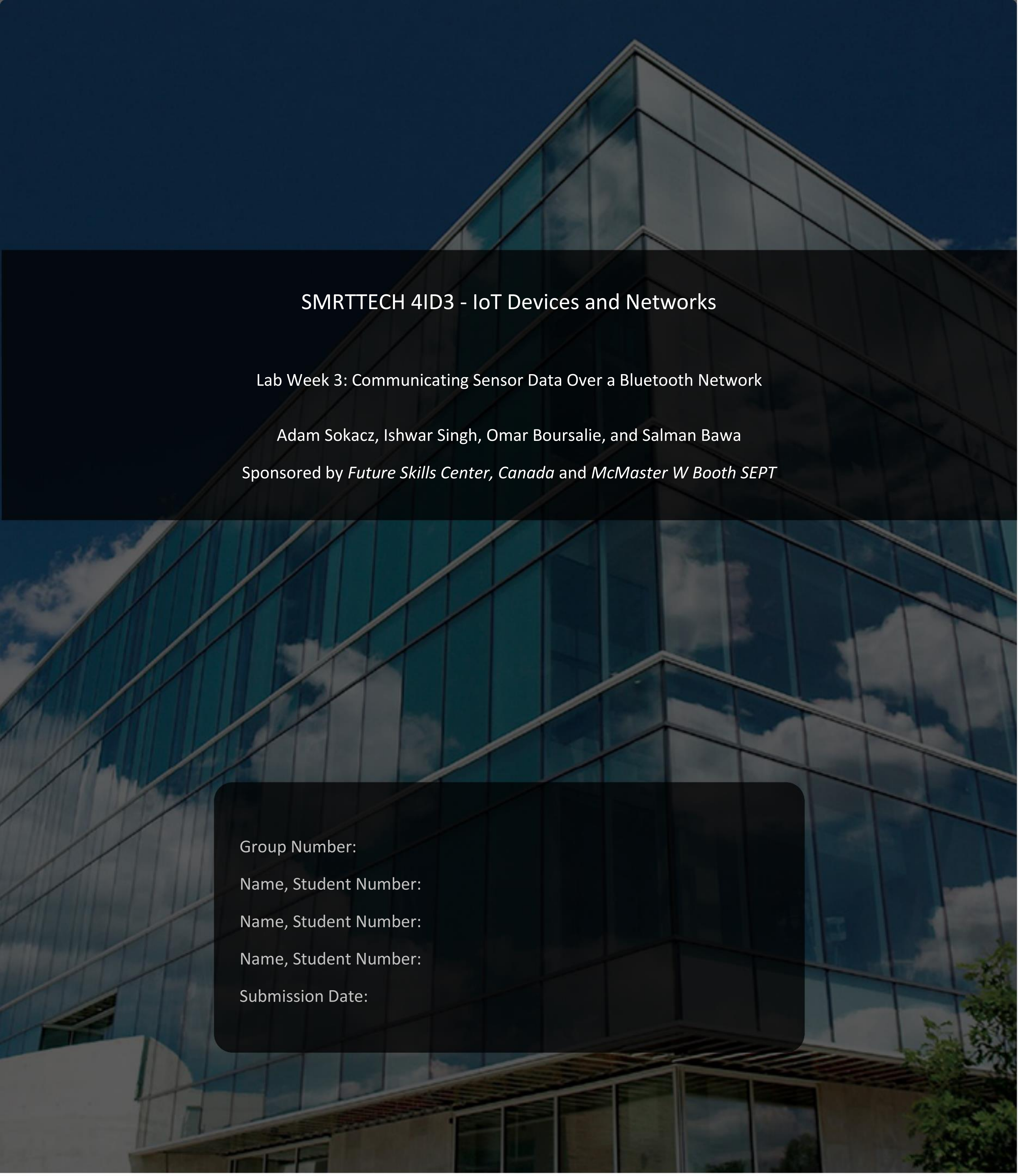


SMRTTECH 4ID3 - IoT Devices and Networks

Lab Week 3: Communicating Sensor Data Over a Bluetooth Network

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:

Objective

A practice-based understanding on the "things" of the Internet of Things, and the "Internet" of the Internet of Things, including temperature, pressure, humidity, and light intensity sensors, connected to a microcontroller (with Publication-Subscription client), communicating over Bluetooth to an MQTT server on the cloud, corresponding with Node-Red application, showcasing outcome in a Node-Red user interface.

Learning Outcomes

By the end of this lab, students will:

1) Students lean, line-by-line, explanation of a generic code, enabling them to edit code as needed for lab and/or project work
2) Students learn about Bluetooth as an Access Technology and appreciate the criteria for choosing Bluetooth 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 piconet messaging pattern for Bluetooth. What role does the primary device play? What role do the secondary devices play? How many secondary devices can be actively coordinated by a single primary device? Include a sketch of the piconet topology in your explanation.

(Suggested: Short paragraph, 4 points)

Q2— In this lab, we will encode data as a JSON string to make it easy to parse using Python

- What is JSON?
- What is a Python Dictionary data structure?
- What does the json.loads function do?

(Suggested: 3 sentences, 3 points)

Q3 – What layer of the OSI model does Bluetooth technology reside in?

(Suggested: Paragraph or screenshot, 1 point)

Q4 - What is a relational database? What is a non-relational database? How do they differ? What type of database is MongoDB?

(Suggested: Short paragraph, 2 points)

Q5 – 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.

(Suggested: Screenshot, 0 points)

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.

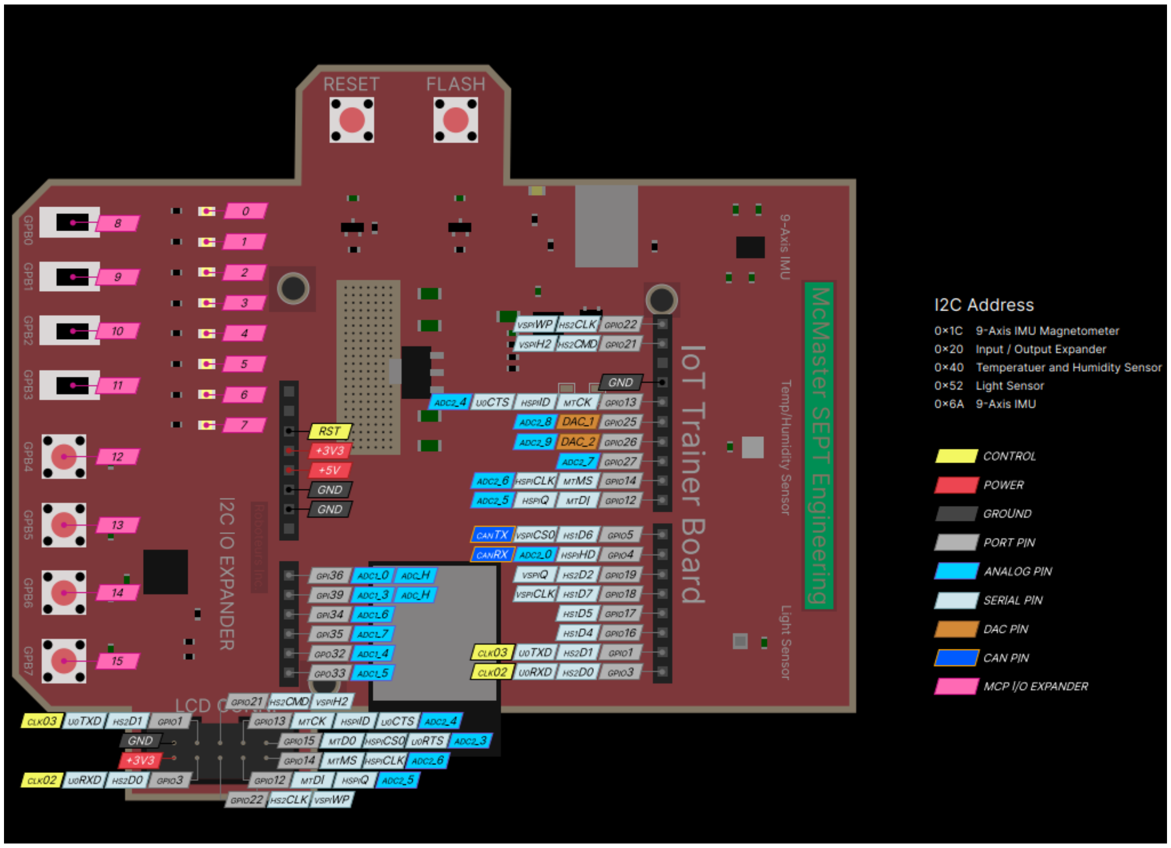
3. Experiment

3.1. Integrated Circuits & Components

The following integrated circuits are to be used in this laboratory

Identification	Description
MacIoT	Development Board

The MacIoT board aims to allow the user to learn step by step about IoT as well as communication protocols such as CAN which is used inside modern day vehicles and I2C which can be used to communicate with various sensors. The MacIoT board brings an onboard 9-axis accelerometer, gyroscope, and magnetometer, temperature and humidity sensor, light sensor as well as LoRa, I2C, and CAN. The ESP32-WROOM-32D is a low-cost, low-power system on a chip micrcontroller. The microcontroller has onboard WiFi and Bluetooth. The IO expander is a very helpful addition to the MacIoT board as it allows for more input and ouput ports to be used. This means that more sensors and devices can be connected than previously allowed from just the ESP32. This is done through I2C communication from the ESP32 to the IO expander.



I2C Address

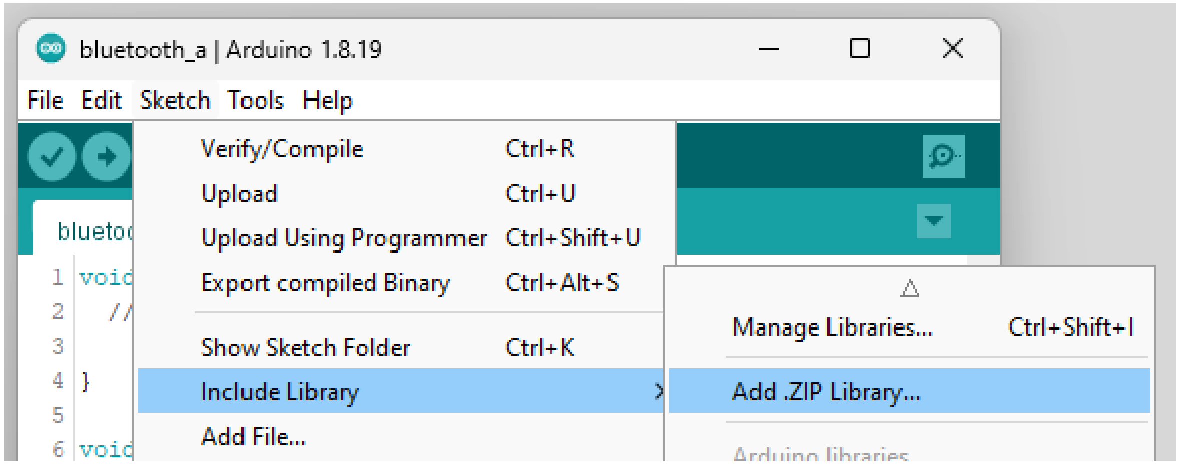
0×1C	9-Axis IMU Magnetometer
0×20	Input / Output Expander
0×40	Temperatuer and Humidity Sensor
0×52	Light Sensor
0×6A	9-Axis IMU

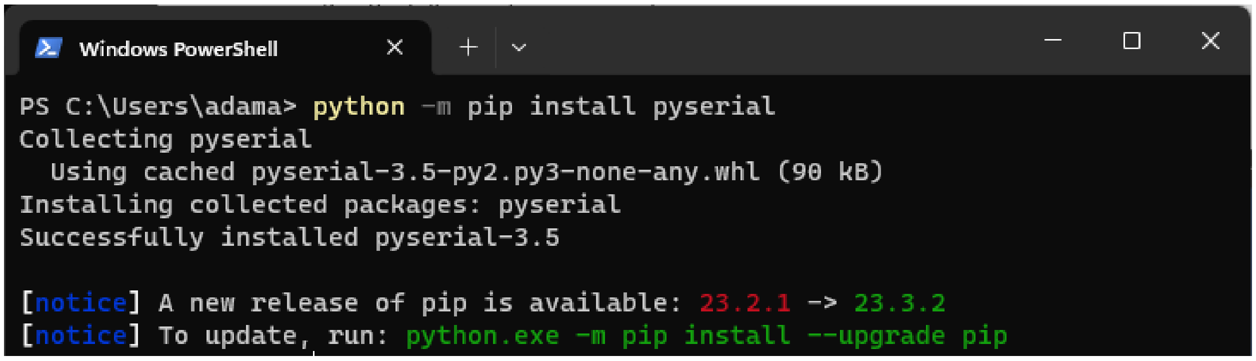
Obtain the manual and data sheets for each of the above devices from avenue. Please familiarize yourself with their logical and electrical characteristics.

3.2. Software to Install

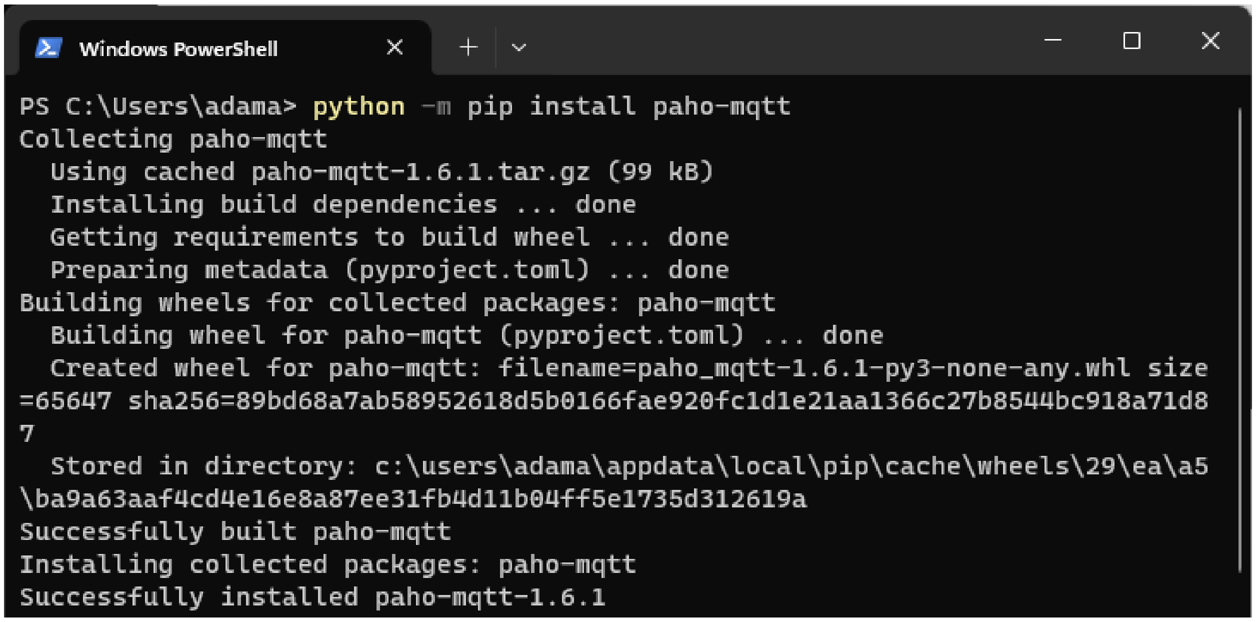
Ensure that the following libraries are installed:

- The APDS9306 library must be downloaded from GitHub.
  - Navigate to the following link: https://github.com/gmarti/AsyncAPDS9306
  - Press Code > Download Zip.
  - Navigate back to the Arduino IDE.
  - Navigate to Sketch > Include Library > Add .ZIP Library.
  - Find and open the downloaded zip library.
- Open Powershell as Administrator and tun the following commands
  - *pip install pyserial*
  - *pip install paho-mqtt*
  - *pip install pymongo*
- Install *MQTTool* on an iOS device (only one team member needs the app). Available at: *https://apps.apple.com/ca/app/MQTTool/id1085976398*

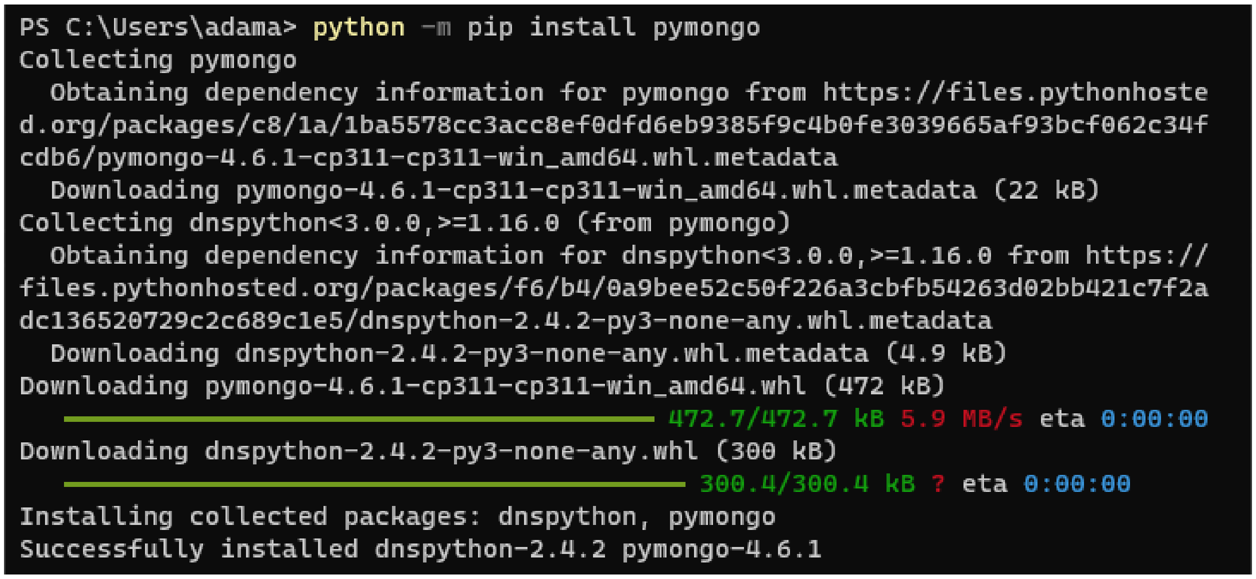

bluetooth_a | Arduino 1.8.19
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries



```
PS C:\Users\adama> python -m pip install pyserial
Collecting pyserial
  Using cached pyserial-3.5-py2.py3-none-any.whl (90 kB)
Installing collected packages: pyserial
Successfully installed pyserial-3.5

[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```



```
PS C:\Users\adama> python -m pip install paho-mqtt
Collecting paho-mqtt
  Using cached paho-mqtt-1.6.1.tar.gz (99 kB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Building wheels for collected packages: paho-mqtt
  Building wheel for paho-mqtt (pyproject.toml) ... done
  Created wheel for paho-mqtt: filename=paho_mqtt-1.6.1-py3-none-any.whl size
=65647 sha256=89bd68a7ab58952618d5b0166fae920fc1d1e21aa1366c27b8544bc918a71d8
7
  Stored in directory: c:\users\adama\appdata\local\pip\cache\wheels\29\ea\a5
\ba9a63aaf4cd4e16e8a87ee31fb4d11b04ff5e1735d312619a
Successfully built paho-mqtt
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-1.6.1
```



```
PS C:\Users\adama> python -m pip install pymongo
Collecting pymongo
  Obtaining dependency information for pymongo from https://files.pythonhoste
d.org/packages/c8/1a/1ba5578cc3acc8ef0dfd6eb9385f9c4b0fe3039665af93bcf062c34f
cdb6/pymongo-4.6.1-cp311-cp311-win_amd64.whl.metadata
  Downloading pymongo-4.6.1-cp311-cp311-win_amd64.whl.metadata (22 kB)
Collecting dnspython<3.0.0,>=1.16.0 (from pymongo)
  Obtaining dependency information for dnspython<3.0.0,>=1.16.0 from https://
files.pythonhosted.org/packages/f6/b4/0a9bee52c50f226a3cbfb54263d02bb421c7f2a
dc136520729c2c689c1e5/dnspython-2.4.2-py3-none-any.whl.metadata
  Downloading dnspython-2.4.2-py3-none-any.whl.metadata (4.9 kB)
Downloading pymongo-4.6.1-cp311-cp311-win_amd64.whl (472 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 472.7/472.7 kB 5.9 MB/s eta 0:00:00
Downloading dnspython-2.4.2-py3-none-any.whl (300 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 300.4/300.4 kB ? eta 0:00:00
Installing collected packages: dnspython, pymongo
Successfully installed dnspython-2.4.2 pymongo-4.6.1
```

3.3. 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)
- ESP32 Overview (https://youtu.be/UuxBfKA3U5M)

3.4. Temperature and Light MacIoT Sensor [15 marks]

This milestone aims to read data from sensors and print them to the serial monitor before we communicate to a server.

3.4.1. MacIoT Program

1. Launch the Arduino IDE and File → New Sketch.
2. File → Save As → Go to your Bluetooth folder in Repositories and save as bluetooth_a (this will create an Arduino folder (called a sketchbook) called bluetooth_a and an Arduino file inside bluetooth_a also called bluetooth_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 "bluetooth_a.h"
6. Install the APDS9306 library (See Section 3.2)

- Note: DO NOT COPY AND PASTE THE PDF CODE (Introduces whitespaces), COPY THE CODE FROM AVENUE

7. In the setup() function, start the serial monitor and print details to the user.

File: bluetooth_a.ino

```
void setup() {
  Serial.begin(9600);
  Serial.print("\n\n------------------------\n"
    + group_name + " : " + device_name + "\n------------------------\n\n");
}
```

8. Configure your I2C temperature sensor.

File: bluetooth_a.ino

```
void setup() {
  Serial.begin(9600);
  Serial.print("\n\n------------------------\n"
    + group_name + " : " + device_name + "\n------------------------\n\n");
  Wire.begin();
  Wire.beginTransmission(ADDR);
  Wire.endTransmission();
  delay(300);
}
```

9. Establish connection to your light intensity sensor.

File: bluetooth_a.ino

```
  light_sensor.begin(apds_gain, apds_time);
}
```

10. Write the polling command to poll the temperature sensor into it’s I2C data register

File: bluetooth_a.ino

```
void loop(){

  //Temp sensor
  Wire.beginTransmission(ADDR);
  Wire.write(TMP_CMD);
  Wire.endTransmission();
  delay(100);
}
```

11. Read 2 bytes from the I2C bus register.

File: bluetooth_a.ino

```
  Wire.requestFrom(ADDR, 2);

  char data[2];
  if(Wire.available() == 2){
    data[0] = Wire.read();
    data[1] = Wire.read();
  }
```

12. Using the sensor datasheet, calculate the temperature then convert it to comfortable units. **Why this formula? (Hint: Look it up in your MacIoT Manual and datasheets!)**

File: bluetooth_a.ino

```
  float temp = ((data[0] * 256.0) + data[1]);
  float temp_c = ((175.72 * temp) / 65536.0) - 46.85;
```

13. Poll the light sensor and convert it to comfortable units.

File: bluetooth_a.ino

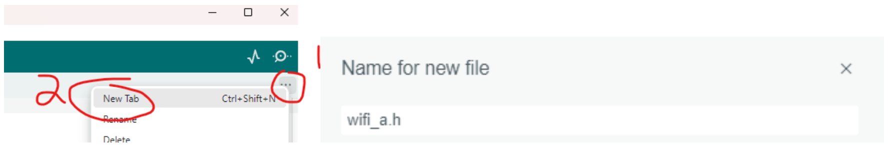
```
   //Sample light sensor
  AsyncAPDS9306Data light_data = light_sensor.syncLuminosityMeasurement();

  //Calculate luminosity
  float lux = light_data.calculateLux();
```

14. Print this data to the serial monitor.

File: bluetooth_a.ino

```
//Format data as a JSON string
  String formatted_data = "{ \"" + group_name + "\": { \"" + device_name + "\": { \"Temp\": 
\"" + String(temp_c) + "\", \"Luminosity\": \"" + String(lux) + "\" } } }" + '\n';

  Serial.println(formatted_data);
  delay(DELAY_BETWEEN_SAMPLES_MS);
}
```

15. Create a new header file by clicking the three dots and selecting the new tab in the dropdown menu. Name this file **bluetooth_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 objects.



16. Include the libraries for each sensor

File: bluetooth_a.h

```
//Libraries
#include <Arduino.h>
#include <Wire.h>
#include <AsyncAPDS9306.h>
```

17. Define the register address for your built-in temperature sensor and the command to poll from it. **Why 0x40 and 0xF3? (Hint: Look it up in your MacIoT Manual!)**

File: bluetooth_a.h

```
//I2C Addresses for Temperature Sensor
#define ADDR (byte)(0x40)
#define TMP_CMD (byte)(0xF3)
```

18. Set the polling rate

File: bluetooth_a.h

```
//Sample frequency
#define DELAY_BETWEEN_SAMPLES_MS 5000
```

19. Set your group and device name. Update this for your group!

File: bluetooth_a.h

```
//Device information
String group_name = "GroupA";
String device_name = "DeviceA";
```

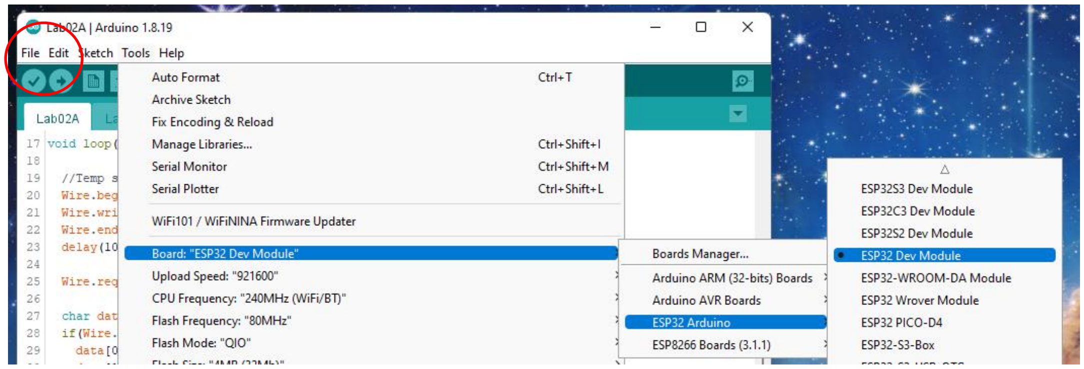
20. Configure your light intensity sensor.

File: bluetooth_a.h

```
//Instantiating sensor object and configuration
AsyncAPDS9306 light_sensor;
const APDS9306_ALS_GAIN_t apds_gain = APDS9306_ALS_GAIN_1;
const APDS9306_ALS_MEAS_RES_t apds_time = APDS9306_ALS_MEAS_RES_16BIT_25MS;
```

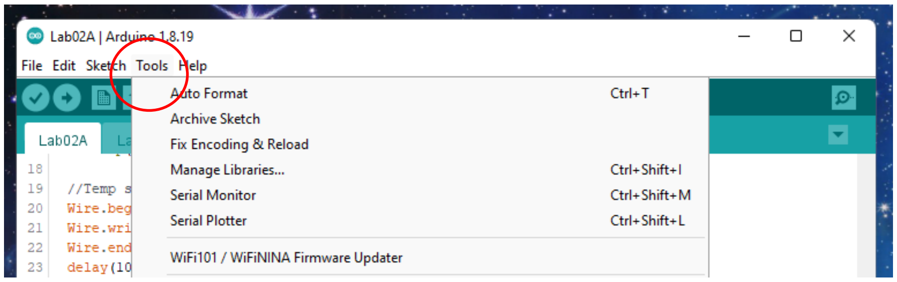
3.4.2. Upload Code to MacIoT

1. Change the board to Generic ESP8266 Module by going to *Tools > Board > ESP8266 Boards > ESP32 Dev Module*



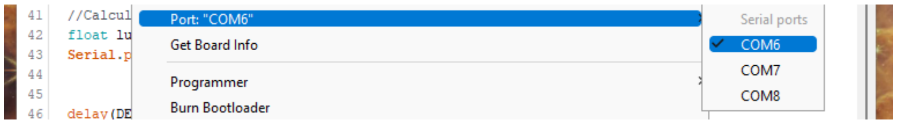
Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"

2. Plug in the MacIoT to your computer and install drivers if prompted
   - Note: You may need to update your security settings (there will be a prompt as a new icon on your taskbar)
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<#>*


Lab02A | Arduino 1.8.19
File Edit Sketch Tools Help
Lab02A
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater

```
//Temp s
Wire.beg
Wire.wri
Wire.end
delay(10
```

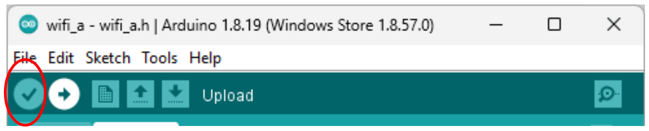
…



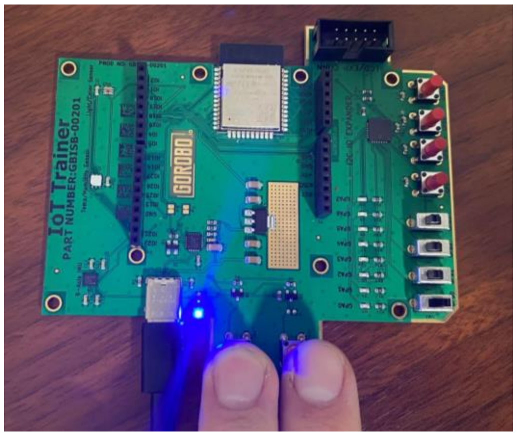
```
//Calcul
float lu
Serial.p

delay(DE
```

Note: You will see three COM ports available; write all of them down. You will be using them in this lab. For this milestone, connect to the lowest COM number. You may need to try all three ports to find the one you need for the microcontroller to communicate.

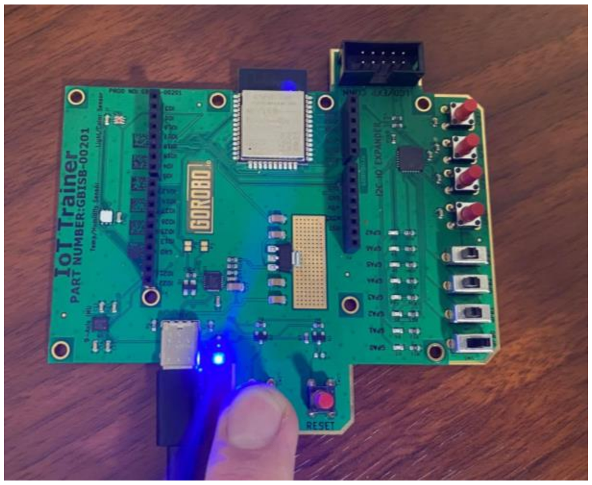
4. Press the verify button to compile the code. Keep an eye on the terminal window for any errors that arise.


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

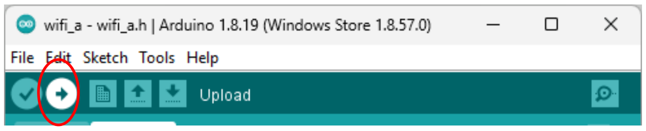
5. Uploading to the MacIoT board can be somewhat unintuitive. Follow the following steps:
   - Hold down **both** the **FLASH** and **RESET** buttons



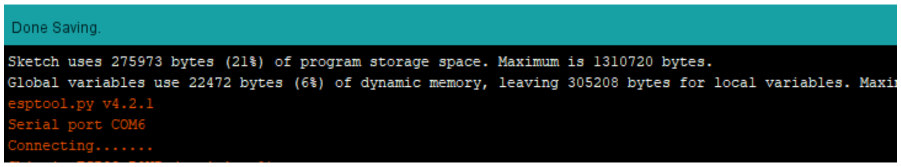
- After 2 seconds, **release** the **RESET** button. Continue to hold the **FLASH** button



- Press the upload button to compile and upload the code. Once the code finishes compiling and begins to upload, **release** the **FLASH** button.
    i. Keep an eye on the terminal window for any errors that arise. *Sketch > Upload*

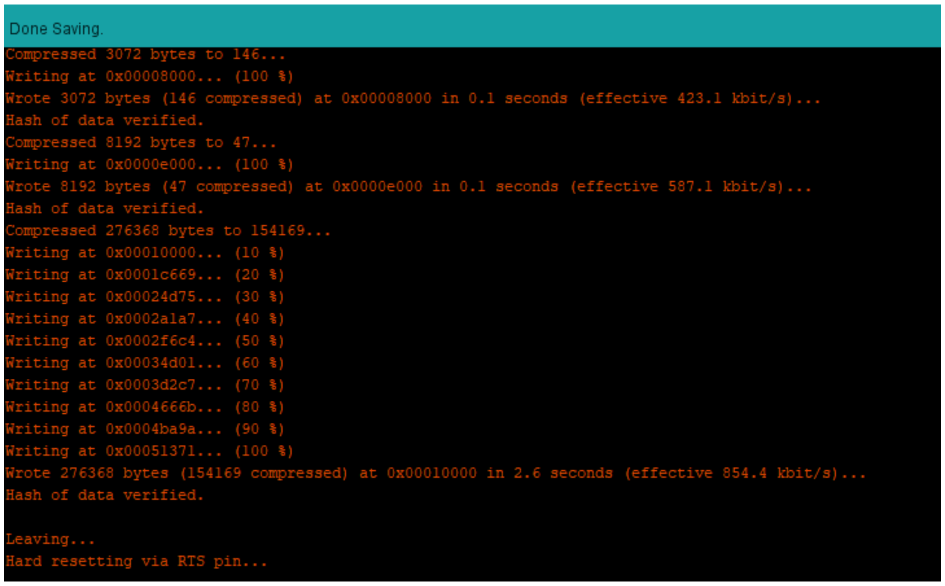

wifi_a - wifi_a.h | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
Upload

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



```
Done Saving.
Sketch uses 275973 bytes (21%) of program storage space. Maximum is 1310720 bytes.
Global variables use 22472 bytes (6%) of dynamic memory, leaving 305208 bytes for local variables. Maxi
esptool.py v4.2.1
Serial port COM6
Connecting.......
```

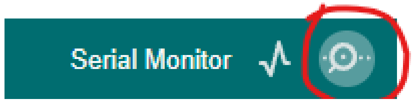
A successful upload should look like this in the Arduino IDE output (bottom left of IDE):

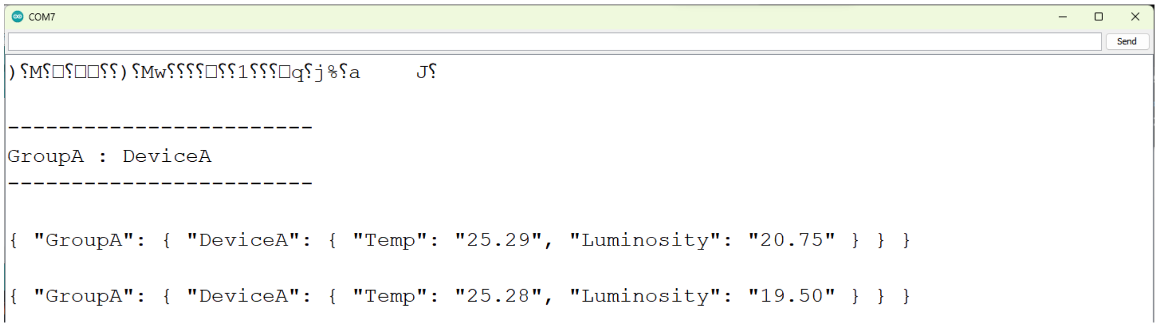


```
Done Saving.
Compressed 3072 bytes to 146...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (146 compressed) at 0x00008000 in 0.1 seconds (effective 423.1 kbit/s)...
Hash of data verified.
Compressed 8192 bytes to 47...
Writing at 0x0000e000... (100 %)
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.1 seconds (effective 587.1 kbit/s)...
Hash of data verified.
Compressed 276368 bytes to 154169...
Writing at 0x00010000... (10 %)
Writing at 0x0001c669... (20 %)
Writing at 0x00024d75... (30 %)
Writing at 0x0002a1a7... (40 %)
Writing at 0x0002f6c4... (50 %)
Writing at 0x00034d01... (60 %)
Writing at 0x0003d2c7... (70 %)
Writing at 0x0004666b... (80 %)
Writing at 0x0004ba9a... (90 %)
Writing at 0x00051371... (100 %)
Wrote 276368 bytes (154169 compressed) at 0x00010000 in 2.6 seconds (effective 854.4 kbit/s)...
Hash of data verified.

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

6. 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).





```
)ʕMʕ□ʕ□□ʕʕ)ʕMwʕʕʕʕ□ʕʕ1ʕʕʕ□qʕj%ʕa      Jʕ

------------------------
GroupA : DeviceA
------------------------

{ "GroupA": { "DeviceA": { "Temp": "25.29", "Luminosity": "20.75" } } }

{ "GroupA": { "DeviceA": { "Temp": "25.28", "Luminosity": "19.50" } } }
```

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

3.4.3. Transmitting Over Bluetooth

1. Duplicate bluetooth_a in your course repository and call the new folder and files bluetooth_b
2. Open bluetooth_b.ino in a new Arduino IDE tab. bluetooth_b.h will also be available
3. Modify your header file to include the BluetoothSerial library. Select the header file tab and include the following libraries above the existing code

File: bluetooth_b.h

```
//Libraries
#include <Arduino.h>
#include <Wire.h>
#include <AsyncAPDS9306.h>
#include "BluetoothSerial.h"
```

4. Below, instantiate the bluetooth serial object.

File: bluetooth_b.h

```
//Instantiating bluetooth serial object
BluetoothSerial bluetooth_serial;
```

5. Navigate to the implementation file and add begin bluetooth communication in the setup() function

File: bluetooth_b.ino

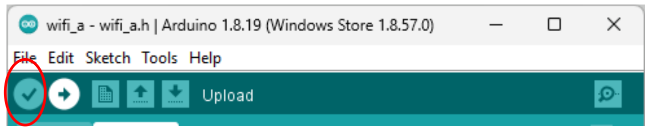
```
  SerialBT.begin(group_name + " : " + device_name);
  Serial.println("Ready for bluetooth connection!");
```

6. Modify the void() loop function to transmit the JSON formatted bluetooth message

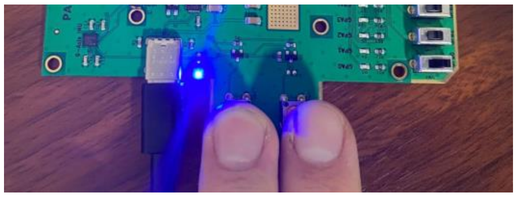
File: bluetooth_b.ino

```
 //Push the string characters onto the bluetooth output buffer
  for (int i = 0; i < strlen(formatted_data.c_str()) + 1; i++){
    bluetooth_serial.write(formatted_data.c_str()[i]);
  }
  Serial.println("Bluetooth sent!");
  delay(DELAY_BETWEEN_SAMPLES_MS);
}
```

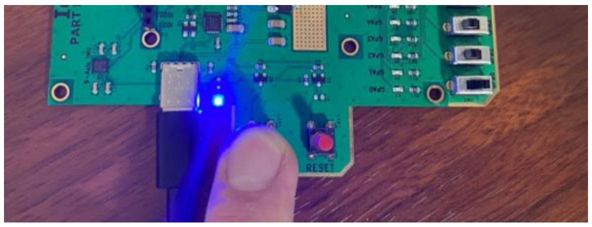
7. Press the verify button to compile the code.


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

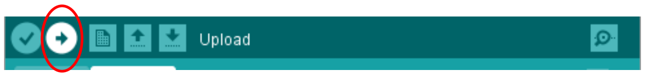
8. Uploading to the MacIoT board using the following steps:
   a. Hold down **both** the **FLASH** and **RESET** buttons



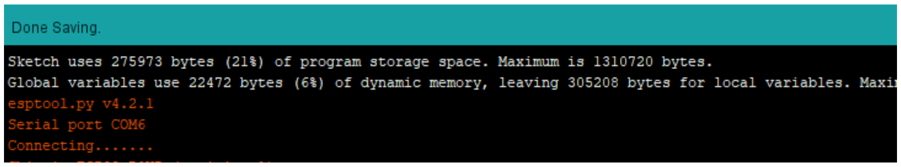
   b. After 2 seconds, **release** the **RESET** button. Continue to hold the **FLASH** button



   c. Press the upload button to compile and upload the code. Once the code finishes compiling and begins to upload, **release** the **FLASH** button.
      1. Keep an eye on the terminal window for any errors that arise. *Sketch > Upload*

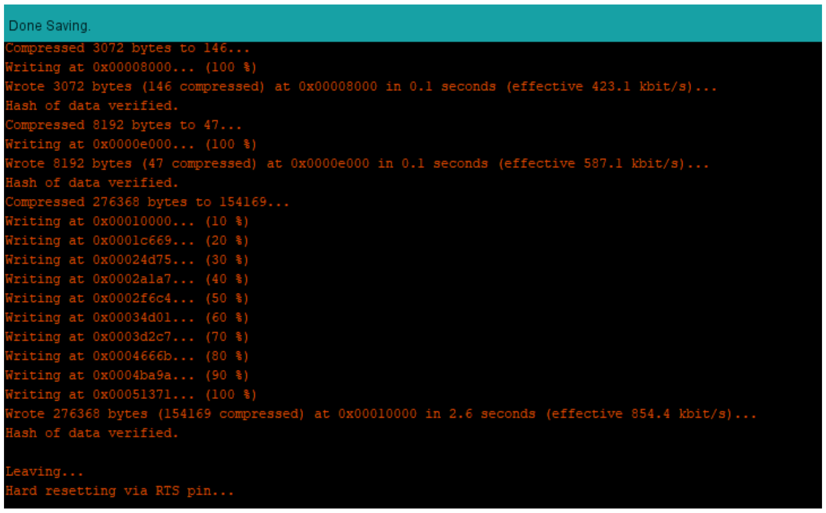


A successful connection should look like this in the Arduino IDE output (bottom left of IDE):



```
Done Saving.
Sketch uses 275973 bytes (21%) of program storage space. Maximum is 1310720 bytes.
Global variables use 22472 bytes (6%) of dynamic memory, leaving 305208 bytes for local variables. Maxi
esptool.py v4.2.1
Serial port COM6
Connecting.......
```

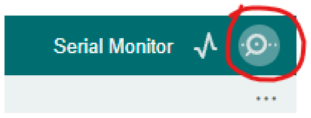
A successful upload should look like this in the Arduino IDE output (bottom left of IDE):

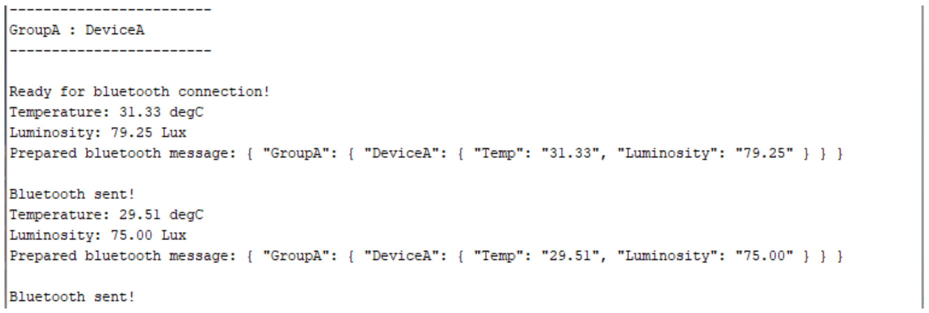


```
Done Saving.
Compressed 3072 bytes to 146...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (146 compressed) at 0x00008000 in 0.1 seconds (effective 423.1 kbit/s)...
Hash of data verified.
Compressed 8192 bytes to 47...
Writing at 0x0000e000... (100 %)
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.1 seconds (effective 587.1 kbit/s)...
Hash of data verified.
Compressed 276368 bytes to 154169...
Writing at 0x00010000... (10 %)
Writing at 0x0001c669... (20 %)
Writing at 0x00024d75... (30 %)
Writing at 0x0002a1a7... (40 %)
Writing at 0x0002f6c4... (50 %)
Writing at 0x00034d01... (60 %)
Writing at 0x0003d2c7... (70 %)
Writing at 0x0004666b... (80 %)
Writing at 0x0004ba9a... (90 %)
Writing at 0x00051371... (100 %)
Wrote 276368 bytes (154169 compressed) at 0x00010000 in 2.6 seconds (effective 854.4 kbit/s)...
Hash of data verified.

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

9. 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*

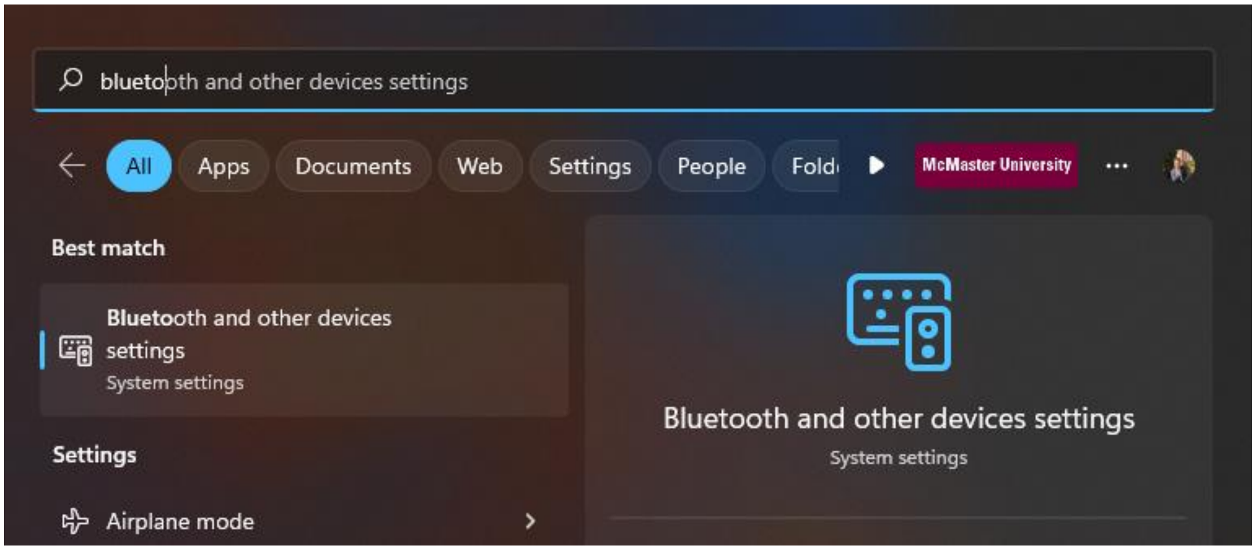
   Note: The corresponding COM port is used while the Serial Monitor runs. You must close the Serial Monitor to release the COM port. Not closing the Serial Monitor may lock you out of this COM port for future use (you need to restart your system to reopen the COM port).

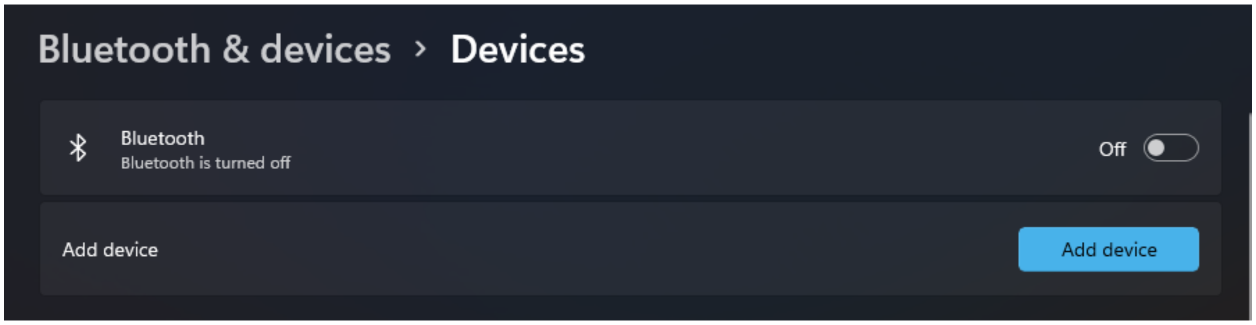




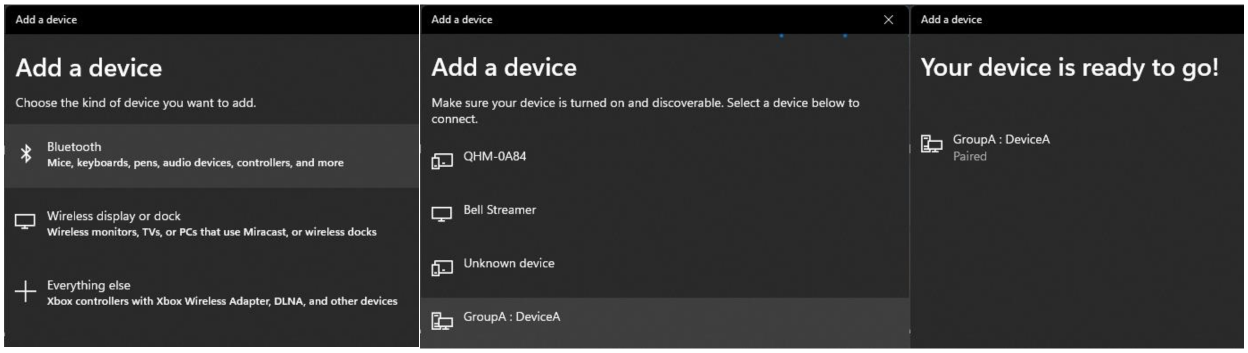
```
------------------------
GroupA : DeviceA
------------------------

Ready for bluetooth connection!
Temperature: 31.33 degC
Luminosity: 79.25 Lux
Prepared bluetooth message: { "GroupA": { "DeviceA": { "Temp": "31.33", "Luminosity": "79.25" } } }

Bluetooth sent!
Temperature: 29.51 degC
Luminosity: 75.00 Lux
Prepared bluetooth message: { "GroupA": { "DeviceA": { "Temp": "29.51", "Luminosity": "75.00" } } }

Bluetooth sent!
```

10. Enable Bluetooth on your PC and connect to the microcontroller. Toggle Bluetooth **On** and press **Add device**





11. Press on your microcontroller to connect.

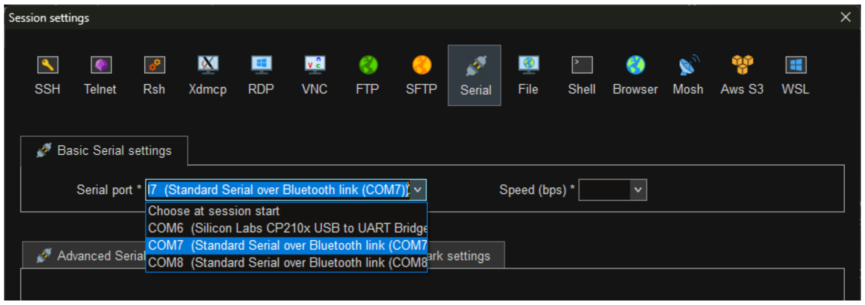

Add a device
Add a device
Bluetooth
Mice, keyboards, pens, audio devices, controllers, and more
Wireless display or dock
Wireless monitors, TVs, or PCs that use Miracast, or wireless docks
Everything else
Xbox controllers with Xbox Wireless Adapter, DLNA, and other devices
Add a device
Add a device
QHM-0A84
Bell Streamer
Unknown device
GroupA : DeviceA
Add a device
GroupA : DeviceA
Paired

Choose the kind of device you want to add.

Make sure your device is turned on and discoverable. Select a device below to connect.

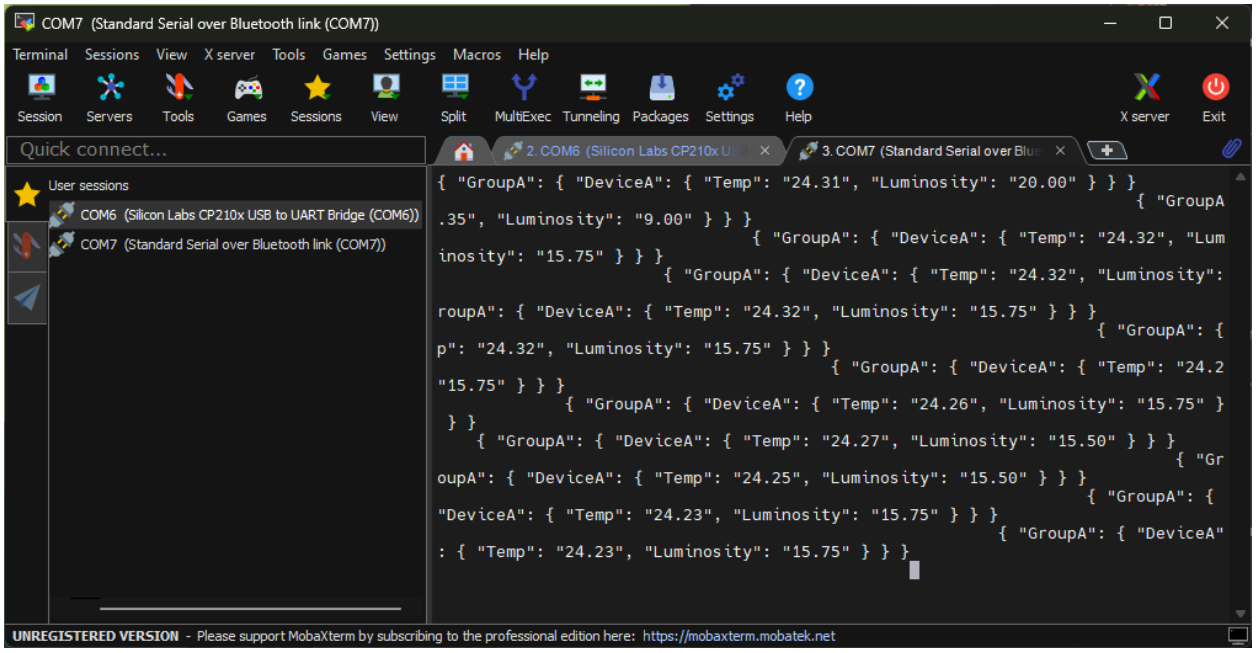
Your device is ready to go!

12. Launch MobaXTerm and press **Session** (Under the Terminal Dropdown menu)
    - MobaXTerm is a feature-filled application for managing servers and ports. We will use it to open the Bluetooth serial port and ensure that data is being transmitted between the MacIoT board and our PC.
13. Select Serial from the top ribbon. Choose a Bluetooth serial port. You may need to try all that are available to find which one is transmitting. For example, COM7 worked below



Serial port * 7 (Standard Serial over Bluetooth link (COM7))
Speed (bps) *

```
Choose at session start
COM6  (Silicon Labs CP210x USB to UART Bridge
COM7  (Standard Serial over Bluetooth link (COM7
COM8  (Standard Serial over Bluetooth link (COM8
```


COM7 (Standard Serial over Bluetooth link (COM7))
Terminal
Sessions
View
X server
Tools
Games
Settings
Macros
Help
Session
Servers
Tools
Games
Sessions
View
Split
MultiExec
Tunneling
Packages
Settings
Help
X server
Exit
Quick connect...
2. COM6 (Silicon Labs CP210x U
3. COM7 (Standard Serial over Blu
User sessions
COM6 (Silicon Labs CP210x USB to UART Bridge (COM6))
COM7 (Standard Serial over Bluetooth link (COM7))

```
{ "GroupA": { "DeviceA": { "Temp": "24.31", "Luminosity": "20.00" } } }
                                                                   { "GroupA
.35", "Luminosity": "9.00" } } }
                               { "GroupA": { "DeviceA": { "Temp": "24.32", "Lum
inosity": "15.75" } } }
                       { "GroupA": { "DeviceA": { "Temp": "24.32", "Luminosity":

roupA": { "DeviceA": { "Temp": "24.32", "Luminosity": "15.75" } } }
                                                                   { "GroupA": {
p": "24.32", "Luminosity": "15.75" } } }
                                        { "GroupA": { "DeviceA": { "Temp": "24.2
"15.75" } } }
             { "GroupA": { "DeviceA": { "Temp": "24.26", "Luminosity": "15.75" }
 } }
    { "GroupA": { "DeviceA": { "Temp": "24.27", "Luminosity": "15.50" } } }
                                                                           { "Gr
oupA": { "DeviceA": { "Temp": "24.25", "Luminosity": "15.50" } } }
                                                                  { "GroupA": {
"DeviceA": { "Temp": "24.23", "Luminosity": "15.75" } } }
                                                         { "GroupA": { "DeviceA"
: { "Temp": "24.23", "Luminosity": "15.75" } } }
```

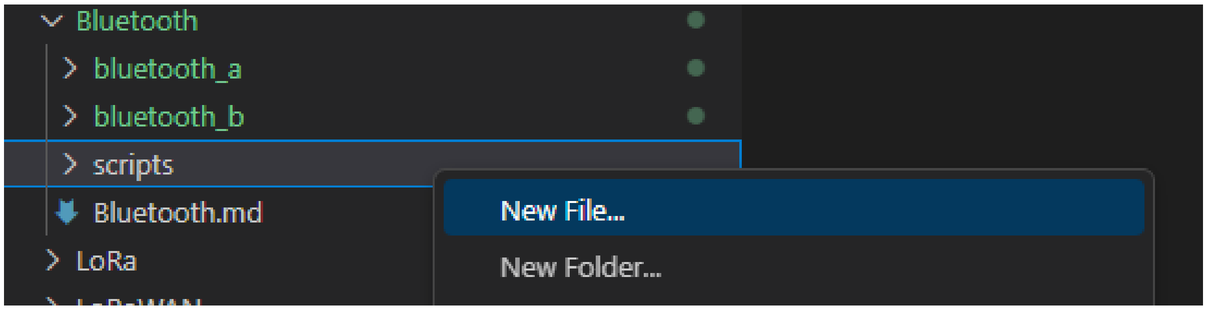
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

Close your serial monitor on MobaXTerm and your Arduino IDE, but keep MacIoT plugged in

3.4.4. Serial To MQTT

It may not be enough to view the data. We want to be able to act on the data being sent, collect it over time, and visualize it on an HMI or familiar format.

1. In your Bluetooth folder, create a new folder called scripts
2. Open VSCode and go File > Open Folder and open your local repository "Bluetooth, you will be able to see the scripts folder in the file explorer on the left hand tab of your VSCode Editor
3. Right-click on the scripts folder and select **New File**.



4. Name it btserial2mqtt.py
5. Double-click to open it in the text editor.
6. Import the required libraries.

File: btserial2mqtt.py

```
#   Imported libraries
import serial
import json
import paho.mqtt.client as mqtt
import time
```

7. Take user input to configure MQTT client
   a. Use broker.hivemq.com for your mqtt_ip
   b. Update the Bluetooth_com_port to your COM PORT

File: btserial2mqtt.py

```
#   Configuring MQTT server and COM
mqtt_ip = None
mqtt_port = None
bluetooth_com_port = None

mqtt_ip = input("MQTT Broker IP: ")
if(mqtt_ip == None or mqtt_ip == ''): mqtt_ip = 'test.mosquitto.org'

mqtt_port = input("MQTT Broker Port: ")
if(mqtt_port == None or mqtt_port== ''): mqtt_port = 1883

bluetooth_com_port = input("Bluetooth COM (e.g. COM7): ")
if(bluetooth_com_port == None or bluetooth_com_port  == ''): bluetooth_com_port  = 'COM7'

print(f'\n-------\nCONFIGURATION\n-------\nIP: {mqtt_ip}\nPORT: {mqtt_port}\nBT COM:
{bluetooth_com_port}')
```

8. Instantiate the MQTT client.

File: btserial2mqtt.py

```
#   MQTT callback functions
def on_connect(client, userdata, flags, rc):
    print(f"Connected, status = {str(rc)}")
def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
#   Connecting client to MQTT broker
print("Connecting to MQTT")
while client.is_connected is False:
    print(".")
    time.sleep(0.7)
    client.connect(mqtt_ip, mqtt_port, 60)
```

9. Connect to Bluetooth serial connection.

File: btserial2mqtt.py

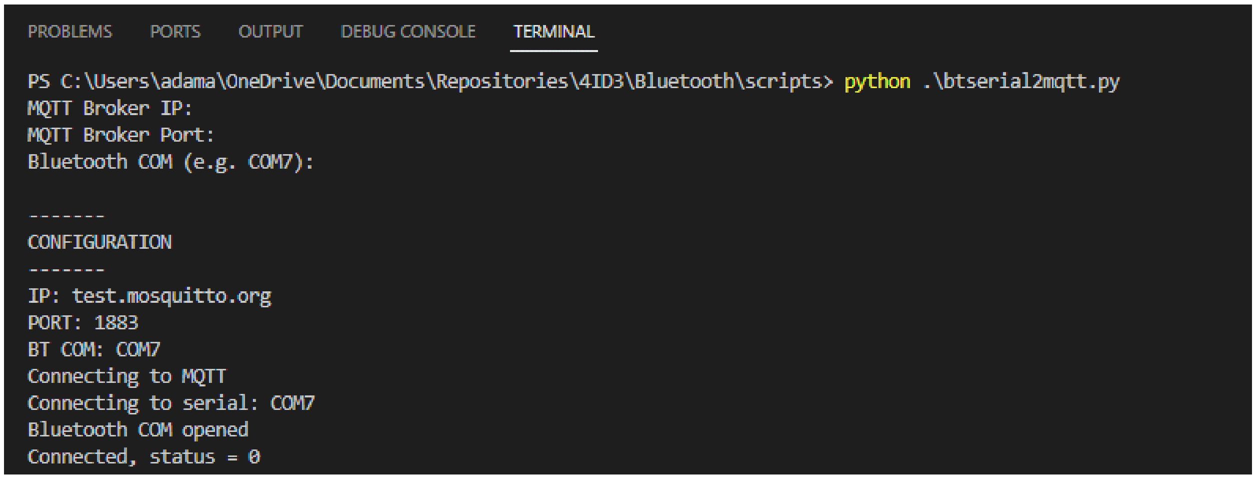
```
#   Connecting to serial
print("Connecting to serial: " + bluetooth_com_port)
time.sleep(1)
ser = serial.Serial(bluetooth_com_port, 9600)
print("Bluetooth COM opened")
```

10. Parse the data as it arrives.

File: btserial2mqtt.py

```
while True:
    #   Refreshing the client connection
    if(client.is_connected() == False):
        client.connect(mqtt_ip, mqtt_port, 60)

    client.loop()
    serial_json_string=str(ser.readline())  #   Reading serial to newline character '\n'
    first_split_index = serial_json_string.find('{')    #   Start of message
    second_split_index = serial_json_string.rfind('}')  #   End of message
    serial_json_string = serial_json_string[first_split_index: second_split_index+1]

    try:
        sensor_data = json.loads(serial_json_string)
        print(f"Received from Bluetooth: {sensor_data}")
        group_name = list(sensor_data.keys())[0]    #   Getting first key in JSON
        device_id = list(sensor_data[group_name])[0]    #   Getting value of first key from
JSON
        for key, val in sensor_data[group_name][device_id].items():
            success = client.publish(f'{group_name}/{device_id}/{key}', val.encode("UTF-8"))
            print(f'MQTT Publish {group_name}/{device_id}/{key} -> {val}\n    : send status =
{success.is_published()}')

    except Exception as e:
        print("Failed to decode: ", e)
        print(serial_json_string)
ser.close()
```

11. Within VSCode, navigate to **Terminal > New Terminal**. A new Powershell terminal window will open inside VSCode. Navigate to your **Bluetooth/scripts/** directory.
12. To launch the python application, run the following command:
    python ./btserial2mqtt.py

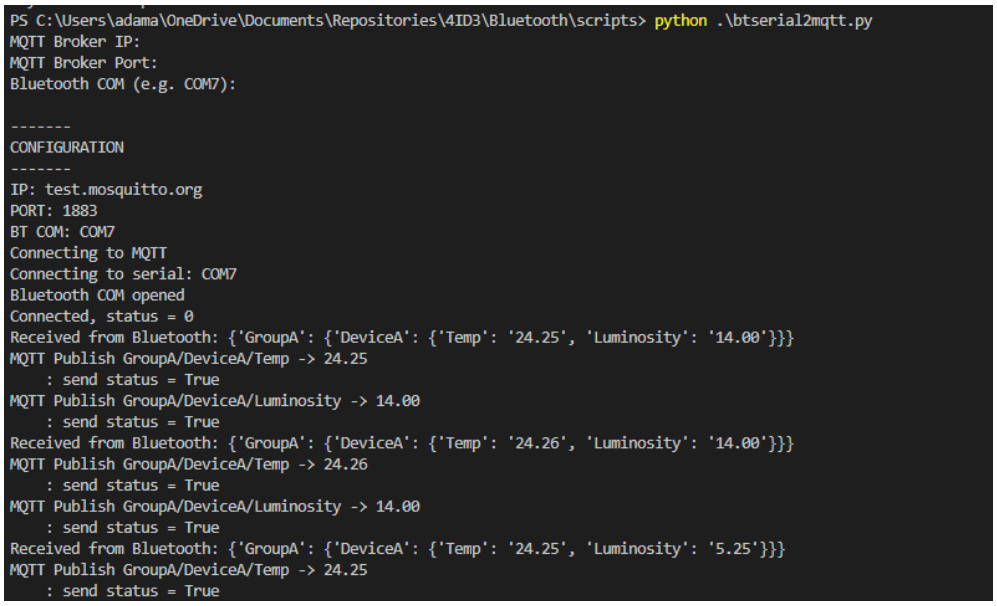


```
PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Bluetooth\scripts> python .\btserial2mqtt.py
MQTT Broker IP:
MQTT Broker Port:
Bluetooth COM (e.g. COM7):

-------
CONFIGURATION
-------
IP: test.mosquitto.org
PORT: 1883
BT COM: COM7
Connecting to MQTT
Connecting to serial: COM7
Bluetooth COM opened
Connected, status = 0
```

13. Fill in the prompted information. Press **enter** to use the default IP, Broker port and COM you hardcoded in step 7

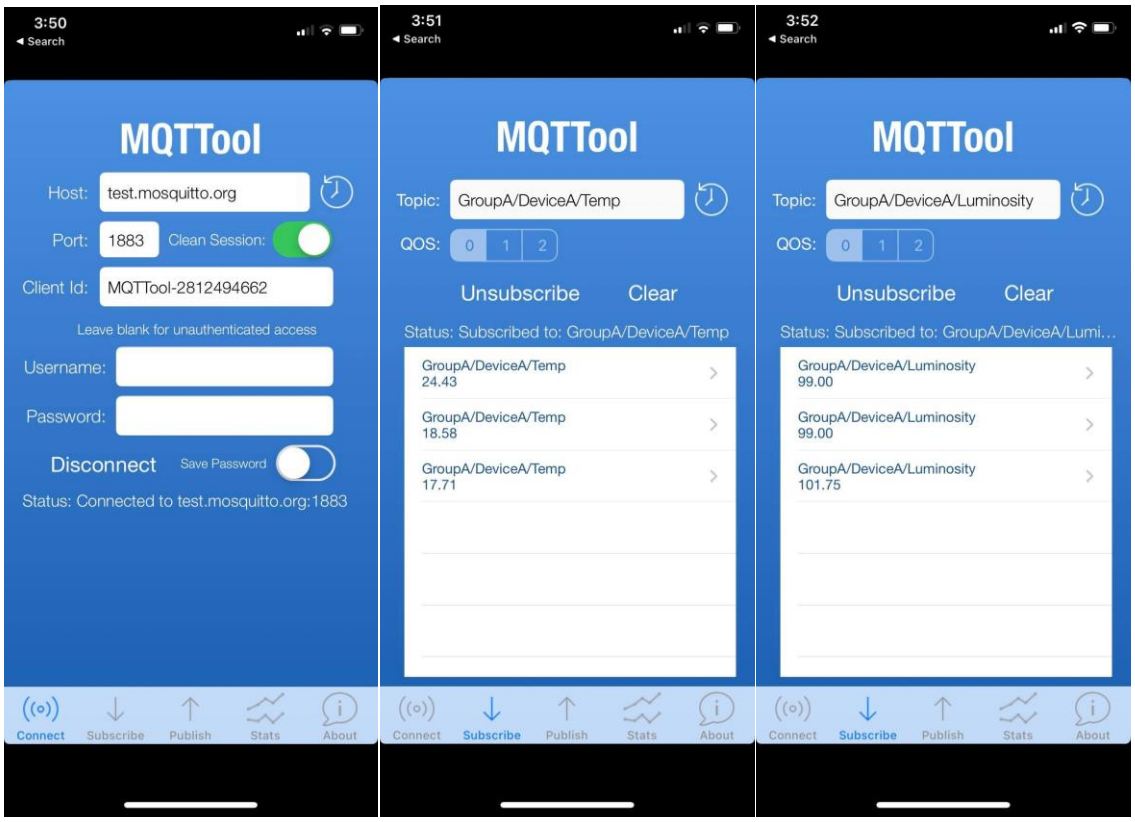
14. Ensure that your MacIoT board is still transmitting. Press RESET to restart it. Wait 10 seconds.
    - ENSURE THAT NO OTHER BLUETOOTH SERIAL MONITORS ARE OPEN, BLOCKING THE PORT

15. Run btserial2mqtt.py using the following command:
    Python ./btserial2mqttt.py



```
PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Bluetooth\scripts> python .\btserial2mqtt.py
MQTT Broker IP:
MQTT Broker Port:
Bluetooth COM (e.g. COM7):

-------
CONFIGURATION
-------
IP: test.mosquitto.org
PORT: 1883
BT COM: COM7
Connecting to MQTT
Connecting to serial: COM7
Bluetooth COM opened
Connected, status = 0
Received from Bluetooth: {'GroupA': {'DeviceA': {'Temp': '24.25', 'Luminosity': '14.00'}}}
MQTT Publish GroupA/DeviceA/Temp -> 24.25
    : send status = True
MQTT Publish GroupA/DeviceA/Luminosity -> 14.00
    : send status = True
Received from Bluetooth: {'GroupA': {'DeviceA': {'Temp': '24.26', 'Luminosity': '14.00'}}}
MQTT Publish GroupA/DeviceA/Temp -> 24.26
    : send status = True
MQTT Publish GroupA/DeviceA/Luminosity -> 14.00
    : send status = True
Received from Bluetooth: {'GroupA': {'DeviceA': {'Temp': '24.25', 'Luminosity': '5.25'}}}
MQTT Publish GroupA/DeviceA/Temp -> 24.25
    : send status = True
```

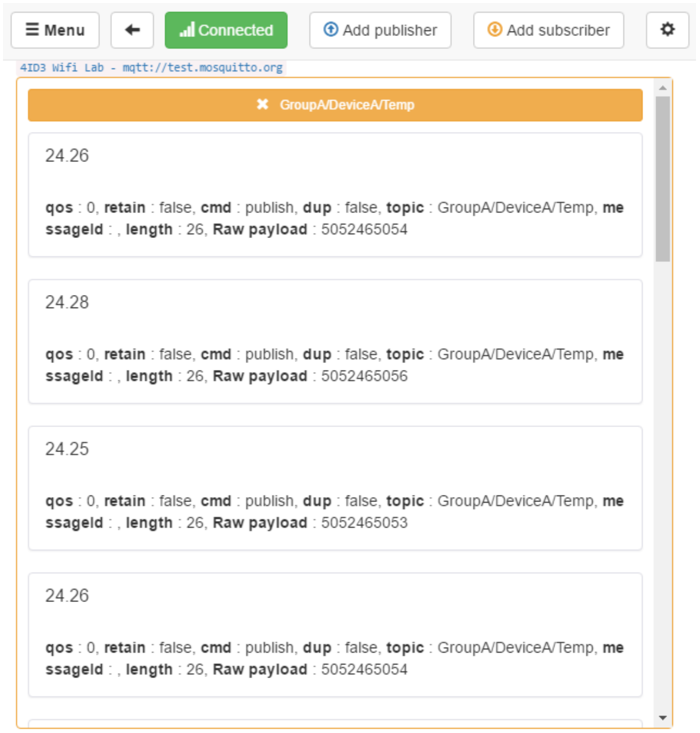
16. Use any MQTT application to verify that data is being transmitted to the correct paths



Host: test.mosquitto.org
Port: 1883 Clean Session:
Client Id: MQTTool-2812494662
Leave blank for unauthenticated access
Username:
Password:
Disconnect Save Password
Status: Connected to test.mosquitto.org:1883

Topic: GroupA/DeviceA/Temp
QOS: 0 1 2
Unsubscribe Clear
Status: Subscribed to: GroupA/DeviceA/Temp
GroupA/DeviceA/Temp 24.43
GroupA/DeviceA/Temp 18.58
GroupA/DeviceA/Temp 17.71

Topic: GroupA/DeviceA/Luminosity
QOS: 0 1 2
Unsubscribe Clear
Status: Subscribed to: GroupA/DeviceA/Lumi...
GroupA/DeviceA/Luminosity 99.00
GroupA/DeviceA/Luminosity 99.00
GroupA/DeviceA/Luminosity 101.75



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

GroupA/DeviceA/Temp

24.26

qos : 0, **retain** : false, **cmd** : publish, **dup** : false, **topic** : GroupA/DeviceA/Temp, **messageId** : , **length** : 26, **Raw payload** : 5052465054

24.28

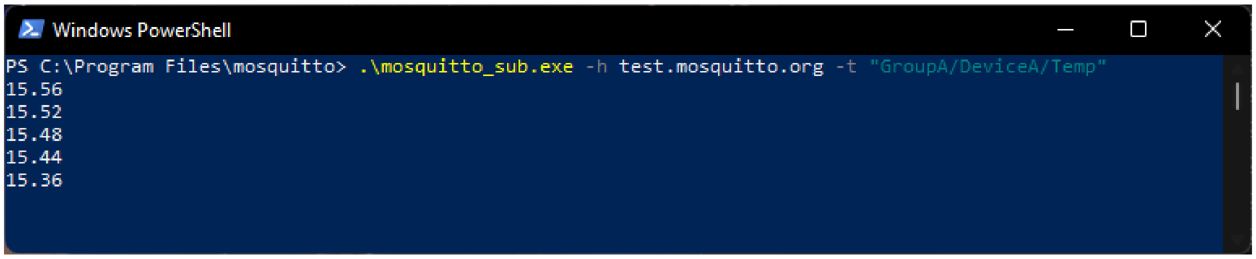
qos : 0, **retain** : false, **cmd** : publish, **dup** : false, **topic** : GroupA/DeviceA/Temp, **messageId** : , **length** : 26, **Raw payload** : 5052465056

24.25

qos : 0, **retain** : false, **cmd** : publish, **dup** : false, **topic** : GroupA/DeviceA/Temp, **messageId** : , **length** : 26, **Raw payload** : 5052465053

24.26

qos : 0, **retain** : false, **cmd** : publish, **dup** : false, **topic** : GroupA/DeviceA/Temp, **messageId** : , **length** : 26, **Raw payload** : 5052465054



```
PS C:\Program Files\mosquitto> .\mosquitto_sub.exe -h test.mosquitto.org -t "GroupA/DeviceA/Temp"
15.56
15.52
15.48
15.44
15.36
```

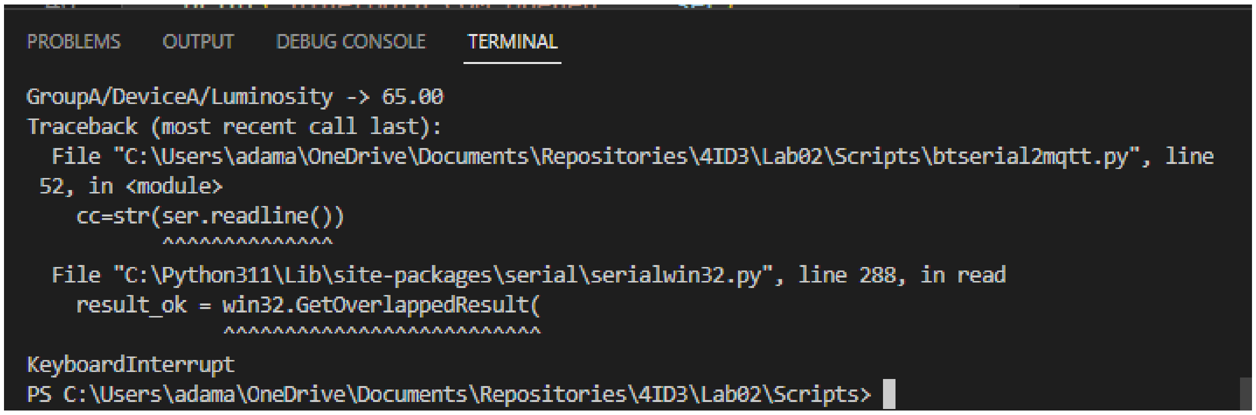
17. Now that data is being transmitted to MQTT, using knowledge from previous labs, set up a NodeRED flow to read this data from the public MQTT broker and visualize it
    1. Note: If you get an error that the PORT 1880 is not available for node-red, try opening your browser to http:127.0.0.1:1880/ (you may have node-red already running)

Milestone 1: Instructor/TA to verify sensor output appearing on VSCode, MQTTool, and NodeRed

Lab Report Q1 – Screenshot of VSCode Serial Monitor, MQTTool Serial Output, and NodeRed

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.

18. Press CTRL + C to exit the Python application in VSCode



```
GroupA/DeviceA/Luminosity -> 65.00
Traceback (most recent call last):
  File "C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab02\Scripts\btserial2mqtt.py", line
 52, in <module>
    cc=str(ser.readline())
           ^^^^^^^^^^^^^^^
  File "C:\Python311\Lib\site-packages\serial\serialwin32.py", line 288, in read
    result_ok = win32.GetOverlappedResult(
                ^^^^^^^^^^^^^^^^^^^^^^^^^^
KeyboardInterrupt
PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab02\Scripts>
```

3.5. Humidity MacIoT Bluetooth Sensor [10 marks]

Duplicate the Bluetooth_a code from Milestone 1 and update it so that Humidity is read from the sensor (call it bluetooth_humidty), formatted as a JSON attribute, and transmitted to MQTT. Hint: You must review the MacIoT User Manual to identify the I2C Humidity Command and formula.

Milestone 2: Instructor/TA to verify sensor output appearing on VSCode, MQTTool, and NodeRed

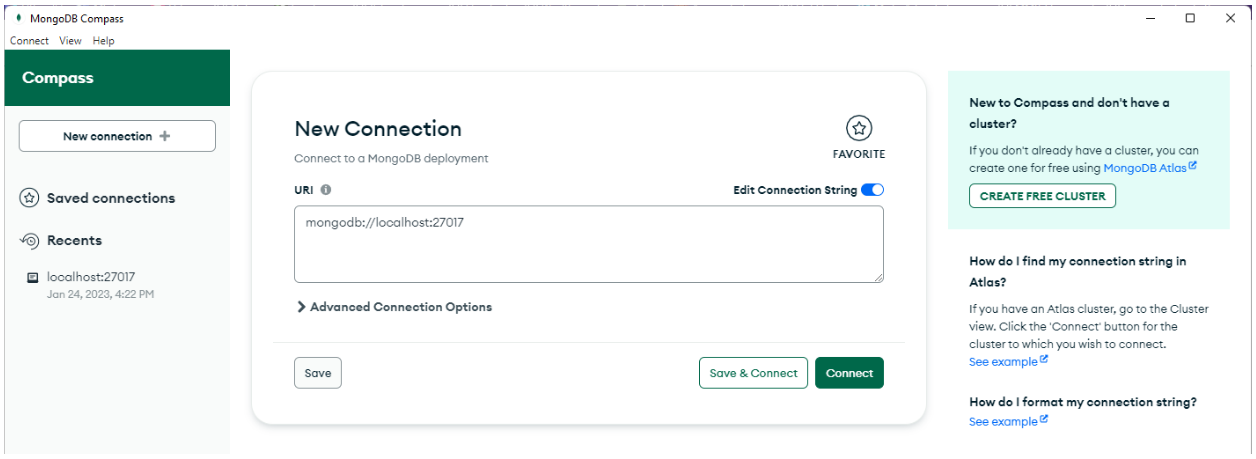
Lab Report Q2 – Screenshot of VSCode Serial Monitor, MQTTool Serial Output, and NodeRed

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.

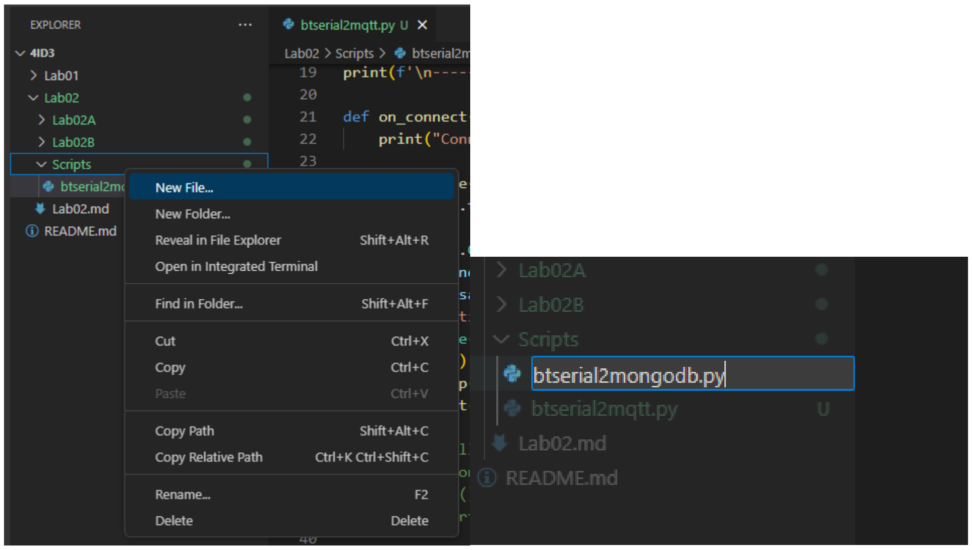
3.6. Connect to Database [25 marks]

We will now set up a Mongo database (db) to record our temperature/light (bluetooth_a) IoT data.

1. Launch MongoDB Compass and Press Connect


MongoDB Compass
Connect View Help
Compass
New connection +
Saved connections
Recents
localhost:27017
Jan 24, 2023, 4:22 PM
New Connection
Connect to a MongoDB deployment
FAVORITE
URI
Edit Connection String
mongodb://localhost:27017
Advanced Connection Options
Save
Save & Connect
Connect
New to Compass and don't have a cluster?
If you don't already have a cluster, you can create one for free using MongoDB Atlas
CREATE FREE CLUSTER
How do I find my connection string in Atlas?
If you have an Atlas cluster, go to the Cluster view. Click the 'Connect' button for the cluster to which you wish to connect.
See example
How do I format my connection string?
See example

2. Navigate back to VSCode and create a new Python script in the Scripts/ folder named btserial2mongodb



3. Import libraries

File: btserial2pymongodb.py

```
#   Importing libraries
import serial
import json
import time
import pymongo
```

4. Set db_address and Bluetooth_com_port

File: btserial2mongodb.py

```
#   Setting the COM and MongoDB targets
db_address = "mongodb://localhost:27017/"
bluetooth_com_port = None
bluetooth_com_port bluetoothCOM = input("Bluetooth COM (e.g. COM7): ")
if(bluetooth_com_port == None or bluetooth_com_port  == ''): bluetooth_com_port = 'COM7'
print(f'\n-------\nCONFIGURATION\n-------\nBT COM: { bluetooth_com_port }\nMongoDB:
{db_address}')
```

5. Connect to Bluetooth and MongoDB

File: btserial2mongodb.py

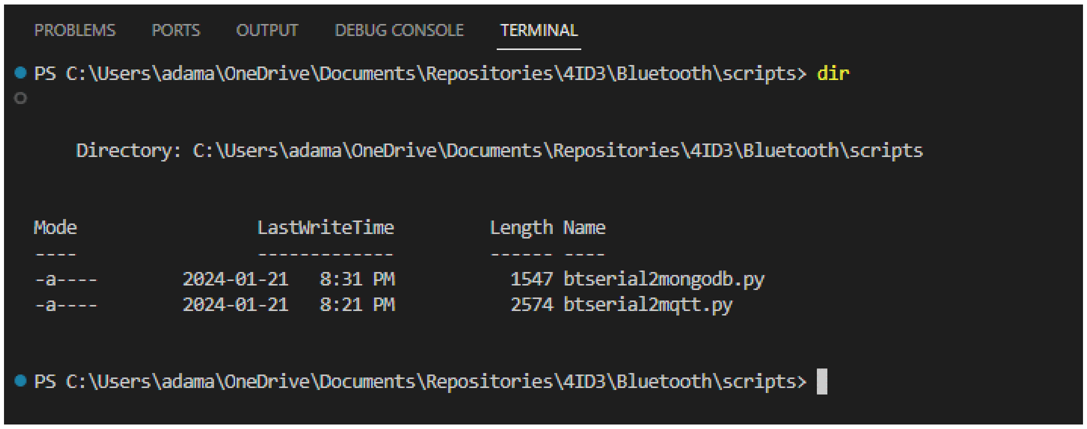
```
#   Connecting to bluetooth
print("Connecting to serial: " + bluetooth_com_port)
time.sleep(1)
ser = serial.Serial(bluetooth_com_port, 9600)
print("Bluetooth COM opened")

#   Instantiating the MongoDB client
mongo_client = pymongo.MongoClient(db_address)
```

6. While data streams in, parse the JSON object and insert a document into the MongoDB database

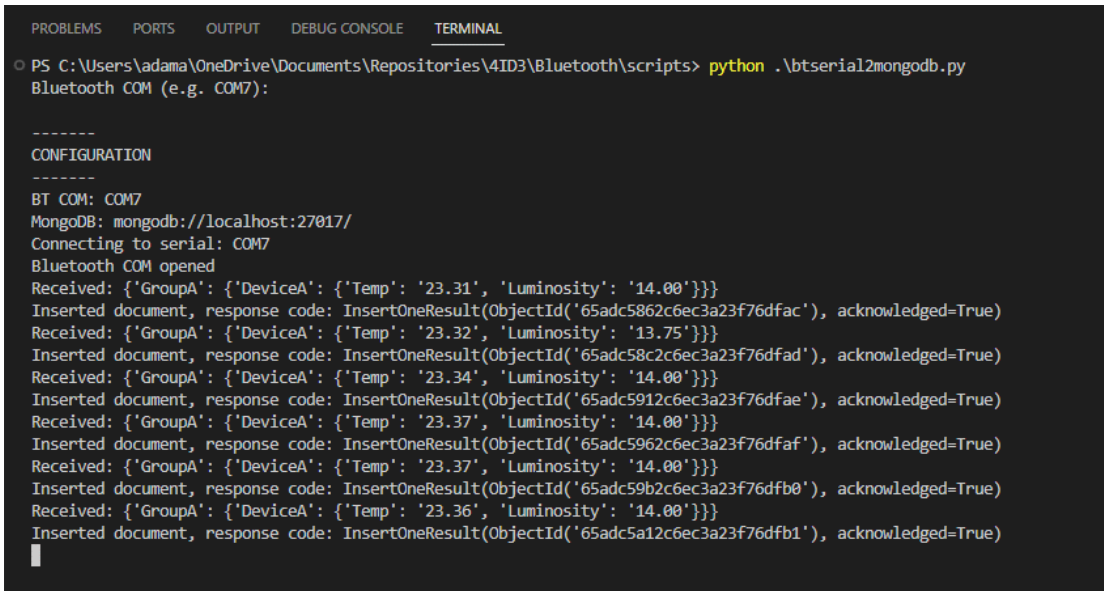
File: btserial2mongodb.py

```
while True:
    serial_json_string=str(ser.readline())
    firstSplitIndex = serial_json_string.find('{')
    secondSplitIndex = serial_json_string.rfind('}')
    serial_json_string = serial_json_string[firstSplitIndex: secondSplitIndex+1]

    try:
        json_data = json.loads(serial_json_string)
        print(json_data)
        groupName = list(json_data.keys())[0]
        deviceId = list(json_data[groupName])[0]

        mydb = mongo_client[groupName]
        mycollection = mydb[deviceId]

        dbDict = dict({})
        for key, val in json_data[groupName][deviceId].items():
            dbDict[key] = val

        ret = mycollection.insert_one(dbDict)
        print("Inserted suserial_json_stringessfully")

    except:
        print("Failed to decode: ")
        print(serial_json_string)


ser.close()
```

7. Run the python program while the MacIoT board is connected with Bluetooth and transmitting data. If the data is being parsed correctly and it can access the MongoDB database, then **Inserted Successfully** should be printed.

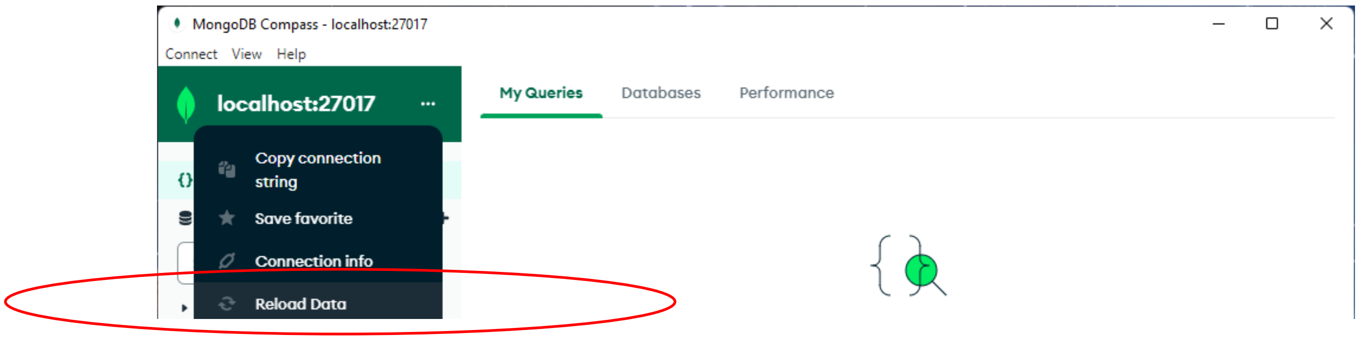


```
PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Bluetooth\scripts> dir


    Directory: C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Bluetooth\scripts


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        2024-01-21   8:31 PM           1547 btserial2mongodb.py
-a----        2024-01-21   8:21 PM           2574 btserial2mqtt.py


PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Bluetooth\scripts>
```

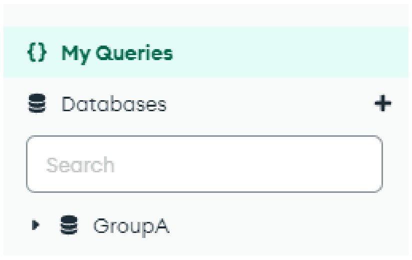


```
PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Bluetooth\scripts> python .\btserial2mongodb.py
Bluetooth COM (e.g. COM7):

-------
CONFIGURATION
-------
BT COM: COM7
MongoDB: mongodb://localhost:27017/
Connecting to serial: COM7
Bluetooth COM opened
Received: {'GroupA': {'DeviceA': {'Temp': '23.31', 'Luminosity': '14.00'}}}
Inserted document, response code: InsertOneResult(ObjectId('65adc5862c6ec3a23f76dfac'), acknowledged=True)
Received: {'GroupA': {'DeviceA': {'Temp': '23.32', 'Luminosity': '13.75'}}}
Inserted document, response code: InsertOneResult(ObjectId('65adc58c2c6ec3a23f76dfad'), acknowledged=True)
Received: {'GroupA': {'DeviceA': {'Temp': '23.34', 'Luminosity': '14.00'}}}
Inserted document, response code: InsertOneResult(ObjectId('65adc5912c6ec3a23f76dfae'), acknowledged=True)
Received: {'GroupA': {'DeviceA': {'Temp': '23.37', 'Luminosity': '14.00'}}}
Inserted document, response code: InsertOneResult(ObjectId('65adc5962c6ec3a23f76dfaf'), acknowledged=True)
Received: {'GroupA': {'DeviceA': {'Temp': '23.37', 'Luminosity': '14.00'}}}
Inserted document, response code: InsertOneResult(ObjectId('65adc59b2c6ec3a23f76dfb0'), acknowledged=True)
Received: {'GroupA': {'DeviceA': {'Temp': '23.36', 'Luminosity': '14.00'}}}
Inserted document, response code: InsertOneResult(ObjectId('65adc5a12c6ec3a23f76dfb1'), acknowledged=True)
```

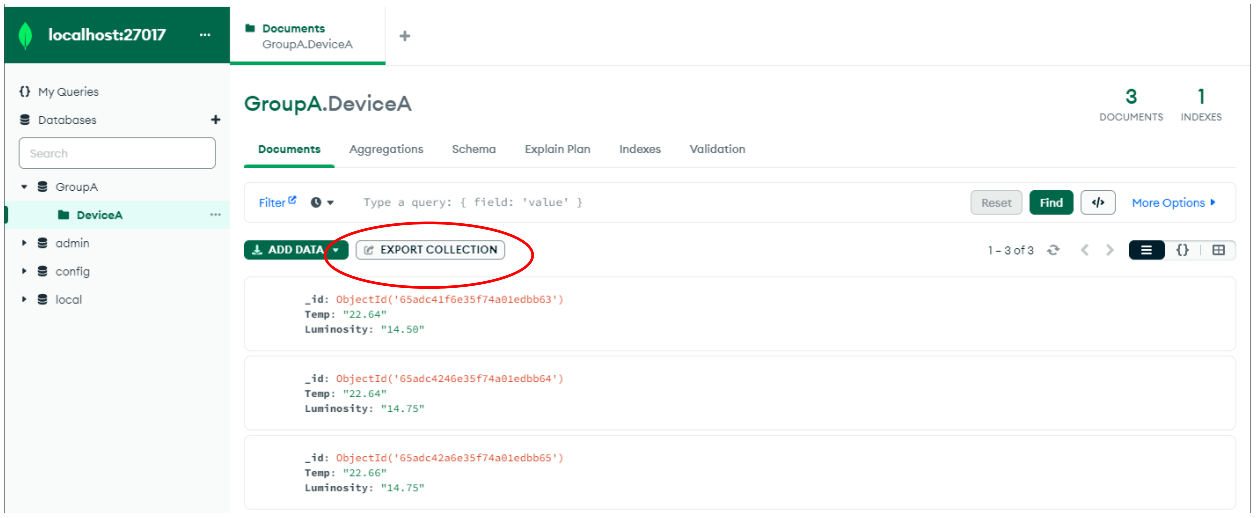
8. Reload your MongoDB Compass client
9.



10. Observe that a new database has been generated. Open this database



11. You should see all of your data being inserted



```
_id: ObjectId('65adc41f6e35f74a01edbb63')
Temp: "22.64"
Luminosity: "14.50"

_id: ObjectId('65adc4246e35f74a01edbb64')
Temp: "22.64"
Luminosity: "14.75"

_id: ObjectId('65adc42a6e35f74a01edbb65')
Temp: "22.66"
Luminosity: "14.75"
```

12. Collect sensor data for 2 minutes then export the dataset from the database (Export Collection, above in red). Using excel or another spreadsheet software, graph the data (and label the axes) and identify any temperature and light intensity trends. Include the graphs with your lab submission.

Milestone 3: Instructor/TA to verify sensor output appearing on VSCode, MongoDB, and Graphs

Lab Report Q3 – Screenshot of the Graphs

3.7. Controlling MacIoT with Bluetooth [25 marks]

Until now we have been receiving messages sent from the MacIoT. In this milestone, you will send messages to the MacIoT to control the LED.

3.7.1. Python LED Control

1. Launch the Arduino IDE and File → New Sketch.
2. File → Save As → Go to your Bluetooth folder in Repositories and save as bluetooth_c (this will create an Arduino folder (called a sketchbook) called bluetooth_c and an Arduino file inside bluetooth_c also called bluetooth_c.ino
3. Close any other Arduino sketch windows that may be open
4. Copy the **MCP23017.h** and **MCP23017.cpp** files for the **MCP23017** library from avenue to the Bluetooth_c folder
5. Add all team members' names and student numbers (one line per student) as comments (%)
6. After team members' names and student numbers, add the following lines : #include "bluetooth_c.h"
7. Initialize the MCP23017 I2C IO expansion IC for PORT A to be set as output

File: bluetooth_c.ino

```
#include "bluetooth_c.h"
void setup() {
    Serial.begin(9600);
    Serial.print("\n\n------------------------\n"
      + group_name + " : " + device_name + "\n------------------------\n\n");

    bluetooth_serial.begin(group_name + " : " + device_name);
    bluetooth_serial.flush();
    Serial.println("Ready for bluetooth connection!");

    //  Initializing I2C interface
    Wire.begin();
    //  Initializing IO port for MCP23017 IO expansion bus
    mcp.init();
    mcp.portMode(MCP23017Port::A, 0); // Configuring port A as OUTPUT
    mcp.writeRegister(MCP23017Register::GPIO_A, 0x00);  //Resetting port A
}
```

8. In the void loop() function, we will be polling the serial buffer for character bytes (char_byte) and concatenating them to a String variable called command. Once we see a newline (‘\n’) character in the register, we know this is the end of the command.

File: bluetooth_c.ino

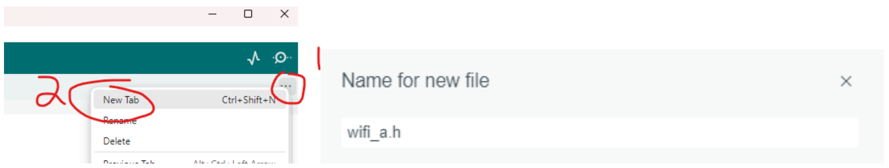
```
void loop() {
    String command = "";
    bool ret = 0;
    char char_byte = ' ';
    while (bluetooth_serial.available()){
      command = "";
      ret = 0;
      char_byte = ' ';
      while(char_byte != '\n'){
        char_byte= bluetooth_serial.read();
        command += String(char_byte);
     }
    bluetooth_serial.flush();
    Serial.println(command);
    // Evaluate command here
    }
}
```

9. Once we see the newline termination character, we can evaluate the command using a series of conditional statements. We then reset the command string and restart this process.

File: blueotooth_c.ino

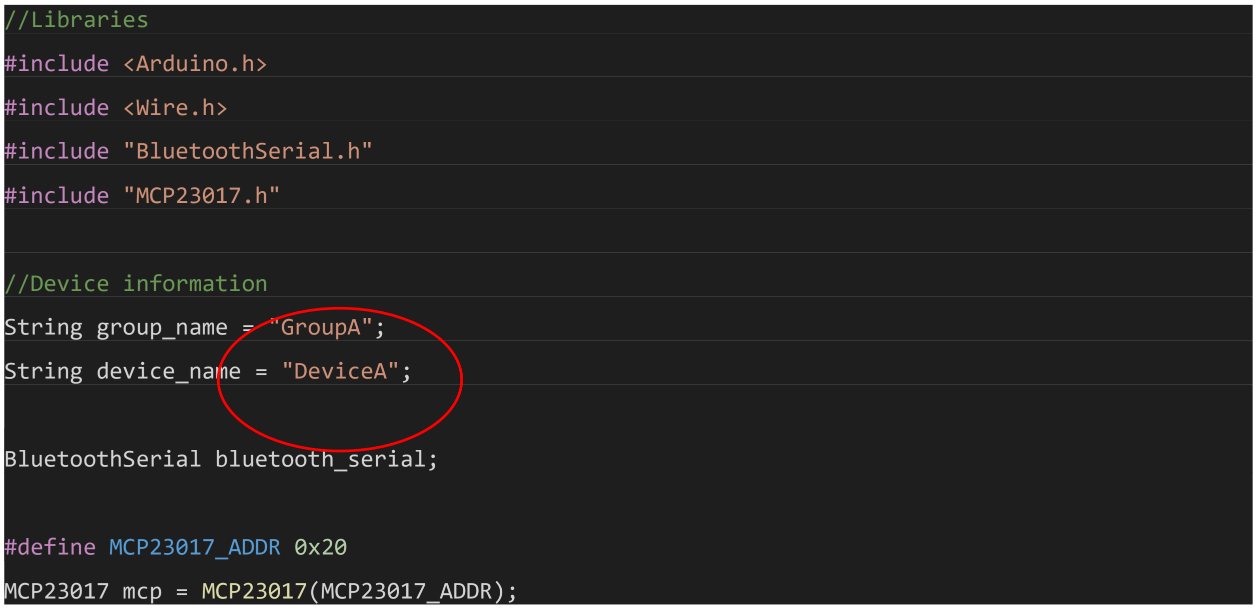
```
    if(command == "LED_ON\n"){
        Serial.println("Value of 1 written to LED's 1 - 7");
        mcp.digitalWrite(0, 1);
     }
    else if(command == "LED_OFF\n"){
        Serial.println("Value of 0 written to LED's 1 - 7");
        mcp.digitalWrite(0, 0);
    }
  }
```

10. Create a new header file by clicking the three dots and selecting the new tab in the dropdown menu. Name this file **bluetooth_c.h**. It will be saved to the same folder as your Arduino .ino file.



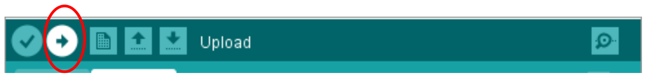
11. Set up the Bluetooth_c.h header file

File: bluetooth_c.h

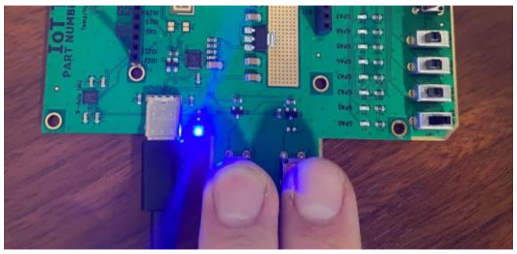


```
//Libraries
#include <Arduino.h>
#include <Wire.h>
#include "BluetoothSerial.h"
#include "MCP23017.h"

//Device information
String group_name = "GroupA";
String device_name = "DeviceA";

BluetoothSerial bluetooth_serial;

#define MCP23017_ADDR 0x20
MCP23017 mcp = MCP23017(MCP23017_ADDR);
```

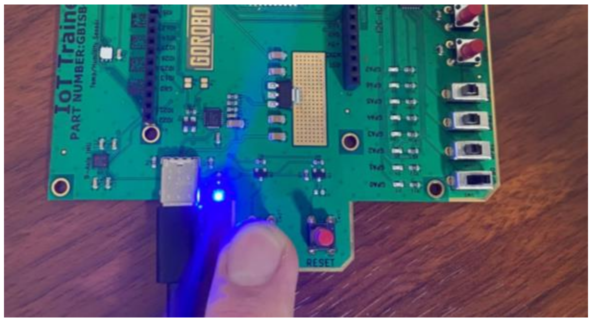
12. Press the verify button to compile the code. Keep an eye on the terminal window for any errors that arise.



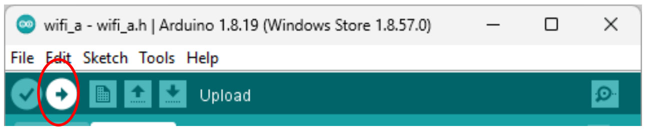
13. Uploading to the MacIoT board using the following steps:
    a. Hold down **both** the **FLASH** and **RESET** buttons



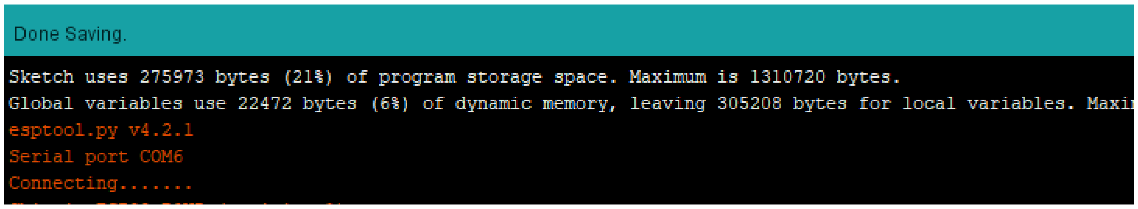
b. After 2 seconds, **release** the **RESET** button. Continue to hold the **FLASH** button



c. Press the upload button to compile and upload the code. Once the code finishes compiling and begins to upload, **release** the **FLASH** button.
   i. Keep an eye on the terminal window for any errors that arise. *Sketch > Upload*

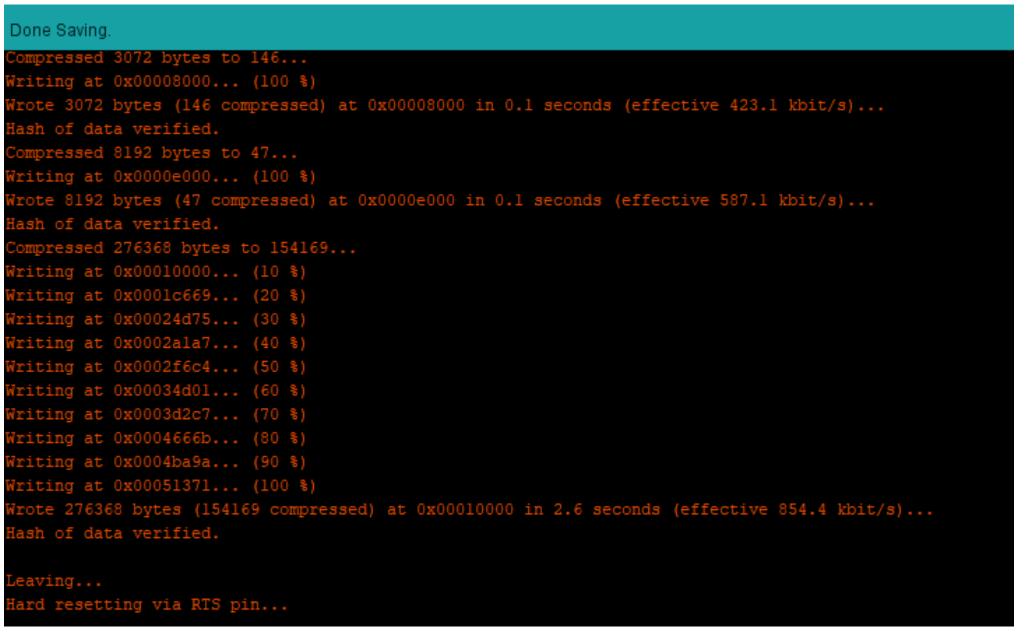

wifi_a - wifi_a.h | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
Upload

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



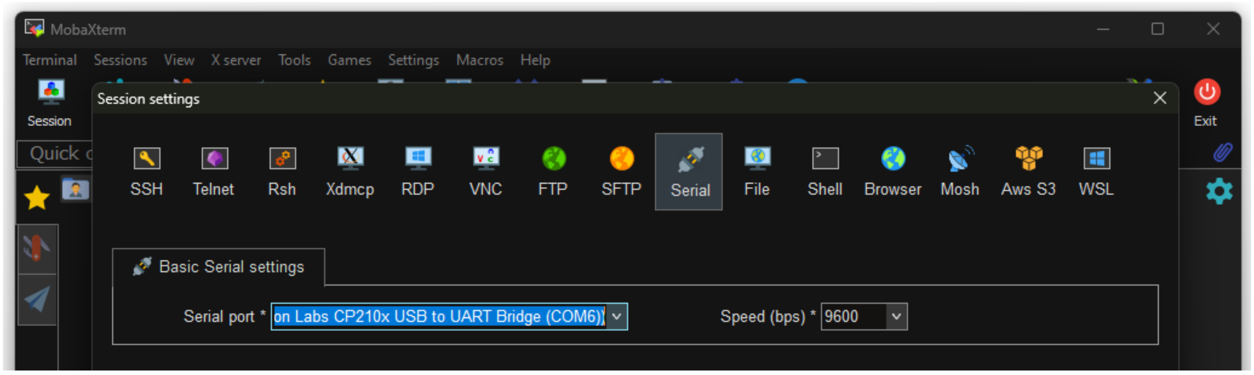
```
Done Saving.
Sketch uses 275973 bytes (21%) of program storage space. Maximum is 1310720 bytes.
Global variables use 22472 bytes (6%) of dynamic memory, leaving 305208 bytes for local variables. Maxi
esptool.py v4.2.1
Serial port COM6
Connecting.......
```

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



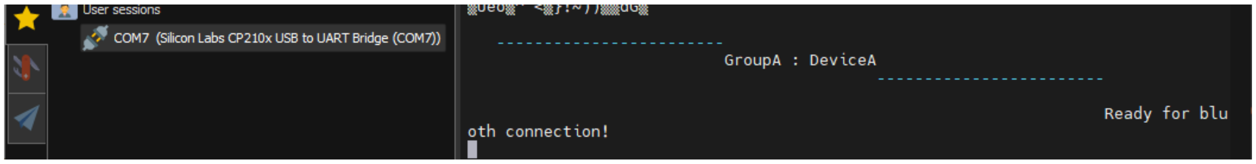
```
Done Saving.
Compressed 3072 bytes to 146...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (146 compressed) at 0x00008000 in 0.1 seconds (effective 423.1 kbit/s)...
Hash of data verified.
Compressed 8192 bytes to 47...
Writing at 0x0000e000... (100 %)
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.1 seconds (effective 587.1 kbit/s)...
Hash of data verified.
Compressed 276368 bytes to 154169...
Writing at 0x00010000... (10 %)
Writing at 0x0001c669... (20 %)
Writing at 0x00024d75... (30 %)
Writing at 0x0002a1a7... (40 %)
Writing at 0x0002f6c4... (50 %)
Writing at 0x00034d01... (60 %)
Writing at 0x0003d2c7... (70 %)
Writing at 0x0004666b... (80 %)
Writing at 0x0004ba9a... (90 %)
Writing at 0x00051371... (100 %)
Wrote 276368 bytes (154169 compressed) at 0x00010000 in 2.6 seconds (effective 854.4 kbit/s)...
Hash of data verified.

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

14. Launch MobaXTerm and press **Session** (Under the Terminal Dropdown menu)
    - MobaXTerm is a feature-filled application for managing servers and ports. We will use it to open the Bluetooth serial port and ensure that data is being transmitted between the MacIoT board and our PC.
15. Select Serial from the top ribbon. Choose a “CP210x USB to UART Bridge” **NOT** “Bluetooth Serial Bridge”.



Serial port * on Labs CP210x USB to UART Bridge (COM6)
Speed (bps) * 9600

16. You should see “Ready for bluetooth connection” printed to your screen



```
 ------------------------
                        GroupA : DeviceA
                                       ------------------------
                                                          Ready for blu
oth connection!
```

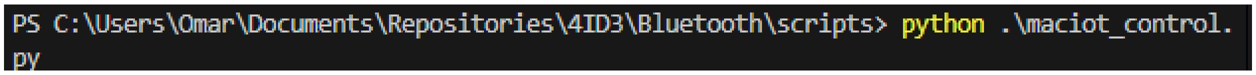
17. Navigate back to VSCode and create a new Python script in the Scripts/ folder named MacIoT_control

18. Connect to the bluetooth serial port and write ALL_ON\n or ALL_OFF\n to the bluetooth output buffer

File: maciot_control.py

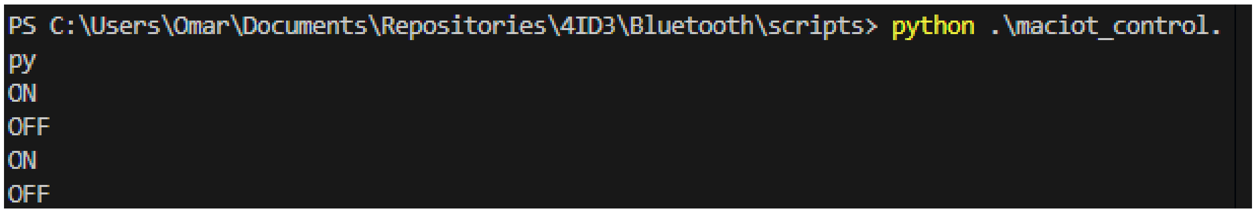
```python
import serial
import time
ser = serial.Serial("COM9", 9600)
while True:
    print("ON")
    ser.write("LED_ON\n".encode('utf-8'))
    time.sleep(5)
    print("OFF")
    ser.write("LED_OFF\n".encode('utf-8'))
    time.sleep(5)
```

19. Run the python code from VSCode Terminal

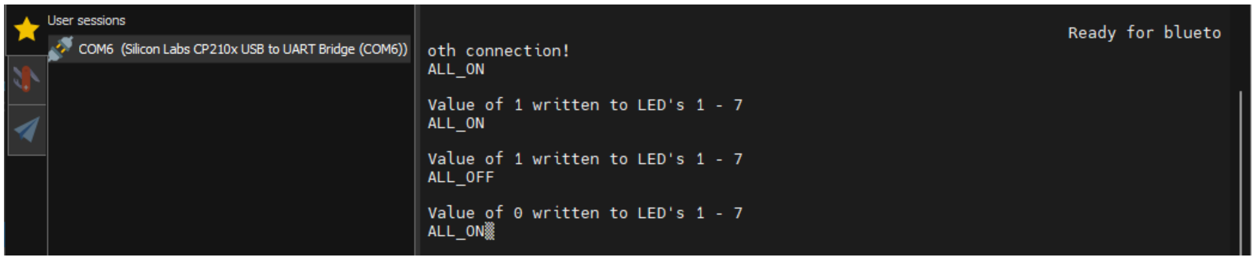


```
PS C:\Users\Omar\Documents\Repositories\4ID3\Bluetooth\scripts> python .\maciot_control.
py
```

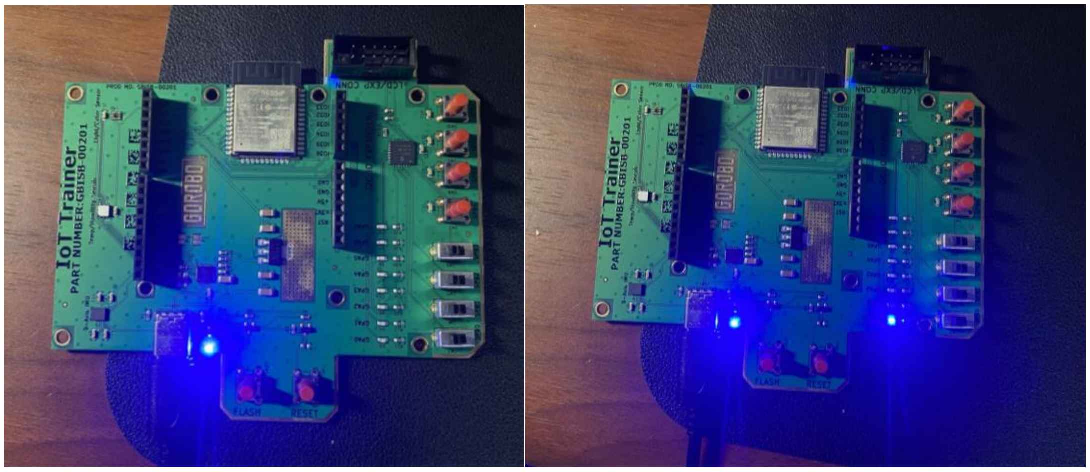
20. The VSCode and MobaXTerm and the MacIoT will output the LED turning ON and OFF



```
PS C:\Users\Omar\Documents\Repositories\4ID3\Bluetooth\scripts> python .\maciot_control.
py
ON
OFF
ON
OFF
```



```
                                                    Ready for blueto
oth connection!
ALL_ON

Value of 1 written to LED's 1 - 7
ALL_ON

Value of 1 written to LED's 1 - 7
ALL_OFF

Value of 0 written to LED's 1 - 7
ALL_ON
```



Milestone 4: Instructor/TA to verify MacIoT control using Python

Lab Report Q4 – Screenshot of Python Output and Pictures of MacIoT turning on and off (two pictures)

21. 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 Q5 – 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.

3.7.2. MQTTool LED Control [Optional]

Now that the connection between your PC and the MacIoT board is established and tested, we can link your PC to MQTT.

1. Navigate back to VSCode and create a new Python script in the Scripts/ folder named qtt2btserial.py
2. Add the following libraries

File: mqtt2btserial.py

```python
import serial
import json
import paho.mqtt.client as mqtt
import time
```

3. Set up MQTT information, use borker.hivemq.com for mqtt_ip and update Bluetooth_com_port

File: mqtt2btserial.py

```python
#   Configuring MQTT server and COM
mqtt_ip = None
mqtt_port = None
bluetooth_com_port = None
mqtt_topic = None

mqtt_ip = input("MQTT Broker IP: ")
if(mqtt_ip == None or mqtt_ip == ''): mqtt_ip = 'test.mosquitto.org'

mqtt_port = input("MQTT Broker Port: ")
if(mqtt_port == None or mqtt_port== ''): mqtt_port = 1883

bluetooth_com_port = input("Bluetooth COM (e.g. COM9): ")
if(bluetooth_com_port == None or bluetooth_com_port  == ''): bluetooth_com_port  = 'COM9'

mqtt_topic= input("MQTT Topic to Subscribe to: ")
if(mqtt_topic == None or mqtt_topic== ''): mqtt_topic = "GroupA/DeviceA/LED"

print(f'\n-------\nCONFIGURATION\n-------\nIP: {mqtt_ip}\nPORT: {mqtt_port}\nBT COM:
{bluetooth_com_port}')
```

4. Configure your Bluetooth communication

File: mqtt2btserial.py

```python
#   Connecting to serial
print("Connecting to serial: " + bluetooth_com_port)
time.sleep(1)
ser = serial.Serial(bluetooth_com_port, 9600)
print("Bluetooth COM opened");
```

5. Set up your callback functions for your MQTT client.

File: mqtt2btserial.py

```python
#   MQTT callback functions
def on_connect(client, userdata, flags, rc):
    print(f"Connected, status = {str(rc)}")
def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))
    if msg.topic == mqtt_topic:
        print("Writing to serial...")
        ser.write(f"{str(msg.payload.decode())}\n".encode('utf-8'))
        print("Written.")
def on_subscribe(mosq, obj, mid, granted_qos):
    print("Subscribed: " + str(mid) + " " + str(granted_qos))
```

6. Initialize your MQTT client and link them to the callback functions you have written

File: mqtt2btserial.py

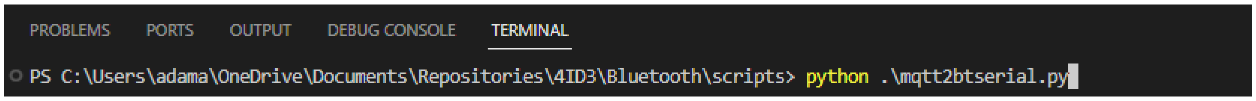
```python
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.on_subscribe = on_subscribe
```

7. In an infinite loop, check if the client is connected. If it is not, reconnect your client and subscribe to your chosen topic. Refresh your connection using the **client.loop()** method each loop iteration.

File: mqtt2btserial.py

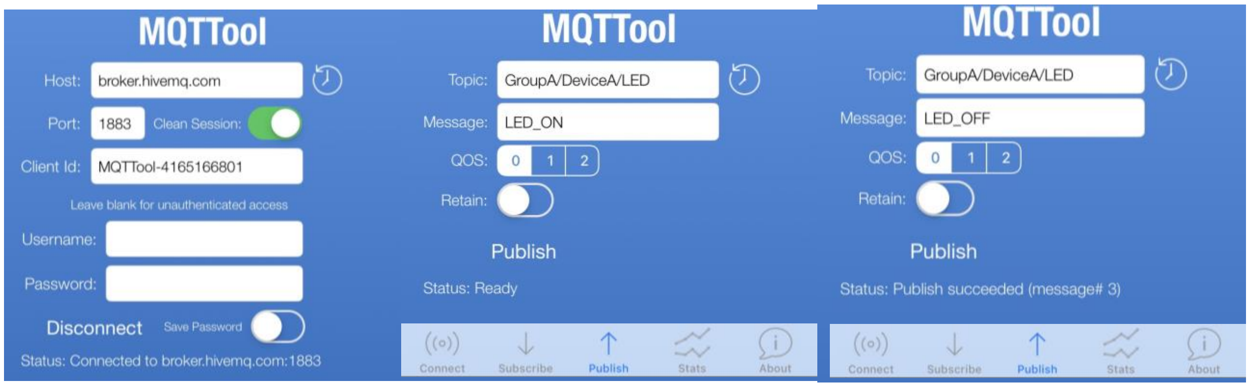
```
while True:
    #   Refreshing the client connection
    if(client.is_connected() == False):
        client.connect(mqtt_ip, mqtt_port, 60)
        client.subscribe(mqtt_topic, 0)
    client.loop()


ser.close()
```

8. Run the script to link MQTT to your Bluetooth Serial port



```
PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Bluetooth\scripts> python .\mqtt2btserial.py
```

9. Use an app to publish your chosen topic; in my case, it is **GroupA/DeviceA/LED.** My implemented commands are:
    a. LED_ON
    b. LED_OFF

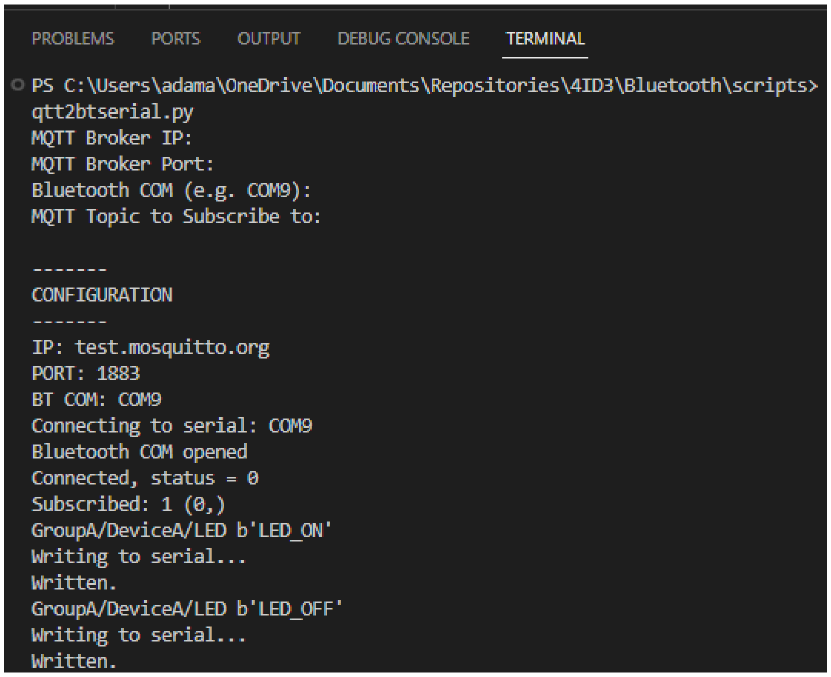


Host: broker.hivemq.com
Port: 1883 Clean Session: on
Client Id: MQTTool-4165166801
Leave blank for unauthenticated access
Username:
Password:
Save Password: off
Status: Connected to broker.hivemq.com:1883

Topic: GroupA/DeviceA/LED
Message: LED_ON
QOS: 0 1 2
Retain: off
Status: Ready

Topic: GroupA/DeviceA/LED
Message: LED_OFF
QOS: 0 1 2
Retain: off
Status: Publish succeeded (message# 3)

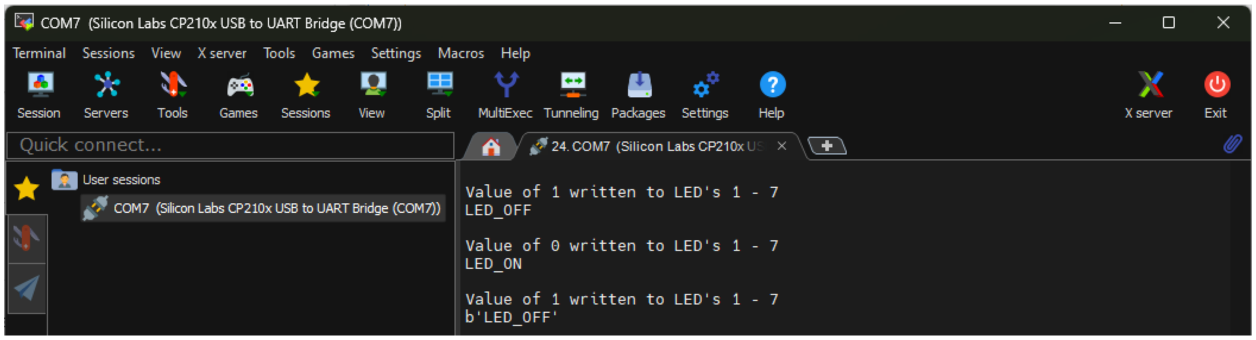
10. Watch it populate on your python mqtt client, MobaXTerm, and your MacIoT device



```
PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Bluetooth\scripts>
qtt2btserial.py
MQTT Broker IP:
MQTT Broker Port:
Bluetooth COM (e.g. COM9):
MQTT Topic to Subscribe to:

-------
CONFIGURATION
-------
IP: test.mosquitto.org
PORT: 1883
BT COM: COM9
Connecting to serial: COM9
Bluetooth COM opened
Connected, status = 0
Subscribed: 1 (0,)
GroupA/DeviceA/LED b'LED_ON'
Writing to serial...
Written.
GroupA/DeviceA/LED b'LED_OFF'
Writing to serial...
Written.
```

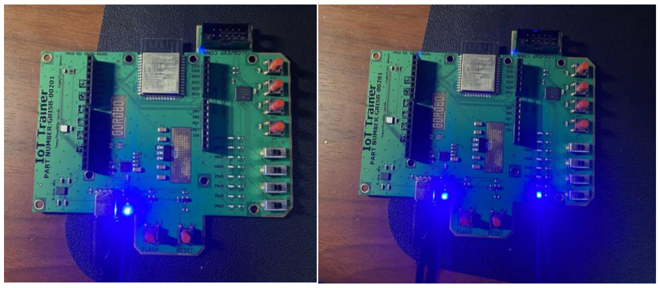
The MobaXTerm will show the messages being echoed back

NOTE: THIS IS NOT THE BLUETOOTH COM PORT. IT IS BEING SENT USING BLUETOOTH THEN RETRANSMITTED TO SERIAL MONITOR. YOU ARE VIEWING THE RETRANSMITTED VERSION.



```
Value of 1 written to LED's 1 - 7
LED_OFF

Value of 0 written to LED's 1 - 7
LED_ON

Value of 1 written to LED's 1 - 7
b'LED_OFF'
```

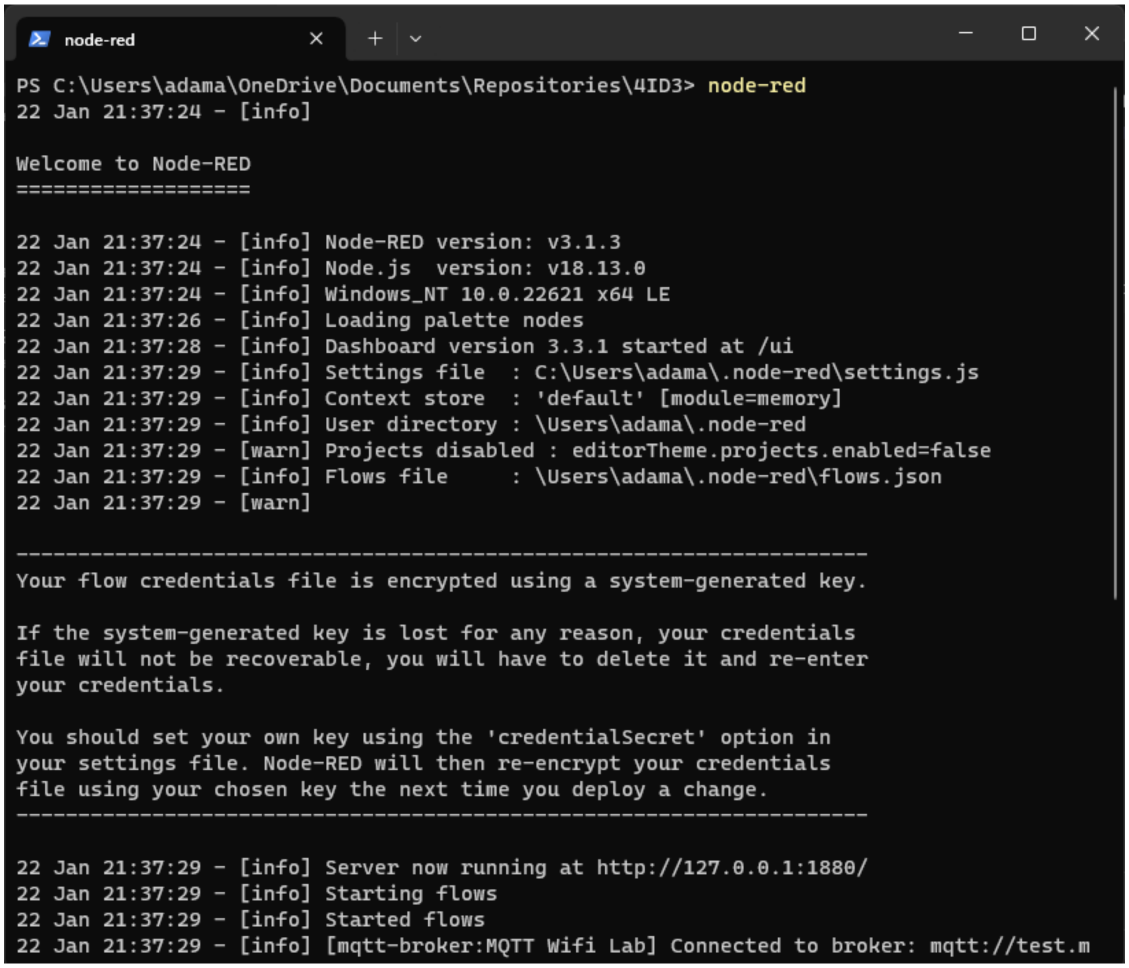
And finally, on your device



3.7.3. NodeRED LED Control [Optional]

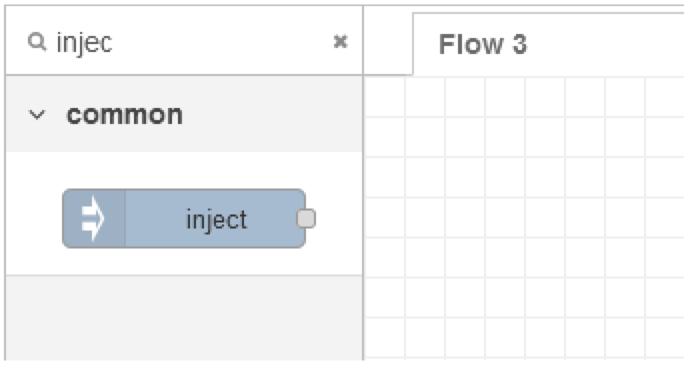
We can also control the MacIoT LED using NodeRED.

1. Launch node-red. Note: If you get an error that the PORT 1880 is not available try opening your browser to http:127.0.0.1:1880/ (you may have node-red already running)

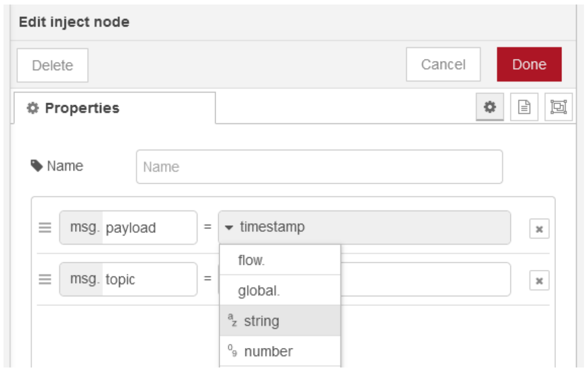


```
PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3> node-red
22 Jan 21:37:24 - [info]

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

22 Jan 21:37:24 - [info] Node-RED version: v3.1.3
22 Jan 21:37:24 - [info] Node.js  version: v18.13.0
22 Jan 21:37:24 - [info] Windows_NT 10.0.22621 x64 LE
22 Jan 21:37:26 - [info] Loading palette nodes
22 Jan 21:37:28 - [info] Dashboard version 3.3.1 started at /ui
22 Jan 21:37:29 - [info] Settings file  : C:\Users\adama\.node-red\settings.js
22 Jan 21:37:29 - [info] Context store  : 'default' [module=memory]
22 Jan 21:37:29 - [info] User directory : \Users\adama\.node-red
22 Jan 21:37:29 - [warn] Projects disabled : editorTheme.projects.enabled=false
22 Jan 21:37:29 - [info] Flows file     : \Users\adama\.node-red\flows.json
22 Jan 21:37:29 - [warn]

---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
---------------------------------------------------------------------

22 Jan 21:37:29 - [info] Server now running at http://127.0.0.1:1880/
22 Jan 21:37:29 - [info] Starting flows
22 Jan 21:37:29 - [info] Started flows
22 Jan 21:37:29 - [info] [mqtt-broker:MQTT Wifi Lab] Connected to broker: mqtt://test.m
```

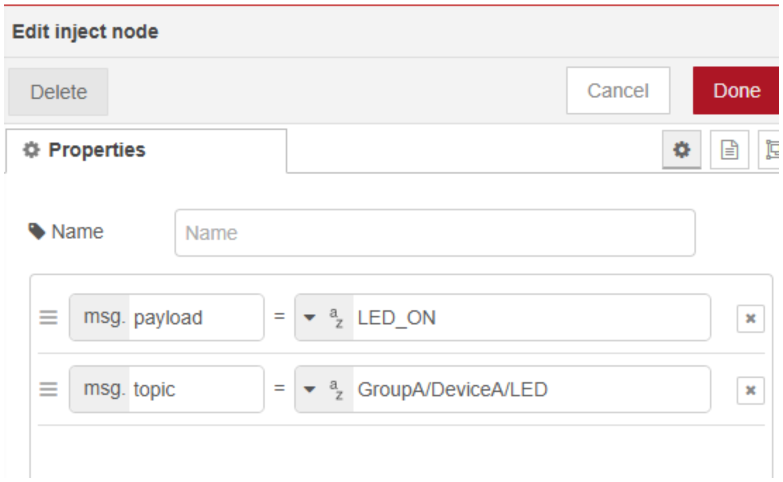
1. Filter nodes to find the **inject** node and drag two into your flow.



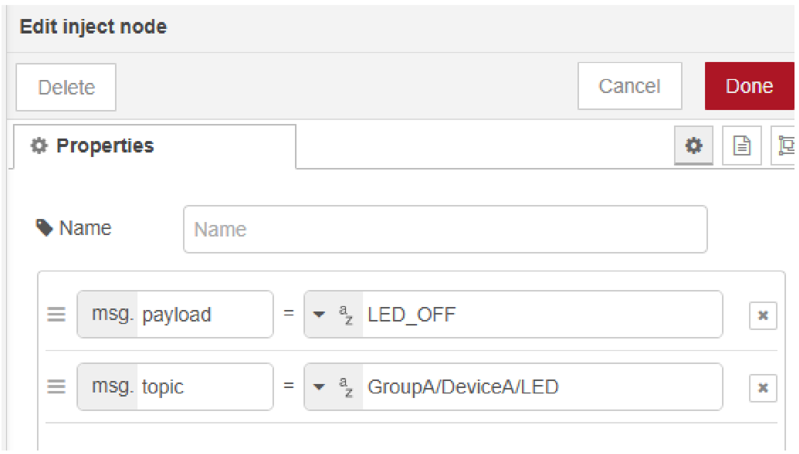
2. In your inject node, change your message.payload to a string type using the dropdown menu



3. Fill in the information for the inject node: update GroupA with group number

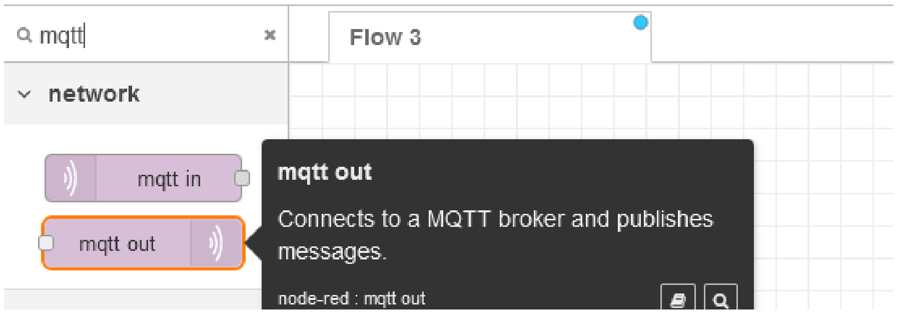


msg. payload = LED_ON
msg. topic = GroupA/DeviceA/LED



msg. payload = LED_OFF
msg. topic = GroupA/DeviceA/LED

4. Search for an mqtt out node

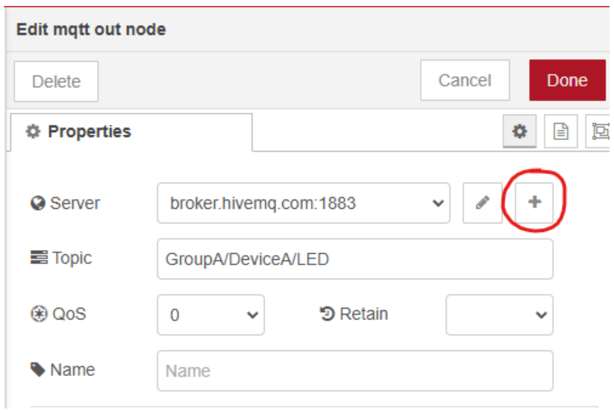


mqtt out

Connects to a MQTT broker and publishes messages.

node-red : mqtt out

5. In your mqtt out node, click plus (+) to add new config file



Server: broker.hivemq.com:1883

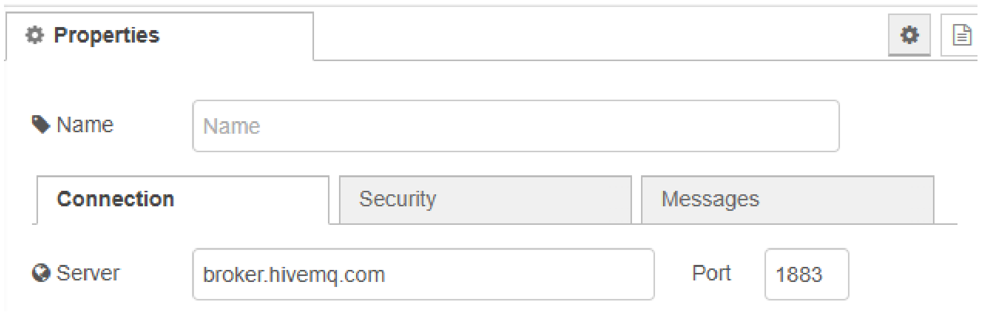
Topic: GroupA/DeviceA/LED

QoS: 0

Retain:

Name: Name

6. Fill out for broker.hivemq.com and click add (top right corner)

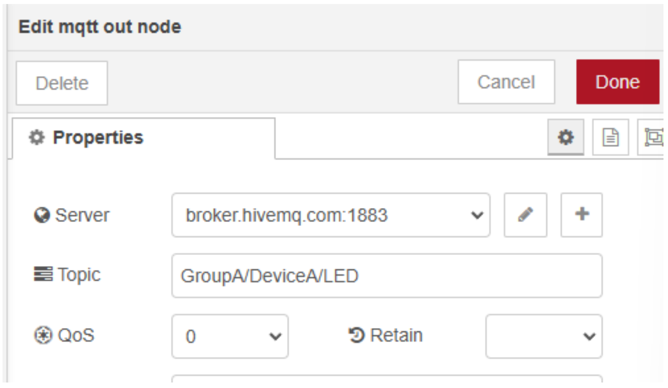


Name: Name

Server: broker.hivemq.com

Port: 1883

1. Under Properties, fill in the Topic field.



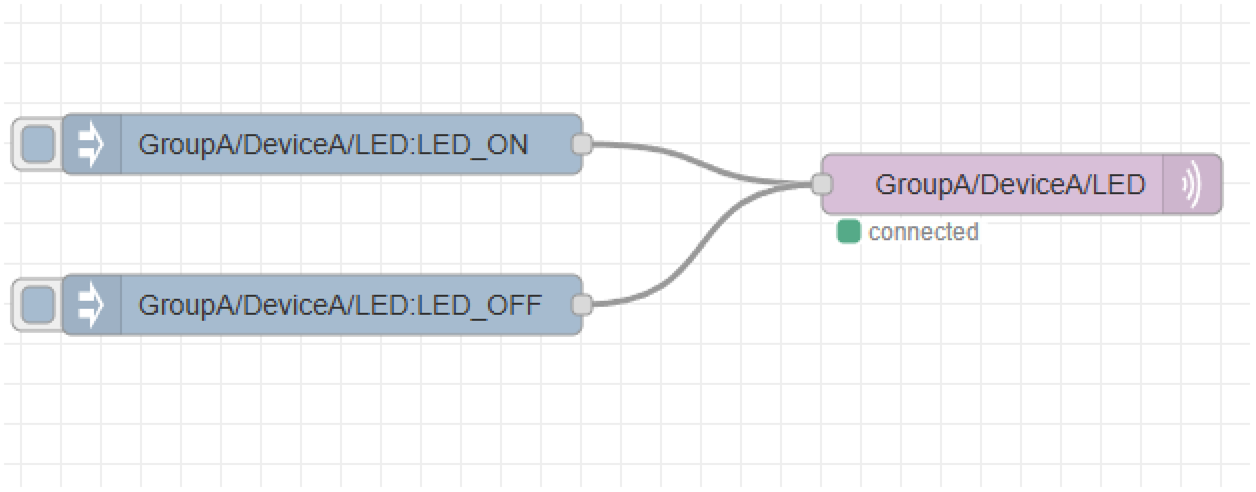
Server: broker.hivemq.com:1883

Topic: GroupA/DeviceA/LED

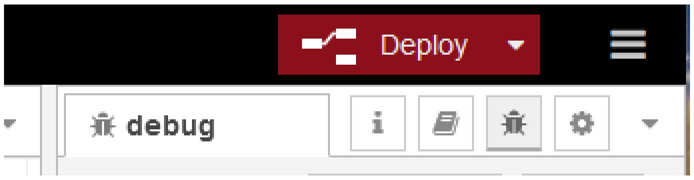
QoS: 0

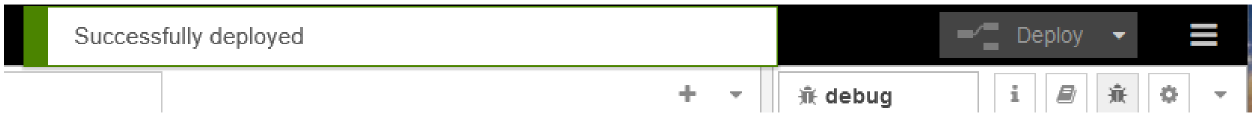
Retain:

7. Connect everything like this:

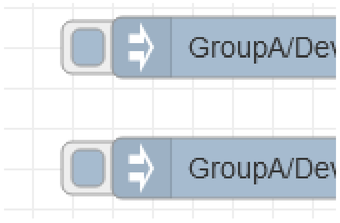


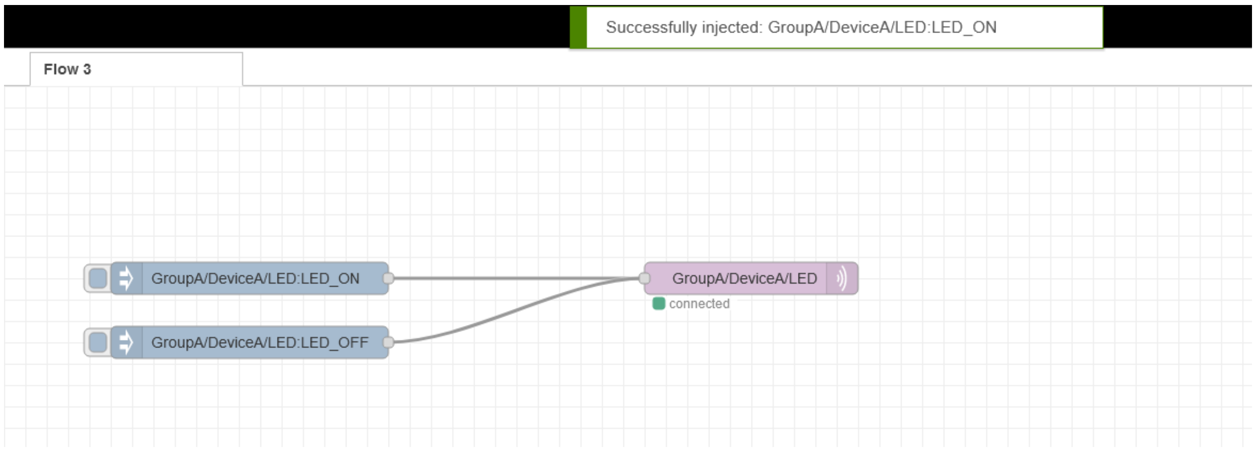
8. Press Deploy.



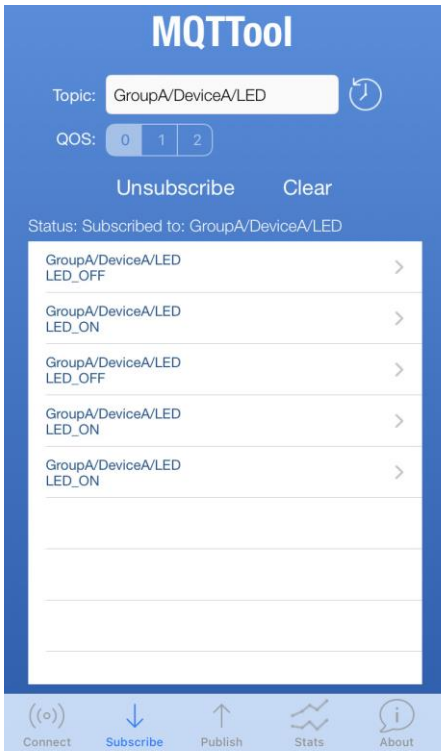

Successfully deployed

9. Click the inject buttons one at a time to publish commands ON or OFF




Successfully injected: GroupA/DeviceA/LED:LED_ON

10. Use any MQTT app to subscribe to your chosen topic to verify the results are being published and check your MacIoT board response


MQTTool

Topic: GroupA/DeviceA/LED

QOS: 0 1 2

Status: Subscribed to: GroupA/DeviceA/LED

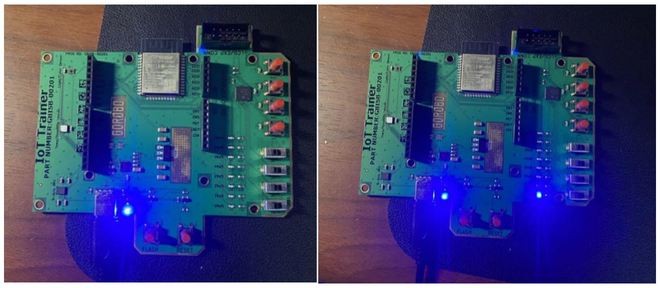
GroupA/DeviceA/LED
LED_OFF

GroupA/DeviceA/LED
LED_ON

GroupA/DeviceA/LED
LED_OFF

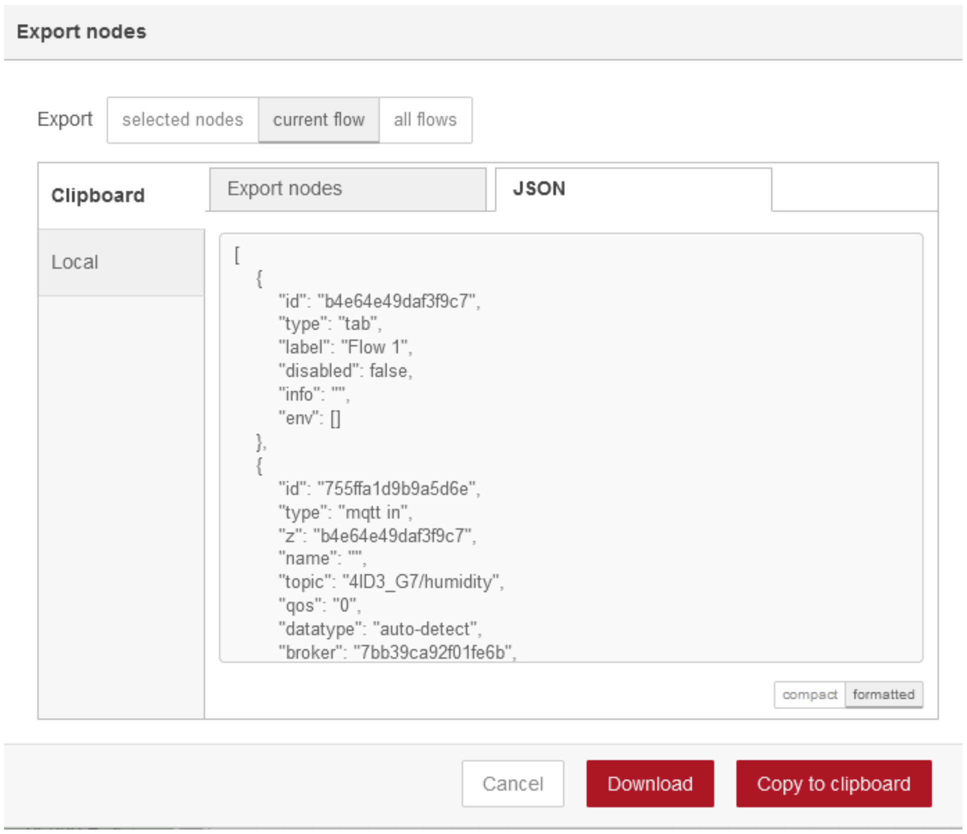
GroupA/DeviceA/LED
LED_ON

GroupA/DeviceA/LED
LED_ON



11. Export your NodeRED Flow as JSON

  - Click on the hamburger menu and select export.
  - Select current flow.
  - Select JSON.
  - Press Download
  - Copy the flows.json file from your Downloads/ folder to the Bluetooth/ 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",
```

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: Screenshot of VSCode Serial Monitor, MQTTool Serial Output, and NodeRed

(Suggested: Paragraph or screenshot, 1 points)

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

(Suggested: Paragraph or screenshot, 1 point)

Q3 – Milestone 3: Screenshot of Labelled Graphs

(Suggested: Paragraph or screenshot, 1 points)

Q4 – Milestone 4: Screenshot of Python Output and Pictures of MacIoT turning on and off (two pictures)

(Suggested: Paragraph or screenshot, 1 points)

Q5 – 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.

Q6 – Complete Mandatory Avenue Safety Quiz

(Suggested: Recorded on Avenue, 0 points)

Q7 – What network topology would best describe the IoT network built in this lab? Explain with a block diagram that is drawn with PowerPoint. Include labels for each node and connection in the network

(Suggested: Diagram, 5 points)

Q8 – If the microcontroller is transmitting data to the PC, but the Python script is not running, how does this impact what data is accessible through MQTT?

(Suggested: 3 sentences, 2 points)

Q9—What is the theoretical range of Bluetooth? How might this impact its use in IoT networks? Explain using an example.

(Suggested: 3 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
4. Milestone 4

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	15%
Milestone 2	10%
Milestone 3	25%
Milestone 4	25%
Lab Report	15%
Total	100%

END