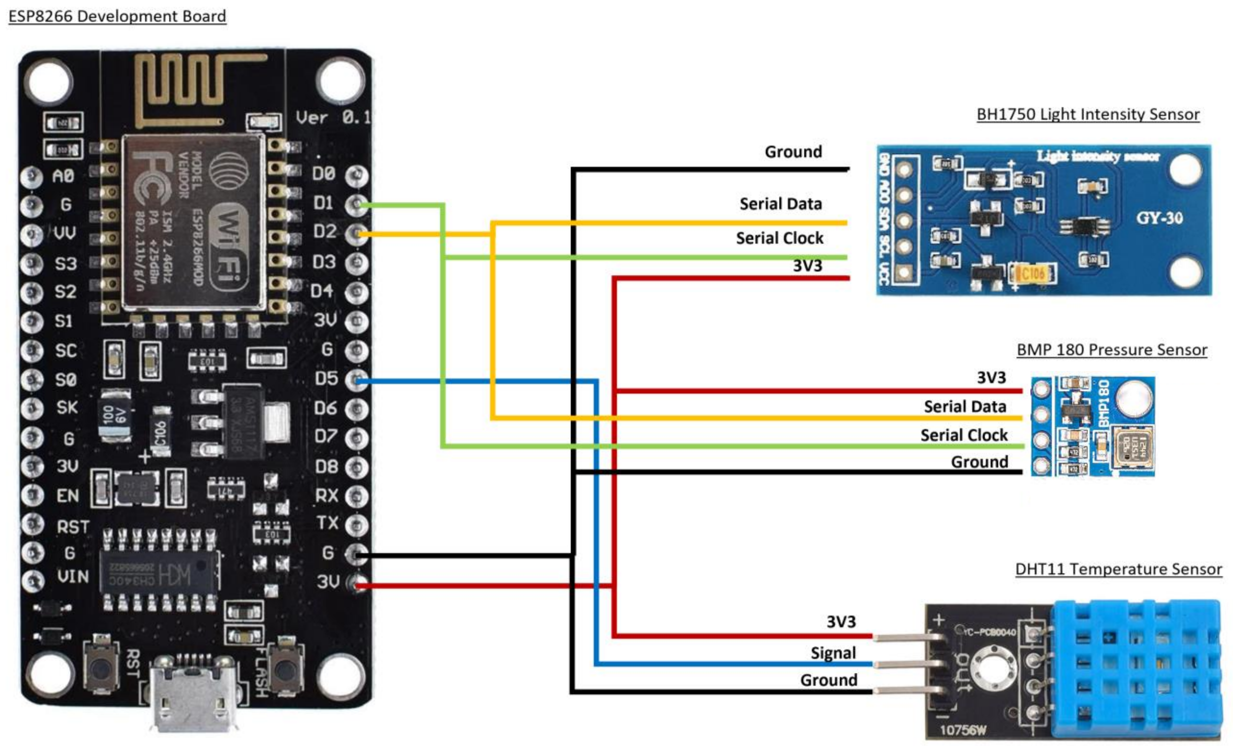
6. Set the Baud Rate to match the baud rate in your setup function in wifi_a.ino. Note: If the Baud Rate in the serial monitor does not match your code, you cannot receive messages correctly.
7. Close your serial monitor and unplug the microcontroller

3.4.3. Wiring Setup for ESP8266 and Sensors

Remember when wiring up your board and sensors:

- **Powering up** should be the **last step** in your setup process
- **Common Ground**: Ensure all components share a common ground to avoid ground loops and potential damage to the sensors or the ESP8266
- **Voltage Regulation:** Ensure that each component receives the correct voltage.
- **Avoid Loose Wires:** Keep wires organized and avoid loose connections that could cause short circuits
- **Incremental Testing**: Connect and test one sensor at a time to isolate any issues and ensure each component works correctly

1. Wire up the sensors using the diagram below



2. Plug the microcontroller back into your computer
3. Launch the Serial monitor to view the microcontroller output

Milestone 1: Instructor/TA to verify sensor output for all three sensors appearing on the serial monitor

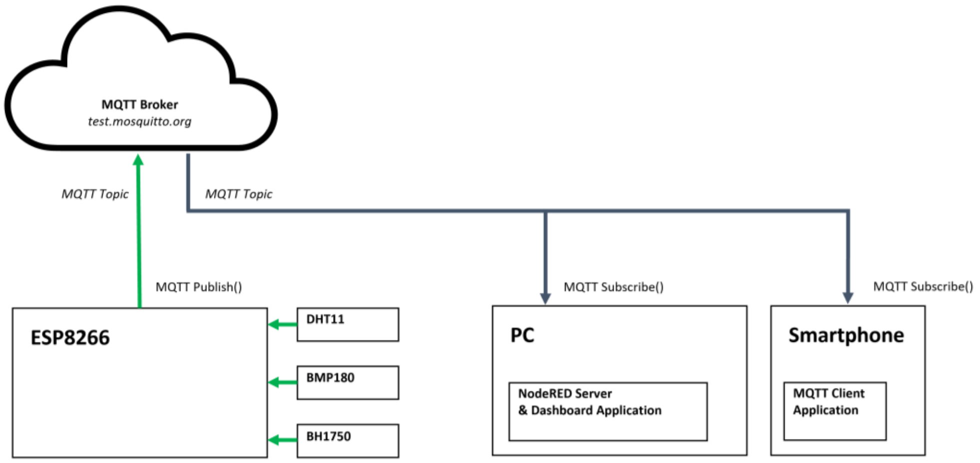
Lab Report Q1- Picture of Circuit and Screenshot of Arduino IDE Serial Monitor.

Note: Team member names and student numbers *must* appear embedded in the pictures written on paper or in a notepad alongside the serial output, not added post-image capture. -5 marks if team member names/student numbers are not shown.

Close your serial monitor and unplug your microcontroller, **but keep everything wired up!**

3.5. Hello World! Publishing to an MQTT Broker [25 marks]

The ESP8266 has a built-in WiFi transceiver, which allows it to connect to the Internet. You will connect to the lab's Wi-Fi network (or your PC or mobile device's hotspot) and publish the sensor data to an MQTT broker, which will broadcast it to subscribed applications on your PC or smartphone. Please ensure that the ESP8266, hotspot (if applicable), and subscribed applications are running on separate devices.

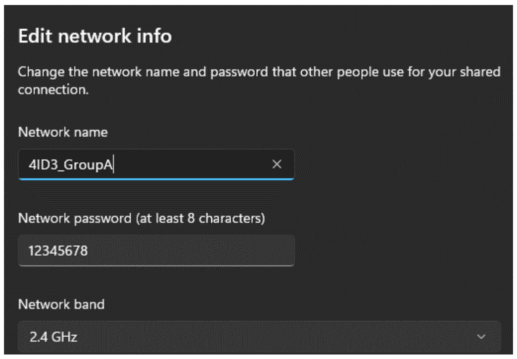


When adding the broker IP address, you have two options:

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

   test.mosquitto.org
   Port is 1883

2. Use a local broker on the same network
   - Open Windows Mobile Hotspot
   - Set your login credentials (Note: Replace GroupA with Group#)
   - Toggle it on and turn off power saving



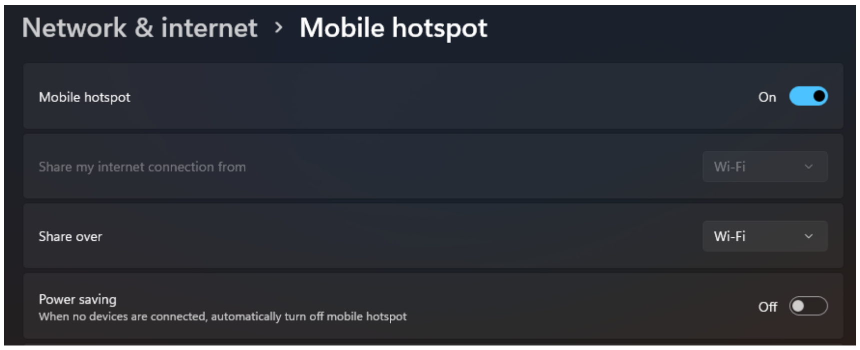
Edit network info

Change the network name and password that other people use for your shared connection.

Network name: 4ID3_GroupA

Network password (at least 8 characters): 12345678

Network band: 2.4 GHz



Network & internet › Mobile hotspot

Mobile hotspot: On

Share my internet connection from: Wi-Fi

Share over: Wi-Fi

Power saving: Off
When no devices are connected, automatically turn off mobile hotspot

3.5.1. Update ESP8266 Program for IoT

1. Duplicate wifi_a in your course repository and call the new folder and files wifi_b
2. Open wifi_b.ino in a new Arduino IDE tab. Wifi_b.h will also be available
3. Modify your header file to include global variables for the WiFi driver and the MQTT libraries, and instantiate the MQTT Client object. Select the header file tab and include the following communication libraries above the existing code

File: wifi_b.h

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

4. Below the existing code, include the following global variables and update the SSID and pass to include the Wi-Fi (or hotspot) ssid (user name) and pass (password)

File: wifi_b.h

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

//Global Variables
char* ssid = "GroupA";
char* pass = "12345678";
const char* brokerAddress = "test.mosquitto.org";
uint16_t addressPort = 1883;
```

5. Instantiate the WifiClient object and PubSubClient object below.

File: wifi_b.h

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

6. Change the DELAY_BETWEEN_SAMPLES_MS in wifi_b.h to 20,000 ms (20 seconds)
7. Update the include statement in the wifi_b .ino implementation file to include the correct header file. #include “wifi_b.h”
8. Modify the setup() function to initialize the WiFi driver and MQTT client.

File: wifi_b.ino

```
#include "wifi_b.h"

void setup() {

  [...]

  //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, addressPort);

}
```

9. If the microcontroller loses connection to the MQTT server, we want to have a callback function that attempts to reconnect. Below the setup() function, create a reconnect function. This function will be called from the void loop().

File: wifi_b.ino

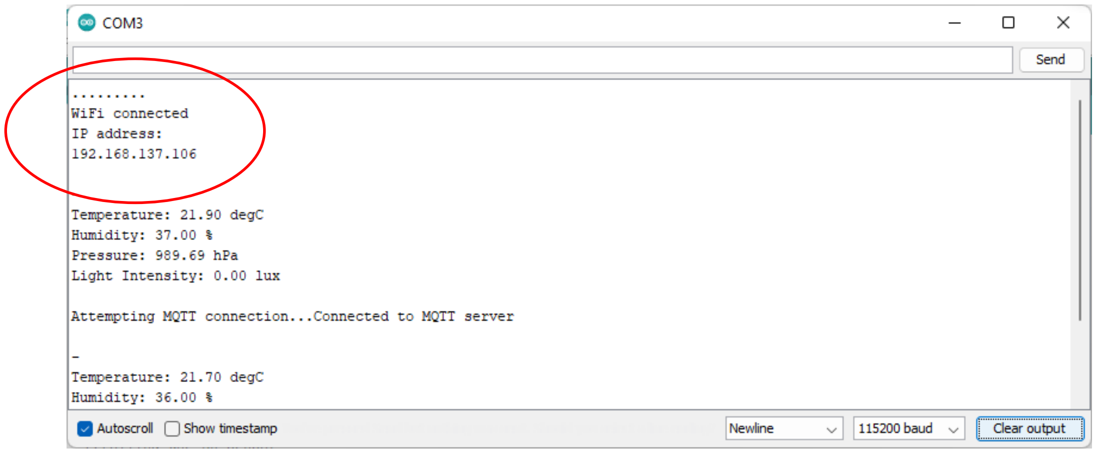
```
void reconnect() {

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

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

10. In void loop(), call the reconnect function if the connection is lost. Once connected, publish each sensor data point to its associated topic.
11. Include the publish command to transmit data to the MQTT Broker for each sensor. Note: You must update GroupA to your Group#.

File: wifi_b.ino

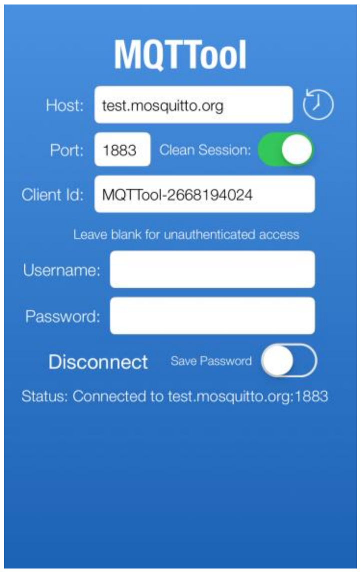
```
void loop() {

   [...]

  if(!client.loop())
    client.connect("ESP8266Client");

  //Publish the sensor data to the associated topics
  client.publish("4ID3_GroupA/temperature", String(dhtTempEvent.temperature).c_str());
  delay(100);
  client.publish("4ID3_GroupA/humidity", String(dhtHumEvent.relative_humidity).c_str());
  delay(100);
  client.publish("4ID3_GroupA/pressure", String(bmpEvent.pressure).c_str());
  delay(100);
  client.publish("4ID3_GroupA/light", String(lux).c_str());
  Serial.println("Published data.");

   delay(DELAY_BETWEEN_SAMPLES_MS);
}
```

12. Ensure the code compiles and uploads it to the microcontroller



```
.........
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 %
```

3.5.2. Connect to Broker Using MQTTool



Host: test.mosquitto.org

Port: 1883 Clean Session:

Client Id: MQTTool-2668194024

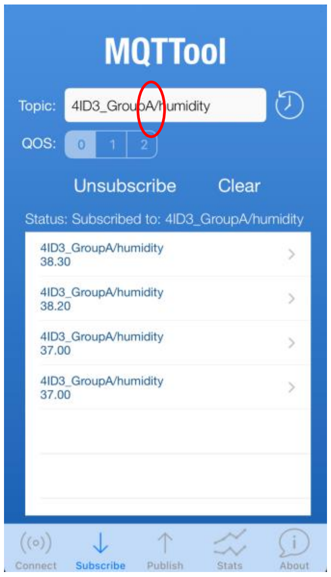
Leave blank for unauthenticated access

Username:

Password:

Disconnect Save Password

Status: Connected to test.mosquitto.org:1883



Topic: 4ID3_GroupA/humidity

QOS: 0 1 2

Status: Subscribed to: 4ID3_GroupA/humidity

```
4ID3_GroupA/humidity
38.30
4ID3_GroupA/humidity
38.20
4ID3_GroupA/humidity
37.00
4ID3_GroupA/humidity
37.00
```

Steps 1 (left) and 2 (right)

1. On the MQTTool connect tab (bottom left), complete the host and port information for the public mosquito broker and hit connect (will show disconnect once after successful connection)
2. In the subscribe tab (second from left), subscribe to the topic of choice. What topics can you subscribe to? Where do you set that information?
    - Hint: It includes your group number
3. Wait for values to populate. How long is the delay between samples? Why is there a delay between samples?
4. You can also try it with other sensor readings to ensure they are working correctly

Milestone 2: Instructor/TA to verify sensor output for all three sensors appearing on MQTool

Lab Report Q2 – Screenshot of Serial Monitor and MQTTool Serial Output

Note: Team member names and student numbers *must* appear in the pictures written on paper, not added post-image capture. -5 marks if team member names/student numbers are not shown.

3.5.3. Connect to MQTT Broker from a Variety of Client Applications

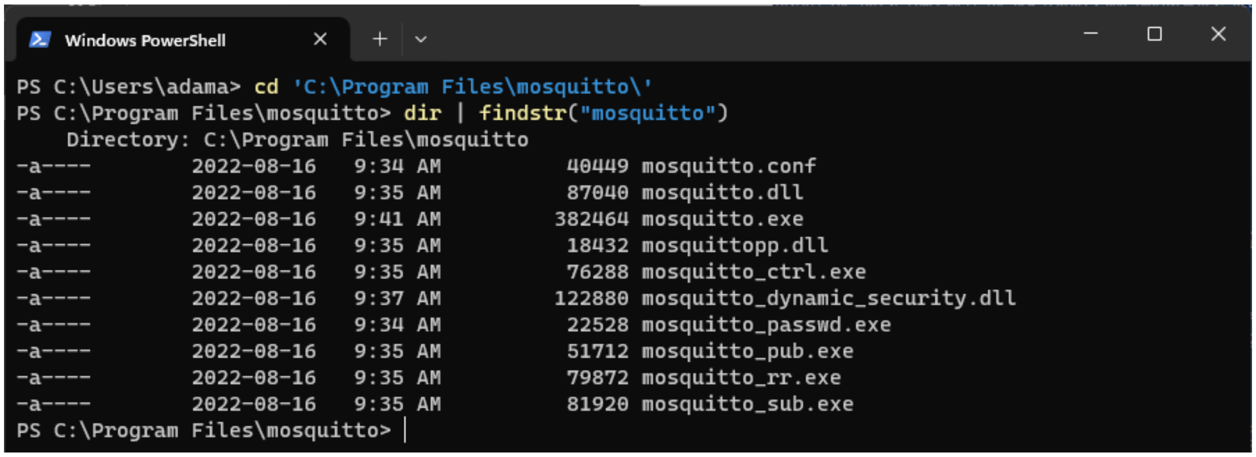
To verify and demonstrate the communication of sensor data over the Internet, we will connect to the public MQTT broker from various client applications. Please try each method on a different device.

Using a Desktop Terminal Application

1. Open a Powershell terminal. *Windows Terminal > Powershell or Windows Powershell*
2. Navigate to the installation directory of the mosquitto MQTT client.

   cd 'C:\Program Files\mosquitto\"

   dir | findstr("mosquitto")



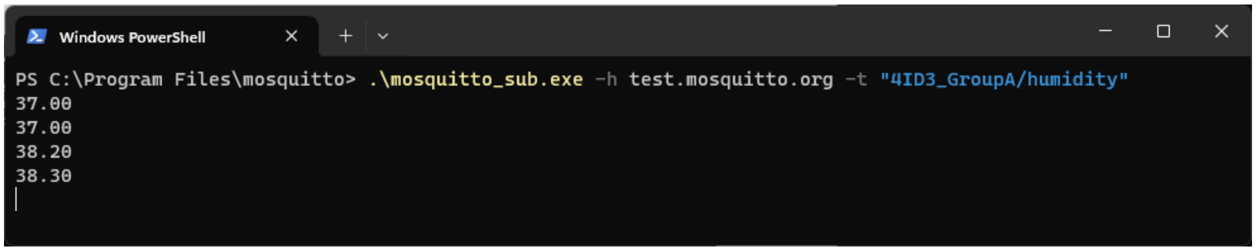
```
PS C:\Users\adama> cd 'C:\Program Files\mosquitto\'
PS C:\Program Files\mosquitto> dir | findstr("mosquitto")
    Directory: C:\Program Files\mosquitto
-a----         2022-08-16   9:34 AM          40449 mosquitto.conf
-a----         2022-08-16   9:35 AM          87040 mosquitto.dll
-a----         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
PS C:\Program Files\mosquitto> |
```

3. Launch the subscribe applications with the following flags:

   .\mosquitto_sub.exe -h test.mosquitto.org -t "4ID3_GroupA/humidity"

   -h - MQTT broker IP address

   -t - the topic that you wish to connect to

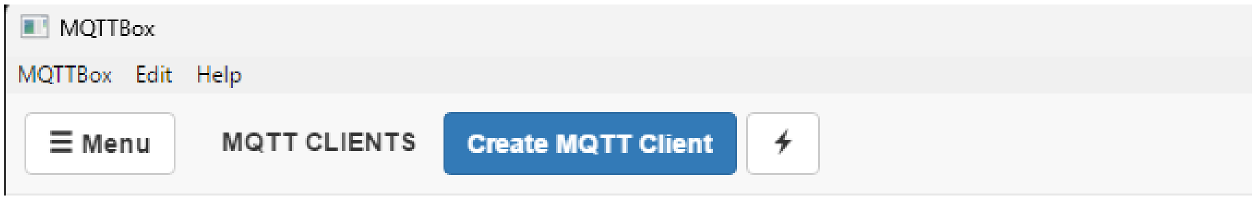


```
PS C:\Program Files\mosquitto> .\mosquitto_sub.exe -h test.mosquitto.org -t "4ID3_GroupA/humidity"
37.00
37.00
38.20
38.30
|
```

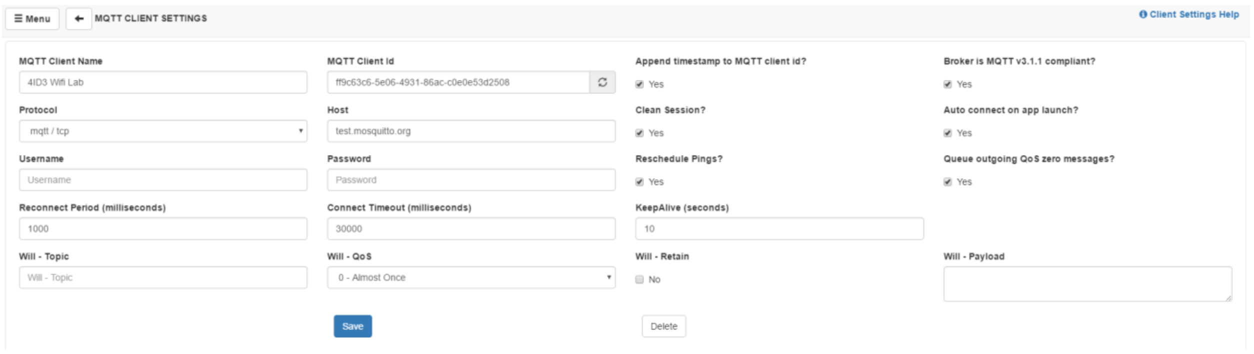
After a short delay, you will see the values printed on the terminal window.

Using MQTTBox

1. Open MQTTBox and select Create New MQTT Client.

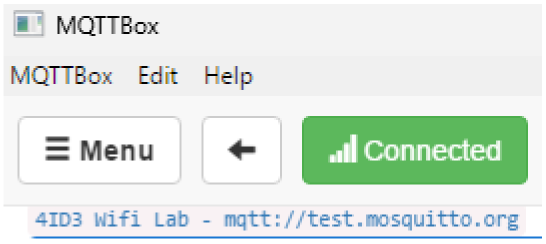


2. Configure the following connection details:



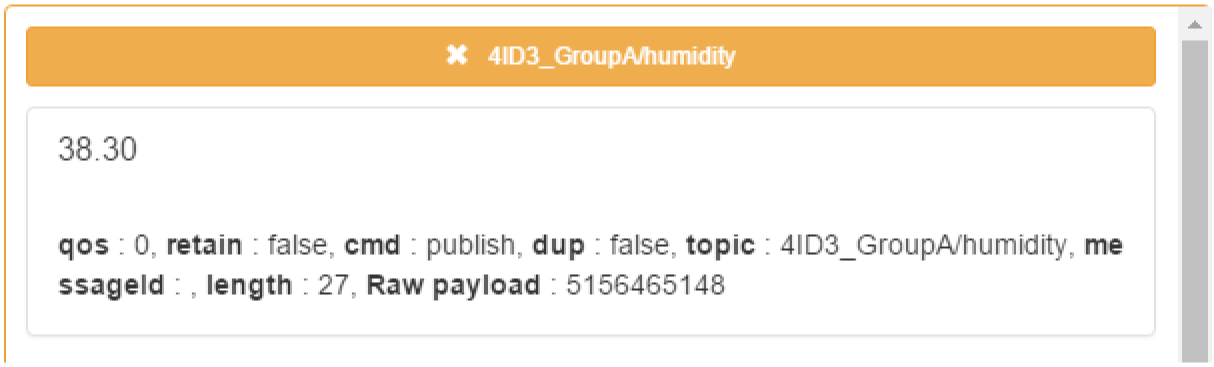
MQTT Client Name: 4ID3 Wifi Lab	MQTT Client Id: ff9c63c6-5e06-4931-86ac-c0e0e53d2508	Append timestamp to MQTT client id? ✓ Yes	Broker is MQTT v3.1.1 compliant? ✓ Yes
Protocol: mqtt / tcp	Host: test.mosquitto.org	Clean Session? ✓ Yes	Auto connect on app launch? ✓ Yes
Username: Username	Password: Password	Reschedule Pings? ✓ Yes	Queue outgoing QoS zero messages? ✓ Yes
Reconnect Period (milliseconds): 1000	Connect Timeout (milliseconds): 30000	KeepAlive (seconds): 10	
Will - Topic: Will - Topic	Will - QoS: 0 - Almost Once	Will - Retain: ☐ No	Will - Payload:

3. Press Save and ensure that the toolbar says Connected.



4ID3 Wifi Lab - mqtt://test.mosquitto.org

4. Subscribe to your desired topic and watch as messages populate.



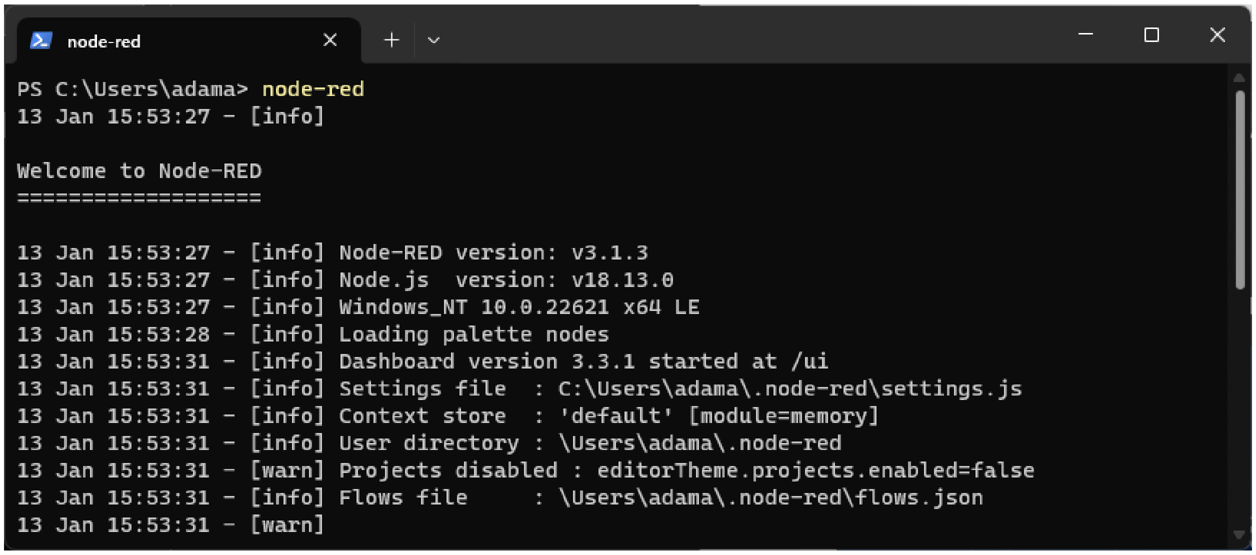
```
38.30

qos : 0, retain : false, cmd : publish, dup : false, topic : 4ID3_GroupA/humidity, me
ssageId : , length : 27, Raw payload : 5156465148
```

3.6. Visualization Using Node-RED [25 marks]

We will use Node-RED to create a live data dashboard.

1. To start NodeRED, open Windows Terminal > Powershell and type the command:

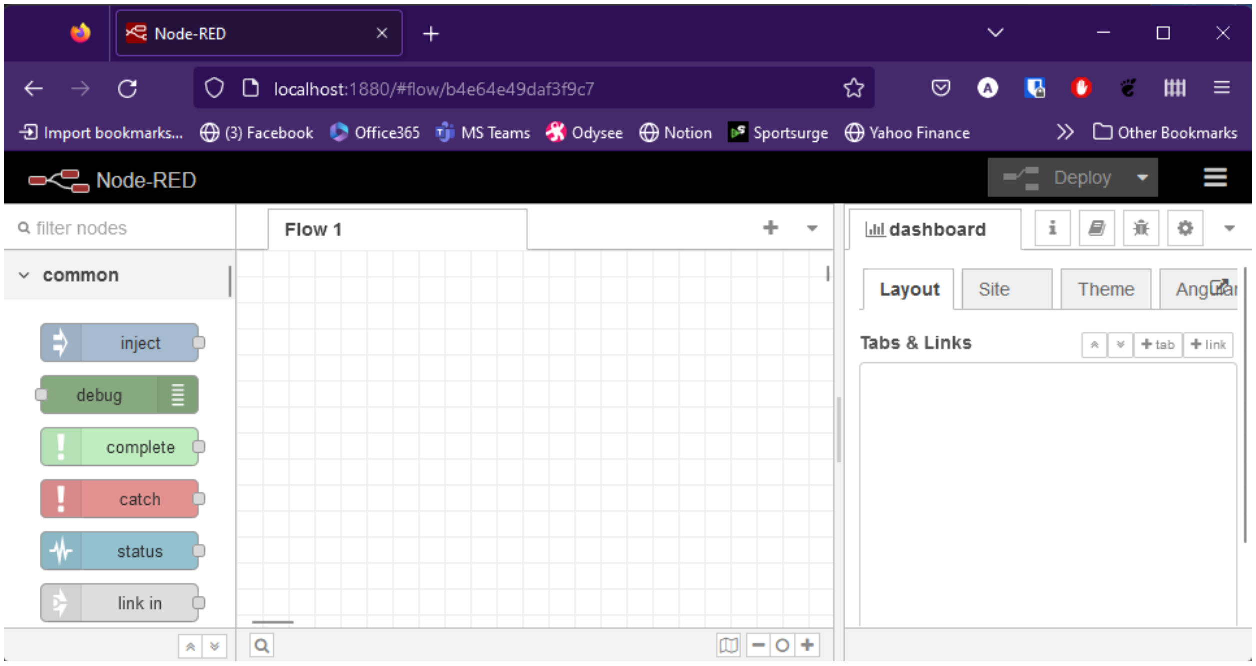
   node-red



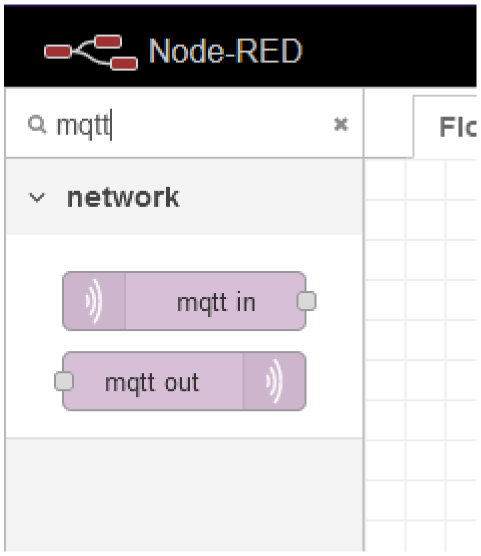
```
PS C:\Users\adama> node-red
13 Jan 15:53:27 - [info]

Welcome to Node-RED
===================

13 Jan 15:53:27 - [info] Node-RED version: v3.1.3
13 Jan 15:53:27 - [info] Node.js  version: v18.13.0
13 Jan 15:53:27 - [info] Windows_NT 10.0.22621 x64 LE
13 Jan 15:53:28 - [info] Loading palette nodes
13 Jan 15:53:31 - [info] Dashboard version 3.3.1 started at /ui
13 Jan 15:53:31 - [info] Settings file   : C:\Users\adama\.node-red\settings.js
13 Jan 15:53:31 - [info] Context store   : 'default' [module=memory]
13 Jan 15:53:31 - [info] User directory  : \Users\adama\.node-red
13 Jan 15:53:31 - [warn] Projects disabled : editorTheme.projects.enabled=false
13 Jan 15:53:31 - [info] Flows file      : \Users\adama\.node-red\flows.json
13 Jan 15:53:31 - [warn]
```

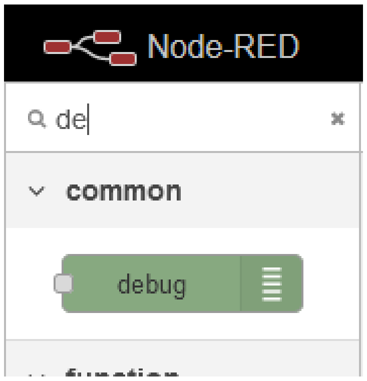
2. Navigate to the URL presented in a web browser:

Web Browser > *127.0.0.1:1880*

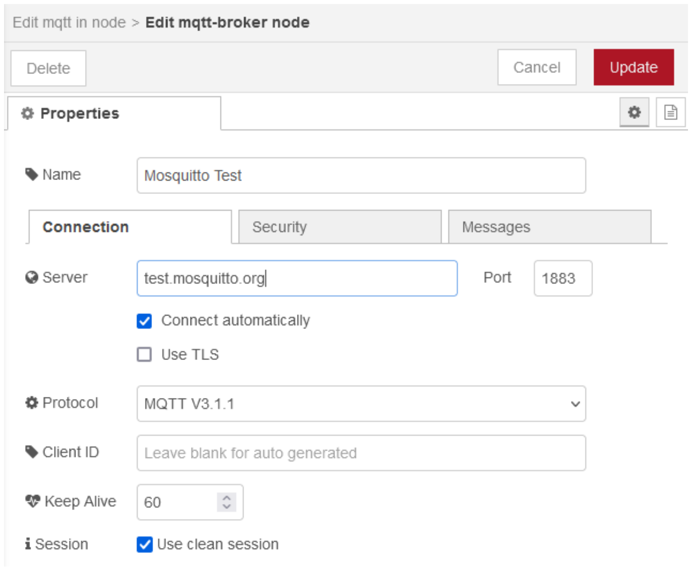


3. Filter nodes to find the MQTT in the node and the debug node. Drag them into your flow diagram.



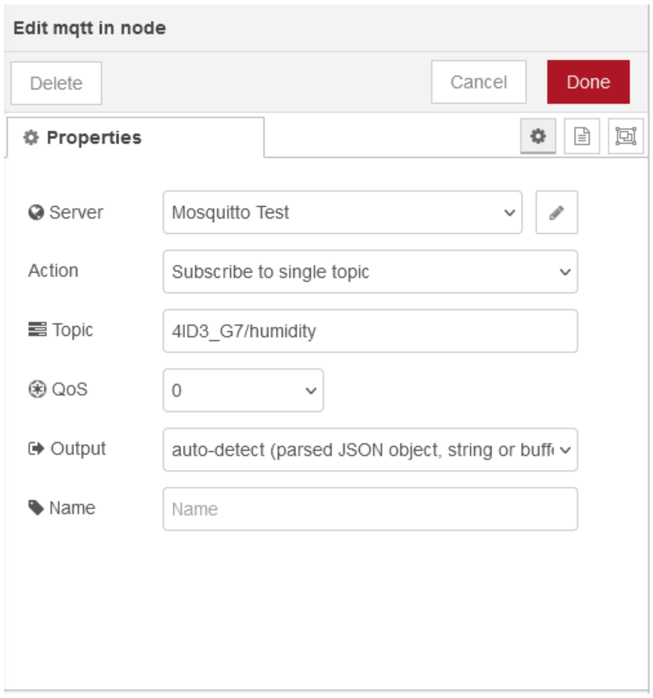


4. Click on the MQTT in the node to begin editing its configuration.
5. Fill in the information for the public broker and Press Update.



Name: Mosquitto Test
Server: test.mosquitto.org
Port: 1883
☑ Connect automatically
☐ Use TLS
Protocol: MQTT V3.1.1
Client ID: Leave blank for auto generated
Keep Alive: 60
Session: ☑ Use clean session

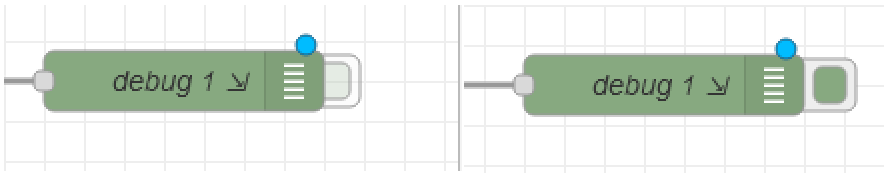
6. Under Properties, fill in the Topic field.



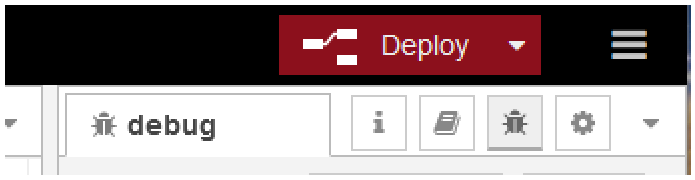
Edit mqtt in node

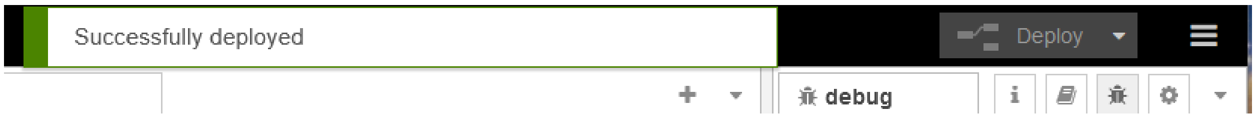
Server: Mosquitto Test
Action: Subscribe to single topic
Topic: 4ID3_G7/humidity
QoS: 0
Output: auto-detect (parsed JSON object, string or buff
Name: Name

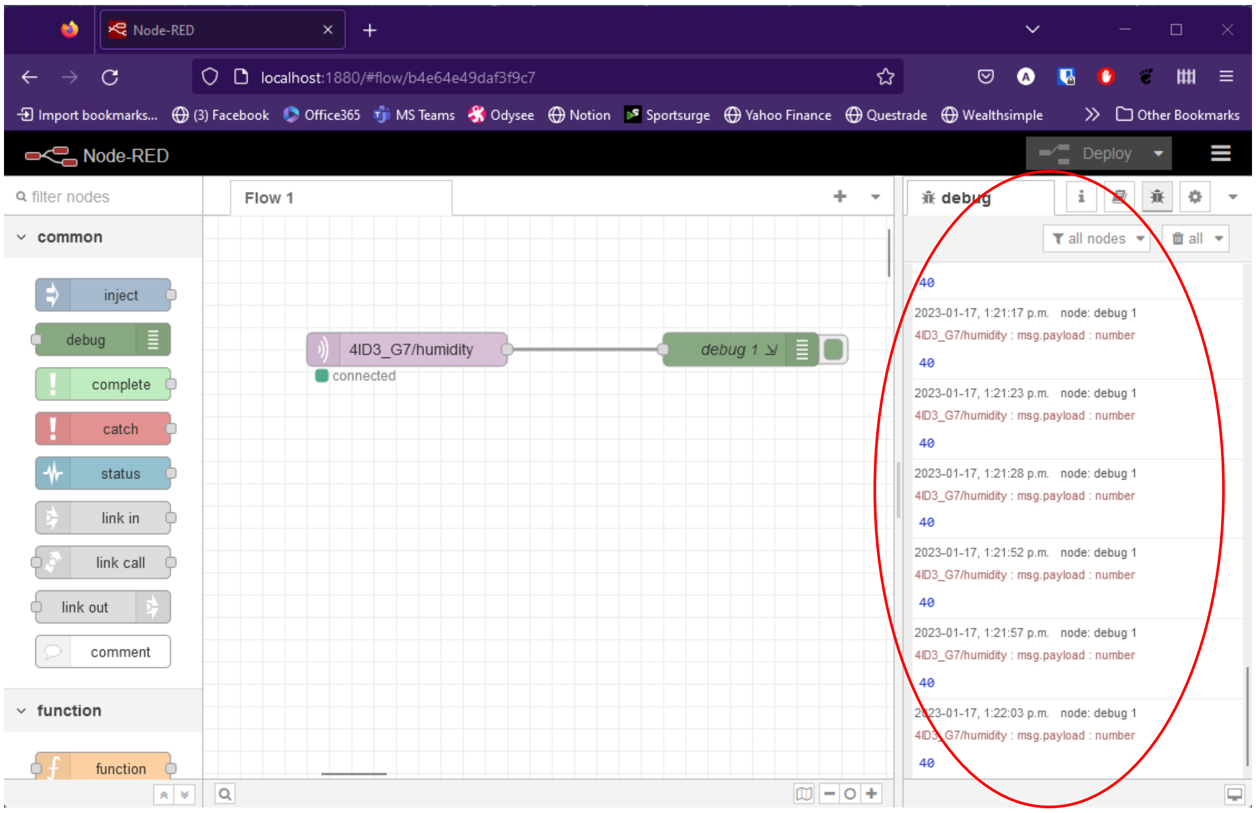
7. Enable the debug node by pressing the green box.



8. Press Deploy and watch the Debug Panel populate with sensor values.

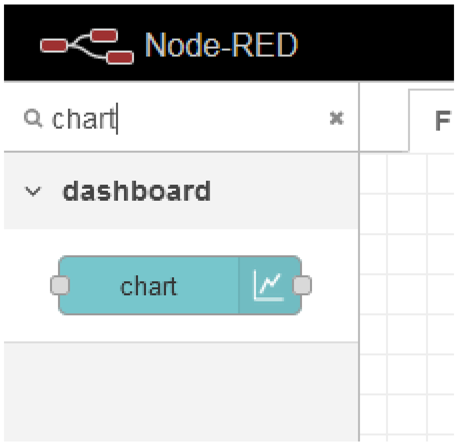



Successfully deployed

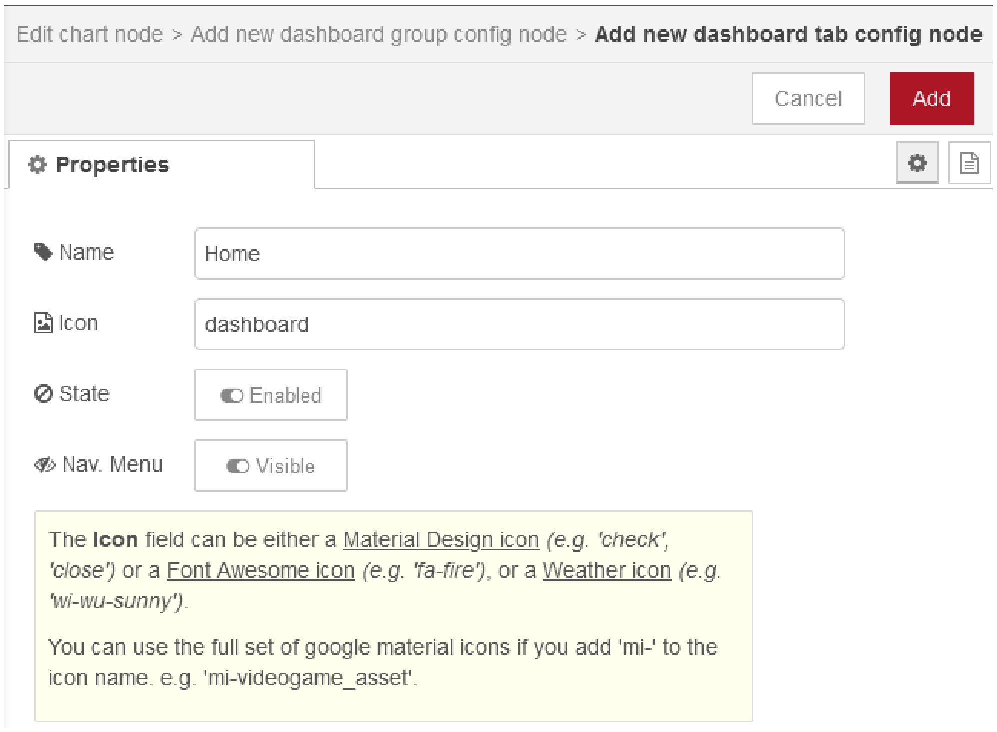


```
40
2023-01-17, 1:21:17 p.m.   node: debug 1
4ID3_G7/humidity : msg.payload : number
40
2023-01-17, 1:21:23 p.m.   node: debug 1
4ID3_G7/humidity : msg.payload : number
40
2023-01-17, 1:21:28 p.m.   node: debug 1
4ID3_G7/humidity : msg.payload : number
40
2023-01-17, 1:21:52 p.m.   node: debug 1
4ID3_G7/humidity : 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
```

9. Search for the chart in the filter nodes field and drag it into your flow diagram.



10. When you click on the node, you must create a new Dashboard Tab. Use the default name and press Add.



Edit chart node > Add new dashboard group config node > **Add new dashboard tab config node**

Name: Home

Icon: dashboard

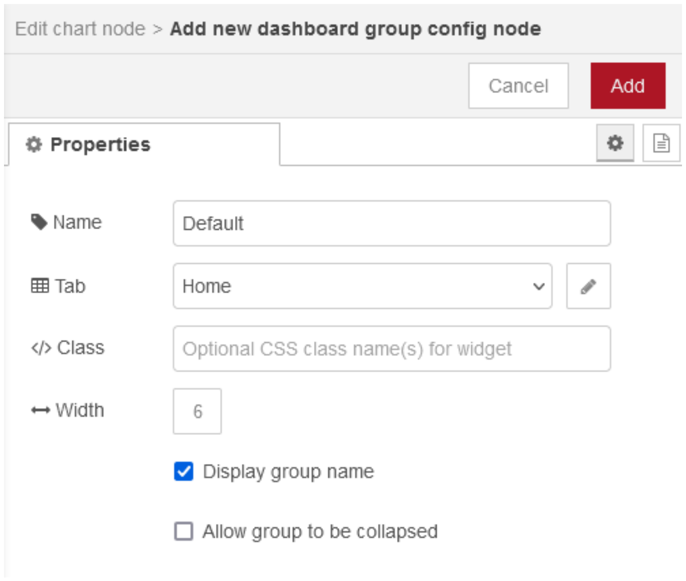
State: Enabled

Nav. Menu: Visible

The **Icon** field can be either a Material Design icon *(e.g. 'check', 'close')* or a Font Awesome icon *(e.g. 'fa-fire')*, or a Weather icon *(e.g. 'wi-wu-sunny')*.

You can use the full set of google material icons if you add 'mi-' to the icon name. e.g. 'mi-videogame_asset'.

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



Edit chart node > **Add new dashboard group config node**

Name: Default

Tab: Home

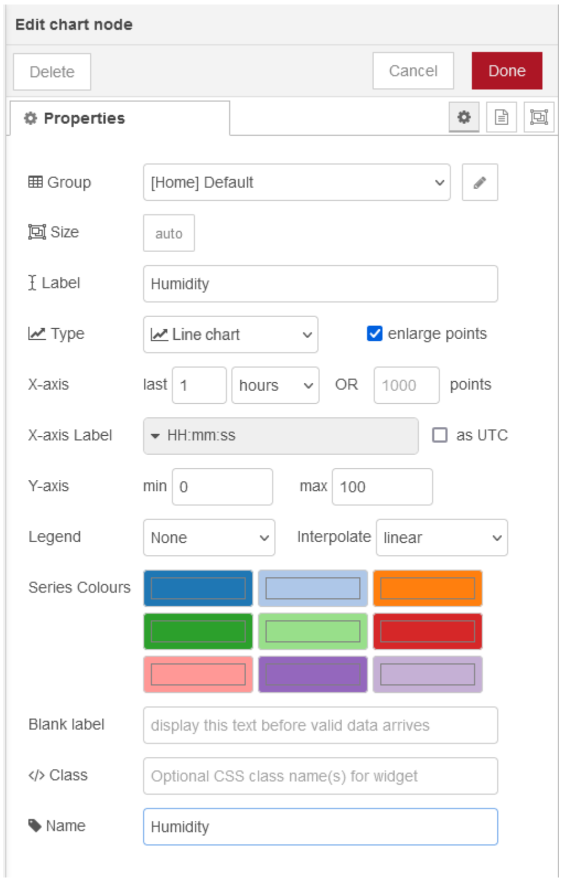
Class: Optional CSS class name(s) for widget

Width: 6

[✓] Display group name

[ ] Allow group to be collapsed

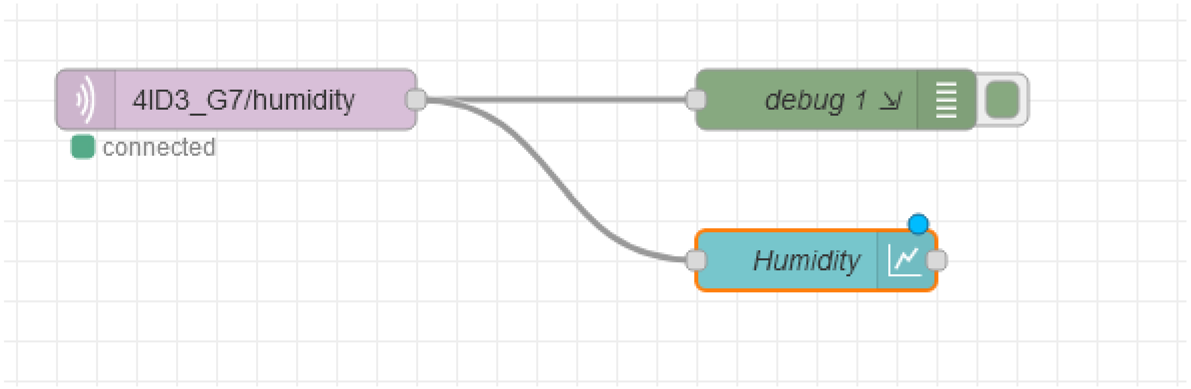
12. Edit the chart node to visualize your data nicely. Press Done when complete.

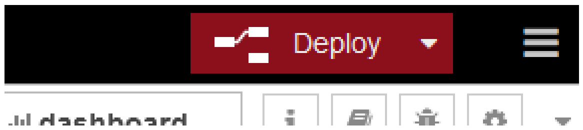


Edit chart node

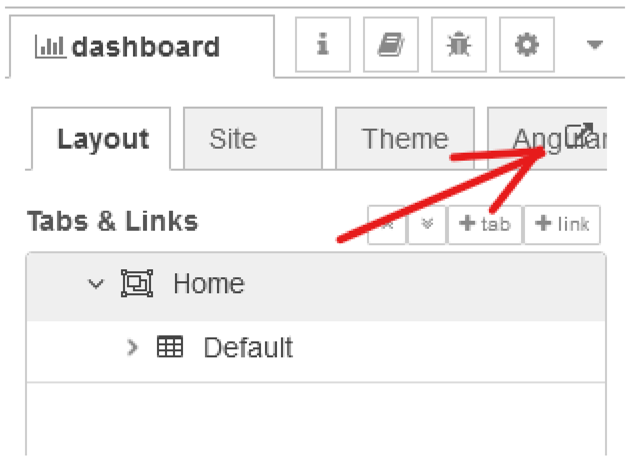
Group: [Home] Default
Size: auto
Label: Humidity
Type: Line chart [✓ enlarge points]
X-axis: last 1 hours OR 1000 points
X-axis Label: HH:mm:ss ☐ as UTC
Y-axis: min 0 max 100
Legend: None Interpolate: linear
Series Colours
Blank label: display this text before valid data arrives
Class: Optional CSS class name(s) for widget
Name: Humidity

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

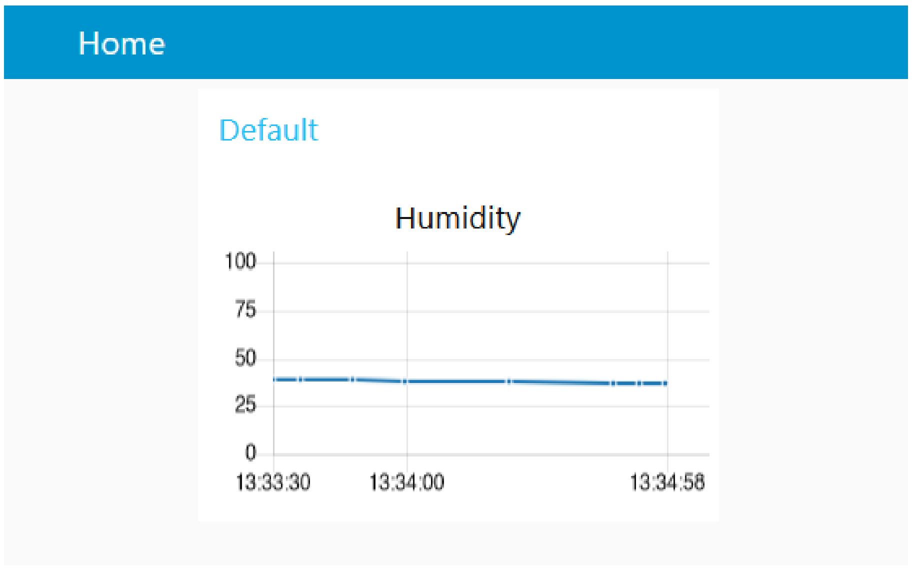




14. To view the dashboard, either append ui/ to your URL (http://localhost:1880/ui) or press the open dashboard icon in the top right corner of the dashboard panel.

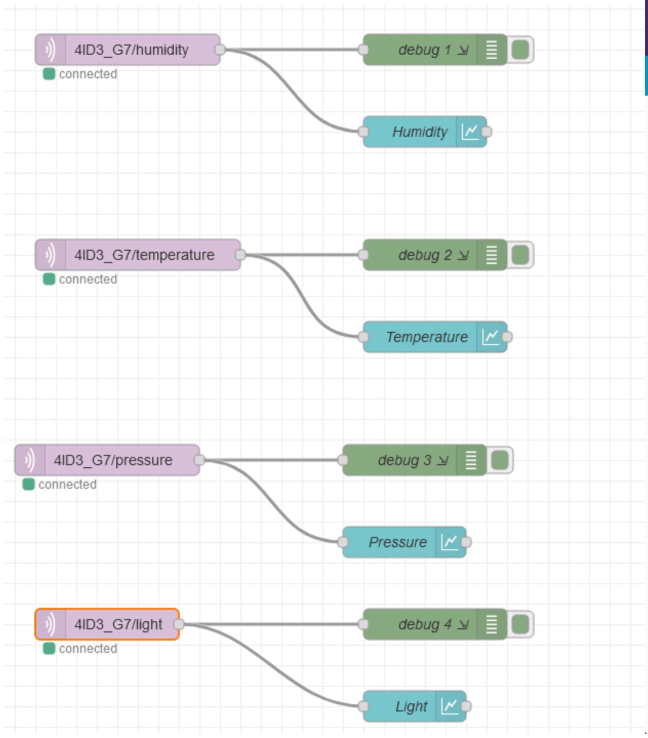


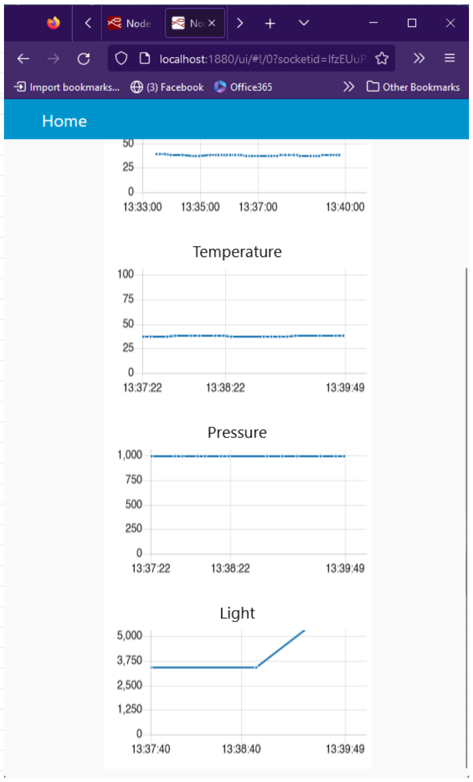
15. Wait for data to populate.



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

16. Now that you have seen how to visualize one topic, visualize the rest. The resultant flow should look like this:





Explain your observations of the Node-Red user interface as you interact with the sensors. Is the change instantaneous? What changes?

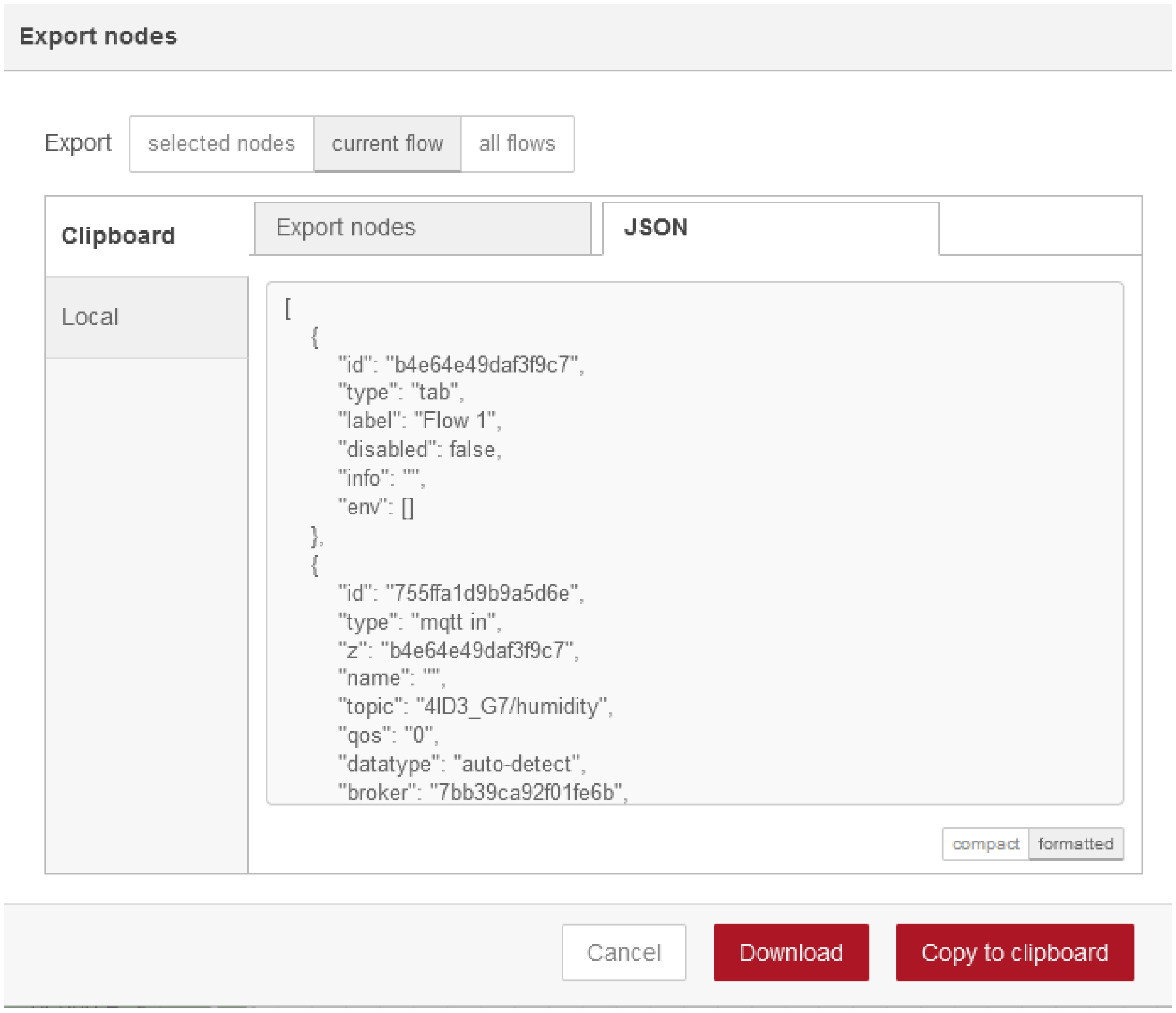
Milestone 3: Instructor/TA to verify sensor output for all three sensors appearing on the dashboard

Lab Report Q3 – Screenshot of Node-Red Graphs

Note: Team member names and student numbers *must* appear embedded in the screenshot, not added to the post-image capture. For example, you can include student names and numbers in a notepad alongside the charts. -5 marks if team member names/student numbers are not shown.

17. Export your NodeRED Flow as JSON

    a. Click on the hamburger menu and select export.
    b. Select current flow.
    c. Select JSON.
    d. Press Download
    e. Copy the flows.json file from your Downloads/ folder to the Wifi/ folder in the local repo



```
[
    {
        "id": "b4e64e49daf3f9c7",
        "type": "tab",
        "label": "Flow 1",
        "disabled": false,
        "info": "",
        "env": []
    },
    {
        "id": "755ffa1d9b9a5d6e",
        "type": "mqtt in",
        "z": "b4e64e49daf3f9c7",
        "name": "",
        "topic": "4ID3_G7/humidity",
        "qos": "0",
        "datatype": "auto-detect",
        "broker": "7bb39ca92f01fe6b",
```

18. View the status of your local Git repository to verify that there aren't any uncommitted changes. Push changes from your local repository to GitHub to ensure both are up-to-date.

    Lab Report Q4 – Git Repository of changes pushed to GitHub

Note: Team member names and student numbers *must* appear embedded in the screenshot, not added to the post-image capture. For example, you can include student names and numbers in a notepad alongside the terminal output. -5 marks if team member names/student numbers are not shown.

4. Lab Report [15 marks]

In addition to the cover page, include the text of the original question followed by your clear and concise answer. Answers to design or calculation questions will receive a zero if they are not accompanied by supporting work.

Lab reports are due one week after the lab.

Q1 – Milestone 1: Picture of Circuit and Screenshot of Arduino IDE Serial Monitor

(Suggested: Paragraph or screenshot, 2 points)

Q2 – Milestone 2: Screenshot of Serial Monitor and MQTTool Serial Output

(Suggested: Paragraph or screenshot, 1 point)

Q3 – Milestone 3: Screenshot of Node-Red Graphs

(Suggested: Paragraph or screenshot, 2 points)

Q4 – View the status of your local Git repository to verify that there aren't any uncommitted changes. Push changes from your local repository to GitHub to ensure both are up-to-date.

(Suggested: Paragraph or screenshot, 2 points)

Q5 – Using software like PowerPoint or Draw.IO, create a diagram representing the nodes in this IoT network and label the data being exchanged between nodes.

(Suggested: Diagram, 4 points)

Q6 – If the microcontroller loses battery, will the Node-RED dashboard still be connected to the MQTT server? Explain your answer.

(Suggested: 3 sentences, 2 points)

Q7—Using a remote MQTT broker on the cloud allows client Node-RED dashboards to visualize sensor data without needing to be in the exact geographical location of the network. Explain two security vulnerabilities with the network established in this lab and how these issues could be mitigated.

(Suggested: 4 sentences, 2 points)

5. Summary of Milestones and Evaluation Rubric

Instructor/TA must visually verify and record your successful completion of:

1. Milestone 1
2. Milestone 2
3. Milestone 3

Criteria	**Mark**
Successfully demonstrate and explain to TA an entirely correct working milestone	100%
Demonstrate and explain a coherent attempt at a milestone but an incorrect result	50%
No demonstration, cannot explain, or non-coherent attempt/demonstration	0%

6. Grading Summary Table

Component	**Weight**
Prelab	10%
Milestone 1	25%
Milestone 2	25%
Milestone 3	25%
Lab Report	15%
Total	100%

END