IOT Personal Portfolio

IOT Block 3

Luthando Mashigo

Week 1

Day 1

07/11

Who am I
Name: Luthando Mashigo

Age: 21

Bio : I am from South Africa , I study Bachelors of Computing at Belgium Campus ITVersity, I major in Software Engineering. I enjoy stuff such as making music and gaming, I occasionally code for fun but not extensively. I have programming experience in C# , C++ (IOT Perspective) , Java , Python , HTML , Javscript. Framworks : Springboot , NodeJS , ExpressJS , ReactJS and VueJS.

What is IOT?

The internet of things is a technology that allows us to add a device to an inert object (for example: vehicles, plant electronic systems, roofs, lighting, etc.) that can measure environmental parameters, generate associated data and transmit them through a communications network.

ESP8266 and Coffee

The esp8266 is quite a versatile device and it is quite cheap in both manufacturing and in price. The connection that is had , has to do with espresso as there was a micro controller could the espresso lite which was a very lightweight controller.

Reflection

Entered the IOT Classroom and introduced each other to the Lecturer. Learnt some concepts about IOT such as edge computing, swarm and the basic definition of Internet of Things. Also, we had a look at some hardware such as the wemos d1 mini as well as a raspberry pi 400. Also, we looked at some sensors we had provide such as temperature sensors, distance sensors and so on. Lastly for the end of the day we flashed the pi 400 in other to make a Wi-Fi server/ gateway in order to connect to it like a regular router but also establish a network based on what the pi is connected to.

Day 1 Reflections

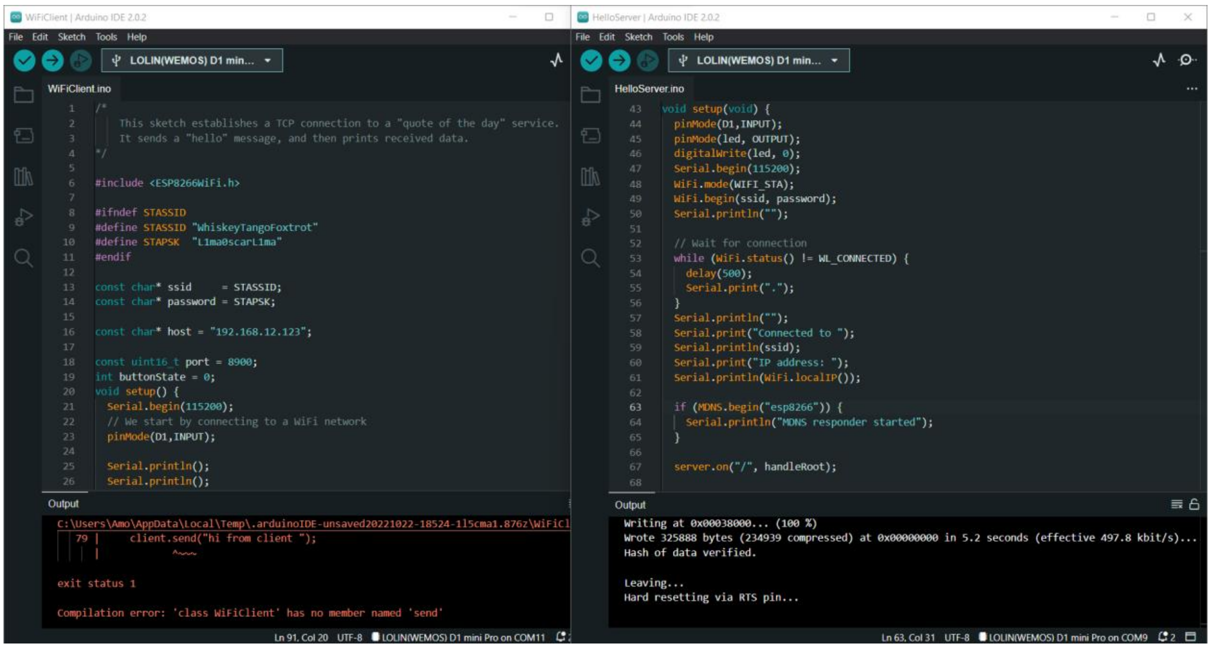
The first day was great though we did not really get our hands that dirty with the IOT devices but overall was a pretty good day. The Lecturer is very appealing and knowledgeable in this field. I feel engaged when conversation.

Day 2

08/11

Reflection

Got the led to work with the wemos d1 mini. We wrote logic to the board to switch the built in led on and off. Webserver also to be configured to be able to turn on light of the other wemos d1 mini. Struggling to get the webserver. Webserver working but light not blinking yet. Looked at documentation for any guidance. I did receive help in that regard, and we managed to get the program up and running entirely. Also, we had to make a story driven IOT scenario for our aquaponics project and not just one but two as for the project we were split into two for the purposes of paired programming. The story had to be really developed and detailed.



WiFiClient | Arduino IDE 2.0.2 — WiFiClient.ino

```
/*
    This sketch establishes a TCP connection to a "quote of the day" service.
    It sends a "hello" message, and then prints received data.
*/

#include <ESP8266WiFi.h>

#ifndef STASSID
#define STASSID "WhiskeyTangoFoxtrot"
#define STAPSK  "L1ma0scarL1ma"
#endif

const char* ssid     = STASSID;
const char* password = STAPSK;

const char* host = "192.168.12.123";

const uint16_t port = 8900;
int buttonState = 0;
void setup() {
  Serial.begin(115200);
  // We start by connecting to a WiFi network
  pinMode(D1,INPUT);

  Serial.println();
  Serial.println();
```

Output

```
C:\Users\Amo\AppData\Local\Temp\.arduinoIDE-unsaved20221022-18524-115cma1.876z\WiFiCl
   79 |     client.send("hi from client ");
      |            ^~~~

exit status 1

Compilation error: 'class WiFiClient' has no member named 'send'
```

Ln 91, Col 20   UTF-8   LOLIN(WEMOS) D1 mini Pro on COM11

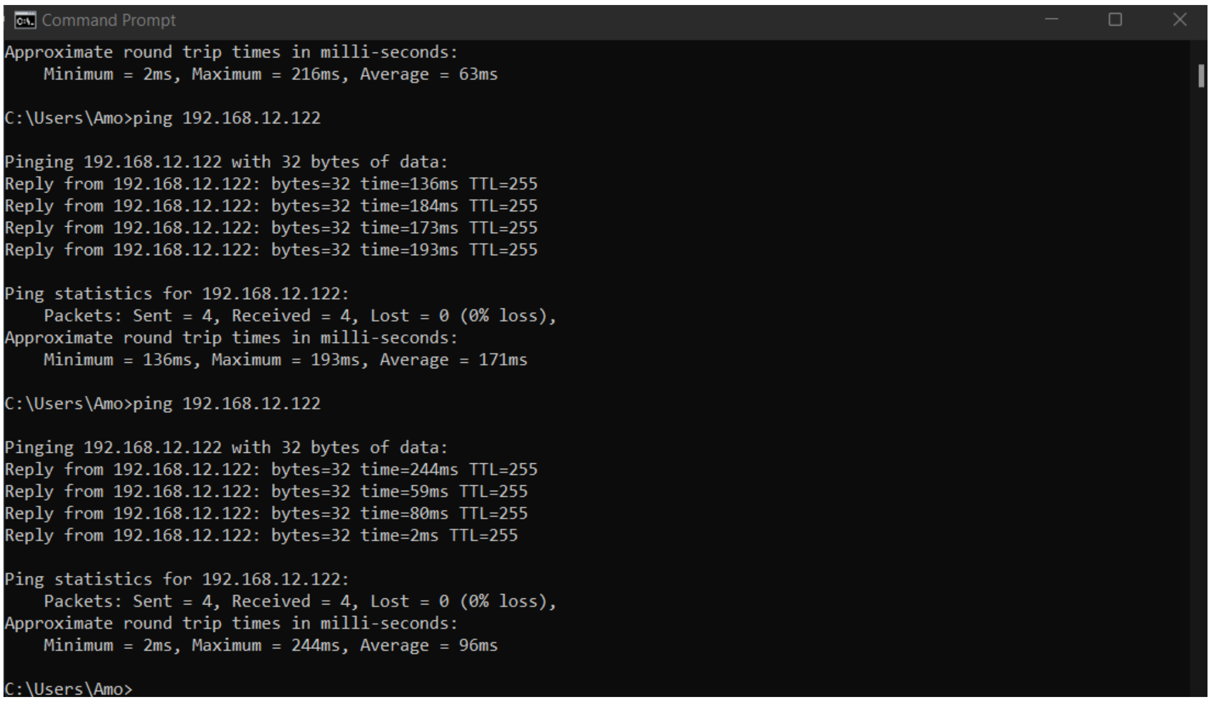
HelloServer | Arduino IDE 2.0.2 — HelloServer.ino

```
void setup(void) {
  pinMode(D1,INPUT);
  pinMode(led, OUTPUT);
  digitalWrite(led, 0);
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  if (MDNS.begin("esp8266")) {
    Serial.println("MDNS responder started");
  }

  server.on("/", handleRoot);
```

Output

```
Writing at 0x00038000... (100 %)
Wrote 325888 bytes (234939 compressed) at 0x00000000 in 5.2 seconds (effective 497.8 kbit/s)...
Hash of data verified.

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

Ln 63, Col 31   UTF-8   LOLIN(WEMOS) D1 mini Pro on COM9



Command Prompt

```
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 216ms, Average = 63ms

C:\Users\Amo>ping 192.168.12.122

Pinging 192.168.12.122 with 32 bytes of data:
Reply from 192.168.12.122: bytes=32 time=136ms TTL=255
Reply from 192.168.12.122: bytes=32 time=184ms TTL=255
Reply from 192.168.12.122: bytes=32 time=173ms TTL=255
Reply from 192.168.12.122: bytes=32 time=193ms TTL=255

Ping statistics for 192.168.12.122:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 136ms, Maximum = 193ms, Average = 171ms

C:\Users\Amo>ping 192.168.12.122

Pinging 192.168.12.122 with 32 bytes of data:
Reply from 192.168.12.122: bytes=32 time=244ms TTL=255
Reply from 192.168.12.122: bytes=32 time=59ms TTL=255
Reply from 192.168.12.122: bytes=32 time=80ms TTL=255
Reply from 192.168.12.122: bytes=32 time=2ms TTL=255

Ping statistics for 192.168.12.122:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 244ms, Average = 96ms

C:\Users\Amo>
```

We could not complete the any objectives to do with the second node because as shown in the Arduino ide screen we can send any requests to our server and do not understand why.

Day 2 Reflections

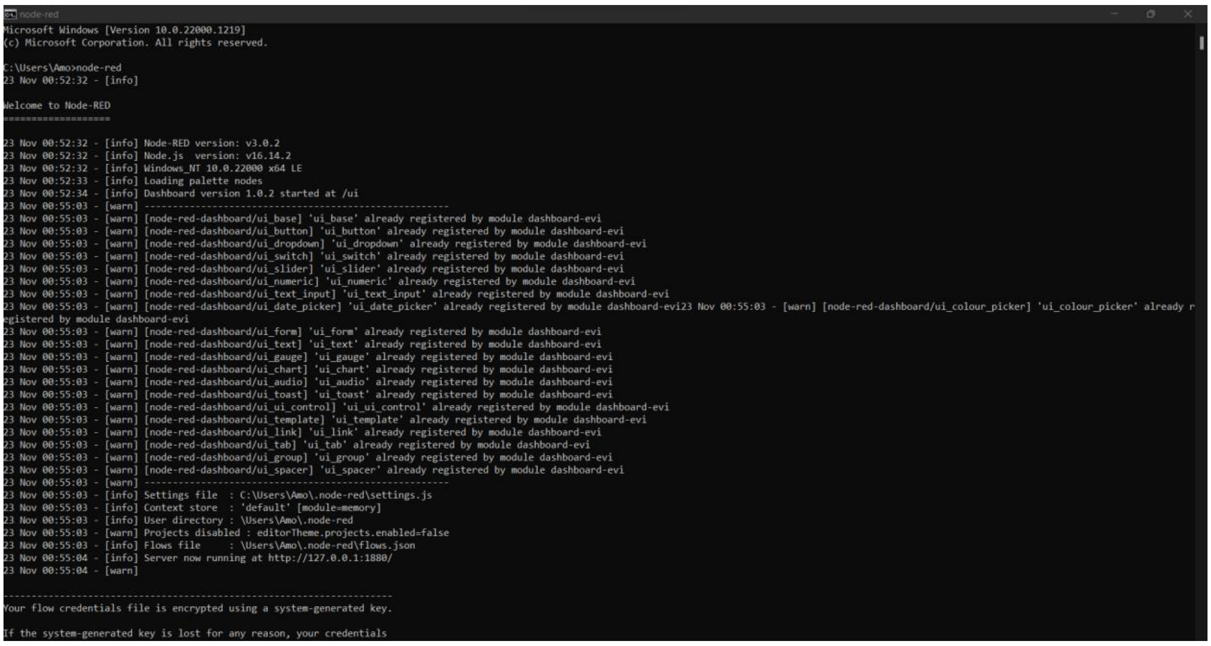
Not a bad day though we got to do a few things, but the day felt rushed. Also, not a major fan of having to take down notes and compile detailed documentation for everything I do.

Day 3

09/11

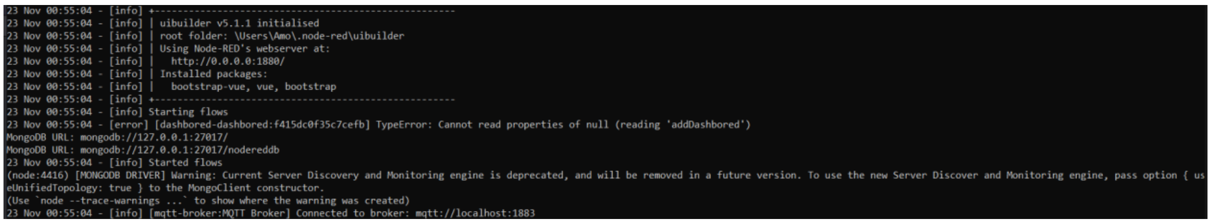
Node Red introduction

I got introduced to node red and also had to have mqtt running on my local system.



```
node-red
Microsoft Windows [Version 10.0.22000.1219]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Amo>node-red
23 Nov 00:52:32 - [info]

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

23 Nov 00:52:32 - [info] Node-RED version: v3.0.2
23 Nov 00:52:32 - [info] Node.js  version: v16.14.2
23 Nov 00:52:32 - [info] Windows_NT 10.0.22000 x64 LE
23 Nov 00:52:33 - [info] Loading palette nodes
23 Nov 00:52:34 - [info] Dashboard version 1.0.2 started at /ui
23 Nov 00:55:03 - [warn] ------------------------------------------------------
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_base] 'ui_base' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_button] 'ui_button' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_dropdown] 'ui_dropdown' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_switch] 'ui_switch' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_slider] 'ui_slider' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_numeric] 'ui_numeric' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_text_input] 'ui_text_input' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_date_picker] 'ui_date_picker' already registered by module dashboard-evi23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_colour_picker] 'ui_colour_picker' already r
egistered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_form] 'ui_form' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_text] 'ui_text' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_gauge] 'ui_gauge' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_chart] 'ui_chart' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_audio] 'ui_audio' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_toast] 'ui_toast' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_ui_control] 'ui_ui_control' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_template] 'ui_template' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_link] 'ui_link' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_tab] 'ui_tab' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_group] 'ui_group' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] [node-red-dashboard/ui_spacer] 'ui_spacer' already registered by module dashboard-evi
23 Nov 00:55:03 - [warn] ------------------------------------------------------
23 Nov 00:55:03 - [info] Settings file  : C:\Users\Amo\.node-red\settings.js
23 Nov 00:55:03 - [info] Context store  : 'default' [module=memory]
23 Nov 00:55:03 - [info] User directory : \Users\Amo\.node-red
23 Nov 00:55:03 - [warn] Projects disabled : editorTheme.projects.enabled=false
23 Nov 00:55:03 - [info] Flows file     : \Users\Amo\.node-red\flows.json
23 Nov 00:55:04 - [info] Server now running at http://127.0.0.1:1880/
23 Nov 00:55:04 - [warn]

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

If the system-generated key is lost for any reason, your credentials
```

Figure 1: Node red local instance



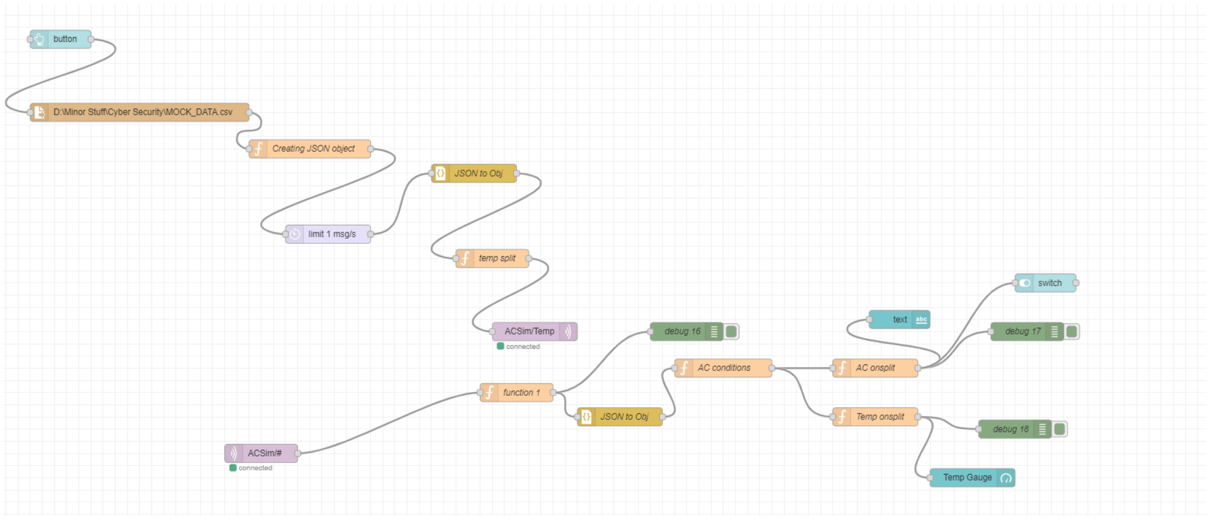
```
23 Nov 00:55:04 - [info] +-----------------------------------------------------
23 Nov 00:55:04 - [info] | uibuilder v5.1.1 initialised
23 Nov 00:55:04 - [info] | root folder: \Users\Amo\.node-red\uibuilder
23 Nov 00:55:04 - [info] | Using Node-RED's webserver at:
23 Nov 00:55:04 - [info] |   http://0.0.0.0:1880/
23 Nov 00:55:04 - [info] | Installed packages:
23 Nov 00:55:04 - [info] |   bootstrap-vue, vue, bootstrap
23 Nov 00:55:04 - [info] +-----------------------------------------------------
23 Nov 00:55:04 - [info] Starting flows
23 Nov 00:55:04 - [error] [dashbored-dashbored:f415dc0f35c7cefb] TypeError: Cannot read properties of null (reading 'addDashbored')
MongoDB URL: mongodb://127.0.0.1:27017/
MongoDB URL: mongodb://127.0.0.1:27017/nodereddb
23 Nov 00:55:04 - [info] Started flows
(node:4416) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { us
eUnifiedTopology: true } to the MongoClient constructor.
(Use `node --trace-warnings ...` to show where the warning was created)
23 Nov 00:55:04 - [info] [mqtt-broker:MQTT Broker] Connected to broker: mqtt://localhost:1883
```

Figure 2:Mqtt on local instance

Reflection

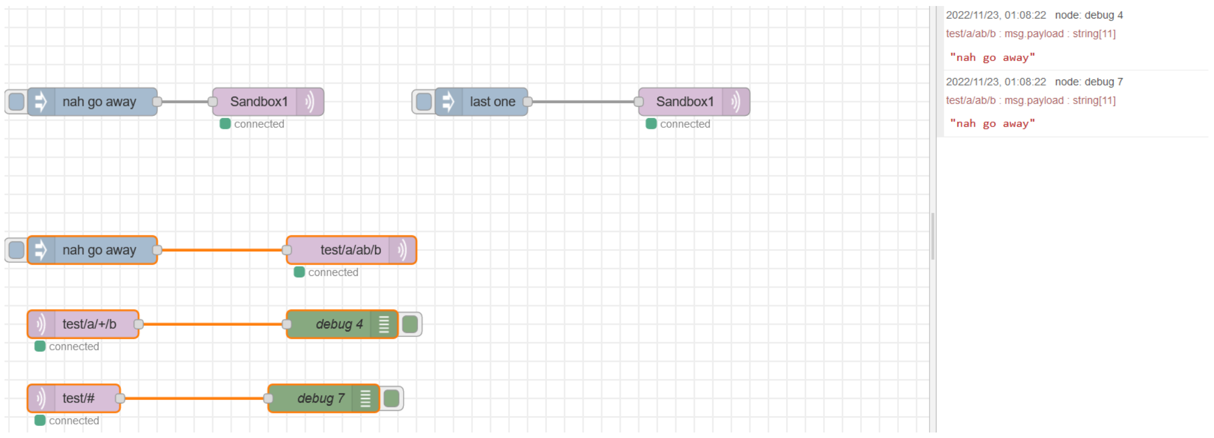
Presented the aquaponics scenario and received feedback for it. Then we continued to fumble around with IOT hardware as we began to embark on complete specific tasks such as having to configure node red via a given platform IOTempower what is remotely hosted and whereby we could collaborate on node red projects. Also, I configured and got the Arduino IDE platform to also function so the I could the flash the wemos d1 mini with specific instructions that I have coded, and it will perform those instructions. However, the IDE was configured barely on Day 2.

 A major task we had to perform in pairs was to make Air Conditioner like simulator digitally via node red and using a MQTT broker to relay messages over a network which I did manage and even did some javascript functions to perform certain decion making.



However, at a certain point we were instructed to swap out some of the digital elements such as the input (temperature sensor data) with the actual sensor itself which was a major challenge as this requires knowledge of MQTT (PubSub) but also the specific sensor you are trying to read data from. We did require help in understanding how to actual use certain libraries to have the wemos d1 mini interact with the sensor and node red via mqtt and got the program working to satisfactory level.

Also, we had to make use of mqtt wildcards within node red which is the # and the +, my input came in with finding out how the hashtag worked. An example of the output is shown in the picture below, both wildcards being used and receive data.



```
2022/11/23, 01:08:22   node: debug 4
test/a/ab/b : msg.payload : string[11]
"nah go away"
2022/11/23, 01:08:22   node: debug 7
test/a/ab/b : msg.payload : string[11]
"nah go away"
```

Really a tough day as it felt like an actual workday. Playing around with the temperature sensor was fun but then trying to integrate the hardware into the AC simulator project was really difficult and took not only research but help from others who understood the concepts a little better. Class though did get tyring with all the debugging and research, and it was quite frustrating at times when I would get errors and could not figure why, or the error was extremely unclear.

Day 4

11/11

Scaling and Testing and nice to haves

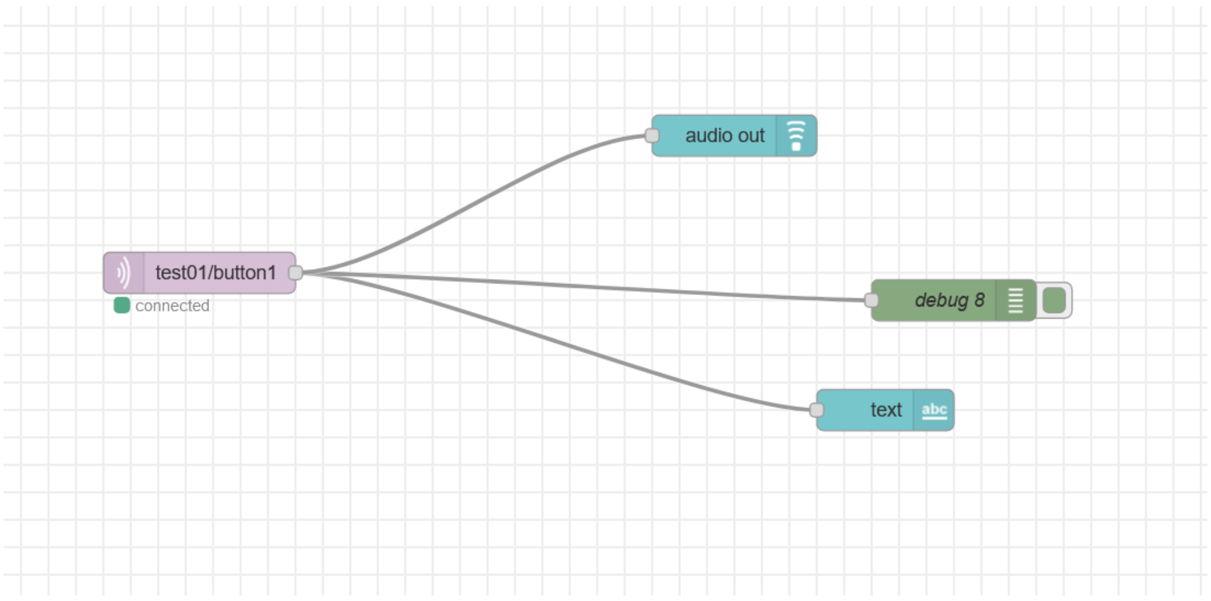
No comment on this as I did not understand how this had to be interpreted.

Features, Questions and recommendations

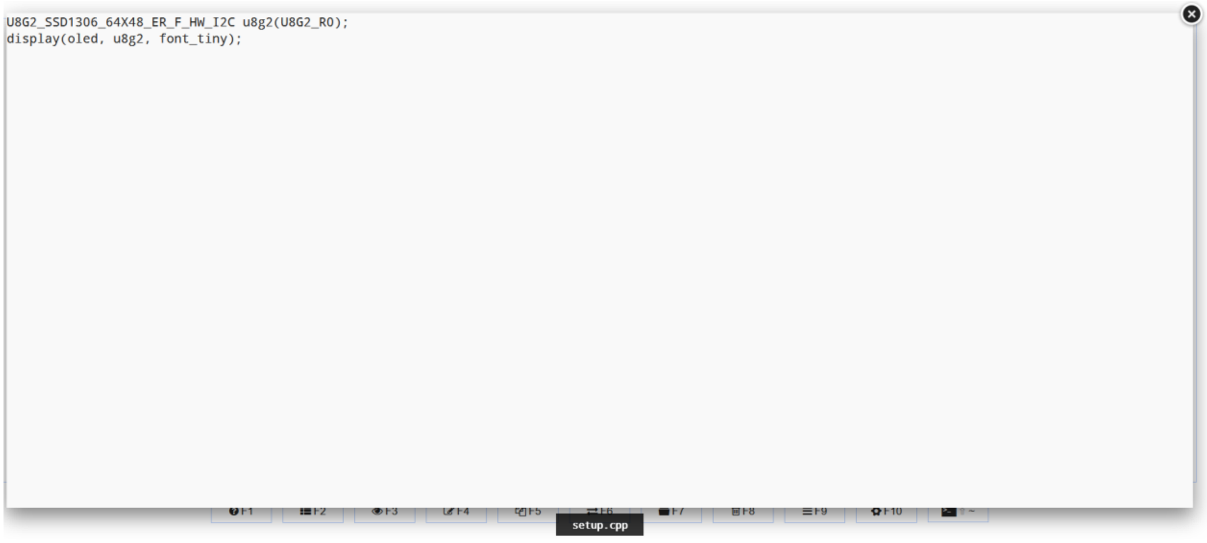
I have no feature requests nor questions nor recommendations.

Reflection

We did move on and looked to working with the other sensors such as doing a simple push button exercise for the wemos d1, though there were a couple of crashes with the program and my own Arduino Crashed and refused to work so I had to uninstall and install the latter version which ended turning out just fine. So, the thing is with the button exercise, its “sounds” simple but it was quite the challenge due to having to output a sound after the button was pressed. The thing is this had to be done via MQTT as well as we had a couple of bugs and crashes and asking for help sessions but finally on a teammate laptop, we got the program to function. My contributions had been very little during this exercise, I was more involved in building the hardware.

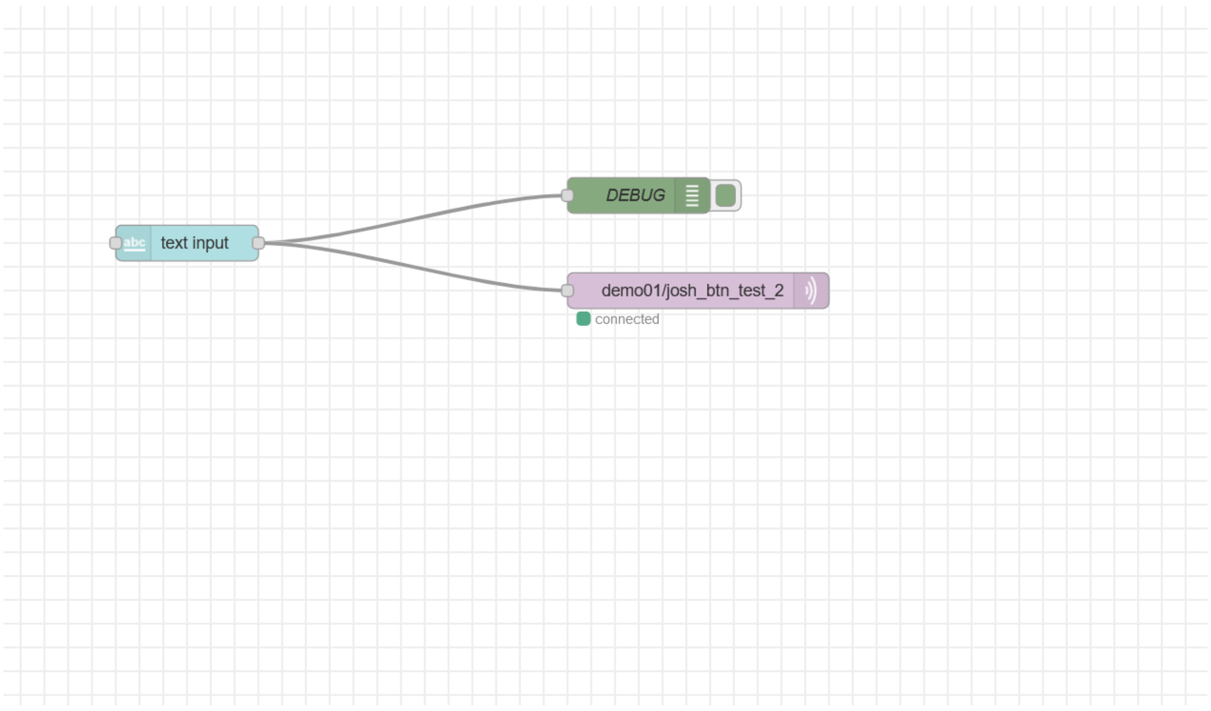


Also, we had to program a function to send input text from node red to an OLED display connected to the d1. The initial struggle was just getting the wiring done correctly as I was responsible for that and had to search up schematics of the OLED display just to understand how to have the thing wired to the d1. After that we got libraries and coded the webserver as well as the OLED display portion as that it was ready for use but no avail it did not work at all. I tried to see if it had something to do with the program, we flashed to the wemos but still nothing, we asked for help and still nothing. So, while my team/pair partner moved on to sending a message via node red via pushing a button. I tried a few more times with the OLED as we were losing time but eventually, I caved in and stopped and moved on to help my pair partner create the program for the button push notification.



```
U8G2_SSD1306_64X48_ER_F_HW_I2C u8g2(U8G2_R0);
display(oled, u8g2, font_tiny);
```

This was our cpp setup file that was supposed to have the oled functional, but it failed.



Before further discussion of the next task of the day, this was supposed be where we give input for our oled to display.

For the push notification task we used simple push; I downloaded a simple push node and setup the node red/ MQTT side, and my partner helped me with flashing the program on the hardware to send a ping via MQTT to node red. Also, there are a few images below just to illustrate some of the explained content. The first image shows a local implementation of the task. The second and third image shows how we tackled the task as intended we had a button that we pressed and that would send a signal as indicated in the setup file so that when received by node red it can then send a push notification.

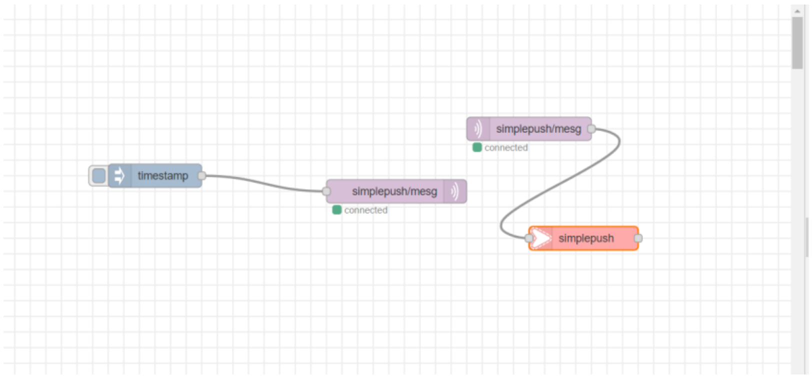
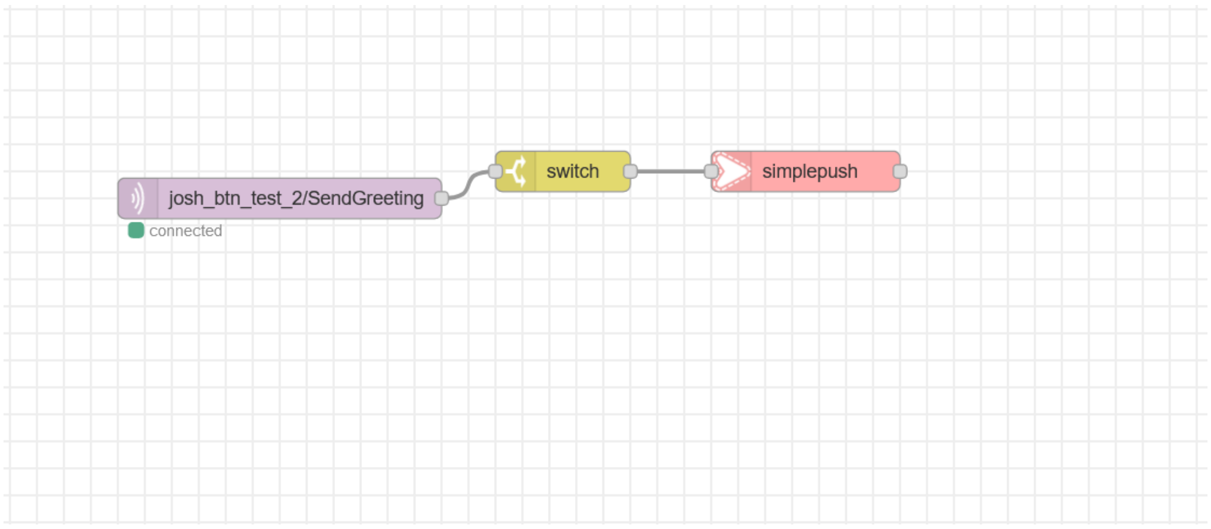
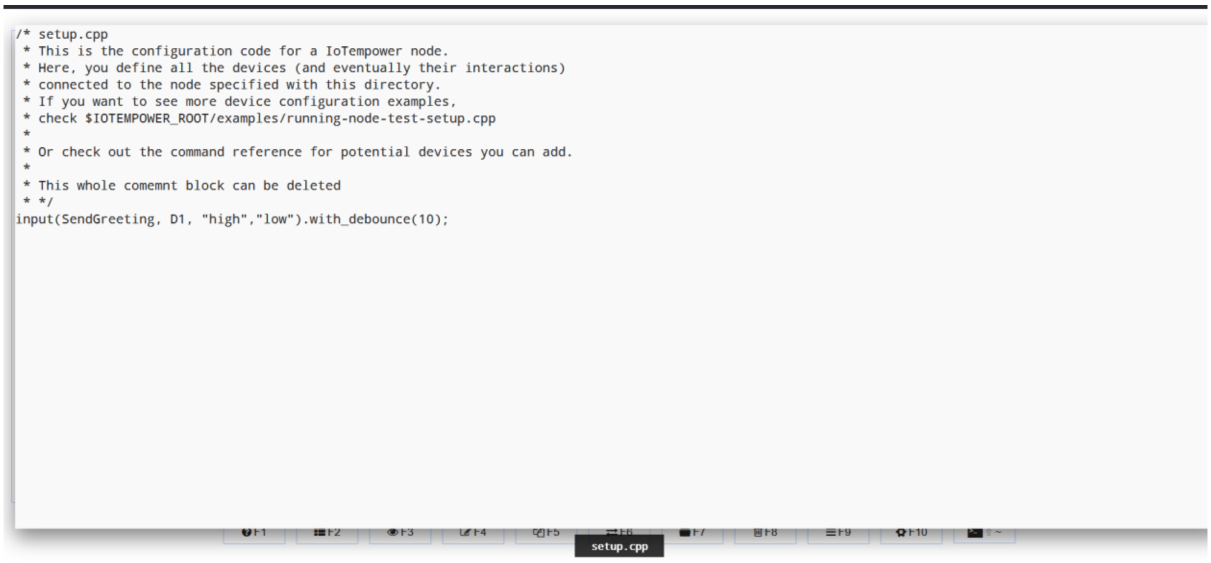
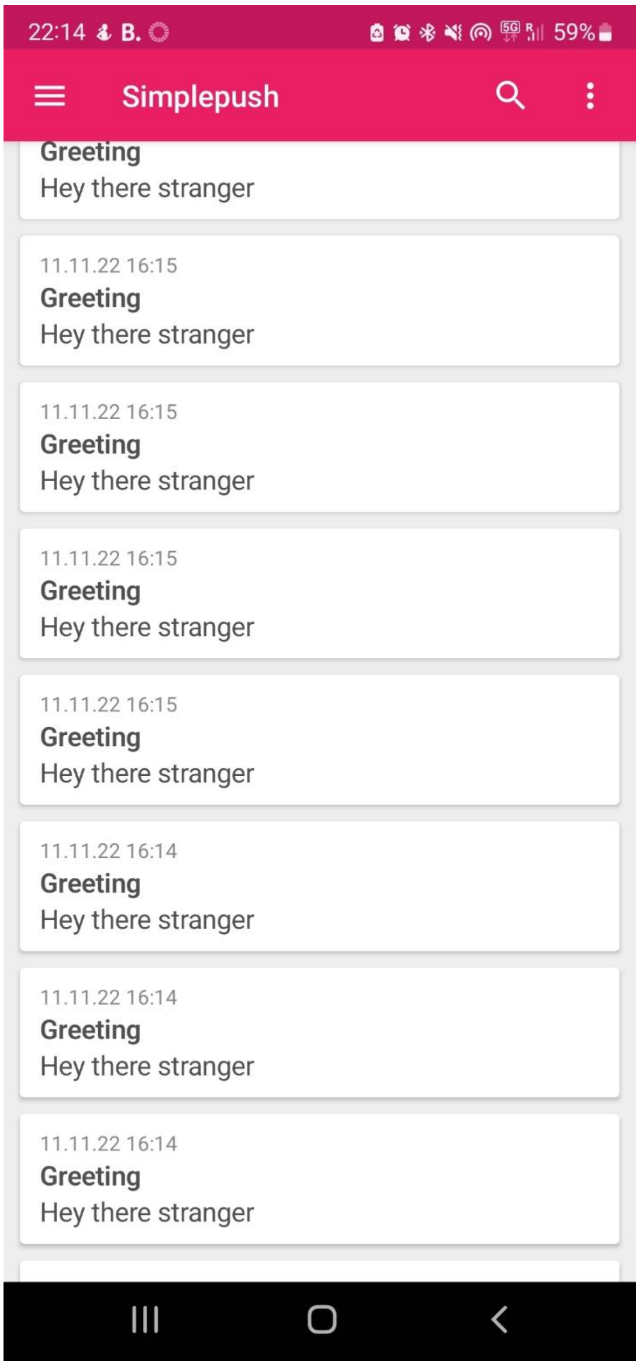


Figure 3: Notification – SimplePush





```
/* setup.cpp
 * This is the configuration code for a IoTempower node.
 * Here, you define all the devices (and eventually their interactions)
 * connected to the node specified with this directory.
 * If you want to see more device configuration examples,
 * check $IOTEMPOWER_ROOT/examples/running-node-test-setup.cpp
 *
 * Or check out the command reference for potential devices you can add.
 *
 * This whole comemnt block can be deleted
 * */
input(SendGreeting, D1, "high","low").with_debounce(10);
```



22:14 B. 59%

Simplepush

Greeting
Hey there stranger

11.11.22 16:15
Greeting
Hey there stranger

11.11.22 16:15
Greeting
Hey there stranger

11.11.22 16:15
Greeting
Hey there stranger

11.11.22 16:15
Greeting
Hey there stranger

11.11.22 16:14
Greeting
Hey there stranger

11.11.22 16:14
Greeting
Hey there stranger

11.11.22 16:14
Greeting
Hey there stranger

Figure 4: SimplePush Application, Andriod

Add text

Add moisture

Add ultrasonic

Add other 2.1.x tasks here as docs.

Day 4 Reflections

It was a really long day as well as sad and frustrating as one of our programs never full functions. So, I was not in the best mood by the end of the lecture as I do not like to leave things incomplete but due to the interest of time it had to be done. Would have loved more time but even a 9-hour work day was not enough which felt strange. However, there was pockets of upliftment during the day when the lecturer brought some nuggets of wisdom in respect to IOT which did help. Also felt like we had to rush to cover a lot of content in a short period of time and I feel like I am not getting the full experience of IOT, but I understand that we only have so much time.

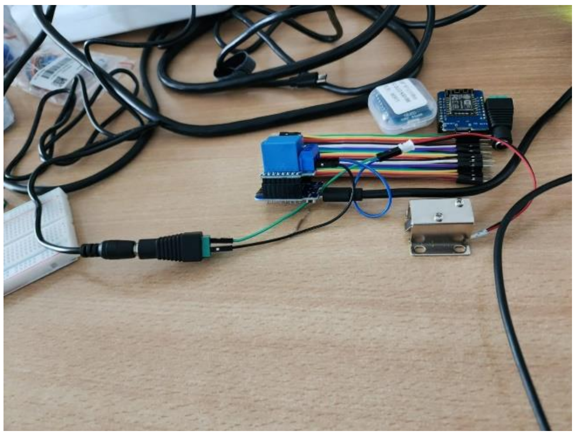
Week 2

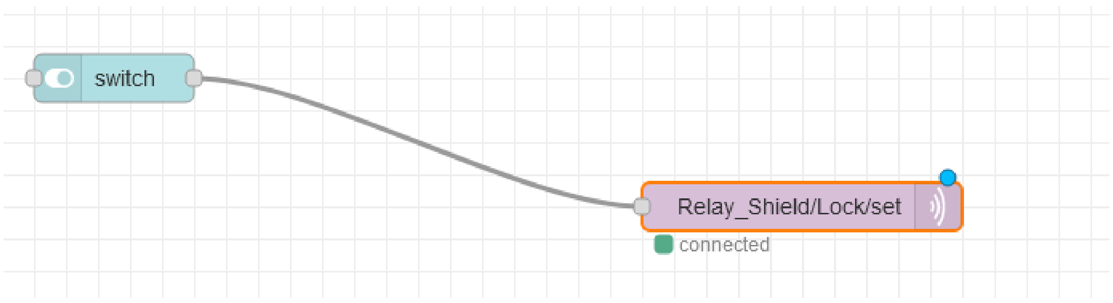
Day 1

14/11

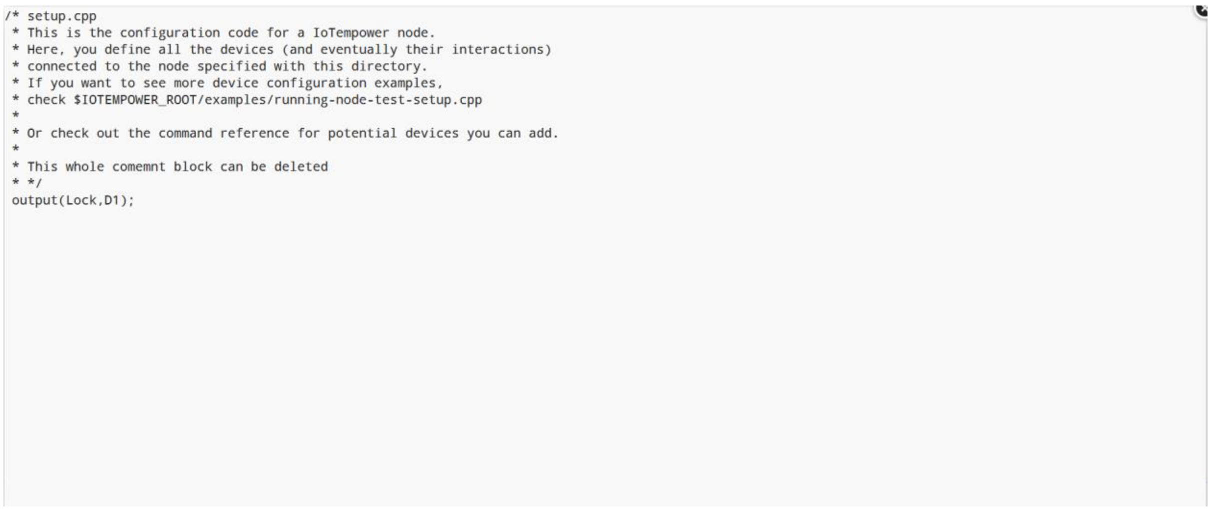
Reflection

iot upgrade to be able to flash and upgrade the firmware of the pi 400. We are now partaking on completing the integration task whereby we have to build a security system with a bunch of sensors. We got assistance with upgrade the pi 400 since we were getting access denied errors. We tried completing the RFID and Distance senor task in our pairs and we had to resize the memory of the pi 400 for we had around 3.7 GB of space left, so with the space resize we have around 11 GB more of space. The resize is taking a bit of time. The resize finally worked and we got the wemos for the Distance sensor exercise. We are getting the RFID Reader wemos d1 flashed. The Wemos d1 that will be used for the RFID scanner is flashed however I am trying to research on how to send data of what the RFID reader is receiving via MQTT. Still didn't work but because our RFID reader is actual damaged in a matter we do not understand, and we tested with another teams RFID reader and the program actual works. The lecturer tried to re soldered er the RFID reader, but it failed, and the RFID reader does not work. We are now trying to connect the relay to a female wemos but the wemos was incorrectly soldered so we are using a different one. We are now getting the relay and lock to work. We have configured the relay to function with the lock so that we could send mqtt messages from node red and receive them on the programmed wemos d1 mini with the topic Relay_Shield/Lock/set as shown in the screen shot with the node red flow.





What would happen when we flip the switch in node red it would send an on signal to the topic and then the lock would respond by either locking or unlocking.

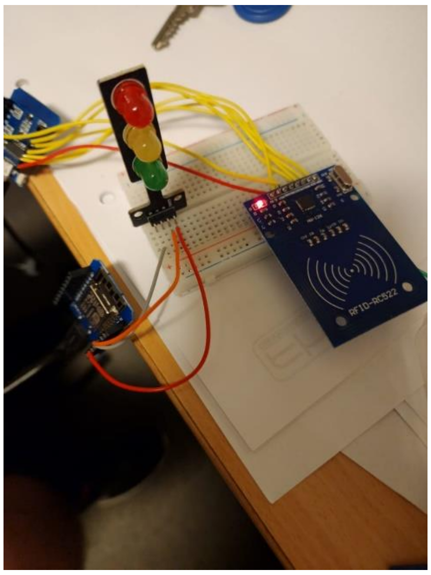


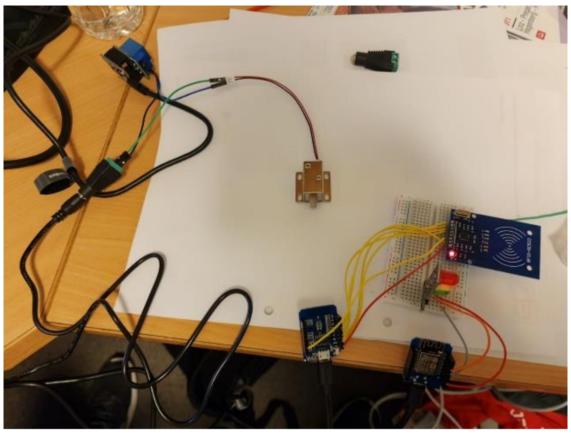
```
/* setup.cpp
 * This is the configuration code for a IoTempower node.
 * Here, you define all the devices (and eventually their interactions)
 * connected to the node specified with this directory.
 * If you want to see more device configuration examples,
 * check $IOTEMPOWER_ROOT/examples/running-node-test-setup.cpp
 *
 * Or check out the command reference for potential devices you can add.
 *
 * This whole comemnt block can be deleted
 * */
output(Lock,D1);
```

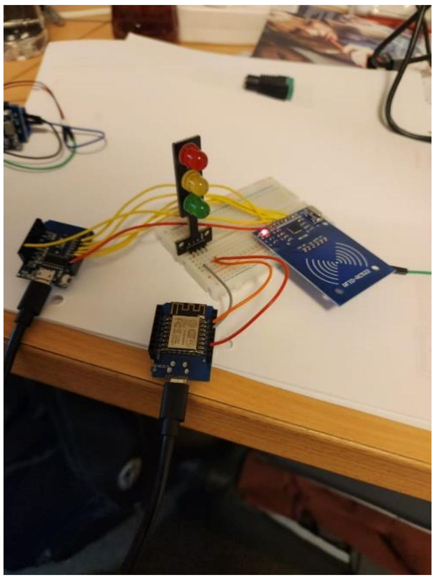
Here is our setup.cpp file for the lock.

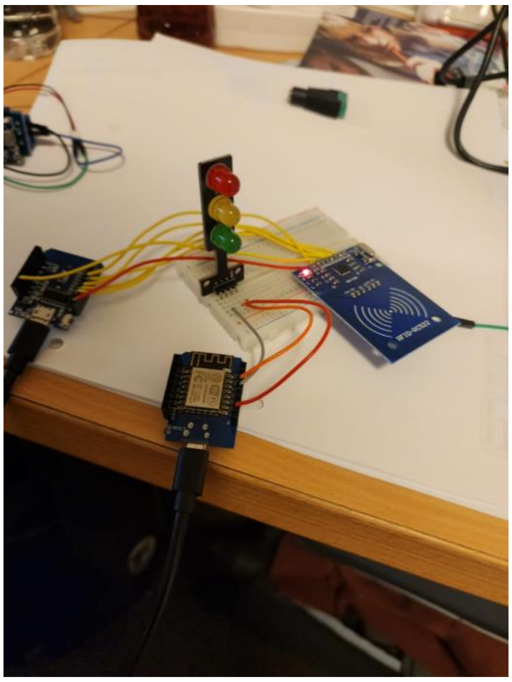
We are building the rest of the integration with the traffic light and the rfid senor works and are able to send data via MQTT and be able to build logic to have the lights of the traffic light switch on based on certain conditions such as turning the light red for access denied and turning green for having the right key card. I wired the traffic light incorrectly by putting the red into ground instead of D2. However regardless the red light constantly remained on but we are rewiring the traffic light hardware to a female wemos. We plugged the wire meant for green into the yellow pin. We got the traffic light working and there will be images to display how the works. We are now re integrating the lock and relay and we are testing using previous complied logic. We used the wrong pin output for the logic and now it works. Our topic now was incorrect. So, we had to use D1 instead of D2 since the data line in the relay is connected to D1. Insert image of the full system.

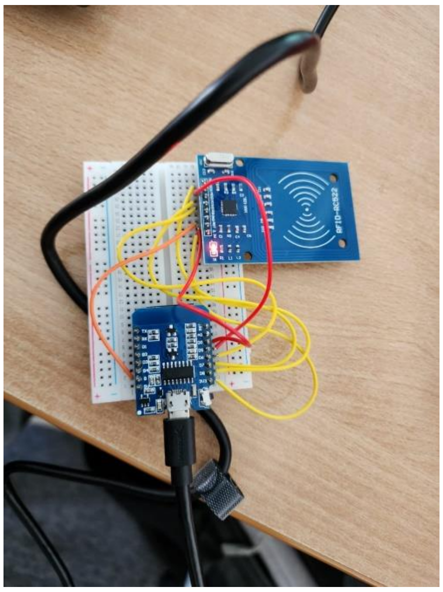
We now trying to now integrate the buzzer and we have to use another wemos and are flashing using the PI for the initial. We are looking into the documentation. When access is granted, the buzzer does make a sound. So Lastly, we have the flow that made the program function as best as it

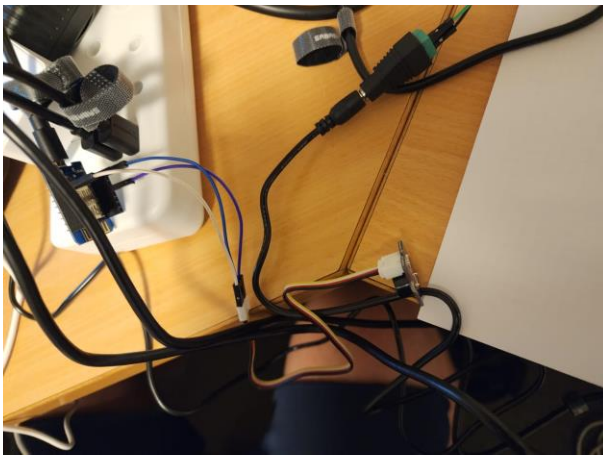

RFID-RC522

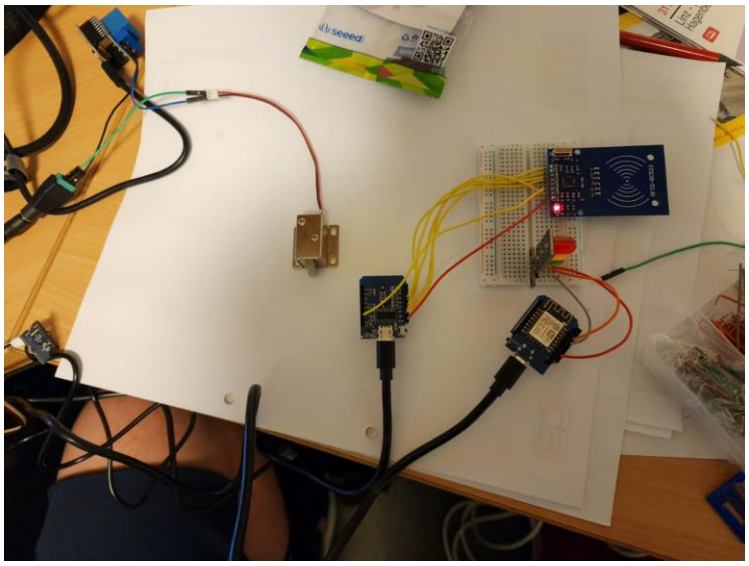


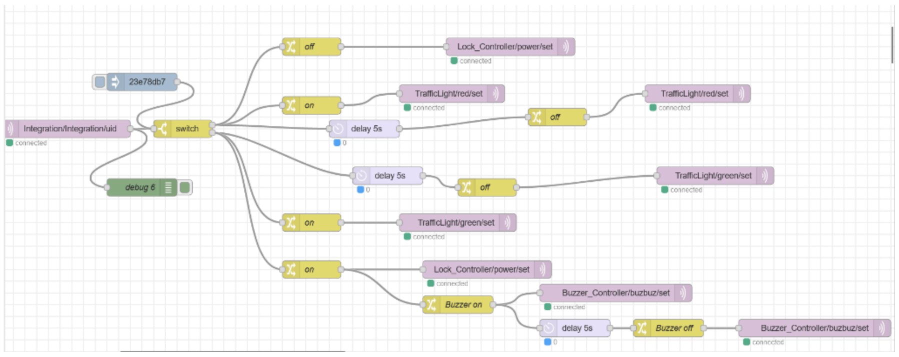

RFID-RC522


RFID-RC522


RFID-RC522







Day 1 Reflection

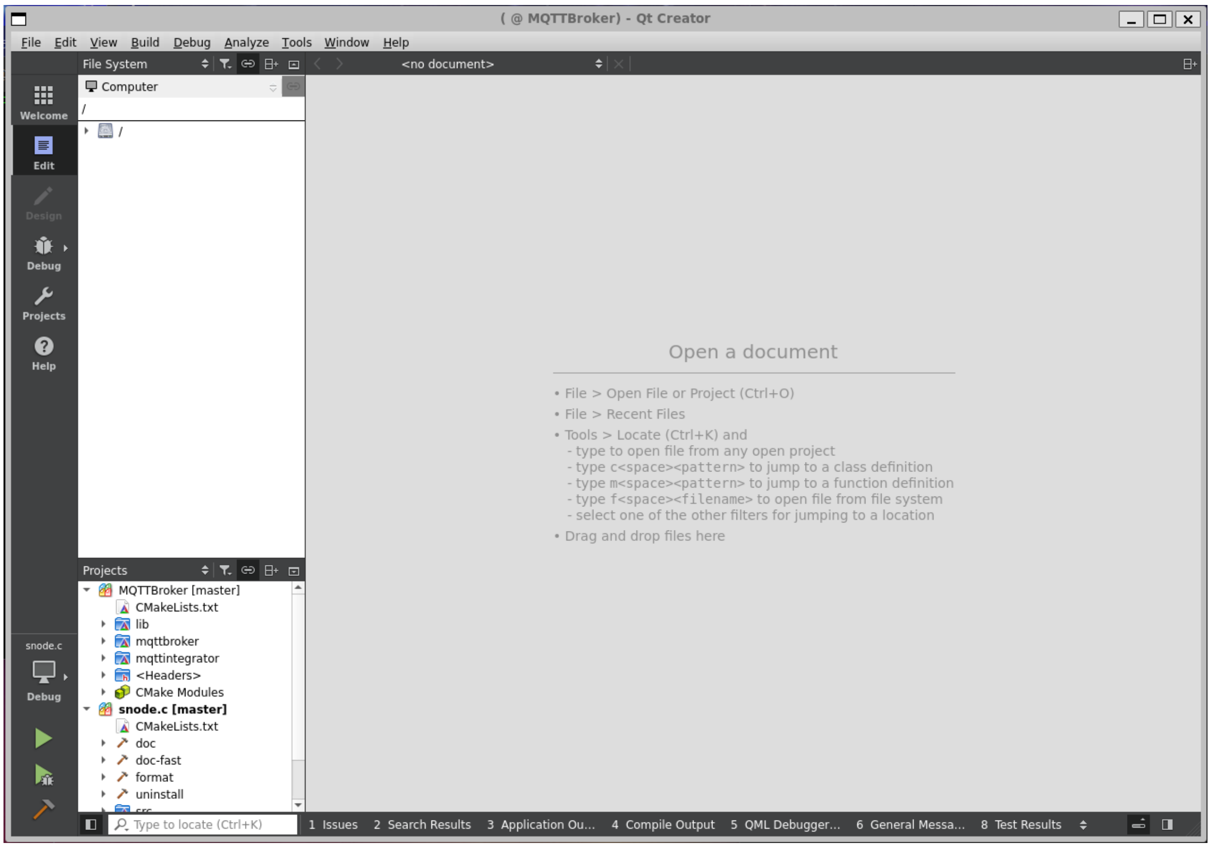
It was a trying day as there was a lot to do as we were wrapping up with Ulrich and we still had an integration task that we had to do. So that took a toll on me.

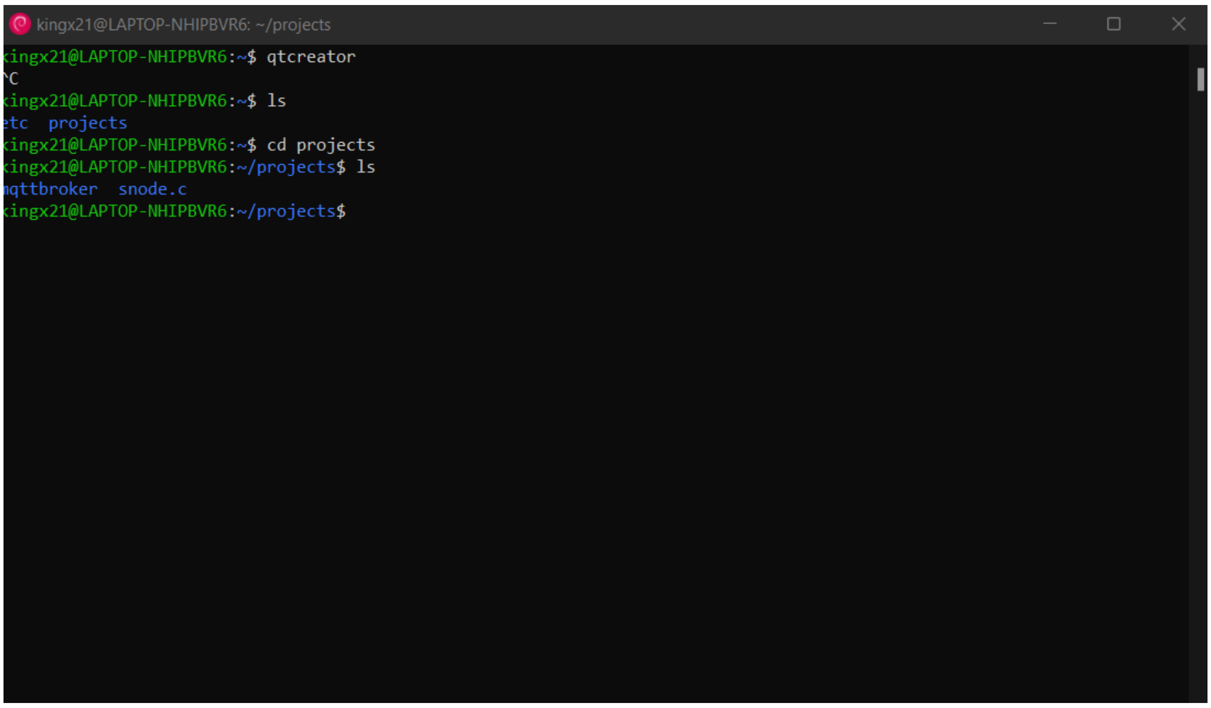
Day 2

15/11

Reflection, Install WSL, Clone Repositories – SNODE.C and MQTT and Install SNODE.C

I installed the windows subsystem for Linux which is a Debian installation and installed the libraries we were assigned to install. I also had to introduce my self as well as participate in an introductory presentation in presenting a scenario about our supposed future IOT system. After having Debian installed, I did a few exercises whereby I had to install the snode.c framework by cloning it and installing it in a local installation using Qt Creator. However, to make use of Qt Creator I had to configure the compiler to use GCC instead of Clang because I had build errors when trying to build the snode.c framework into my local projects folder. After the build was correctly completed successfully. There will be an folder with the build components alongside the original snode.c folder. After that I did an exercise where by I had to run an http client but had to run a legacy server first before I could run the client. After getting the build to work with the help of the lecturer since I was new to the snode.c framework. After that I went Qt Creator and deploy the snode.c project which install the entire framework globally , meaning on the entire sub system (root folder) so that the framework is not constrained to an interior folder. Lastly for the day after deploying the framework I had to clone another repository which is the mqtt broker repo. After that I had to build like with the snode.c framework and after the build was complete. I had to then dig with the folders of the build to find and configure the mqtt broker before I could run the broker before any messages could be sent to the broker. Use screenshots for more detail.


(@ MQTTBroker) - Qt Creator
File Edit View Build Debug Analyze Tools Window Help
File System
Computer
<no document>
Welcome
Edit
Design
Debug
Projects
Help
Open a document
• File > Open File or Project (Ctrl+O)
• File > Recent Files
• Tools > Locate (Ctrl+K) and
- type to open file from any open project
- type c<space><pattern> to jump to a class definition
- type m<space><pattern> to jump to a function definition
- type f<space><filename> to open file from file system
- select one of the other filters for jumping to a location
• Drag and drop files here
Projects
MQTTBroker [master]
CMakeLists.txt
lib
mqttbroker
mqttintegrator
<Headers>
CMake Modules
snode.c [master]
CMakeLists.txt
doc
doc-fast
format
uninstall
snode.c
Debug
Type to locate (Ctrl+K)
1 Issues 2 Search Results 3 Application Ou... 4 Compile Output 5 QML Debugger... 6 General Messa... 8 Test Results



```
kingx21@LAPTOP-NHIPBVR6:~$ qtcreator
^C
kingx21@LAPTOP-NHIPBVR6:~$ ls
etc  projects
kingx21@LAPTOP-NHIPBVR6:~$ cd projects
kingx21@LAPTOP-NHIPBVR6:~/projects$ ls
mqttbroker  snode.c
kingx21@LAPTOP-NHIPBVR6:~/projects$
```

Day 2 reflections

The new framework was overwhelming but manageable, but I did have some fun with the subsystem within the given boundaries. Though the novelty of the subsystem did wear off toward the end of the lecturer as stuff became more challenging as a more serious approach had to be adopt and less of the experimental approach.

Day 3

17/11

Reflection

I was sick and unwell, but I was informed that our team had to install the snode.c framework on the Raspberry Pi.

Day 3 reflections

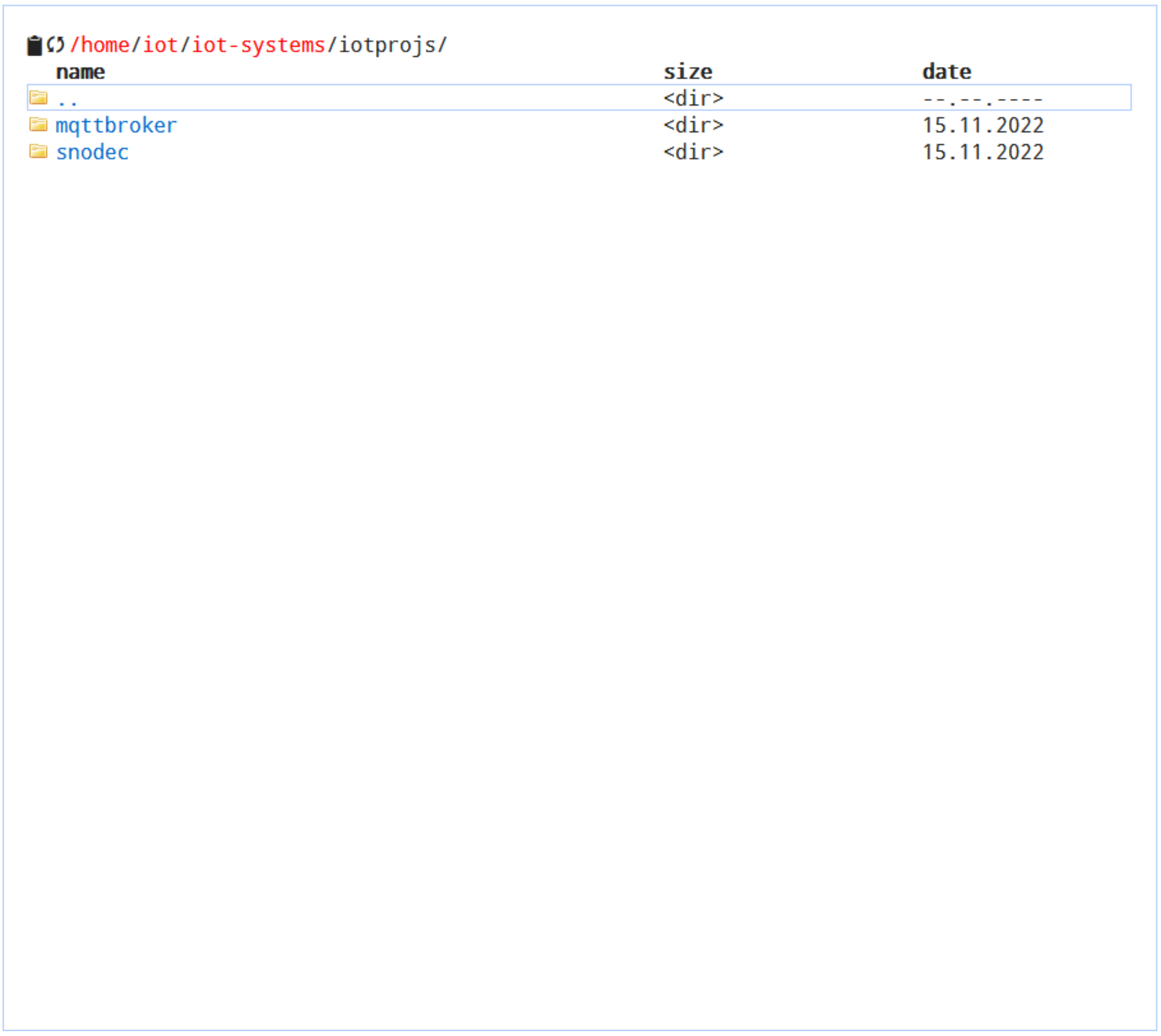
I have nothing much to say at the moment since I was absent.

Week 3

Day 1

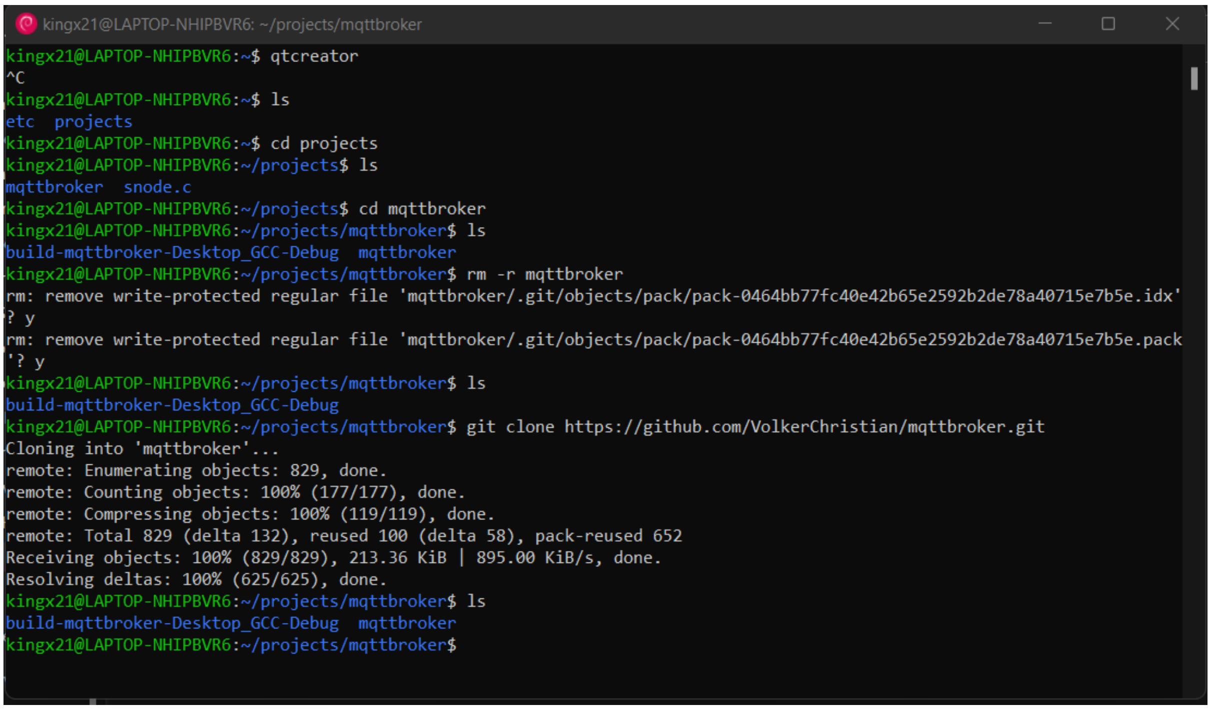
Reflection, Mqttbroker Webserver Application

I am cloning an updated version of the snode.c framework and then also cloning an updated version of the mqtt framework. A colleague (Josias Hoffman) has received help from another team to install the updated clones of the two repos onto our raspberry pi. However, we are having troubles installing stuff system wide such as the snod.c framework but so we as a team cannot continue further but I hope to catchup later. I am learning about the mqtt broker architecture and how this links with snode.c framework but since our pi is not properly configured, I cannot continue on the actual pi.



```
/home/iot/iot-systems/iotprojs/
  name                                     size         date
  ..                                       <dir>        --.--.----
  mqttbroker                               <dir>        15.11.2022
  snodec                                   <dir>        15.11.2022
```

But at the end of the day I and my team had to get help from the lecture and he sent us his configured mqtt broker program and pushed it to git hub. So, I clone that updated file.



```
kingx21@LAPTOP-NHIPBVR6: ~/projects/mqttbroker
kingx21@LAPTOP-NHIPBVR6:~$ qtcreator
^C
kingx21@LAPTOP-NHIPBVR6:~$ ls
etc  projects
kingx21@LAPTOP-NHIPBVR6:~$ cd projects
kingx21@LAPTOP-NHIPBVR6:~/projects$ ls
mqttbroker  snode.c
kingx21@LAPTOP-NHIPBVR6:~/projects$ cd mqttbroker
kingx21@LAPTOP-NHIPBVR6:~/projects/mqttbroker$ ls
build-mqttbroker-Desktop_GCC-Debug  mqttbroker
kingx21@LAPTOP-NHIPBVR6:~/projects/mqttbroker$ rm -r mqttbroker
rm: remove write-protected regular file 'mqttbroker/.git/objects/pack/pack-0464bb77fc40e42b65e2592b2de78a40715e7b5e.idx'
? y
rm: remove write-protected regular file 'mqttbroker/.git/objects/pack/pack-0464bb77fc40e42b65e2592b2de78a40715e7b5e.pack
'? y
kingx21@LAPTOP-NHIPBVR6:~/projects/mqttbroker$ ls
build-mqttbroker-Desktop_GCC-Debug
kingx21@LAPTOP-NHIPBVR6:~/projects/mqttbroker$ git clone https://github.com/VolkerChristian/mqttbroker.git
Cloning into 'mqttbroker'...
remote: Enumerating objects: 829, done.
remote: Counting objects: 100% (177/177), done.
remote: Compressing objects: 100% (119/119), done.
remote: Total 829 (delta 132), reused 100 (delta 58), pack-reused 652
Receiving objects: 100% (829/829), 213.36 KiB | 895.00 KiB/s, done.
Resolving deltas: 100% (625/625), done.
kingx21@LAPTOP-NHIPBVR6:~/projects/mqttbroker$ ls
build-mqttbroker-Desktop_GCC-Debug  mqttbroker
kingx21@LAPTOP-NHIPBVR6:~/projects/mqttbroker$
```

So, this is an example of the updated file cloning but on the local installation.

Day 1 Reflections

Day 2

23/11

Reflection

We are going to disable the mosquitto broker from running automatically on the raspberry pi. So we are going to use our mqtt broker that was coded the day before. Such as by going into the system files. Also we are trying to have node red run but we are running into issues , the starter does not function as intended, it seems it was already in use. To see how we can see what is running , we used a command , "ps -aef | grep node -red". To make sure the snode.c and the mqttbroker files install system wide on the pi we run the command sudo "filename" make install. We had to compile the files beforehand. We focused on some project work wereby we had to implement some of the IOT concepts we learnt into our overarching project.

Day 3

24/11

Investigate Public-Key crypotography

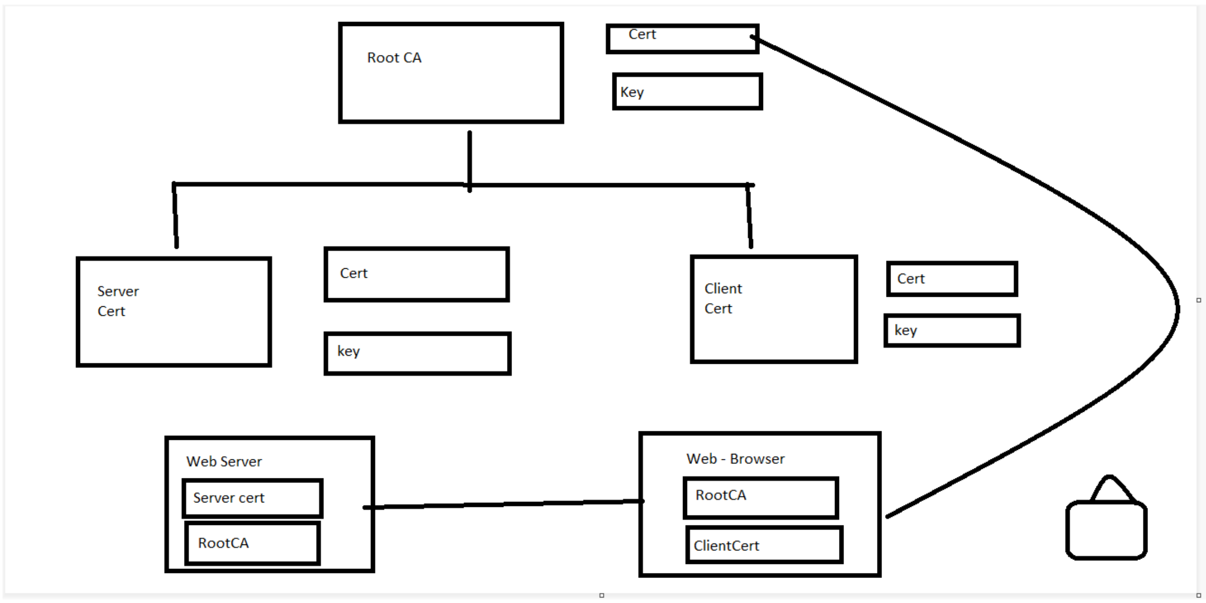
Public key cryptography is a class of cryptographic protocols based on algorithms. This method of cryptography requires two separate keys, one that is private or secret, and one that is public. Public key cryptography uses a pair of keys to encrypt and decrypt data to protect it against unauthorized access or use.

Investigate X.509

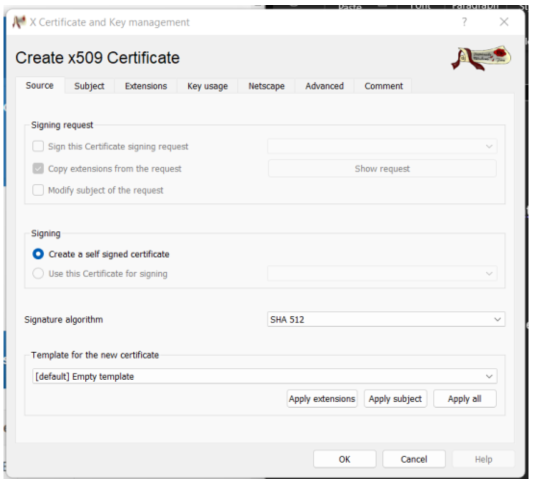
X.509 is a standard format for **public key certificates**, digital documents that securely associate cryptographic key pairs with identities such as websites, individuals, or organizations.

Reflection

I installed XCA which is a certificate management software and was instructed to make a root certificate and a client and server certificate. I along with my classmates explored the architecture/system of how certificates work.



I am also learning how to actually create the certificate within our local linux subsystems and not via the raspberry pi for tutorial purposes.



Create x509 Certificate

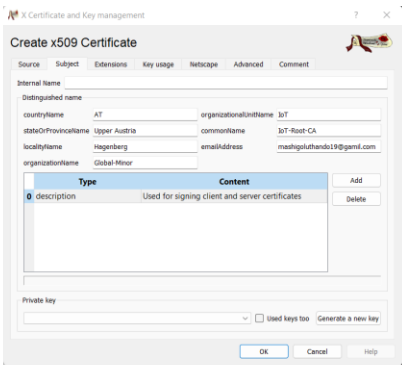
Signing request
- [ ] Sign this Certificate signing request
- [x] Copy extensions from the request
- [ ] Modify subject of the request

Signing
- [✓ Create a self signed certificate]
- Use this Certificate for signing

Signature algorithm: SHA 512

Template for the new certificate: [default] Empty template

We are creating a certificate which will be a root CA certificate and defining its basic characteristics



Internal Name:

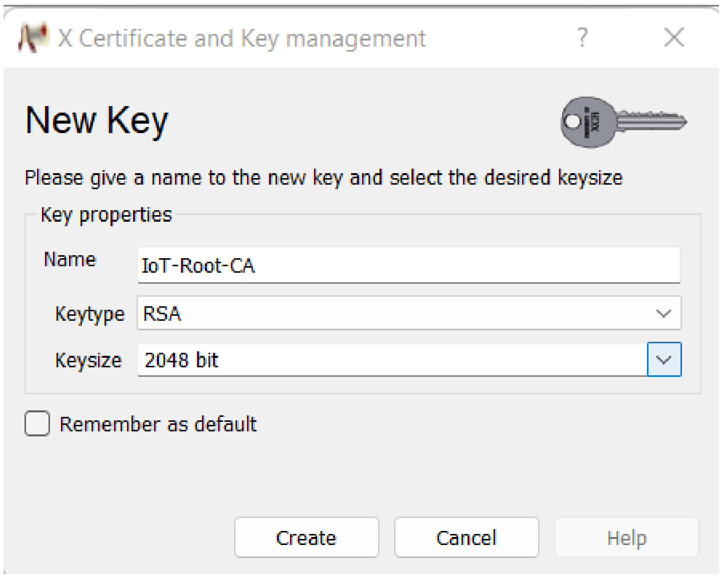
Distinguished name

countryName	AT	organizationalUnitName	IoT
stateOrProvinceName	Upper Austria	commonName	IoT-Root-CA
localityName	Hagenberg	emailAddress	mashigoluthando19@gamil.com
organizationName	Global-Minor		

	Type	Content
0	description	Used for signing client and server certificates

Private key: [ ] Used keys too | Generate a new key

We are always generating a new key



New Key

Please give a name to the new key and select the desired keysize

Key properties

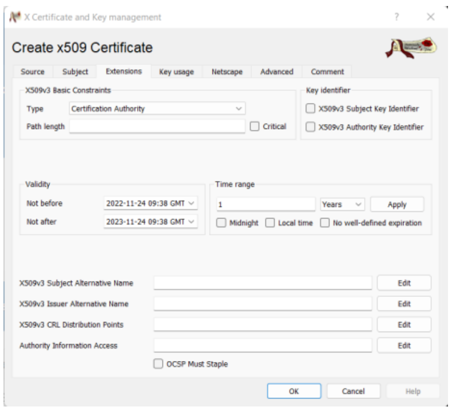
Name: IoT-Root-CA

Keytype: RSA

Keysize: 2048 bit

[ ] Remember as default

More definition we are giving to the Root CA



X509v3 Basic Constraints

Type: Certification Authority

Path length: [ ] Critical

Key identifier

[ ] X509v3 Subject Key Identifier

[ ] X509v3 Authority Key Identifier

Validity

Not before: 2022-11-24 09:38 GMT

Not after: 2023-11-24 09:38 GMT

Time range: 1 Years

[ ] Midnight [ ] Local time [ ] No well-defined expiration

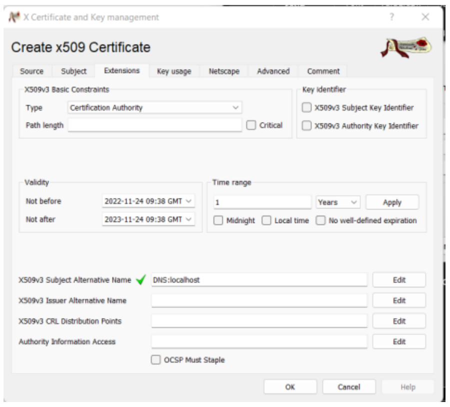
X509v3 Subject Alternative Name

X509v3 Issuer Alternative Name

X509v3 CRL Distribution Points

Authority Information Access

[ ] OCSP Must Staple


X Certificate and Key management

Create x509 Certificate

X509v3 Basic Constraints

Type: Certification Authority

Path length: ☐ Critical

Key identifier

☐ X509v3 Subject Key Identifier

☐ X509v3 Authority Key Identifier

Validity

Not before: 2022-11-24 09:38 GMT

Not after: 2023-11-24 09:38 GMT

Time range: 1 Years Apply

☐ Midnight ☐ Local time ☐ No well-defined expiration

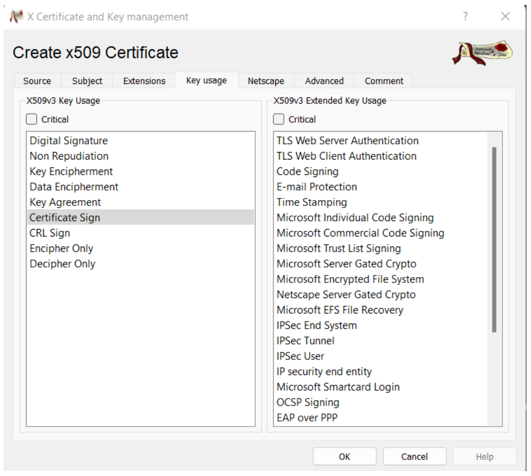
X509v3 Subject Alternative Name ✓ DNS:localhost

X509v3 Issuer Alternative Name

X509v3 CRL Distribution Points

Authority Information Access

☐ OCSP Must Staple


X Certificate and Key management

Create x509 Certificate

X509v3 Key Usage

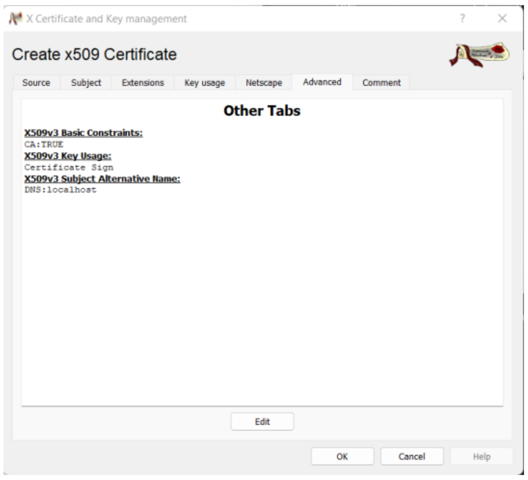
☐ Critical

Digital Signature
Non Repudiation
Key Encipherment
Data Encipherment
Key Agreement
Certificate Sign
CRL Sign
Encipher Only
Decipher Only

X509v3 Extended Key Usage

☐ Critical

TLS Web Server Authentication
TLS Web Client Authentication
Code Signing
E-mail Protection
Time Stamping
Microsoft Individual Code Signing
Microsoft Commercial Code Signing
Microsoft Trust List Signing
Microsoft Server Gated Crypto
Microsoft Encrypted File System
Netscape Server Gated Crypto
Microsoft EFS File Recovery
IPSec End System
IPSec Tunnel
IPSec User
IP security end entity
Microsoft Smartcard Login
OCSP Signing
EAP over PPP


X Certificate and Key management

Create x509 Certificate

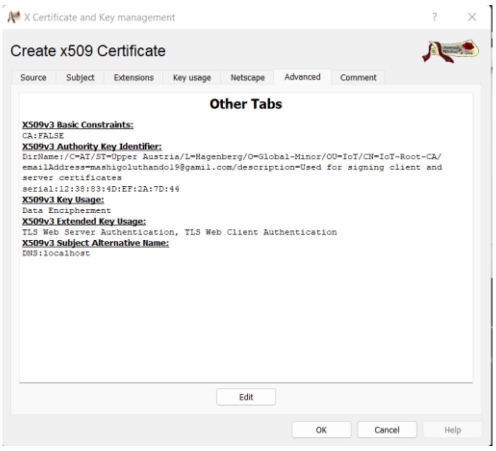
Other Tabs

X509v3 Basic Constraints:
```
CA:TRUE
```
X509v3 Key Usage:
```
Certificate Sign
```
X509v3 Subject Alternative Name:
```
DNS:localhost
```

Summary of the details of my root CA

Now we are going to create to end certificates for web application purposes

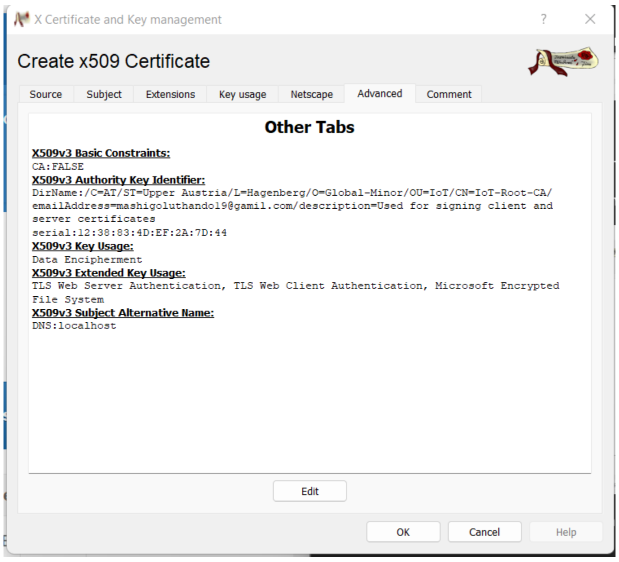
This is my MQTT Server Certificate Summary



Other Tabs

X509v3 Basic Constraints:
```
CA:FALSE
```
X509v3 Authority Key Identifier:
```
DirName:/C=AT/ST=Upper Austria/L=Hagenberg/O=Global-Minor/OU=IoT/CN=IoT-Root-CA/
emailAddress=mashigoluthando19@gamil.com/description=Used for signing client and
server certificates
serial:12:38:83:4D:EF:2A:7D:44
```
X509v3 Key Usage:
```
Data Encipherment
```
X509v3 Extended Key Usage:
```
TLS Web Server Authentication, TLS Web Client Authentication
```
X509v3 Subject Alternative Name:
```
DNS:localhost
```

MQTT Client Certificate Summary



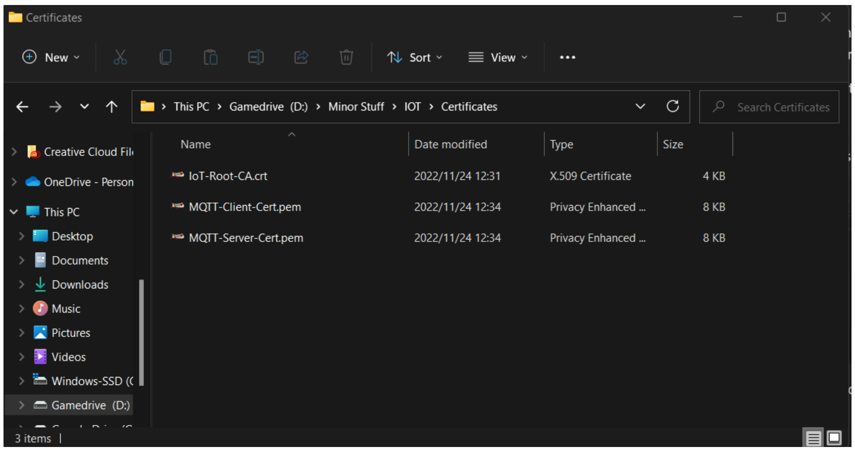
Other Tabs

X509v3 Basic Constraints:
```
CA:FALSE
```
X509v3 Authority Key Identifier:
```
DirName:/C=AT/ST=Upper Austria/L=Hagenberg/O=Global-Minor/OU=IoT/CN=IoT-Root-CA/
emailAddress=mashigoluthando19@gamil.com/description=Used for signing client and
server certificates
serial:12:38:83:4D:EF:2A:7D:44
```
X509v3 Key Usage:
```
Data Encipherment
```
X509v3 Extended Key Usage:
```
TLS Web Server Authentication, TLS Web Client Authentication, Microsoft Encrypted
File System
```
X509v3 Subject Alternative Name:
```
DNS:localhost
```

We are show two scenarios with an httpserver and the interactions with the certificates.

We are exporting the root CA using the PEM format

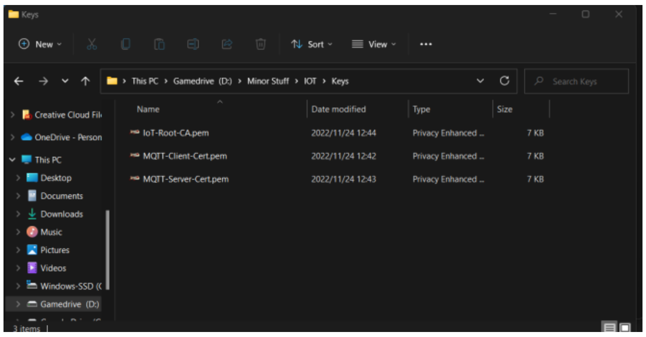
We are also exporting the end entity certificate (Server and Client) but we a different method which is the pem chain which has all certificates upto the root.



Name	Date modified	Type	Size
IoT-Root-CA.crt	2022/11/24 12:31	X.509 Certificate	4 KB
MQTT-Client-Cert.pem	2022/11/24 12:34	Privacy Enhanced ...	8 KB
MQTT-Server-Cert.pem	2022/11/24 12:34	Privacy Enhanced ...	8 KB

We also need to export the keys to unlock the certificates.

We export with encryption because then it is password encrypted.



This PC › Gamedrive (D:) › Minor Stuff › IOT › Keys

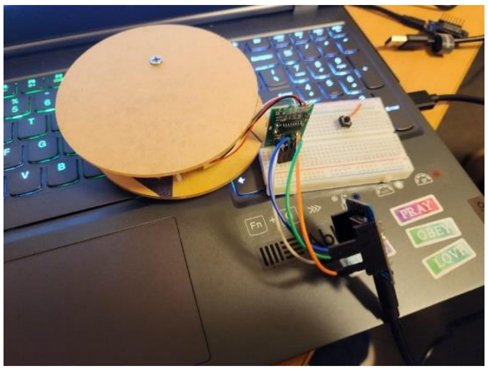
Name	Date modified	Type	Size
IoT-Root-CA.pem	2022/11/24 12:44	Privacy Enhanced ...	7 KB
MQTT-Client-Cert.pem	2022/11/24 12:42	Privacy Enhanced ...	7 KB
MQTT-Server-Cert.pem	2022/11/24 12:43	Privacy Enhanced ...	7 KB

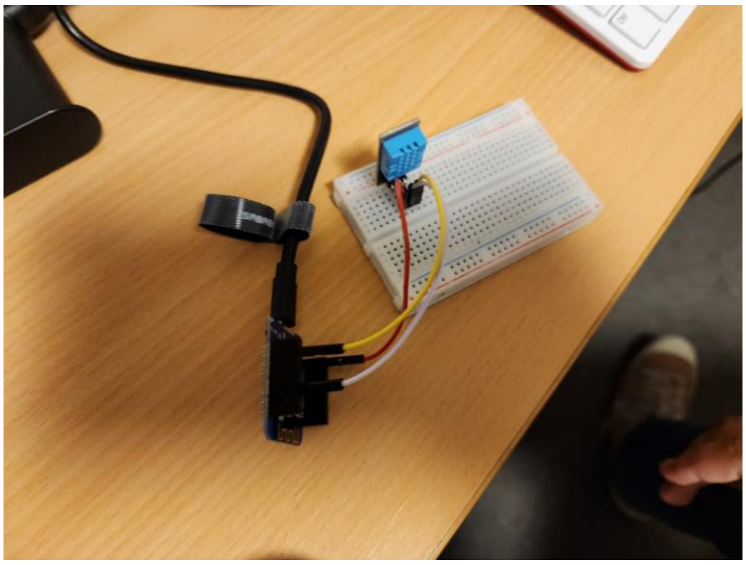
3 items

We are configure the mqtt broker to work , server with the certificates but the lectuere is busy with that.

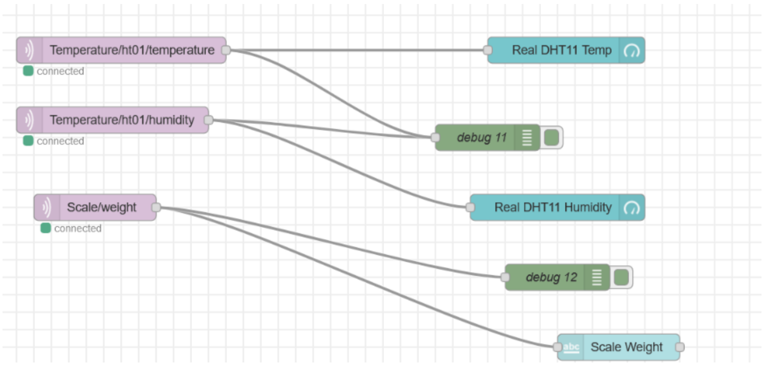
Project Implementation:

We also did part of the project implementation with some assistance such as wiring the two esp8266’s with the temperature and scale.

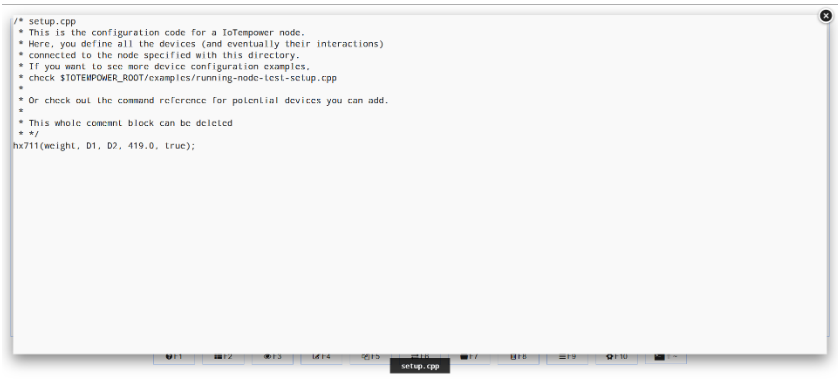




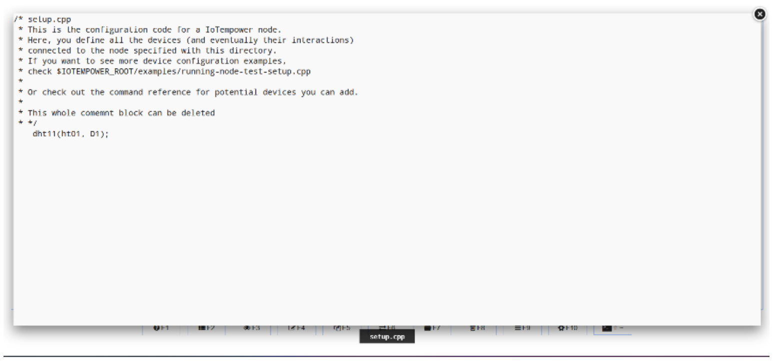
Also, We worked as a team to get node red running and have the mqtt topics configured as shown in the node red screen shot. We had to make use of the IOTEmpower framework as we had no framework of our own.



Also, along my team we flashed the two esp's with code that would have us read the temperature , humidity and weight of the scale and pass that to node-red as shown in the setup.cpp files shown below.

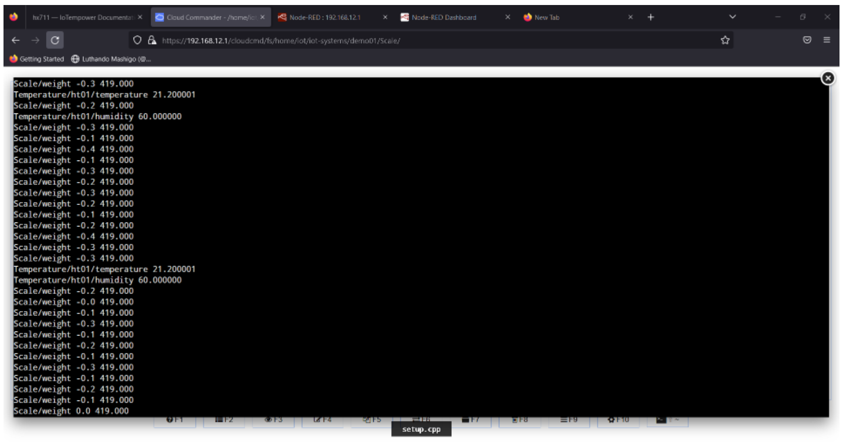


```
/* setup.cpp
 * This is the configuration code for a IoTempower node.
 * Here, you define all the devices (and eventually their interactions)
 * connected to the node specified with this directory.
 * If you want to see more device configuration examples,
 * check $IOTEMPOWER_ROOT/examples/running-node-test-setup.cpp
 *
 * Or check out the command reference for potential devices you can add.
 *
 * This whole comemnt block can be deleted
 * */
hx711(weight, D1, D2, 419.0, true);
```

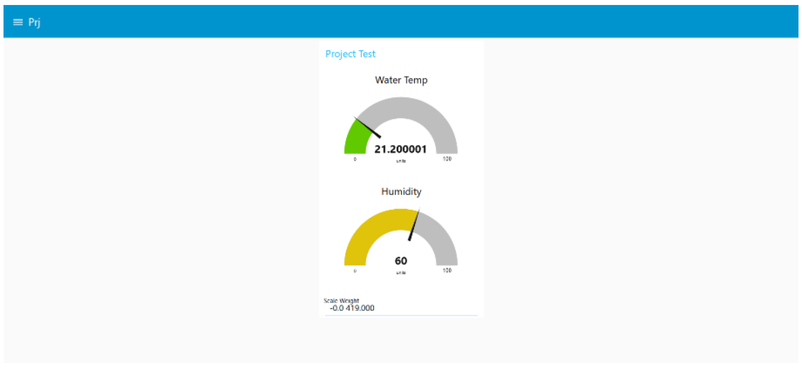


```
/* setup.cpp
 * This is the configuration code for a IoTempower node.
 * Here, you define all the devices (and eventually their interactions)
 * connected to the node specified with this directory.
 * If you want to see more device configuration examples,
 * check $IOTEMPOWER_ROOT/examples/running-node-test-setup.cpp
 *
 * Or check out the command reference for potential devices you can add.
 *
 * This whole comemnt block can be deleted
 * */
   dht11(ht01, D1);
```

Lastly After the flashing and after receiving data we were able to pass data using these topics and having the program running and then having a temporary node-red UI as a stand in for front end display the data. I just quickly put that together since there was not enough time to fill flesh out the system.


hx711 — IoTempower Documentat
Cloud Commander - /home/io
Node-RED : 192.168.12.1
Node-RED Dashboard
New Tab
https://192.168.12.1/cloudcmd/fs/home/iot/iot-systems/demo01/Scale/
Getting Started
Luthando Mashigo (@...
setup.cpp

```
Scale/weight -0.3 419.000
Temperature/ht01/temperature 21.200001
Scale/weight -0.2 419.000
Temperature/ht01/humidity 60.000000
Scale/weight -0.3 419.000
Scale/weight -0.1 419.000
Scale/weight -0.4 419.000
Scale/weight -0.1 419.000
Scale/weight -0.3 419.000
Scale/weight -0.2 419.000
Scale/weight -0.3 419.000
Scale/weight -0.2 419.000
Scale/weight -0.1 419.000
Scale/weight -0.2 419.000
Scale/weight -0.4 419.000
Scale/weight -0.3 419.000
Scale/weight -0.3 419.000
Temperature/ht01/temperature 21.200001
Temperature/ht01/humidity 60.000000
Scale/weight -0.2 419.000
Scale/weight -0.0 419.000
Scale/weight -0.1 419.000
Scale/weight -0.3 419.000
Scale/weight -0.1 419.000
Scale/weight -0.2 419.000
Scale/weight -0.1 419.000
Scale/weight -0.3 419.000
Scale/weight -0.1 419.000
Scale/weight -0.2 419.000
Scale/weight -0.1 419.000
Scale/weight 0.0 419.000
```



Scale Weight
-0.0 419.000