IoT Project

Embedded Systems Project

Bachelor's report

Electrical and Automation Engineering

Spring 2022

Animesh Joshi

Anna Kuznetsova

Emanuela Foica

Dumitru Blaga

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Electrical and Automation Engineering Abstract
Author Animesh Joshi, Anna Kuznetsova, Emanuella Foica, Dumitru Blaga Year 2022
Subject
Supervisors Juha Sarkula, Ari Lingdren

In this project, the students were provided with a Raspberry Pi with a SenseHat addon and an Arduino Uno WiFi with three sensors. The main goal of the project was to automate the room that was given to students in the campus building, to implement IOT system and get data from it for further use. The basic features that were requested were getting data, formatting it to requested by teacher format and send it on common broker topic as MQTT message in JSON format. And implement visualization for real-time metering. Besides basic features additional functionality was added, such as light-bulb intensity value depending on brightness, data base storing (of data + alarm log), other rooms viewing and remote access. Work done satisfy goals set, summary on project was analysed and written.

Keywords Arduino, Raspberry Pi, MQTT, Node-Red.
Pages XX pages and appendices XX pages

Contents

Appendices

1 Functional definition

1.1 System Function and Interfaces

1.1.1 Arduino

The Arduino is connected to three different sensors, a temperature and humidity sensor, a light intensity sensor and a motion sensor. These sensors combined provide data which is then stored in variables. Using the mqttClient.print(); command from the ArduinoMQTTClient library this data is then sent to the MQTT broker in the form of a MQTT message. This data is sent in two types, float and String. In order to format the variables properly as an MQTT message the following print commands are used. To send a float value, the formatting was limited to the title afterwhich the variable could be sent, however, for strings the formatting required us to add a \" before and after the string variable so that it could be parsed and converted to JSON format in NodeRed. The data from the motion sensor is sent as a string and the variable for it is called "pirStatMsg" it has two values depending on the output of the sensor. If the output is "HIGH" then the message sent says "present" and if it is "LOW" it says "not present".

```
//sending MQTT message
mqttClient.beginMessage(topic);
mqttClient.print("{\"temperature\": ");
mqttClient.print(fTemp);

mqttClient.print(",\"humidity\": ");
mqttClient.print(fHum);
mqttClient.print(", \"lightstatus\": ");
mqttClient.print("\"");
mqttClient.print(light_stat);
mqttClient.print("\"");
mqttClient.print(", \"brightness\": ");
mqttClient.print(light_val);
mqttClient.print(", \"presence\": ");
mqttClient.print("\"");
mqttClient.print(pirStatMsg);
mqttClient.print("\"");
mqttClient.print("}");
mqttClient.print(",\"light bulb intensity value\": ");
mqttClient.print(light_dim_send);
mqttClient.endMessage();
```

In addition to the basic implementation, on the Arduino side, we added a dynamic light control feature. This feature uses data from the Light Intensity sensor that was connected to the Arduino and uses it to set the value of the lights between 0 and 100, 0 being the lights being off and 100 being the lights being at full brightness. The underlying calculation for this was done by using information from https://www.engineeringtoolbox.com/light-level-rooms-d_708.html which said that the ideal light intensity in a classroom should be 300 Lux as shown in the table below. The formula was that of a linear equation but with a negative slope as the two values have to be inversely proportional.

Activity	Illuminance *(lx, lumen/m²)*
Public areas with dark surroundings	20 - 50
Simple orientation for short visits	50 - 100
Areas with traffic and corridors - stairways, escalators and travelators - lifts - storage spaces	100
Working areas where visual tasks are only occasionally performed	100 - 150
Warehouses, homes, theaters, archives, loading bays	150
Coffee break room, technical facilities, ball-mill areas, pulp plants, waiting rooms,	200
Easy office work	250
Class rooms	300
Normal office work, PC work, study library, groceries, show rooms, laboratories, check-out areas, kitchens, auditoriums	500
Supermarkets, mechanical workshops, office landscapes	750
Normal drawing work, detailed mechanical workshops, operation theaters	1000
Detailed drawing work, very detailed mechanical works, electronic workshops, testing and adjustments	1500 - 2000
Performance of visual tasks of low contrast and very small size for prolonged periods of time	2000 - 5000
Performance of very prolonged and exacting visual tasks	5000 - 10000
Performance of very special visual tasks of extremely low contrast and small size	10000 - 20000

The calculation for the value of light intensity is shown in the code snippet below. As the values can become negative, an if statement is added in order to limit the minimum value to 0.

```
//Dynamic light control function
//Calculation based on ideal lighting for a class being 300 lux
float light_dim = (-0.33334*light_val) + 100;
float light_dim_send = light_dim;
 if(light_dim<0)
 {
  light_dim_send = 0;

 }
```

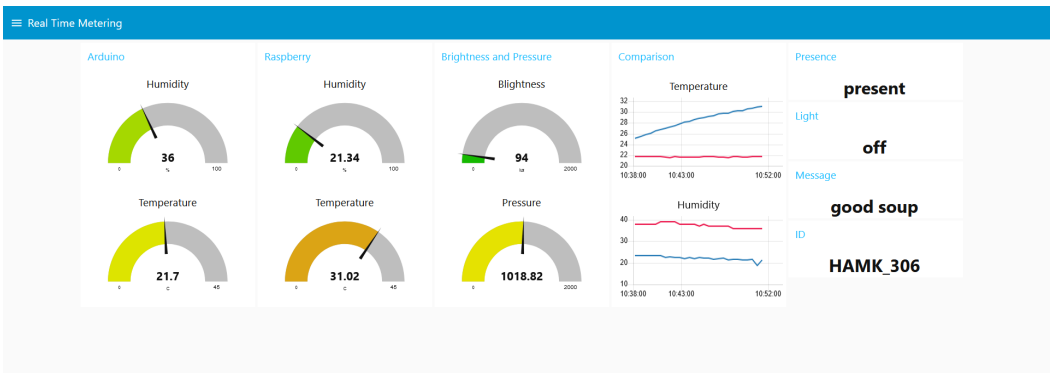
1.1.2 Raspberry

Raspberry Pi is connected to a power source and to a laptop. In the Node-Red through the nodes the program receives messages from the arduino as it is subscribed to the corresponding topic of the broker. Also sense hat is connected to Raspberry Pi, it measures temperature, humidity and pressure. Messages are combined into one in the format set by the teacher and sent to the common topic of the broker. Further, the data is visualized using dashboard nodes.

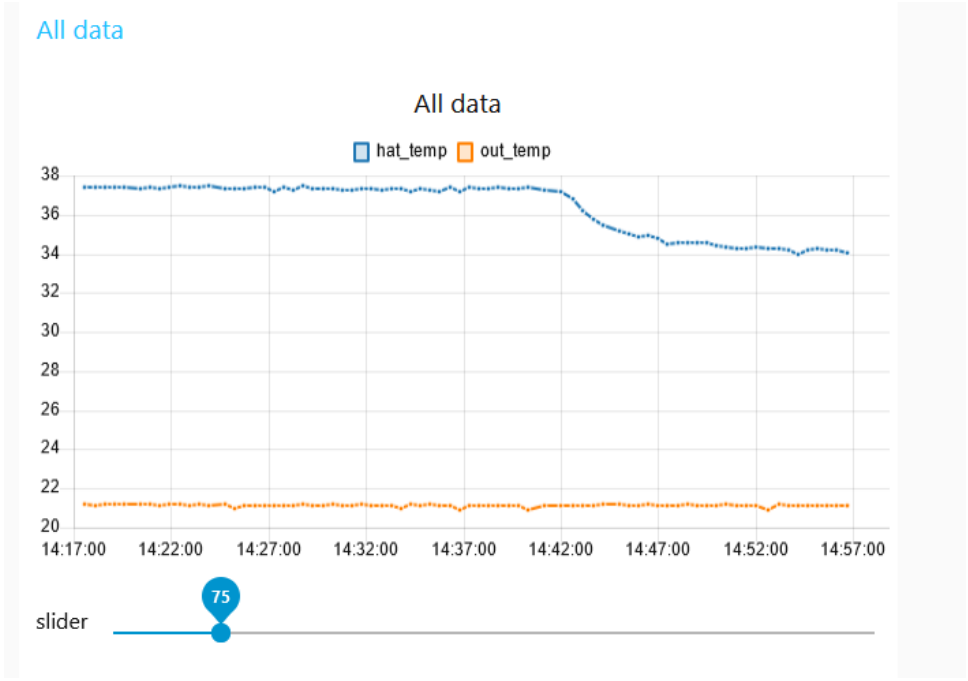
1.1.3 Interface

Node-red interface has four tabs for specific topic.

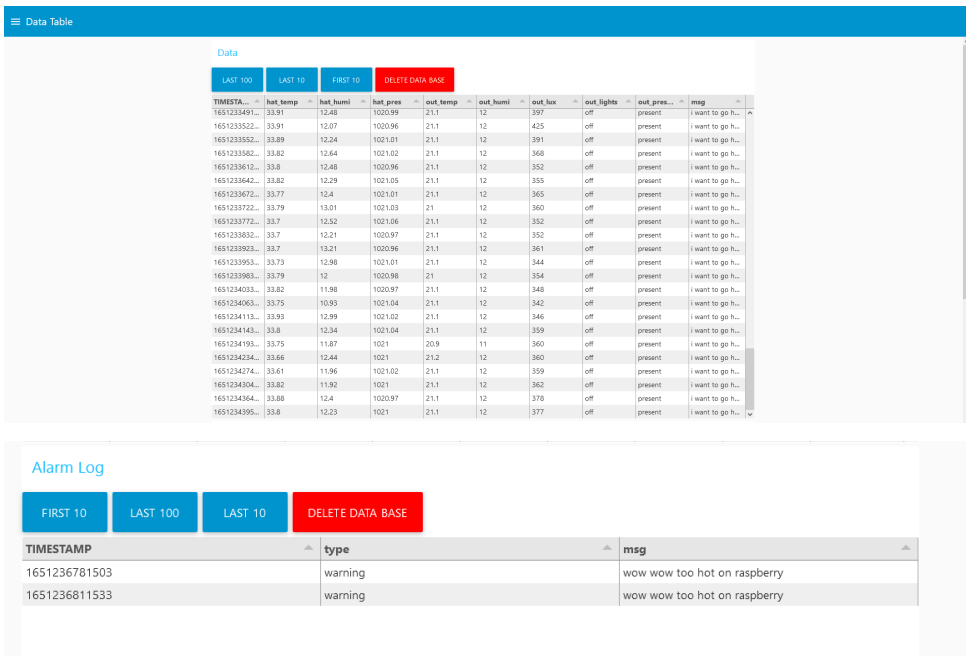
Firs tab Real-Time metering for our devices and all real time data. It also has two graphs for comparing arduino temperature with sense hat temperature and arduino humidity with sense hat humidity in real time.



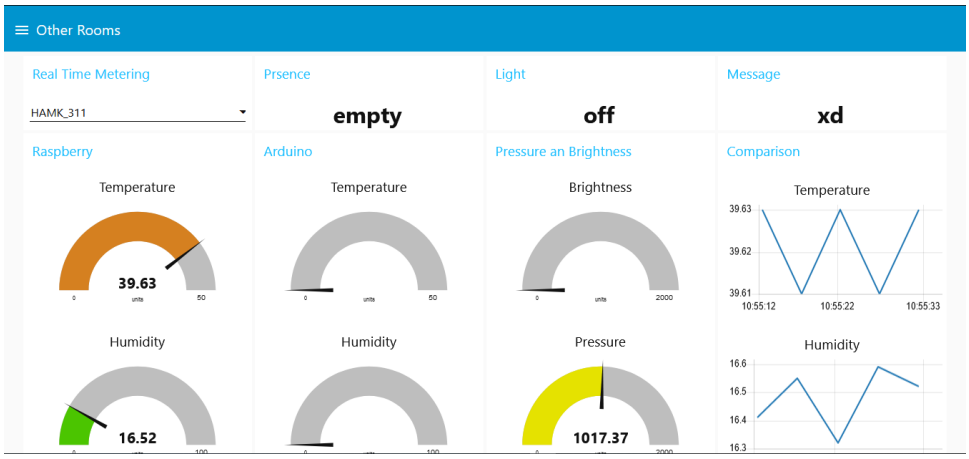
Second tab is Analysis for graph wich takes data from data base. Options of what user want to be displayed can be selected as well as range of points. Destriy button clears the graph.



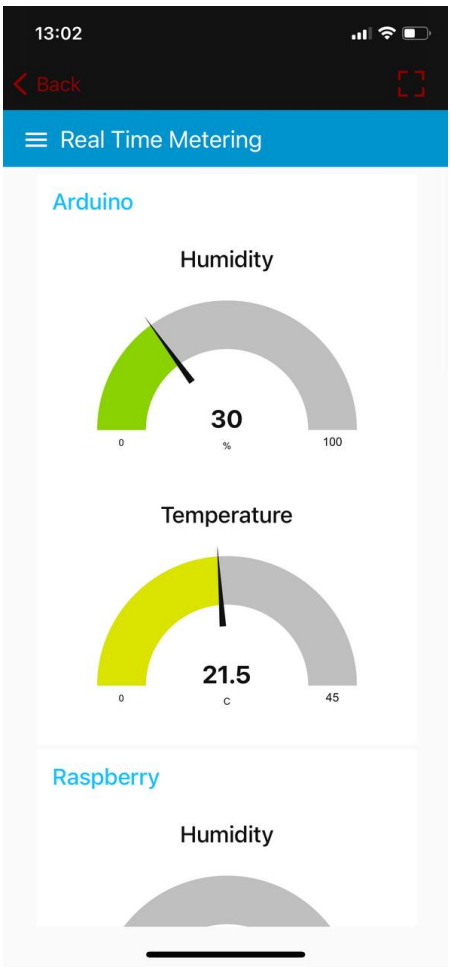
Data Table tab displays two data bases, myDB.db and AlarmLog.db, as tables with four buttons. Three for changing number of enties and one for clearing data base.



Last tab, Others Room, is for exploring other rooms data, room ID can be selected from dropdown menu and values will change with next message from this room.



Also, remote acess was implemented with “node-red-contrib-remote" pallete nodes. Dashboard can be view on your phone.



1.2 Functional Requirements

The functional requirements for this project were roughly defined in the document provided with the project. They were -

1. The Arduino was to measure the temperature, humidity, brightness and determine if the lights are on along with information about someone being in the room

2. Arduino was to forward the data to the raspberry using MQTT protocol in JSON format

3. The Raspberry was to transmit the data from the arduino and SenseHat to the server and was to display the results of other groups too

We successfully implemented all these. And for this we used Analog Ambient Light Sensor V2.1, DHT11 Temperature and Humidity sensor, DFROBOT Motion sensor and SenseHat. The main functionality of the project was to accurately display the data collected in a user friendly way which was done through NodeRed. The System must also collect the data in meaningful intervals which was taken care of by using an interrupt node in nodered.

The program must collect data on the arduino side and send it via MQTT which, as mentioned before was done using the ArduinoMQTTClient library. On the Raspberry Pi side, the program must must receive the data from the arduino in the form of a MQTT message via the special topic of the broker and collect the data with the sense hat, format it into a single message in the JSON format and send it to the general topic of the broker, visualization then made.

Additional features included storing data and creating an alarm log database, Visualisation of the stored data in a graph and table format, an additional tab with an option to select other rooms and then see the visualistions, remote access and dynamic light control based on the light intensity inside the classroom.

1.3 Required Connections

For editing the program, the raspberry should be connected to power supply and to laptop through Ethernet cable. Sense hat should be connected to raspberry.

2 System description

2.1 Arduino IDE code

The Arduino code can roughly be divided in x categories

- Libraries imported
- Defining variables
- Setup function

- Loop function

2.1.1 Libraries imported

We imported libraries that were relevant to the program. If statements were used to import libraries based on hardware specifications of the device used.

```
#include <ArduinoMqttClient.h>
#include <DHT.h>
#if defined(ARDUINO_SAMD_MKRWIFI1010) || defined(ARDUINO_SAMD_NANO_33_IOT) || defined(ARDUINO_AVR_UNO_WIFI_REV2)
  #include <WiFiNINA.h>
#elif defined(ARDUINO_SAMD_MKR1000)
  #include <WiFi101.h>
#elif defined(ARDUINO_ESP8266_ESP12)
  #include <ESP8266WiFi.h>
#endif
```

2.1.2 Defining variables

Variables were defined to store a range of data including WiFi authentication data, the MQTT broker to be used, the topic and the sensor values. We used variables of the types – char, int, float, String amongst others.

int port defined the port that had to be used, const long interval and unsigned long previousMillis were used to create breakpoints in the program without using the delay() function as it is not efficient.

```
const char broker[] = "iot.research.hamk.fi";
int        port     = 1883;
const char topic[]  = "HAMK/VLK/students/306"; //Topic of our device

const long interval = 10000;
unsigned long previousMillis = 0;

int count = 0;
float fTemp, fHum, light_val; //Sensor value variables
int t_attempt;
int h_attempt;
```

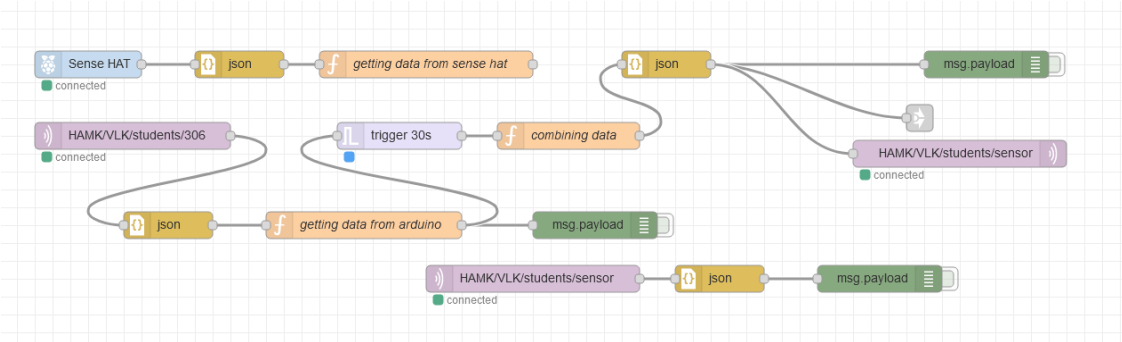
2.2 Node-Red

Node-Red program for Raspberry Pi consist of 5 flows:

- Getting data
- Storing data
- Visu
- Analysis
- Other Rooms

Each flow performs the function corresponding to the name and consists of logical blocks.

2.2.1 Getting data

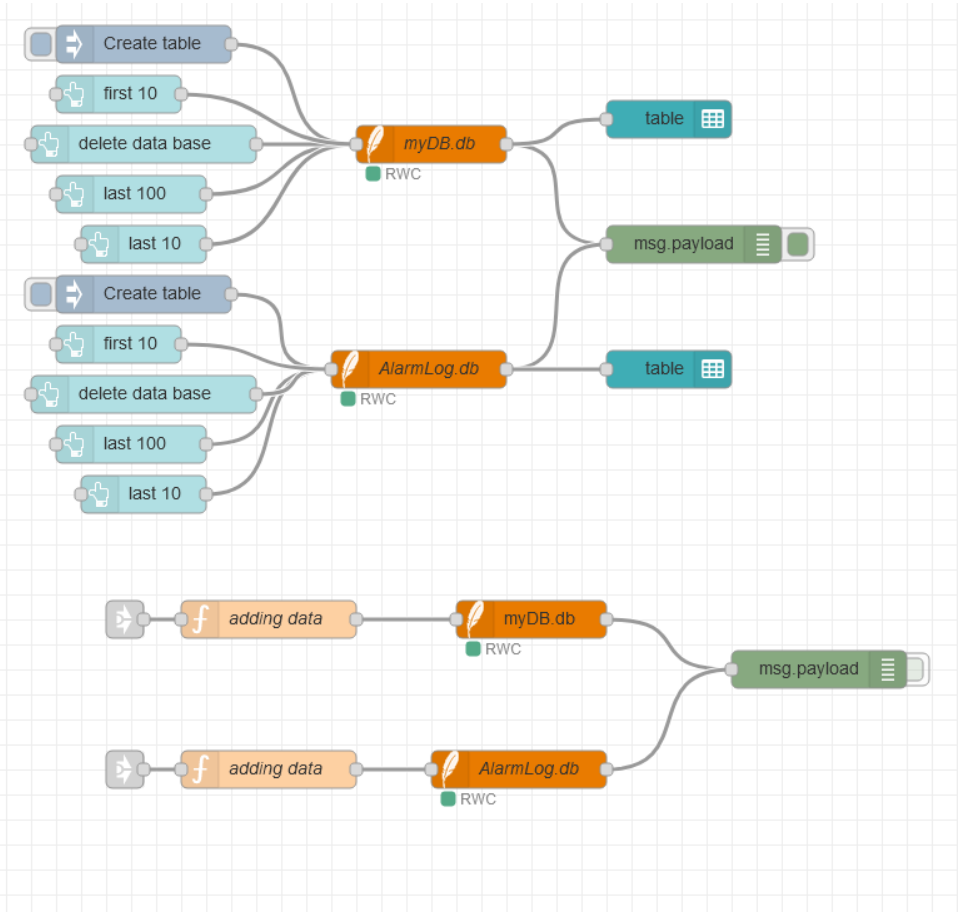


Functionality:

- Gets “environment events” data from Sense Hat of Raspberry Pi through “node-red-node-pi-sense-hat" pallete node and then converts to JSON object. Then implements program for getting this data and setting it to the flow with key name (see flows in appendix).
- Gets arduino MQTT message from personal broker topic (not common/shared one), converts to JSON object format and implements program for getting data and setting it to the flow with key name (see flows in appendix).

- Programm waits every 30 seconds and only then sends latest message. This done for optimizing the program. It also done not to pollute data base with incomplete data (program sends only when both arduino and sense hat are sending data).
- Implementing program for combining arduino and sense hat messages in right JSON format defined by teacher (see flows in appendix).
- Sends complete formatted message to common broker topic HAMK/VLK/students/sensor and to other flows for further processing.

2.2.2 Storing data

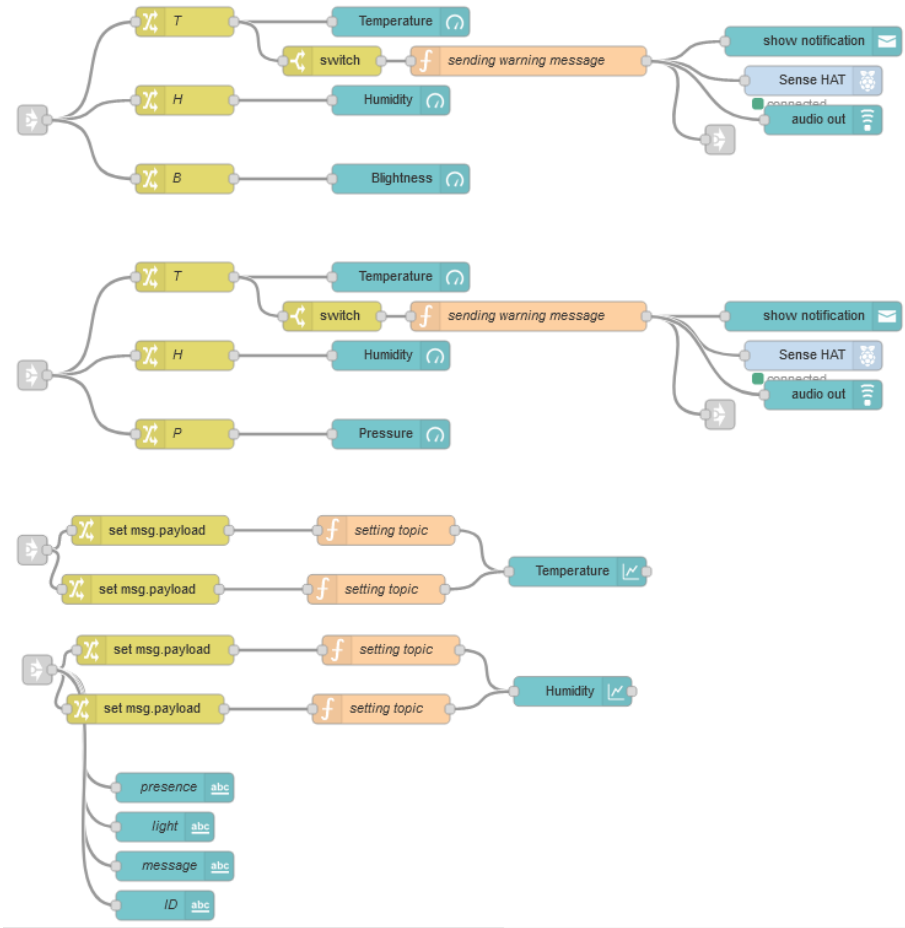


Functionality:

- Gets every message and stores it to myDB.db which is also displayed on dashboard on a “Data Table” tab with four buttons for displaynd different number of recods and clearing data base. SQL Query carried out via message topic.

- Gets every alarm message and stores it to AlarmLog.db which is also displayed on dashboard on a “Data Table” tab with four buttons for displaynd different number of recods and clearing data base. SQL Query carried out via message topic.

2.2.3 Visu



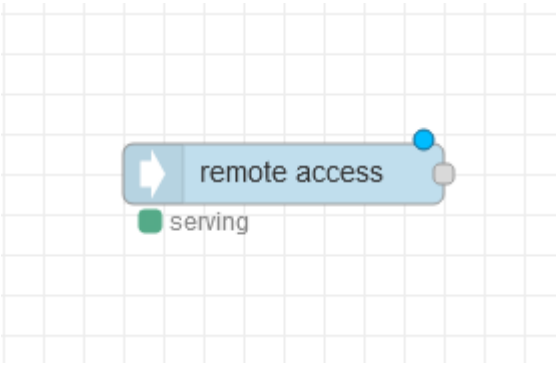
Functionality:

- Gets data and using “change” node changes payload to necessary value and shows it on gauge dashbording node from “node-red-dashboard" pallete. This done for six values: sense hat temperature, sense hat humidity, sense hat pressure, arduino temperature, arduino humidity and arduino brightness.

- Gets data and using “change” node changes payload to necessary value, change topic to correspoding device (arduino/raspberry) and puts it on a graph. This done because we want to display two values on a same graph (sense hat temperature and arduino temperature) thus they need to have different topic.

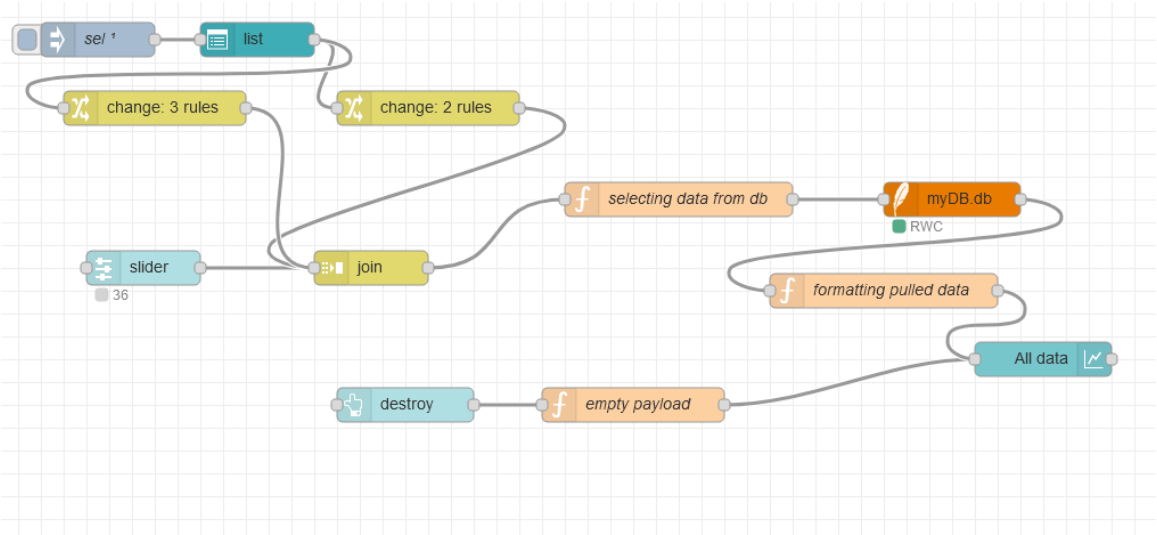
- Four values (presence, light, message, ID) displayed as text using text node andshort html line ({{msg.payload.something}})

- For temperature alarm is implemented. When value is bigger or equel to value set on “switch” node “function” node sets warning message which is then dispayed as notification on a screen and on a sense hat, and stored to AlarmLog.db



This node serves remote acess. Dashboard can be viewd on smartphone.

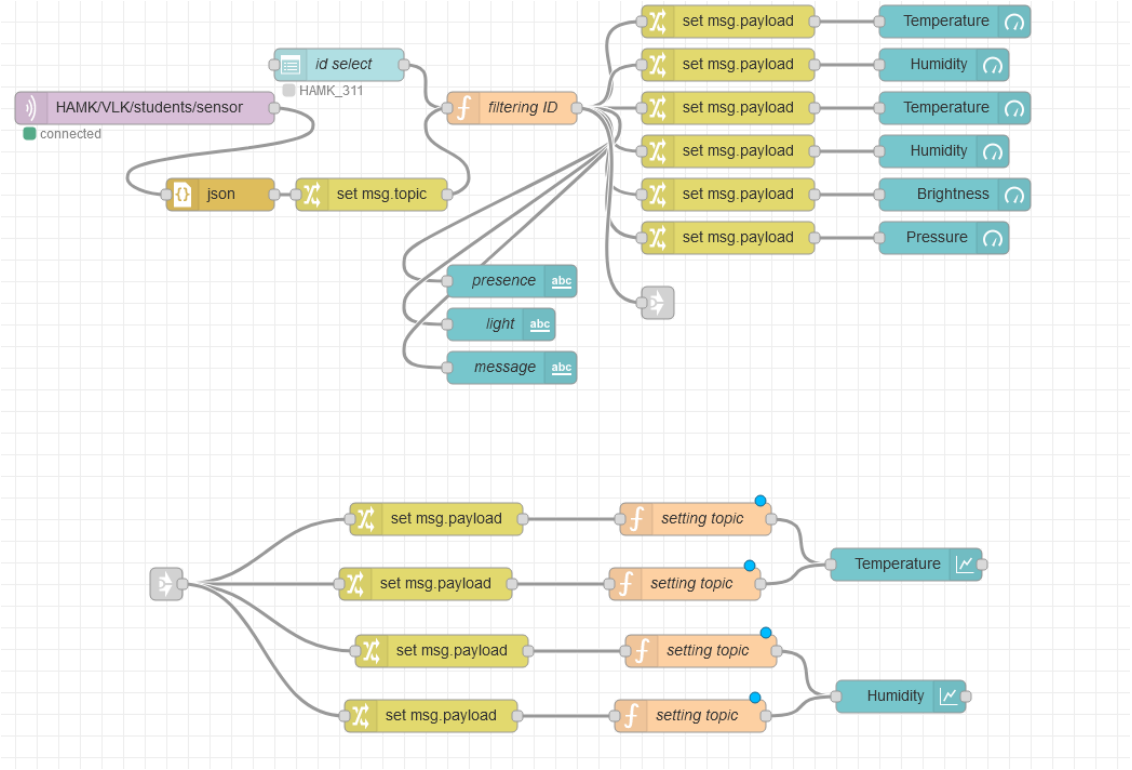
2.2.4 Analysis



Functionality:

- JSON list is passed to list node using inject node to create checkbox.
- Values passed to change node to separate them for easier joining with slider value. After joining message format is : {title: hat_hum, isChecked: true, limit: 20}
- Next passed to function node which implements program for selecting neccesary columns from myDB.db with DESC limit of slider value. (see flows in appendix)
- After pulling data from data base program for formatting message is implemented. Format is : {series: “hat_humi”, “sen_humi”, data: [0,...20], [0,...20], labels: “hat_humi”, “sen_humi”}
- Data is then graphed.
- Destroy button is added to dashboard to clear graph.

2.2.5 Other Rooms



Funcionality:

- Gets data from common broker topic and sets message topic to “data”.
- Adds dropdown menu to dashboard with “id” topic.
- Program then filters messages acording to selection of user in dropdowm menu.
- Plots and graph values in the same way as in Visu tab.

3 System testing

Mainly, all the work was done with Node-RED and Arduino IDE. As the code was written in Arduino to send data from the sensors that are connected to the board to the broker. So, to test that the data sent is right, we tested the sensors with the “Serial Monitor”.

We tried different rooms, to see if the code actually works and if it is right. Hence, we went outside to test the temperature data that was sent was right, we used a thermometer to measure the temperature. Then we went inside the campus, in a room, to also test the sensors. Everything seemed to work just perfect, the temperature data sent was the same as the one measured with the termometer, with some minor errors, of course.

Humidity was also tested in different conditions. Like the temperature, we went outside and indoors to test it.

For the brightness and the “light status”, it was easier to test, since all we had to do was to go in a dark room and observe the data sent and then turn on the lights and observe again the data received. We repeated the process multiple times to make sure it works good, without any errors. For the “light intensity” we tried bringing a flashlight near the sensor and taking it away. The data sent seemed to be correct.

For the Rasberry Pi sensors, we also tried the same testing ideas. Therefore, we took the Rasberry board outside and then in a warmer room to test the temperature and humidity.

4 Summary

4.1 Analysis

The main requirements have been satisfied: the Raspberry and the Arduino is sending data from the sensors to the broker as MQTT message in JSON format. We implemented visualization for real time meetering. Also, the data received were managed into a nice visualisation into Node-RED. Additional features like: storing data to data base, alarm log database, additional tab with other rooms selection and visualization, remote acces, were also implemented in the project.

A problem that we encountered, was getting out the SD card from the small port from the Raspberry Pi which actually broke. The cards are very sensitive and all the data was lost. Then we rebuilt everything in a nicer format.

4.2 Teamwork and peer evaluation

Each member of the team had its own contribution.

4.3 Shortcomings and improvements

The operation of the final system complies with the specifications set earlier, however, in the operation of some functions, shortcomings can be identified.

The graph reflecting the content of the data base works intermittently due to the fact that the destroy button is not connected to the list and therefore does not reset the previous values.

The alarm log needs improvement as at this stage it only saves warnings about too high temperatures and not about errors or any other failures.

The datebase also has a limit on the number of records, and since messages are sent every 30 seconds, it quickly becomes clogged. It would be useful to apply a trigger with a delay to save every tenth message or save mean of every hour. We have not yet developed this feature due to the lack of time.

References

Appendix 1: Name of Appendix

Appendix 2: Node-Red Flows

https://github.com/anna20121/RaspberryPi_Project/blob/main/flows.json