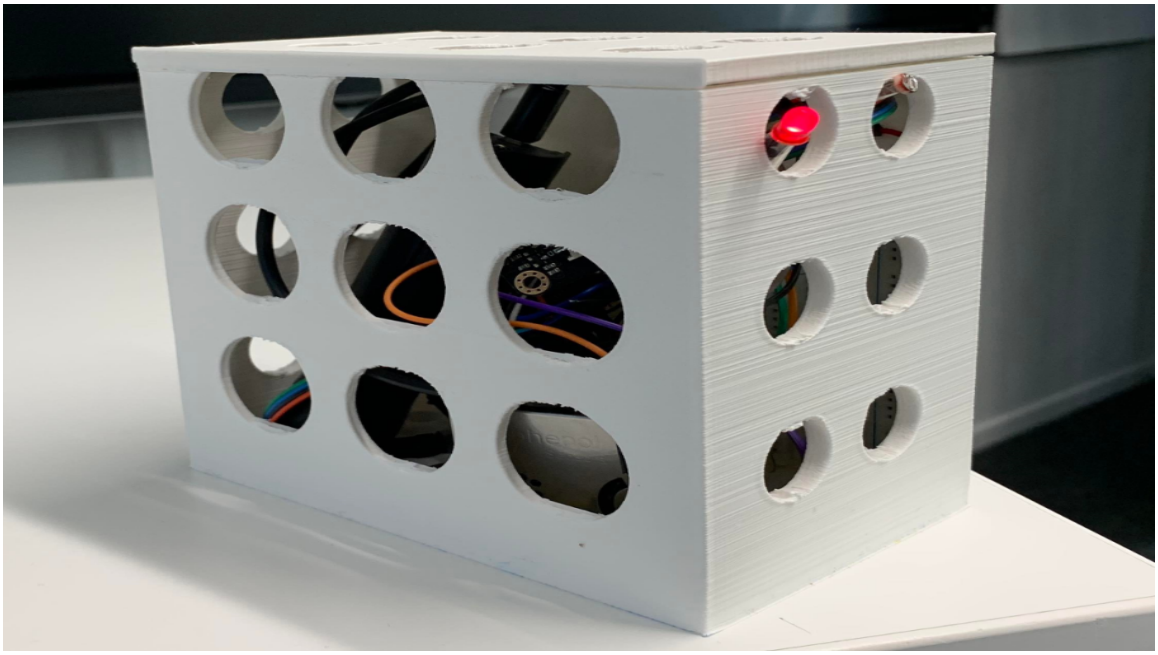
2. Semester IT-Technology, Project III ITEK-E21a

Sensor in the study room.



Author: Jaroslav Komel

Name of master's supervisor:

Anders Toft Eriksen

Peter Jeppesen

Jonas Wehding

Kaj Norman Nielsen

Kristian Pontoppidan

Submission date: 25.05.2022

Characters: 34461

2. Secondary front page

In this project, I will show how to build the measurement sensor station using the acquired knowledge, which I have gotten in Project management, Embedded Systems, Network Technology, Python and C programming. In this project I am required to work 8 hours each day. My goal is to finish the project before the deadline. The reason for this occurrence is that it will help me look for bugs, problems and mistakes in the report and the project. My goal is to finish 2-3 days before the date of the deadline with full response of methods, formalities, design, block design for electrical schematics, describing solution, video, proving, technical solution, block diagram, mind map with Click up, installation WIFI connection, MariaDB, Mqtt, configuration Raspberry pi4, testing and test methods after it is done than check grammar.

3. Summary.

Project's goal is to return our sensors and rebuild it with the new sensors and skills we have gained from the 2nd semester project management, WIFI connection, security, embedded and programming classes. To compete in new requirements. The task is to make connections to sensors that take measurements from the room where it's placed, whilst reporting its environmental data. The next challenge is to send measurement to the serial monitor and Mqtt. Then data sent from the MQTT to the database. Get data from the database to the web page. To complete these challenges, I need to implement new components Temperature, Light, AIR, Dust and GPS sensors with integration on microcontroller ESP32. The sensors are needed for the challenge to report the environmental data. Then I focused on getting a connection from all sensors to the ESP32 and adding the measurements to sending data using MQTT. Next, I tested the sensors to make sure that they worked, after which I connected them to ESP32. Used the Raspberry pi4 as mqtt broker.

4.Table of Contents.

5. Introduction

Sensors are based on the C code to make the measurements, I can see this data in the Serial monitor. The data from sensors send data to Mqtt and then send the data to a database called measurements. The Python code collects data from the database and then reports them back to the web page and displays the data in the web window. This project is to highlight the new skills I have gained from our 1-2 -semester classes including project management, wireless, networking, programming and embedded.

6. Methods and tools.

The solution is following the plan with date by using click up and remember the time can be up to 8 hours a day. Writing the code for the sensors in the C language. Testing C code on the Arduino io. The connection code in python. Database on MariaDB. Web page is in html languages. Writing and being safe with projects use google doc. Diagram I made a United diagram in the drawer. Using the different test methods, some: Exploratory, Usability, Regression. When I made all the codes on my pc . All the code is saved in Github.com on my page name Sensors-C code.I used the Fusion 360 3D Printing to make the white box for all sensors.

7. Requirements.

Required properties for the Project. Case 2: Sensor in study rooms.From the beginning of project,because I am writing a project alone. The agreement with the masters is was, I need to do only 30% of the entire project. I am not using a humidity sensor in the project.

That's why I took only three sensors: Temperature, Air and Light for the full project. The requirements in the project are sensors will make recordings of measurements as the day goes along, the data from sensors is collected then transferred through the ESP32 to mqtt and to MariaDB database. The measurements from sensors should be sent through the mqtt and then the sent data from MariaDB database should be displayed on the web page.

Sensors should collect data every 5 minutes, use the server MQTT and show this in the Serial monitor window or a web-window. Program will go into deep sleep mode for 20 minutes when the Night or value is under the 200 lux and Led will off.

GPS sensor + antenna will show the position of the measurement location in the serial monitor.

Light sensor will show if it is day or night and measure the light level.

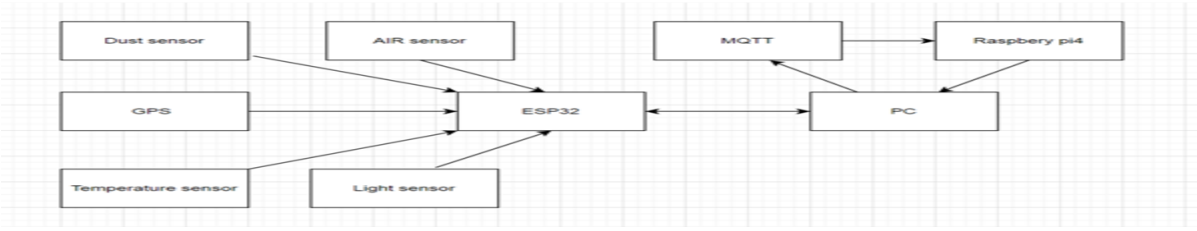
AIR sensors will make measurements of AIR quality in the room.

Dust sensor will have taken the measurements of dust. Then the dust reflected off the sensor fan. Doing this would result in a spike in the serial plotter or serial monitor. The data can seem thick because all 3 particle size values are shown together (PM1, PM2.5, PM10).

Temperature sensor will show the data of measurements in the room. I'm adding the new features with led is on. When the light level shows that it is night outside the led is off. The program then would sleep for 20 minutes. At this point this means that the extra feature which is meant to send a signal to for example a cleaning robot, so the room is empty from visitors and ready for cleaning.

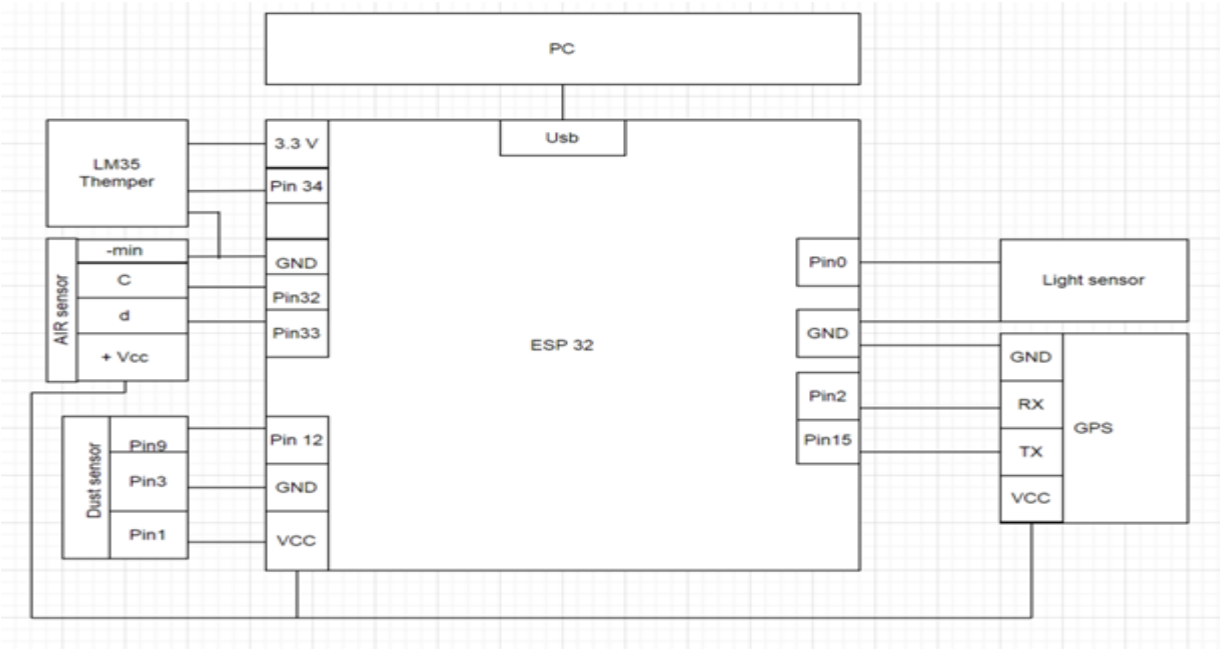
All the Software for sensors built for the C code that include this function to collect measurements: temperature, light, Air, Dust, Gps position and all of them will connect to the microcontroller Esp32, then send data to the Mqtt broker based on the Raspberry pi4. Collect the data from mqtt and show it in a web page with Python code. Make the test of all sensors and general block system tests.The administrative system will run us on a central Linux OS.

8.Design Block diagram.

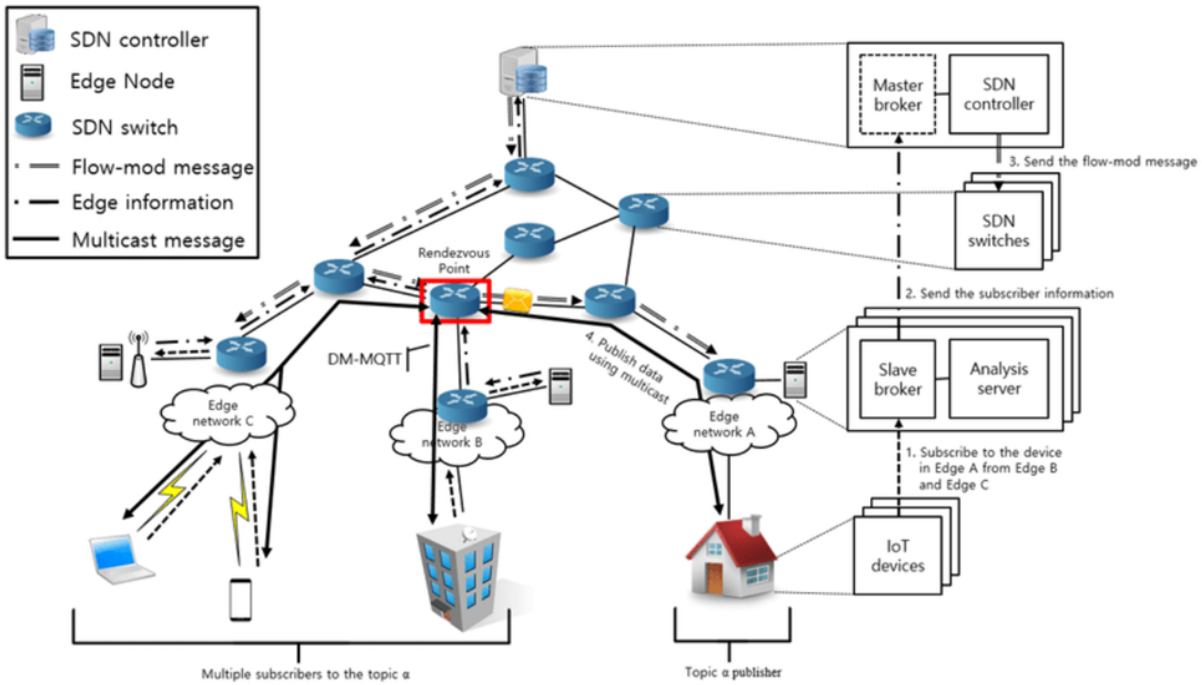


All sensors are wired to the Esp32, the 5 volt will be supplied from the usb port or the pc or can be a power bank. Pc is connected to the mqtt and the raspberry pi4 via wireless connection.

9.Design Electronic Schematic. Connection Temperature sensor, Light photo sensor, AIR sensor, Dust sensor, GPS to the Esp32 and PC as the usb 5 volts.



9.1 DM-MQTT architecture in edge computing, it's to show how data flows from IOT devices.

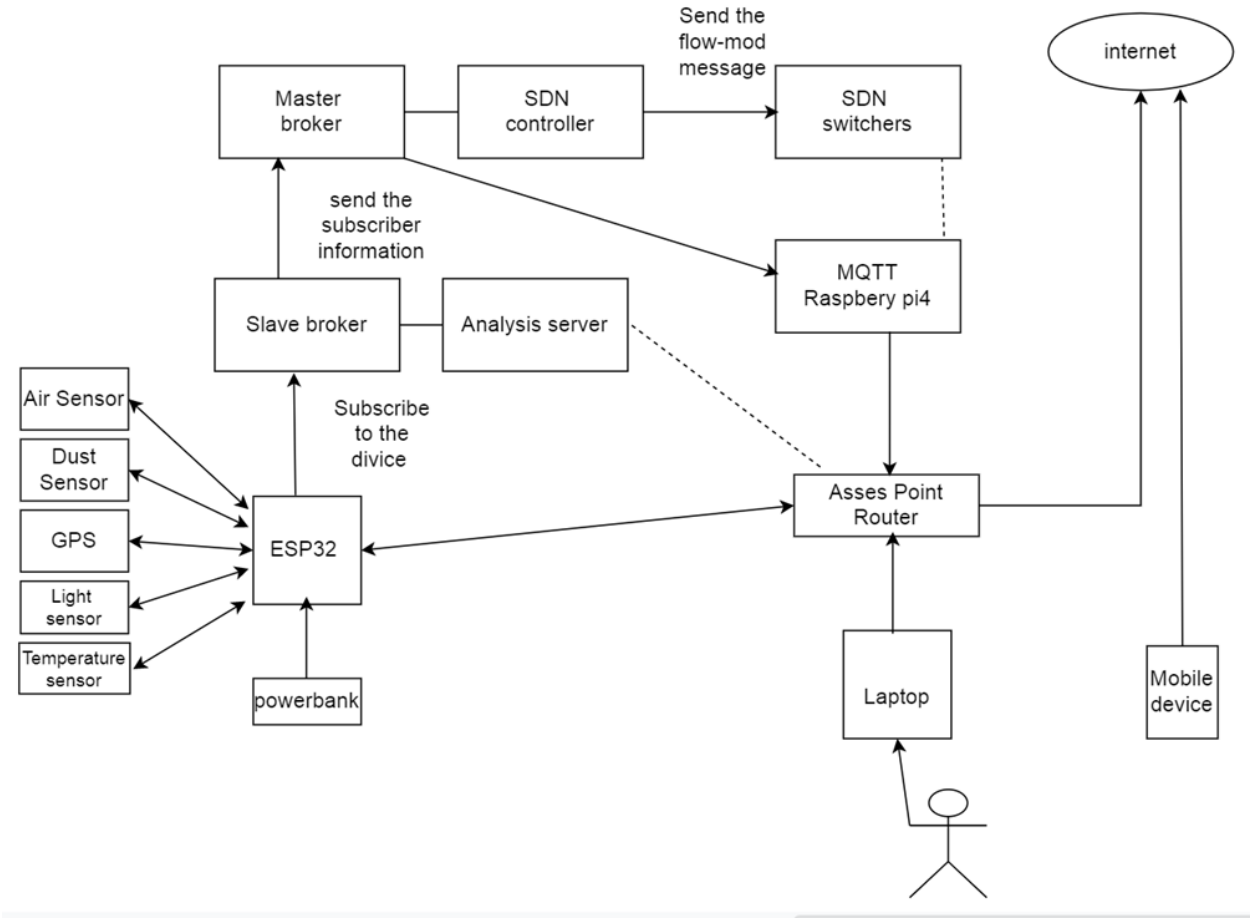


DM-MQTT architecture in edge computing.

https://www.researchgate.net/figure/DM-MQTT-architecture-in-edge-computing_fig1_327610389

In this picture one of the examples is being shown to demonstrate the architecture in edge computing. How the IOT devices subscribe to the master mqtt broker and send the flow mod message. The same connection applies for my IOT device that sends the data from the sensors to the server and to the users, which are subscribed to the mqtt topic.

9.2 Mqtt connection, on the diagram, is a software-based way to automating and ensuring network services. In a software-defined networking architecture, an SDN controller oversees data flow to optimize network management and application performance. The SDN controller platform is typically installed on a server and communicates with switches via protocols. (SDN stands for software-defined network.)



Laptop.

The laptop has both C code and python code. The C code is meant to connect to the WIFI, then connect to mqtt,which receives measurements from all sensors. Mqtt sends

this data to the Maria DB database, and the python code receives this data from the database and shows it on the Web page.

Wireless connection. Local area network, known as LAN, is an access point (WAP). In this case it is the ITEK 2nd modem. This connection will be used between the laptop, broker, Esp32, Raspberry pi4.

Raspberry Pi4.

The Raspberry pi4 is being used in this project. It will connect the mqtt broker with Esp32 using MQTT protocol. Then it will make the connection for messages to be sent and received by the Esp32 and laptop on the different tasks.

Esp32.

Is a microcontroller for connecting all sensors, model of this Esp32-Wrover-Dev, which what I'm programming on C. Esp32 will use gpio pins to power the sensors which send the data to the mqtt broker trough Esp32 and showing to the web page.

Temperature sensor LM35.

Temperature sensor that outputs an analog signal which is proportional to the instantaneous temperature. The output voltage can easily be interpreted to obtain a temperature reading in Celsius. Sensor, it does not require any external calibration.Data will be seen finally on the web page.

Air sensor (SGP40)

Air sensor measures the quality of the air and sends it back to the Esp32 and the broker and to the database.Data will be seen finally on the web page.

Dust sensor (SM-UART-04L)

Dust sensor is reporting measurements of the dust in the air, the amount and size of the particles such as PM1; PM2; PM10.Sends data only to the serial monitor.

Light LDR Photoresistor.

Light sensor is necessary to detect the presence or the level of light in lux. It is an electronic component sensitive to light. When light falls on it, the resistance changes. The resistance values of the LDR can vary by many orders of magnitude of the resistance dropping as the multiplicity level increases.Data will be seen finally on the web page.

GPS NEO-6M

It can rely on 22 satellites on 50 channels and the highest level of sensitivity, i.e. -161dB tracking, while consuming only 45mA of supply current.

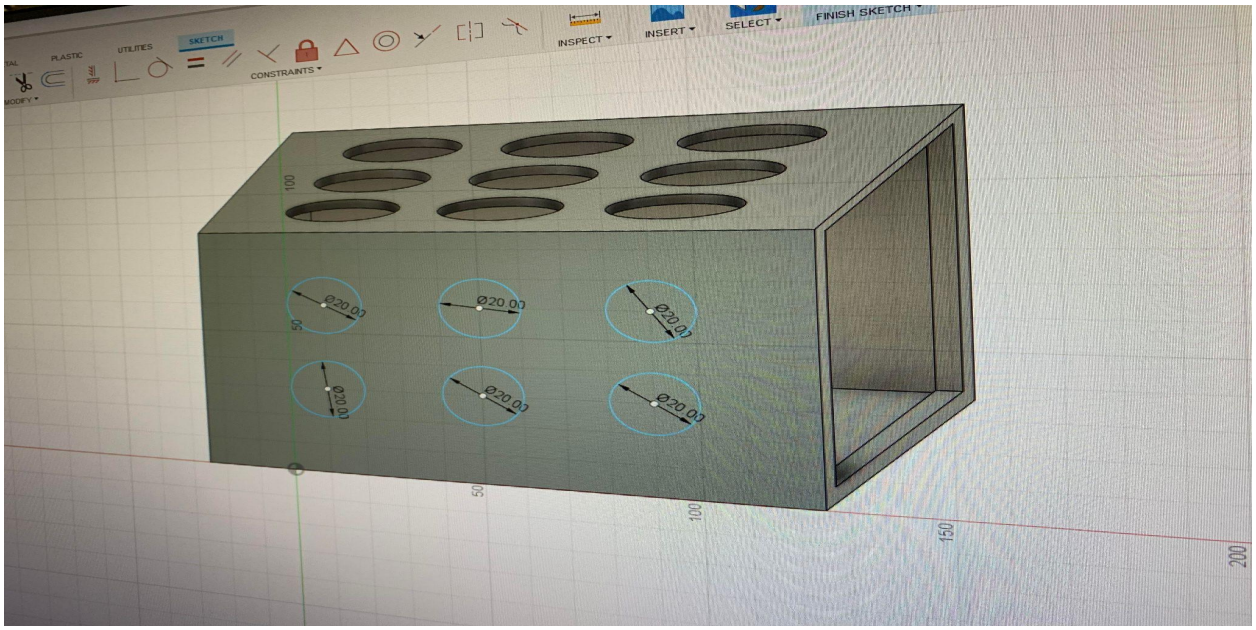
Unlike other GPS modules, it can perform up to 5 updated positions in a few seconds with a 2.5m horizontal position suspect. The u-block 6 detection engine can also maintain a time to first appearance (TTFF) of less than 1 second.

Power bank

The power bank has one 5 volts output, the 2.1 Amp to power Esp32.

My own idea was to use the 3d print to make the box for sensors in Programm the Fusion 360 3D print.

Is one more extra future. I have time to make the white magic box for all the sensors, for that I'm using the size of a box 140 mm width,140mm length and 80mm tall.



Made the hole(30 mm * 9hole) on the front, (20 mm *12 hole from both side and under box is 10mm * 24 hole). Thickness of all the walls is 5mm.

10. Implementation / integration test.

Firstly, I started with the Light **LDR sensor** to make a connection. I selected the pin GND to GND, another pin divided or split to Pin name (VP = Pin36) on Esp32 and 3.3 volts, one Led to Pin 4 and GND.

The sensor used is a photoresistor, which is also called light dependent resistor or photocell. After checking this connection, I make a C code with Led to confirm, when the sensor is measured under 10 % brightness and printing in the serial monitor its Night led is OFF. I'm using a sensor not only to detect light but also to measure the brightness level of the ambient light. This sensor has two pins. It is a kind of resistor, we do not need to distinguish those pins. Pins are symmetric. Sensor works if the more light the photoresistors face is exposed, the smaller its resistance is. Because, by measuring the photoresistors resistance can now know bright ambient light is in lux.

When I'm connecting a pin of the photoresistor to an analog input pin, I can read the analog value from the pin by using analog Value () function, then I can know the light levels relatively.

10.1 Test .1 Light LDR sensor.

1. Connect theEsp32 with a Light sensor, one pin to the GND, second to the Pin 0 and through the resistor(220 ohm) to the 5 volts.

2. Check the connection from usb to Esp32.

3. Run the program on C code.

4. Open serial monitor to see revising measurements.

5.Check connected Led ,through resistor(220 ohm) the GND and Pin 4.

Actual result: When I cover the light sensor to make the Night, the LED is off.

Expected result = Actual result.

Special please from the requirements I need to explain for the sleep function in the main C code. When the Light gets measurements, if analogValue < 200, it will write the Night and Led will switch off, then the program will sleep for 20 minutes. In the future Led can adopt the signal for cleaning robots for on the night robot will come to clean the room or another solution to shoot down the air condition to save electricity.

esp_sleep_enable_timer_wakeup (1200 * 1000 * 1000);

Where is 20 minutes equal for 1200 in second *1000 millisecond and *1000 microseconds, because I'm using microseconds for the program.

Secondly, my choice was on the **SM-UART-04L Laser Dust Sensor**, **tested the Dust sensor, connected the Pin 1 to the VCC, Pin 3 to the GND, pin9 to the pin14 on the ESP32.**

Dust Sensor called "SM-UART-04L" was tested by taking off paper, by rubbing the papers on the ground to collect dust. Then rubbing the dust off in the vicinity of the sensor fan.Doing this would result shown in the Serial monitor. The 3 particle size values are shown together (PM1, PM2.5, PM10).

10.2 Test 2. SM-UART-04L Laser Dust Sensor.

1.Navigate to check connection wires (from sensor Pin1 + to the Vcc (5 volts), Pin 3 - to the GND and Pin 9 to the Pin 12 on ESP32).

2.Upload code from Arduino id to the microcontroller ESP32.

3.Open serial monitor to see consumption and measurement.

4.On the monitor indicators change value up.

Actual result: as grown-up consumption of dust increases, we can see on the serial monitor.

The Air sensor SGP40 was easiest because it is coming with me from the second project. To test the Air Quality sensor, I used “hand spiritus” in the vicinity of the sensor to see spikes in the Serial monitor. In general, the SGP40 was relatively easy to set up, because I’m have a working library called I2C.Then I’m connect AIR sensor from D to Pin33, from C to Pin32, -(minus) to GND, +(plus) to VCC as 5 volt.

10.3 Test 3. SGP40Air sensor.

1. Check the connection to wires from the Air40 sensor and the pins (+ to VCC 5 volts) and (- to the GND).

2. Be sure that it's connected to the wire from Air 40d Sensor to D to pins 33 and C to 32 on ESP32.

3. Upload the SensorAIR40code inclusive library to the ESP microcontroller.

4. Wait for around two minutes to warm up the Air40 Sensor.

5. Open the Arduino to see measurement in the serial monitor.

6. Spray near the sensor spiritus or other stuff.

7. Register if the measurement is going up.

Actual result: show that the pollution is changing.

This result was what I expected.

GPS NEO-6M comes to me with an external antenna, and has header pins, so I don't need to solder them**.** I'm connecting the pins from GPS to the Esp32, from VCC to VVC, from Tx to Pin 15, from Rx to Pin 2, from GND to GND.

Default baud rate:9600 bps. Start to get the raw data reading. I need to connect all the wiring and use a C code open serial monitor. Measurement can work with the raw data from the GPS or can convert those NMEA messages into a readable and useful format, by saving the characters sequences into variables. To do that, I'm going to use the Tiny GPS++ library.

This library makes it simple to get information on location in a format that is useful and easy to understand. I use the Tiny GPS++ Library.

10.4 Test 4. GPS NEO-6M

1. Navigate to the connection GPS NEO-6M with Esp32.
2. Check all wires connected, some are written up.
3. Open the Arduino id.
4. Upload the C code.
5. Wait about 5 minutes.
6. Open the serial monitor to check received data from the module Gps, it sends location to the serial monitor.

Actual result: time and gps position is shown in serial monitor baud rate 9600.

Temperature sensor LM35. I use LM 35 this sensor because it was the most often used in the second semester. Before I start connecting the sensor, Esp32 needs to be disconnected from usb or another power supply. Then I connect the sensor to the breadboard, with the face of the sensor from the left side is 3.3 volts, on the pin in the middle connect to Pin 34 in Esp32, and right pin through resistor(220 ohm) to GND.

10.5 Test 5. Temperature sensor.

1. Check connection from Esp32 to the LM35.

2. Upload the C cod to Esp32.

3. Wait for 1-2 minutes.

4. Select to open the serial monitor.

5. Control of measurements is up and down, compared to the temperature outside of the sensor.

Actual result: problem with data temperature showing result in serial monitor temperature = 00.00 `C.

Expected result: measuring in serial monitor receives the data equal to the temperature outside of sensor.

For some reason, the problem came from the temperature sensor, who stopped to send measurements to the serial monitor. I think the Esp32 sends the wrong current to the sensor pins, because when I check on Arduino the self-program for the temperature works well.

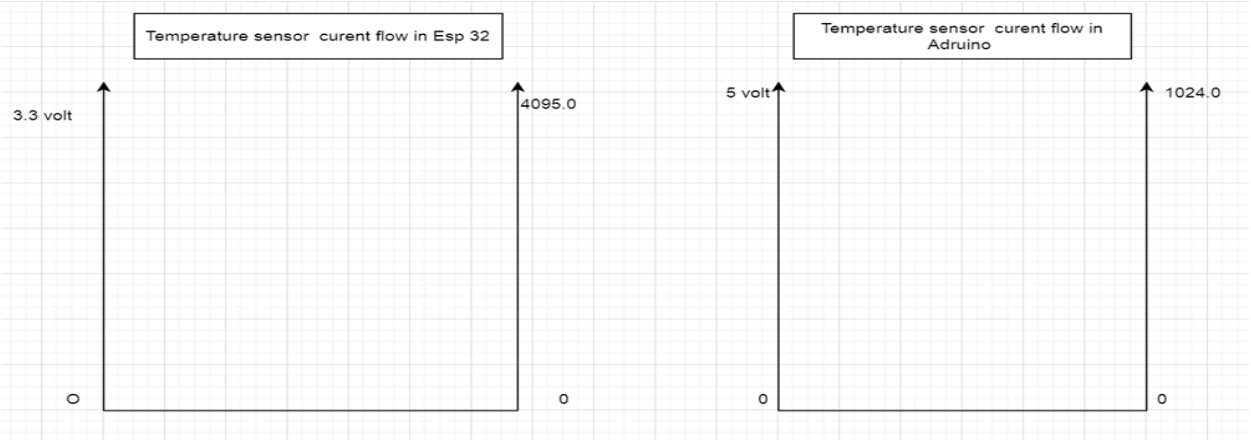
I'm trying to start from the beginning with new temperature sensors named LM35, and make the new version of cod, and temperature sensor works, connection for Wi-Fi and mqtt work, then when I'm connected to another sensor all is crashing. What is the best solution I'm asking colleges and other. Answer was that the code is too long and needs to be converted into a function. I hope this can solve the problems.

I'm sinking that to return with the code where all sensors work and make the program with a new temperature sensor.

Then I'm continuously searching for a problem with the temperature sensor. Why the pins in ESP32 get the wrong current, and it was pin number 14 who made the problem, because in calculation for the Dust sensor it makes a mistake for the general code. Measurements with this wrong position of pin14 a current flow gives the effect of the sensor sending data and showing on Serial monitor 00.00 degree. After I'm researching with Arduino, 2 more temperature sensors, I see that one is showing variable temperature so I'm starting two use this third number LM35.

Changing output of the temperature sensor from Pin 14 to the Pin 34 gives 100% success. My program works well, taking the measurements from Temperature, Light, Air, GPS, and Dust sensors.

Next problem which I got from the temperature sensor was the calculation of current which sends to the pins power in volt. I'm using Arduino for All sensors to check the C code, because the waiting time is short in Arduino, then I wait too long time receive an answer from Esp32.



From the temperature sensor in Arduino at the maximum operating temperature for the LM35 of 100 'C, the output pin would be about 1 volt DC. The temperature accuracy is high, with +/- .5 at 25 degrees C, and about =+/- .9 at max.

In an arduino I have 5 volts divided into 1024 divisions, so the input number divided by 1024 gives a fraction of 5 volts. This number is now easily used to convert to Centigrade using the 10 mV per degree C linearly proportional scale that the sensor operates within. For the LM35 the temperature range is smaller than others in the series, at 0 to 100 degrees C.

The C code I'm uses the formula to advance float temp = (sensor / 4095.0) * 1.05; Where 1.05 is the current in volts.This calculation is proportional to the Esp32, only 4095 divisions to the 3,3 volts.

When all the sensors were tested, I started to combine the C code for one long spaghetti code. Sorry for that, I was very busy this month and worked very hard every evening on the project and other student work. For me it was difficult from the beginning to organize the structure on C code of which one sensor needs to be first or next, following that list of the sensors and Gps with the antenna. After the consultation with Master of project in class and explanation self-project it's all project real big for one member, conclusion and advice was continued without GPS, Humidity, Dust sensor.

Continue to the implementation by sending data only from Temperature, AIR, and Light sensors to Mqtt and so on.

Description of the Raspberry Pi installation/Configuration.

What is installed/configured (why and how)

IP Addresses to Raspberry pi4. A simple Network description

1. IP-number configuration for Client, Server, and Gateway

2. Network

3.Netmask

Raspberry pi programming my special static IP Address to the Raspberry pi for the sensors in the room is 10.120.0.220.

Password: xxxxxxxxx.

A simple Network description.

1. IP-number configuration for Client, Server, and Gateway.

2. Network is access-point “ITEK 2nd” password: “Four_Sprints_F21v”.

3.Subnet mask is 255.255.0.0

I will start installing the system through a special utility Raspberry Pi Imager.

Firstly, put the SD card in the SD card reader slot and erase the data from it with the help of the SD card formatter program. After the SD card is empty, run the Raspberry Pi Imager program. Here I’m to choose the Ubuntu version 20.04 (Pi3/4),64 bit and download it.

After completing the download, I must insert the card in the Raspberry Pi. Then connect the Raspberry to a power source. After getting this done, the Raspberry Pi must be connected to a keyboard and a monitor. Afterwards I got into Putty, using our special IP Address. When I'm finished, then I can use the Putty interface. To get into my Raspberry Pi I need to enter the default username ‘ubuntu’ and password "ubuntu". After entering the password, the system immediately tells me to create a new password. (The

password cannot be the same as the username). When changing, first we enter the old once again, and then 2 times the new one.

Now I'm continuing with the Internet connection set up. To configure Wi-Fi, I need to switch to superuser (root) mode. To do this I need to type sudo before the command. If I do not use sudo I cannot access the set-up.

The command: sudo nano /etc/netplan/50-cloud-init.yaml and remember to install IP address is 10.120.0.220 and access-point "ITEK 2nd" password "Four_Sprints_F21v".

Next, press the key combination of Ctrl + O, then Enter to save the file. With the key combination Ctrl + X exit the text editor "nano" and press 'y' to save.

After booting the system, enter the username and password again. I will update the program to the latest version with command:

sudo apt update and sudo apt upgrade.

After each command, press Enter and wait.

Next, install the graphical shell using: sudo apt install ubuntu-desktop

Then I am waiting, because this download takes an exceptionally long time. Next, reboot the system using the sudo reboot command, and as a result, the graphical shell is loaded automatically. Installation of WIFI is completed.

The following commands I'm used to install mosquito and mosquito-clients to set up MQTT broker on RPi:

- sudo apt-get update
- sudo apt-get install mosquitto
- sudo apt-get install mosquitto-clients

The Python-client and ESP32 will connect to the broker since their respective client handlers will be told the broker's IP address.

Installation of database in MariaDB.

Firstly, I make the database name "measurements ", then directory for the data in table "temperatures" with measure_id, temperature, vocindex, light in varchar 255 characters, which may be more than 255 bytes. In the database will incoming data from all the sensors.

```
MariaDB [measurements]> create Table measurements.temperatures(
    ->  measure_id INT(10) NOT NULL AUTO_INCREMENT,
    ->  temperature VARCHAR(255) default NULL,
    ->  vocIndex VARCHAR(255) default NULL,
    ->  light VARCHAR(255) default NULL,
    ->  ts timestamp NOT NULL default CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    ->  PRIMARY KEY (measure_id)
    ->  );
Query OK, 0 rows affected (0.043 sec)

MariaDB [measurements]> desc temperatures;
+-------------+--------------+------+-----+---------------------+-------------------------------+
| Field       | Type         | Null | Key | Default             | Extra                         |
+-------------+--------------+------+-----+---------------------+-------------------------------+
| measure_id  | int(10)      | NO   | PRI | NULL                | auto_increment                |
| temperature | varchar(255) | YES  |     | NULL                |                               |
| vocIndex    | varchar(255) | YES  |     | NULL                |                               |
| light       | varchar(255) | YES  |     | NULL                |                               |
| ts          | timestamp    | NO   |     | current_timestamp() | on update current_timestamp() |
+-------------+--------------+------+-----+---------------------+-------------------------------+
5 rows in set (0.006 sec)

MariaDB [measurements]>
```

10.6 Test 6. Check Database in MariaDB.

1. Open MySQL Client (MariaDB)
2. Enter password: xxxxxxxxxxxxx.
3. In the line MariaDB[(none)]> write “Use measurements” in line.
4. Select to add after MariaDB[(measurements)]>SHOW DATABASES;
5. Select to add after MariaDB[(measurements)]> show open tables;
6. Select to add after MariaDB[(measurements)]>desc temperatures;
7. Select to add after MariaDB[(measurements)]> SELECT * FROM temperatures;
8. To stop running the data press CTRL + C.

Actual result: I can see the data from table temperatures.

Expected result = Actual result.

My travels continuously follow the connection between the database in mariaDB and mqtt broker, to make the good start it's helpful to use the test.mosquitto.org as host which I change with my IP 10.120.0.220. On these options in Python code, I'm using the library to import the paho mqtt client as mqtt and use MariaDB database.

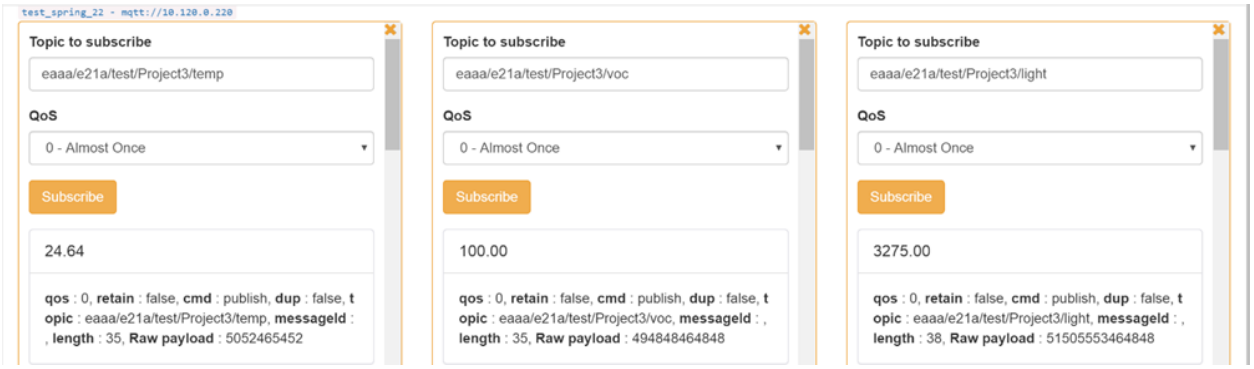
Useful was the function of add in C code the three measurements to database and three Topic to subscribe add three different topics at the:

eaaa/e21a/test/Project3/temp;

eaaa/e21a/test/Project3/voc;

eaaa/e21a/test/Project3/light;

I think it's a good topic-cod for security, small and big letters and numbers. Because in the project I will use my Raspberry pi with IP 10.120.0.220 address to change the mosquito on my Host in Mqtt box.



10.7 Test.7 Check the Mqtt.

1. Open Mqtt box.
2. Write the three topic to subscribe:
    a) eaaa/e21a/test/Project3/temp;
    b) eaaa/e21a/test/Project3/voc;
    c) eaaa/e21a/test/Project3/light;

3.Press on the three orange buttons "Subscribe".

4. Check the data published every 5 minutes.

Actual result: mqtt box receives the last data from MarieDB database.

My developing an IoT application should be security from the very beginning of the implementation process. In mqtt security, it is divided into several levels.Can be the different types of attacks each level prevents. The purpose of the IP address is to provide a lightweight and easy to use communication protocol for the Internet of Things. The IP protocol itself defines only a few security mechanisms.

IP implementations typically use other modern security standards: for example, mqtt/tcp for transport security. Since security is complex, it makes sense to rely on generally accepted standards.

Here is a summary of the security levels, the designated posts for each level are found in this part.

One option for using a secure and reliable connection is to use a physically secure network or VPN to communicate between clients and brokers. This solution is suitable for gateway applications where the gateway is connected to the device on the one hand and to the broker via VPN on the other hand.

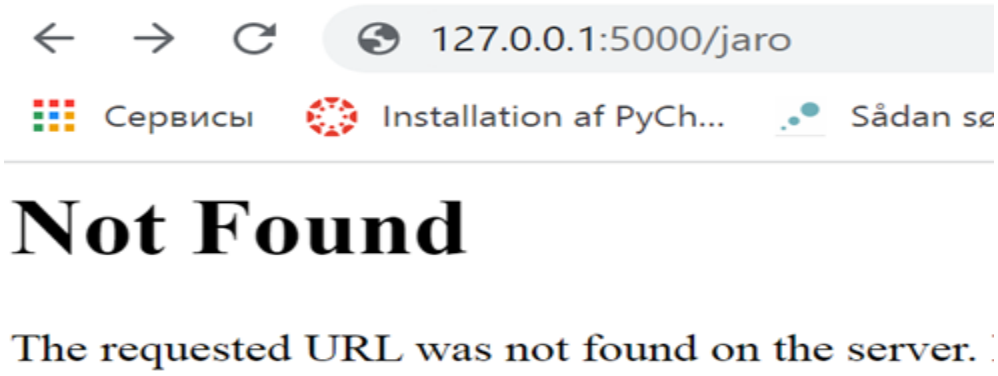
Transport layer when privacy is the primary view, mqtt/tcp is usually used to enable blocking. This security method is a proven way to ensure that data cannot be read during transmission and authentication based on a client certificate.

Application layer, at the transport layer, communications are encrypted, and identities are authenticated. MQTT client ID and username/password credentials for device authentication at the application level I call this in my script Log_to_DB python code. These properties are provided by the protocol itself. Authorization or control of what is allowed to be done on a custom, broker specific implementation. In addition, application layer encryption of the payload can be used to protect against hacking.

Of Course, code was written in python with a “cursor” to connect a database from MariaDB and output from sensors, then receive the data and can be printed to console in PyCharm. Connection to MariaDB platform was true: the user= root, password, host= 127.0.0.1:5000, port= 3306, and name measurements for database in MariaDB. When the client connects to MQTT and adds a subscriber and receives the message in my case is Get_from_DB python code data it can be shown in the web brother page.

Another code made in HTML is cached data with python code Get_from_DB and transferred and shown in the Web page data from temperature, air voc and light sensors with server on raspberry pi4.

On the part between connection from the database and publish the result in web page, it was another problem, in my web bra user as google chrome will not give permission to me use this host =127.0.0.1:5000



So, after so many evenings of trying to fix this bag, the solution was to ask for help from the professionals' teacher, and the problem was solved 😉. Thanks. Problem was because I am testing so many times this host =”127.0.0.1:5000” my old AVG antivirus free make block for this host. Of Course, when this bag is fixed all is working and I get totally another good pictures in my web page with temperature, air voc index, and light suites.

In this situation it can be all of us, who make the project self. Especially important, do not panic and try to find the solution to solve the problem. I remember always now or from project management we need to always have plan b, c, e. In this case I do not make the prognose for that risk with a web browser some known unknown risk.

10.8 Test 8. Check system test overall programs.

1.Open in Arduino io the C code name “Project3_WIFI_MQTT_AIR_DUST_GPS_LIGHT_TEMPERATURE_SENSORS”

2.Connect the usb cable from Esp32 to the laptop.

3.Upload the C code to Esp32.

4.Check in the serial monitor writing the connection to WIFI and MQTT.

5.Check in the serial monitor it sends the data from sensors every 5 minutes.

6.Open mqtt and see the data coming for the three topics every 5 minutes .

7.Open PyCharm, select the code Log_to_DB.py press on run, see the print data from sensors in Pycharm console.

8.Open in PyCharm file name Get_from_DB.py press on run,see the data from sensors in Pycharm console.

9. Select in the PyCharm console the host address http://127.0.0.1:5000 press on it.

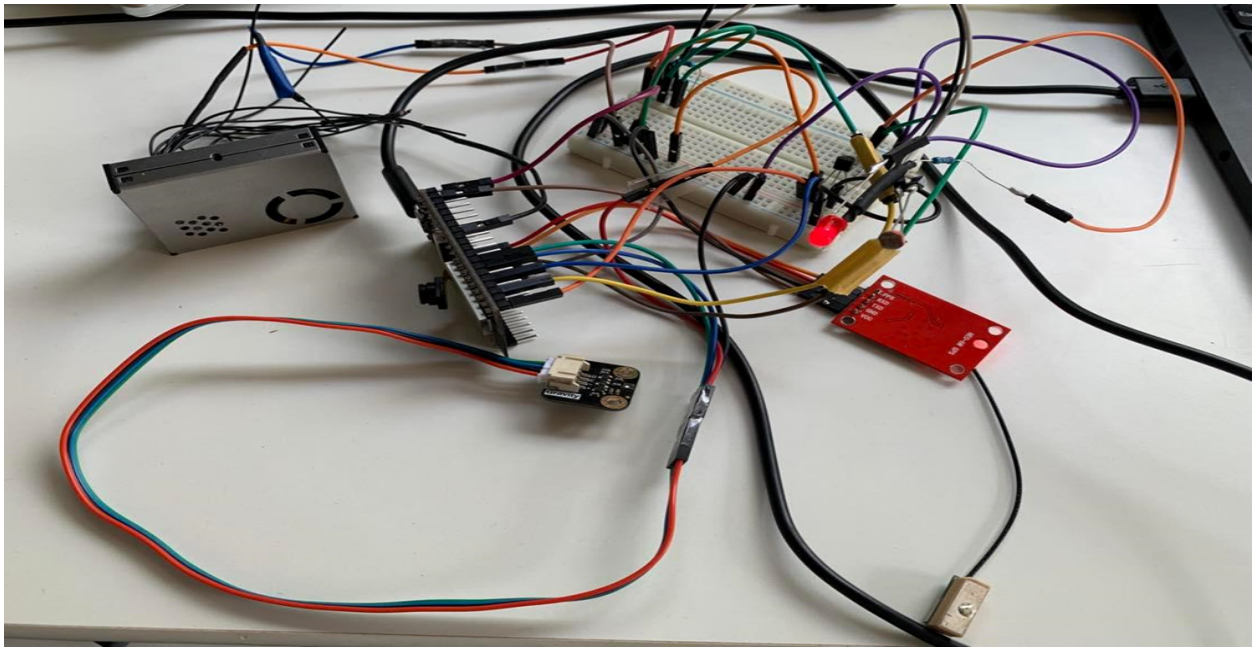
10. Check that you can see the web page with measurements as Temperature, AIR sensor voc index % and Value Light.

Actual result: It is visible last data on the web page name “Home page Sensors” with data from Temperature, AIR sensor voc index % and Light Value.

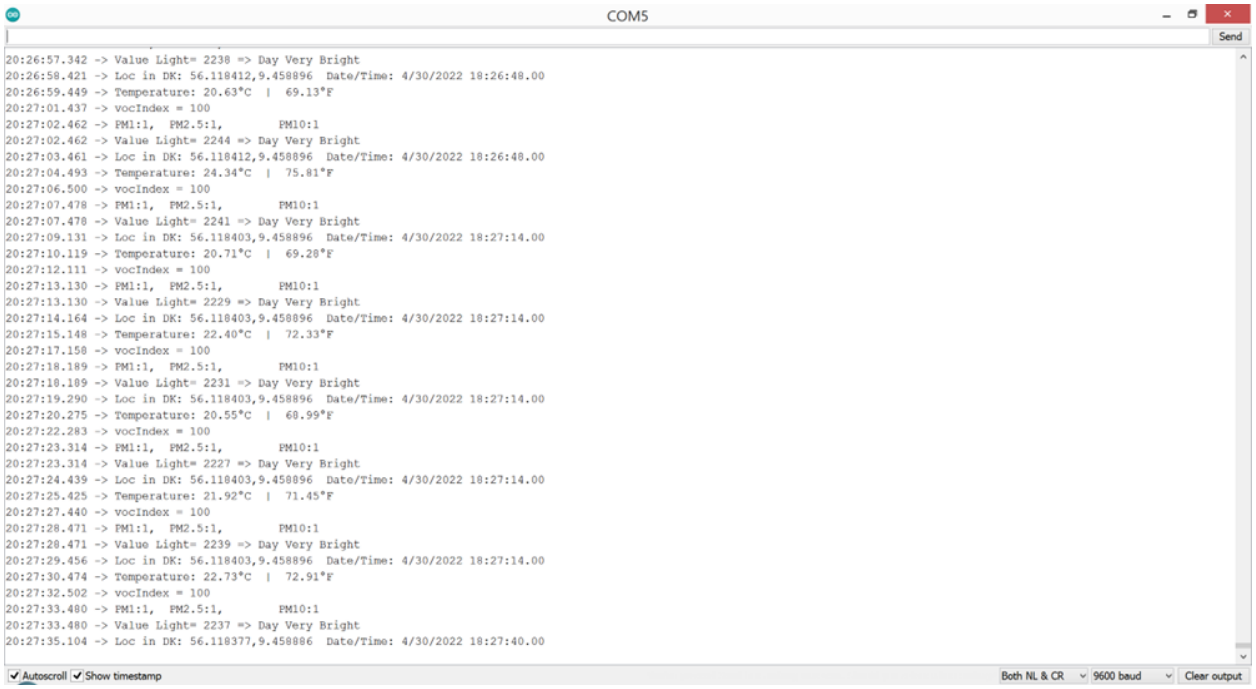
11. Prove.

https://drive.google.com/file/d/1jEswfs4yYX_3if4sL2qOM3s2DkipTYij/view?usp=sharing

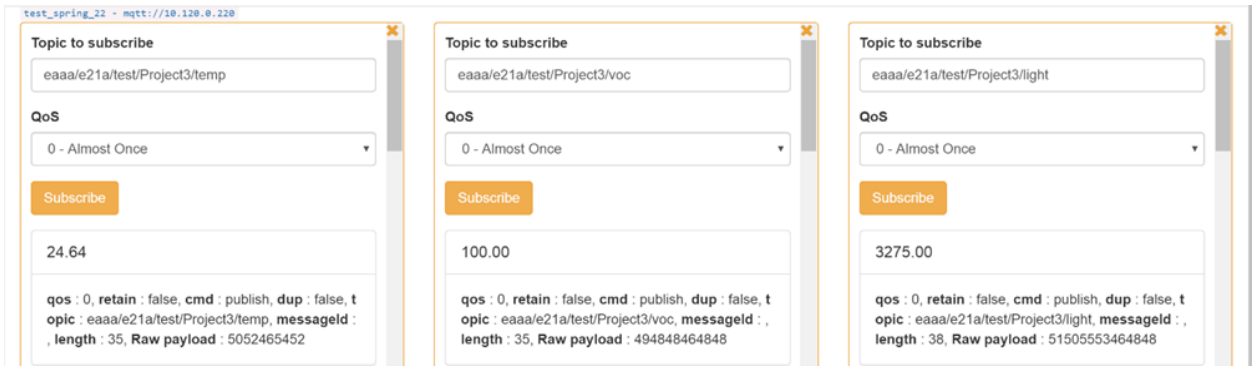
Picture with my connection of the Temperature, Light, Air, Dust, Gps sensors.



Printing results from sensors in serial monitor.



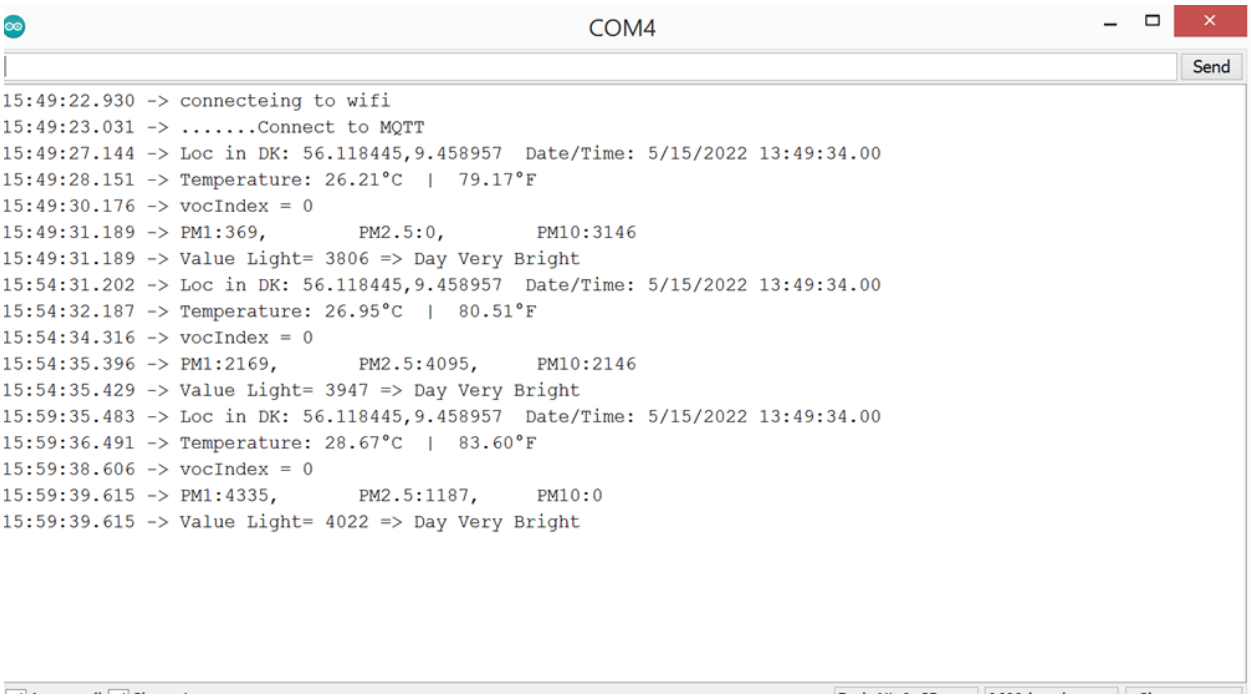
In mqtt three topics receive data from sensors.



Maria DB database.

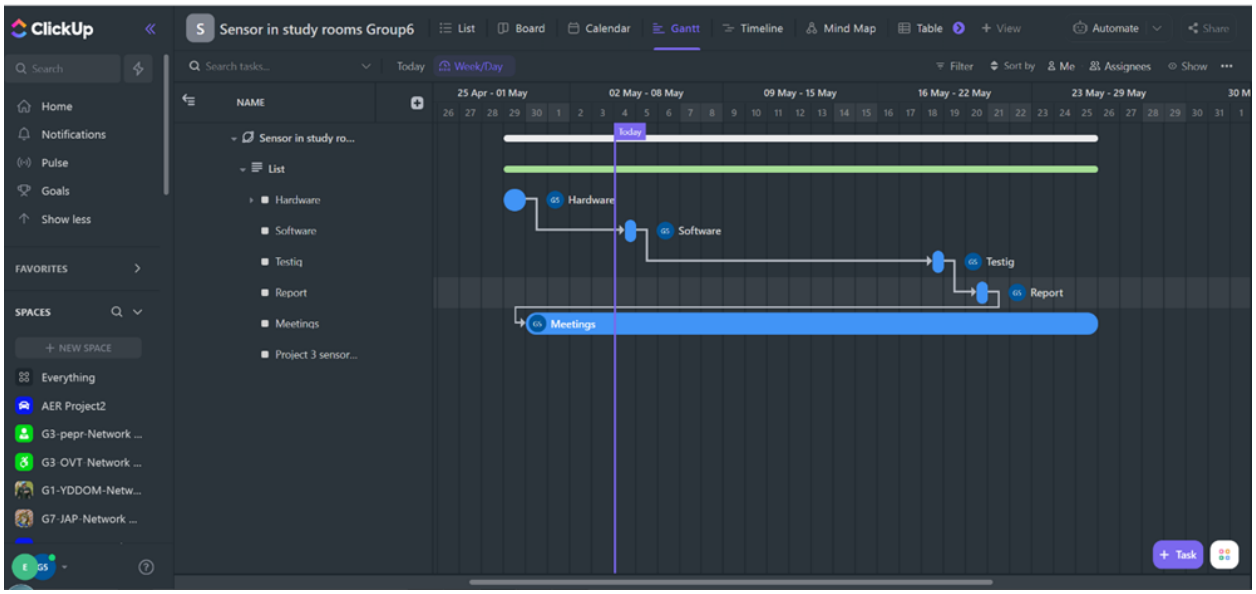
```
MariaDB [measurements]> desc temperatures;
+-------------+-------------+------+-----+--------------------+-----------------------------+
| Field       | Type        | Null | Key | Default            | Extra                       |
+-------------+-------------+------+-----+--------------------+-----------------------------+
| measure_id  | int(10)     | NO   | PRI | NULL               | auto_increment              |
| temperature | varchar(255)| YES  |     | NULL               |                             |
| vocIndex    | varchar(255)| YES  |     | NULL               |                             |
| light       | varchar(255)| YES  |     | NULL               |                             |
| ts          | timestamp   | NO   |     | current_timestamp() | on update current_timestamp() |
+-------------+-------------+------+-----+--------------------+-----------------------------+
5 rows in set (0.006 sec)

MariaDB [measurements]>
```

Receive the data from sensors every 5 minutes.



12. Project management.

Meetings in class and plan in the Gantt calendar from click up.



In this project, I will show how to build the system which connects sensors to the ESP32 and transfers data to the mqtt then in the MariaDB database and receives measurements in the web page.

In that case I can work 8 hours each day on the project. I set the day to every task for better orienteering in the plan and made dependences. Best solution is that I want to finish the project before the deadline, so I have time to check all the work, report and fix bugs and mistakes if they are coming. (1-2 days before the deadline).

The project includes the major task:

1.Hardware.

2.Software.

3.Testing.

4.Report

5.Meetings.

In Hardware include:

1.Temperatura sensor LM35

2.Light sensor LDR Photoresistor.

3.SGP40 Air Quality Sensor v1.0.

4. SM-UART-04L Laser Dust Sensor.

5.GPS NEO-6M

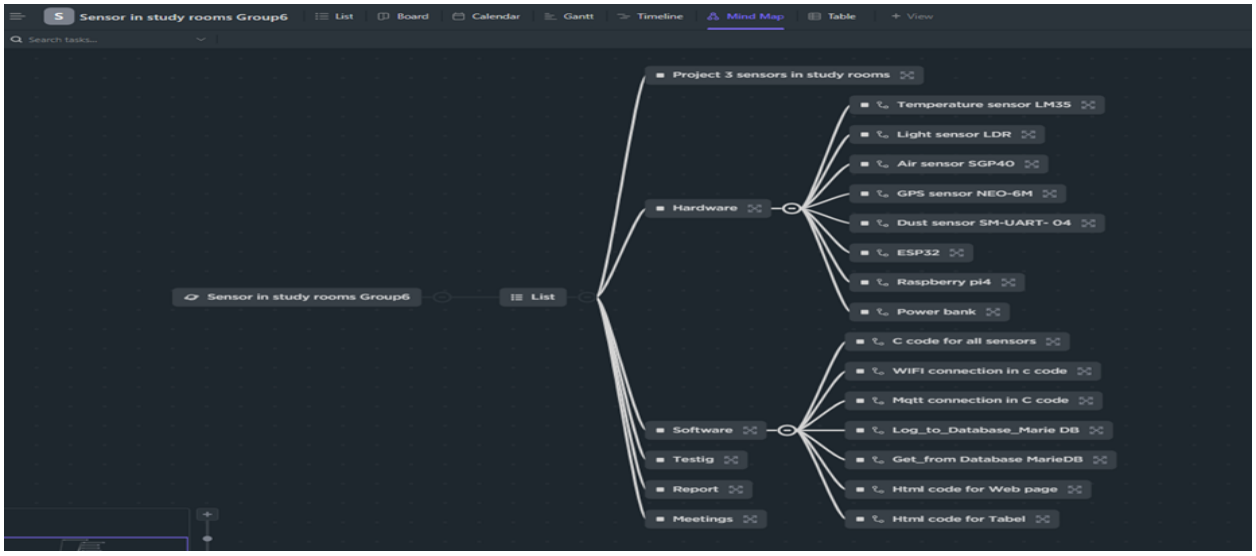
6.Esp32.

7.Raspberry pi4.

8.Power bank.

9.Led and wire.

Review the Mind Map from Click Up.



In software: Build on the C code and based console application with python code that will include the transfer data from sensors true the mqtt and in database to the web page.

Test included these options:

Test cases for the project I'm use the tools and methods learnt in the practice:

1.Im use the classical model like in the good old days model.

2. Divide the project into smaller stages.

3. Agree with the master of department individual jobs in the project.

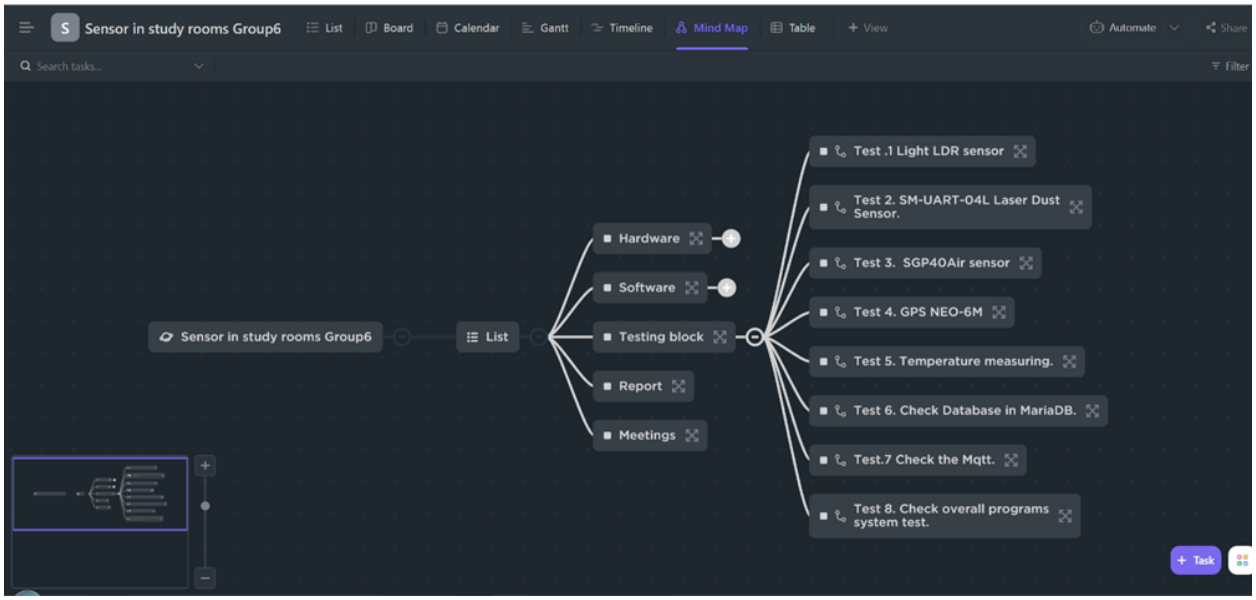
4. Planning the time and cost of resources.

5. Review the budget of the project.

6. Agree on the start date and end date of the project.

7. Used to test different methods and test Technik at the same time when I connect sensors for a room project.

This overview of the Testing block.



Because I'm studying in Business academy Aarhus, for me it was interest in Calculation **on the Budget.** I know this was not a requirement for the order to make a ready measurement station.

Project budget is not fully developed. Project will include the approximate cost of materials and software. If I need to produce for example the 1000 stations, for Business. The deadline for delivery is one month after signing the contract for the order. I will create a WBS and estimate all costs for the work. I'm creating a budget proposal – based on a selected template including cost for labor, materials costs – as well as delivery costs.

	A	B	D	E	F	G	H	I	J	K	L
1	Project Budget										[Eaa/21 / Logdao]
2											
3											
4		Project Lead: [Sensors in study room]							BUDGET	ACTUAL	Under(Over)
5		Start Date: [25.05.2022]							Total ########	$ -	#REFERENCE!
6											
7			Labor		Materials		Fixed				
8	WBS	Task	Hrs	Rate	Units	$/Unit	Costs		Budget	Actual	Under(Over)
9	1	Buying components							########		#REFERENCE!
10	1.1	ESp32			1.000,00	$14,00			14.000,00		14.000,00
11	1.2	LEDs			1.000,00	$0,15			150,00		150,00
12	1.3	Raspberry pi4			1.000,00	$100,00			#########		100.000,00
13	1.4	Bread board and wires			1.000,00	$15,00			15.000,00		15.000,00
14	1.5	Temperature sensor LM35			1.000,00	$2,00			2.000,00		2.000,00
15	1.6	Light sensor			1.000,00	$2,00			2.000,00		2.000,00
16	1.7	AIR sensor			1.000,00	$30,00			30.000,00		30.000,00
17	1.8	Totals			7000,0	$163,15			#########		1.142.050,00
18									-		-
19									##########		#REFERENCE!
20	2	Workers							########	$ -	$ 24.480,00
21	2.1	Reaseach and development	1000,0	$20,00					20.000,00		20.000,00
22	2.2	Software	1,0	#######					1.000,00		1.000,00
23	2.3	Hardware	160,0	$5,00	188,0	$10,00			2.680,00		2.680,00
24	2.4	Coordinator	40,0	$20,00					800,00		800,00
25									-		-
26											
27											
28											
29											
30											
31											
32											
33	3	Extra expences							########	$ -	$ 151.200,00
34	3.1								-		-
35	3.2	Lease			1,0	$1.200,00			1.200,00		1.200,00
36	3.3	Transport			100,0	$1.200,00			#########		120.000,00
37	3.4	el Manuel			1.000,00	$20,00			20.000,00		20.000,00
38		Box			1000,0	$10,00			10.000,00		10.000,00
39									-		-

Total price to produce 1000 stations is 1.142.050,00 dollars plus the extra expenses and workers.

Calculation of Budget is using the components with prices in Denmark. General to manage this uncertainty in a project is called Risk Management. Risk management is the process of identifying, assessing, and controlling threats to an organization's capital and earnings. These risks stem from a variety of sources including financial uncertainties, legal liabilities, technology issues, strategic management errors, accidents, and natural disasters. Risk management advantage, all projects experience the unexpected and most importantly. The advantage is that there are fewer unforeseen problems and if they do, then the future project team may be ready to deal with them. The problems can be basically divided into two types:

1. Known Unknowns: Represents identified potential problems, such as a workers' strike or extreme weather conditions. It can happen, and you can prepare for them.

2. Unknown unknowns: These are unexpected problems that no one could have foreseen the approach of, pilot project managers and specialists would expect them, because something unexpected always happens.

13. Conclusion.

In this project, I had the merits of making my own measuring station software, which in turn, can measure the data from the Temperature, Air, Lights, Dust, GPS position sensors, then store data only from Temperature, AIR, Light in the mqtt. Extract data from the mqt to the database, after subscribing from mqtt to certain topics. Transfer the data from the database to the local network and display on the web page. So, I was able to calculate the approximate cost, what would be the market price of such a measuring station from components with prices in Denmark. I think that the knowledge acquired in this project will be very useful to me in my life and at my student workplace in an IOT company. All knowing that humans cannot perceive all pollutants reliably. So, sensors and their performance play a crucial role in detecting dangerous these components. Solutions in sensor measurements enable precise and accurate monitoring of these quality indicators.

14. Applications.

This is how I write C code, Python and Html codes.

Main C code starts here.

```
 //wifi

#include <WiFi.h>

#include <WebServer.h>


const char* ssid = "ITEK 2nd"; //Else IP address from raspberypi4

const char* password = "Four_Sprints_F21v";   // password for the network


//const char* ssid = "NETGEAR98";

//const char* password = "quickink024";


WiFiClient espClient;


//mqtt connection

#include <PubSubClient.h>
```

```
const char* mqtt_server = "10.120.0.220"; // My raspberi pi  or another mqtt server

//const char* mqtt_server = "test.mosquitto.org";


int       port       = 1883;

const char* inTopictemp = "eaaa/e21a/test/Project3/temp";

const char* inTopicvoc = "eaaa/e21a/test/Project3/voc";

const char* inTopiclight = "eaaa/e21a/test/Project3/light";


PubSubClient client(espClient);


// GPS start const

#include <TinyGPS++.h>

#include <SoftwareSerial.h>

static const int RXPin = 15, TXPin = 2;

static const uint32_t GPSBaud = 9600;

TinyGPSPlus gps;

SoftwareSerial ss(RXPin, TXPin);

// GPS finish const
```

```
//temperatura start
const float hot = 25; //hot parameter
const float cold = 15; //cold parameter
// temperatura finish

// AIR sensor start
//wire library
#include <Wire.h>

//air quality sensor library
#include <DFRobot_SGP40.h>

//the new pins
int const I2C_SDA = 33;
int const I2C_SCL = 32;

TwoWire I2CBME = TwoWire(0);

DFRobot_SGP40 air_sensor;
// AIR sensor finish.
```

```
//Dust sensor start

#include <Wire.h>

#define dust_sensor 12

//Dust sensor finish


//Light sensor start

int const LIGHT_SENSOR_PIN = 36 ;

// Light sensor finish




void setup()

{ // GPS setup start


  ss.begin(GPSBaud);

  Serial.println(F("NEO-6M GPS MODULE QUICK TEST"));

  Serial.print(F("Testing with TinyGPS++ library v. "));
Serial.println(TinyGPSPlus::libraryVersion());

  Serial.println(); // GPS setup finish
```

```
                    // Temperatura setup start
pinMode(12, INPUT );
pinMode(13, OUTPUT);
                    //Temperatura setup finish
 // deep sleep setup start , initialize the digital pin as an output.
pinMode(LIGHT_SENSOR_PIN, OUTPUT);
// deep sleep setup finish

//AIR sensor setup start
I2CBME.begin(I2C_SCL, I2C_SDA);//last param is clock freq
air_sensor.begin(10000);
//AIR sensor setup finish

// Dust sensor setup start
Serial2.begin(9600, SERIAL_8N1, dust_sensor, -1);
// Dust sensor setup finish

//Light sensor start setup
// initialize serial communication at 9600 bits per second:
```

```
 Serial.begin(9600);  //115200
 pinMode(4, OUTPUT);
 //Light sensor finish
 // Connecting to WIFI start
 Serial.println("connecting to wifi");
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
   Serial.print(".");
   delay(500);                          // delay(500);
 }             //connecting to WIFI finish
 // conecting to mqtt
 client.setServer(mqtt_server, 1883);
 Serial.println("Connect to MQTT");
 client.connect("ESP32Client");
 if (!client.connected()) {
 Serial.println("NOT CONNECTED");
       // conecctting to mqtt finish
 }
}
void loop()
```

```
{ //GPS setup start
  while (ss.available() > 0)
        if (gps.encode(ss.read()))
      displayInfo();
   if (millis() > 5000 && gps.charsProcessed() < 10)
  {
    Serial.println(F("No GPS Module Found! Check Hardware!!"));
        while (true);
  }
}
// mqtt publishValue start
 void publishValue(float val, int topic) {
 if (!client.connected()) {
    client.connect("ESP32Client");
  }
  client.loop(); //loop
  if (topic == 1) {
    client.publish(inTopictemp, String(val).c_str(), true);
  }
  else if (topic == 2) {
```

```
    client.publish(inTopicvoc, String(val).c_str(), true);
  }
  else if (topic == 3) {
    client.publish(inTopiclight, String(val).c_str(), true);
  }
  // mqtt publishValue finish
}

void displayInfo()
{
  Serial.print(F("Loc in DK: "));
  if (gps.location.isValid())
  {
    Serial.print(gps.location.lat(), 6);
    Serial.print(F(","));
    Serial.print(gps.location.lng(), 6);
  }
  else
  {
    Serial.print(F("INVALID"));
```

```
}
Serial.print(F("  Date/Time: "));
if (gps.date.isValid())
{
  Serial.print(gps.date.month());
  Serial.print(F("/"));
  Serial.print(gps.date.day());
  Serial.print(F("/"));
  Serial.print(gps.date.year());
}
else
{
  Serial.print(F("INVALID"));
}
Serial.print(F(" "));
if (gps.time.isValid())
{
      if (gps.time.hour() < 10) Serial.print(F("0"));
  Serial.print(gps.time.hour());
  Serial.print(F(":"));
```

```
      if (gps.time.minute() < 10) Serial.print(F("0"));
  Serial.print(gps.time.minute());
  Serial.print(F(":"));
      if (gps.time.second() < 10) Serial.print(F("0"));
  Serial.print(gps.time.second());
  Serial.print(F("."));
      if (gps.time.centisecond() < 10) Serial.print(F("0"));
  Serial.print(gps.time.centisecond());
}
else
{
  Serial.print(F("INVALID"));
}
Serial.println();         //Gps loop finish

// Temperatura loop start
float sensor = analogRead(34);
//Serial.println(sensor);
float temp =  (sensor / 4095.0) * 1.05;       //4095.0
// Convert the voltage into the temperature in Celsius
```

```
  float temperatureC = temp * 100;


   delay(1000);
  // swith termometr
  // Print the temperature in Celsius


  Serial.print("Temperature: ");
  Serial.print(temperatureC);
  Serial.print("\xC2\xB0"); // shows degree symbol
  Serial.print("C  |  ");


  // Print the temperature in Fahrenheit
  float temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;
  Serial.print(temperatureF);
  Serial.print("\xC2\xB0"); // shows degree symbol
  Serial.println("F");
  publishValue(temperatureC, 1);
//publishValue(temperatureC); to mqtt


  delay(1000); // wait a second between readings
```

```
            // temperatura loop finish


  // AIR sensor loop start
  uint16_t index = air_sensor.getVocIndex();
  Serial.print("vocIndex = ");
  Serial.println(index);
  publishValue(index, 2);
//publishValue(index);        to mqtt
  delay(1000);
  // AIR sensor loop finish.


  // Dust sensor loop start
  byte dust_byte[30];
  int dust_index = 0;
  int dust_data[3];
  int l = 0;
  //  while(!Serial2.available()) {
  //  }
  if (Serial2.read() == 0x42 && Serial2.read() == 0x4D) {
    //Serial.println("header done");
```

```
      for (int i = 0; i < 14; i++) {
      if (Serial2.available()) {
     dust_byte[dust_index] =  Serial2.read();
     //Serial.println(dust_byte[dust_index]);
     dust_index++;
      }
      }
      for (int i = 14; i < 30; i++) { //flush the no-data bytes[
      if (Serial2.available()) {
     Serial2.read();
      }
      }
}
//int h_length = dust_byte[0];
//int l_length = dust_byte[1];
//l = h_length + l_length + 2;
//Serial.print("length: ");
//Serial.println(l);
for (int i = 0; i < 3; i++) {
      int value = (dust_byte[2 + i * 2] << 4) + dust_byte[2 + 1 + (i * 2)];
```

```
        dust_data[i] = value;
    }
    Serial.print("PM1:");
    Serial.print(dust_data[0]);
    Serial.print(",\t");
    Serial.print("PM2.5:");
    Serial.print(dust_data[1]);
    Serial.print(",\t");
    Serial.print("PM10:");
    Serial.print(dust_data[2]);
    Serial.println("");
    // Dust loop finish.

    //Light sensor loop start
    // reads the input on analog pin (value between 0 and 4095)
    int analogValue = analogRead(LIGHT_SENSOR_PIN);
    Serial.print("Value Light= ");
    Serial.print(analogValue);   // the raw analog reading

    // We'll have a few threshholds, qualitatively determined
```

```
  if (analogValue < 200) {
    digitalWrite(analogValue, HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(5000);                       // wait for a second //delay(5000);
    digitalWrite(analogValue, LOW);    // turn the LED off by making the voltage LOW
    delay(5000);                       // wait for a second //delay(5000);
    esp_sleep_enable_timer_wakeup(1200 * 1000 * 1000);
    //esp_sleep_enable_ext0_wakeup(GPIO_NUM_34);//could be woken by a pin going high
    esp_deep_sleep_start();
        // deep sleep finish
  }
  //deep sleep start
  else if (analogValue < 200 ) {
    Serial.println(" => Night");
    digitalWrite(4, LOW);

  } else if (analogValue < 500) {
    Serial.println(" => Day Dim");
    digitalWrite(4, LOW);
```

```
  } else if (analogValue < 1000) {
    Serial.println(" =>Day Light");
    digitalWrite(4, LOW);
  } else if (analogValue < 2000) {
    Serial.println(" =>Day Bright");
    digitalWrite(4, HIGH);
  } else {
    Serial.println(" => Day Very Bright");
    digitalWrite(4, HIGH);
  }
  publishValue(analogValue, 3);                    //LIGHT SENSOR
publishValue(analogValue);
  delay(300000); // delay(300000); = 5 minut.

  // Light sensor loop finish
}
```

Then I'm make Python code to Log_to_DB.py

```
import paho.mqtt.client as mqtt
```

```
import mariadb, sys


data_in = [None, None, None] # 0: temp , 1: voc, 2: light
def add_measurement_to_database(cur, payload, topic):
        all_data = True


         if 'eaaa/e21a/test/Project3/temp' in topic:
            data_in[0] = payload
        elif 'eaaa/e21a/test/Project3/voc' in topic:
            data_in[1] = payload
        elif'eaaa/e21a/test/Project3/light' in topic:
            data_in[2] = payload



         for data in data_in:
            if data is None:
                all_data = False


         if all_data is True:
```

```
        cur.execute("INSERT INTO
measurements.temperatures(temperature, vocIndex, light)"
                    "VALUES (?,?,?)", (data_in[0], data_in[1],data_in[2]))
        for index in range(0,3):
            data_in[index] = None


# The callback for when the client receives a connect response from
the server
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("eaaa/e21a/test/Project3/temp")
        client.subscribe("eaaa/e21a/test/Project3/voc")
    client.subscribe("eaaa/e21a/test/Project3/light")


# The callback for when a PUBLISH message is received from the
server
def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))
        add_measurement_to_database(cur, float(msg.payload),
msg.topic)
```

```
# Connect to MariaDB Platform
try:
        conn = mariadb.connect(
       user="root",
       password="xxxxxxxxxxxxxxxxxxxxxxx",
       host="127.0.0.1",
       port=3306,
       database="measurements",
       autocommit=True
        )
except mariadb.Error as e:
   print(f"Error connecting to MariaDB Platform: {e}")
   sys.exit(1)

# Get Cursor
cur = conn.cursor()
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect("test.mosquitto.org")
```

```
#client.connect("10.120.0.220")
client.loop_forever()
```

Then I'm make Python code to Get_from_DB.py

```
from flask import Flask, redirect, url_for, render_template
import mariadb, sys
# Connect to MariaDB Platform
print("connecting to DB")
try:
        conn = mariadb.connect(
       user="root",
       password="xxxxxxxxxxxxxxxxxxx",
       host="127.0.0.1",
           port=3306,
       database="measurements",
       autocommit=True
        )
except mariadb.Error as e:
   print(f"Error connecting to MariaDB Platform: {e}")
   sys.exit(1)
```

```
# Get Cursor

print("get data")

cur = conn.cursor()

cur.execute("SELECT temperature, vocIndex, light FROM
temperatures ORDER BY measure_id DESC LIMIT 1;")

                                #cur.execute("SELECT temperature FROM
temperatures ORDER BY id DESC LIMIT 1;")  #last row from sensors
data

                                   #cur.execute("SELECT  * FROM
temperatures")

                                #result = cur.fetchall()

                                #result = cur.fetchone()


result = []

for set in cur:

        for field in set:

       result.append(field)


#print(result)

temp = result[0]

voc = result[1]
```

```
light = result[2]

print(temp)

print(voc)

print(light)



app = Flask(__name__)

app.debug = True



@app.route("/")

def execute():

  val1 = temp

  val2 = voc

  val3 = light


  return render_template("multipleVal.html", first=val1, second=val2,
third=val3)


app.run(host='0.0.0.0', port=5000)
```

Then I'm make code in HTML the Tabel.html

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=devicewith,
initial-scale=1.0"/>

   <title>Table</title>

</head>

<body>

   <table border=1>

      <tr>

         {% for header in headings %}

         <th>{{ header }}</th>

            {% endfor %}

      </tr>

       {% for row in data %}

         <tr>

            {% for cell in row %}

            <td> {{ row[cell] }}</td>
```

```
            {% endfor %}
          </tr>
        {% endfor %}
</table>
</body>
</html>
```

Then I'm make html code name multipleVal.html

```
<!doctype html>
<html>
<head>     <head>
<meta name="viewport" content="width=device-width,
initial-scale=1">
<link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"


integrity="sha384-fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7Rrh
N7udi9RwhKkMHpvLbHG9Sr" crossorigin="anonymous">
   <style>
       html {
    font-family: Arial;
```

```
    display: inline-block;
    margin: 0px auto;
    text-align: left;
        }
        h2 { font-size: 2.0rem; }
        p { font-size: 2.0rem; }
        .units { font-size: 1.2rem; }
   .dht-labels{
     font-size: 1.5rem;
     vertical-align:middle;
     padding-bottom: 15px;
        }
 </style>
  </head>
 <h1>Home page Sensors </h1>
 <h2>Server Raspberry pi4 </h2>
   <title></title>
   <body>
   <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
```

```
    <p>
        <i class="fas fa-thermometer-half" style="color:#059e8a;"></i>
    <span class="dht-labels"></span>
    <span id="temperature">TEMPERATURE</span>
    <p>{{second}}</p>
        <sup class="units"></sup>
    <h1></h1>
  </p>
</body>
<body>
      <p>
        <i class=" "> AIR SENSOR </i>
    <span class=""> VOC</span>
    <span id=" ">INDEX%</span>
     <p>{{third}}</p
 <sup class="units"></sup>
  </p>
 </body>
<body>
 <p>
```

```
    <span id="light">Value Light</span>
  <p>{{first}}</p>
  </p>
</body> Jaroslav ;)
</html>
```