AIR Quality Monitoring Embedded Systems Project

By -Marian Aquilizan, Brhamjot Singh

What is air pollution?

Air pollution is the presence of excessive amounts of undesirable and unsafe solid or gaseous substances such as Carbon Monoxide, C02, Nitrogen Oxide, VOCs, Particulate Matter, Amonnia, etc., atmosphere.

Air pollution cause and effect

Air pollution has become an increasingly hazardous problem over the past few years. This factor is directly related to human health.

Global warming has become a severe concern for many countries; one widely faced issue is air pollution.

Other effects of air pollution also include various diseases like lung cancer, ischemic heart disease, asthma attacks, etc.

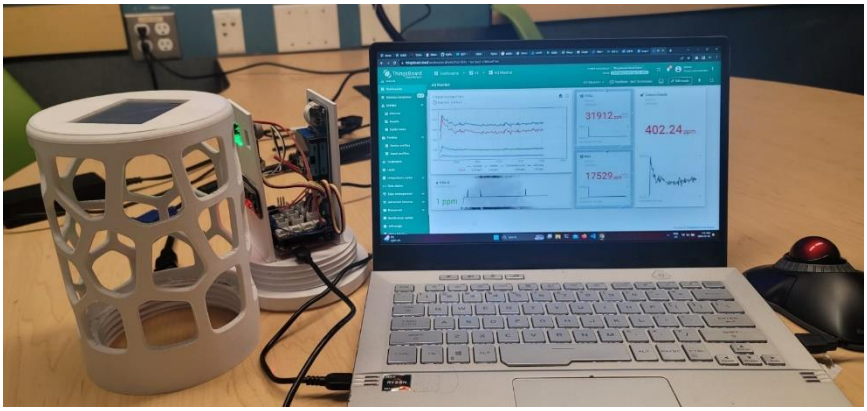
What is an air pollution monitoring system?

The Air pollution monitoring system is a facility to measure air pollutants using sensors, processing using microcontrollers, and showing results using various displays.

How can air pollution be monitored?

Air quality is a measure of how clean or polluted the air is. Air pollution is usually measured as Air Quality Index (AQI) in the PPM unit.

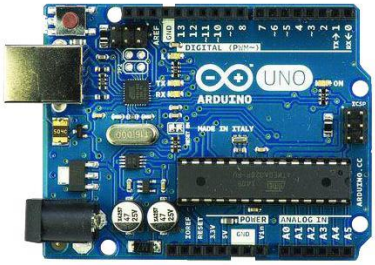
PROJECT OVERVIEW



The primary objective of the project is to meticulously design and implement a fully functional prototype of an indoor air quality monitor, specifically engineered to identify and quantify a diverse range of volatile organic compounds (VOCs), nitrogen oxides (NOx), and particulate matter concentrations within the surrounding environment. This imperative arises from the profound health hazards associated with airborne contaminants, emphasizing the critical need for timely detection and mitigation measures. The gravity of the situation is underscored by the potential fatality resulting from prolonged exposure to elevated concentration levels. Notably, exposure limits are contingent upon the specific contaminant, thereby necessitating a nuanced and adaptable approach to safeguarding human healt

DESIGN-Hardware

3D-Case: We have designed a three-dimensional casing**,** The holes strategically positioned ventilation openings to facilitate sufficient air intake, promoting an efficient cooling process for microcontrollers and ensures unimpeded access for air sensors to effectively sense and analyse the surrounding air quality.



ARDUINO UNO:

The Arduino Uno is a open source microcontroller board based on the 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, ATmega328 .It has 14 digital input/output pins, an ICSP header, and a reset button. Instead, it features the Atmega328 programmed as a USB-to-serial converter. This arduino board is programmed with arduino IDE.

ESP-32 :

The ESP32-WROOM is a versatile System-on-Chip (SoC) module developed by Espressif Systems for IoT applications. It features a dual-core processor, Wi-Fi, and Bluetooth connectivity, ample memory, various peripheral interfaces, and security features. Known for low power consumption, it's widely used in IoT projects for its compact form factor and extensive development support through frameworks like ESP-IDF.

SENSORS:

MQ-135: The MQ-135 is a gas sensor module commonly used to detect various air pollutants in the atmosphere. It is sensitive to gases such as ammonia, alcohol, acetone, toluene, carbon monoxide, and carbon dioxide. The module is often employed in air quality monitoring applications, providing a cost-effective solution for detecting and measuring the concentration of different gases. The MQ-135 sensor operates on the principle of resistance changes in response to the presence of specific gases

HM3301: A dust sensor measuring PM concentration with high sensitivity for various particle sizes (PM1.0, PM2.5, PM10). It operates on the light scattering principle, offers digital output for integration with microcontrollers, and finds applications in air quality monitoring for both indoor and outdoor environments.

SGP41

VOC and NOx sensor for indoor air quality applications

The SGP41 is Sensirion's new VOC and NOx sensor designed as digital smart switch and regulation unit for air treatment devices such as air purifiers.

Software Requirement:

Here in this project we are using Arduino IDE it is a open source IDE developed by Arduino.cc it supports c++ embedded language. C++ is platform dependent language. Any language is said to be platform dependent whenever the program is being execute in the same operating system where that was developed and compiled but it don't execute on other operating system.

Library Used:

Arduino

```
//Include sensor libraries
#include <MQUnifiedsensor.h>
#include <Seeed_HM330X.h>
#include <SensirionI2CSgp41.h>

// Include Dependencies
//___SPG41___//
#include <Arduino.h>
#include <Wire.h>
//__________//
```

ESP-32

```
#include <WiFi.h>
#include <ArduinoMqttClient.h>
#include <ThingsBoard.h>

#define TOKEN "5LCEPkTulUA2jpgA8WQ9"
#define THINGSBOARD_SERVER "mqtt.thingsboard.cloud"
#define THINGSBOARD_PORT "1883"

// y2pd3bsoa6jnuhnmfham
// k6bpvt27ygthqli03zdz
// asnprb6e3xjviqb4f9o5

#define RXD0 3
#define TXD0 1

const char* ssid = "henlow";
const char* password = "aquilizan!*";
```

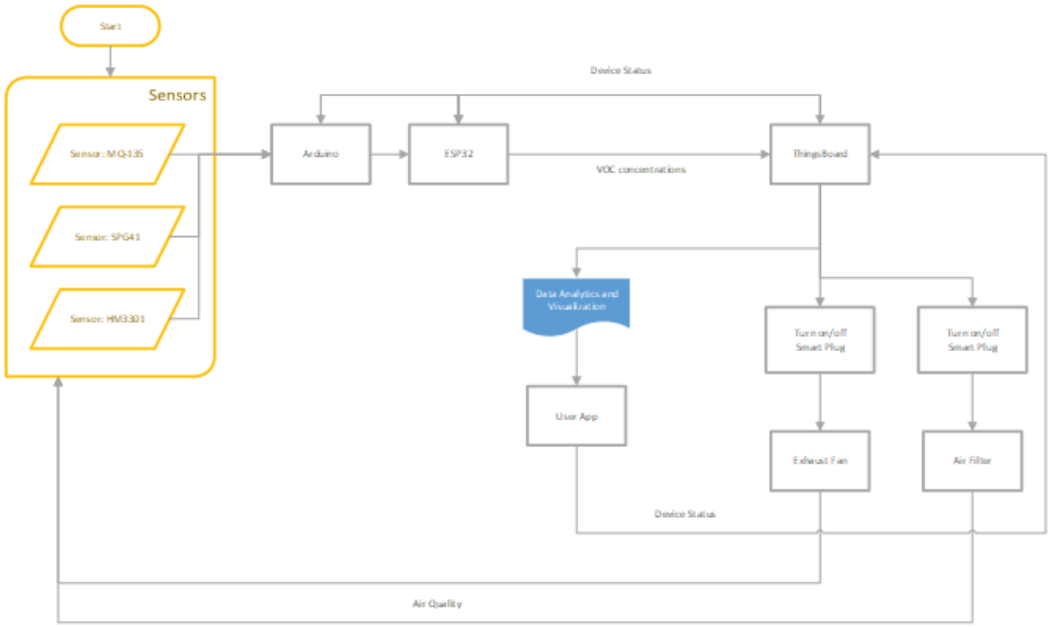
Project Design

The project utilizes three state-of-the-art air quality sensors—MQ-135, SGP41, and HM3301—carefully chosen based on their exceptional capacity to detect a wide spectrum of gases, including but not limited to carbon dioxide (CO2), sulfur dioxide (SO2), ozone (O3), and methane (CH4). Moreover, these sensors excel in measuring total volatile organic compounds (VOCs) and nitrogen oxides (NOx), providing comprehensive insights into air pollution. Additionally, the sensors are adept at detecting various particulate matter sizes, enabling a thorough analysis of airborne particles ranging from fine to coarse. This diverse sensor array ensures a comprehensive and precise evaluation of the air quality parameters essential for a holistic understanding of environmental conditions.

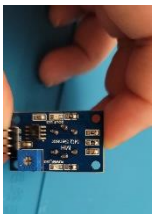
Flow-Diagram of Design:

The initial phase involves the acquisition of data by the first set of sensors, which is subsequently transmitted to an Arduino unit. Following this, the data is formatted into JSON and directly transmitted to the ESP-32. A pivotal step ensues as the API is invoked, facilitating the seamless transfer of data to the ThingsBoard cloud through HTTP. This systematic process culminates in the presentation of data, accompanied by graphical representations and insightful analyses, within the designated platform.

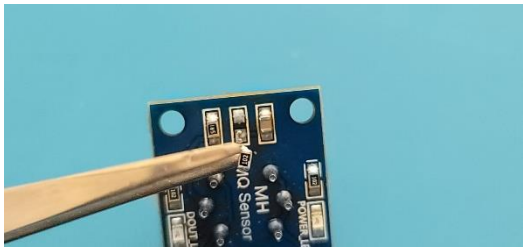
Additionally, it's important to note that the integration of this comprehensive data flow enhances the overall efficiency and reliability of the system, ensuring real-time monitoring and informed decision-making. The synchronized collaboration between the sensors, Arduino unit, ESP-32, API, and ThingsBoard creates a robust framework for data acquisition, transmission, and presentation in a streamlined and user-friendly manner.



Challenges:



The first challenge encountered during the development of the project is calibrating the MQ-135 to get individual values for the specific gases it can detect. A resistance chart is given within its datasheet, for this we soldier a 20k ohm resistor to the sensor.



A set of obstacles were encountered when setting up the ESP32. An appendix has been attached to outline the steps that were taken to resolve the errors. After the setup and connectivity to the ESP32 was verified, the next step was to get data sent from the Arduino to the ESP32. The Arduino and the ESP32 were connected using serial pins, from the ESP32, we are able to read the data by calling Serial.readString().

We set up the ESP32 microcontroller to securely communicate with Thingsboard Cloud using HTTPS, and data is transmitted between them using the REST API.

```
//Set math model to calculate the PPM concentration and the value of constants
MQ135.setRegressionMethod(1); //_PPM =  a*ratio^b
```

The line `MQ135.setRegressionMethod(1);` in the Arduino code configures the MQ135 gas sensor to use a power-law regression model (_PPM = a * ratio^b) for calculating gas concentration in parts per million (PPM) based on the sensor's output ratio. The constants 'a' and 'b' must be calibrated for accurate readings.

We calibrate MQ-135 using that ratio.

```
/*****************************  MQ CAlibration ********************************************/
// Explanation:
// In this routine the sensor will measure the resistance of the sensor supposedly before being pre-heated
// and on clean air (Calibration conditions), setting up R0 value.
// We recomend executing this routine only on setup in laboratory conditions.
// This routine does not need to be executed on each restart, you can load your R0 value from eeprom.
// Acknowledgements: https://jayconsystems.com/blog/understanding-a-gas-sensor
Serial.print("Calibrating please wait.");
float calcR0 = 0;
for(int i = 1; i<=10; i ++)
{
  MQ135.update(); // Update data, the arduino will read the voltage from the analog pin
  calcR0 += MQ135.calibrate(RatioMQ135CleanAir);
  Serial.print(".");
}
MQ135.setR0(calcR0/10);
MQ135.setRL(17.5);
Serial.println("  done!.");

if(isinf(calcR0)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected) please check your wiring and supply"); while(1);}
if(calcR0 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to ground) please check your wiring and supply"); while(1);}
/*****************************  MQ CAlibration ********************************************/
```

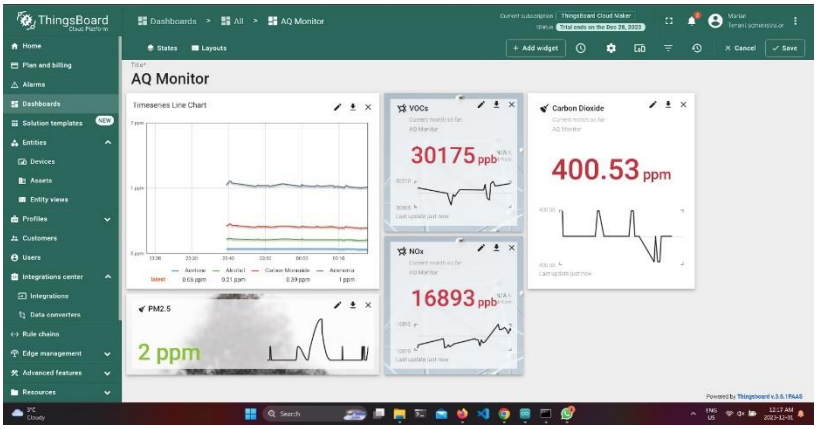
- The code initiates a calibration process for the MQ135 gas sensor, printing a message to the serial monitor.
- It enters a loop where the sensor data is updated, and the calibration value (**calcR0**) is accumulated over ten iterations. Progress is indicated by printing dots to the serial monitor.
- After the loop, the average calibration value is set as the R0 value for the sensor, and the load resistance (**RL**) is set to 17.5.
- A completion message is printed to the serial monitor.
- The code includes error handling to check for extreme calibration results. If the calibration value is infinite (indicating an open circuit) or zero (indicating a short to ground), warning messages are printed, and the program enters an infinite loop.

Lessons Learnt

In undertaking this project, it is imperative to acknowledge that obtaining a functional code for a singular sensor and seamlessly integrating it into the broader framework is not a straightforward endeavour. Numerous challenges arise, requiring comprehensive research and a proactive approach to problem-solving in order to achieve optimal results.

A noteworthy instance is the complex task of consolidating output data from various gases into a unified and meaningful metric. Unlike commonly encountered scenarios where a singular value suffices to delineate air quality, this particular project demanded a more sophisticated approach. Many existing resources, such as videos and example codes, oversimplified the intricacies of this process. Consequently, we were compelled to delve into extensive research and devise a tailored solution to effectively capture and interpret data pertaining to different gases from a singular output source. This exemplifies the meticulous and nuanced nature of our work in ensuring the accuracy and reliability of our project outcomes.

Conclusion and Future Directions



In conclusion, our project has navigated the complexities of sensor integration, coding challenges, and the synthesis of disparate data streams to deliver a robust framework. Looking ahead, our envisioned future direction encompasses the consolidation of all sensors and Arduino components onto a unified printed circuit board (PCB). Additionally, we will make a UI for the screen attached to the top of our case and with that we also recognize the importance of user interface and accessibility. Finally, we will focus on refining control mechanisms. We aspire to implement advanced control features that enable users to not only monitor but also actively manage and optimize the environmental parameters captured by our sensor array.

References

The code in this project has been modified and compiled from the following example code from the following authors.
Libraries and Example code used:
MQUnifiedsensor by Miguel A Califa, Yersson Carrillo, Ghiordy Contreras, Mario Rodriguez – MQ135.
Seeed_HM330X by Downey at Saeed Technology Co., Ltd. Copyright (c) 2018 - for the HM3301PM2.5
SensirionI2CSgp41 by Sensirion AG Copyright (c) 2021 - for the SPG41

https://thingsboard.io/
https://youtu.be/tb2EhGVNjZ4?si=9czrDxvXaZ5lMk_z
https://thingsboard.io/docs/pe/reference/mqtt-api/
https://thingsboard.io/docs/paas/devices-library/esp32-dev-kit-v1/
https://forum.arduino.cc/t/serial-input-basics-updated/382007
https://community.dfrobot.com/makelog-312138.html
http://www.programmingboss.com
https://forum.allaboutcircuits.com/threads/calibration-procedure-of-a-mq135.189025/
https://Wokwi.com
https://www.elprocus.com/mq135-air-quality-sensor/

Appendices:

Arduino Code

INCS3610_FINAL-PROJECT_IAQ-Monitor.

ESP-32

esp32.ino

We meticulously calibrated our MQ-135 sensor to obtain five distinct values from a single analog output. This calibration process involved precision adjustments to the ratios of resistance values, ensuring accurate and reliable sensor readings.

```
void loop() {
//___MQ135___//
MQ135.update(); // Update data, the arduino will read the voltage from the analog pin

  MQ135.setA(605.18); MQ135.setB(-3.937); // Configure the equation to calculate CO concentration value
  float CO = MQ135.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ135.setA(77.255); MQ135.setB(-3.18); //Configure the equation to calculate Alcohol concentration value
  float Alcohol = MQ135.readSensor(); // SSensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ135.setA(110.47); MQ135.setB(-2.862); // Configure the equation to calculate CO2 concentration value
  float CO2 = MQ135.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ135.setA(44.947); MQ135.setB(-3.445); // Configure the equation to calculate Toluen concentration value
  float Toluen = MQ135.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ135.setA(102.2 ); MQ135.setB(-2.473); // Configure the equation to calculate NH4 concentration value
  float NH4 = MQ135.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup

  MQ135.setA(34.668); MQ135.setB(-3.369); // Configure the equation to calculate Aceton concentration value
  float Aceton = MQ135.readSensor(); // Sensor will read PPM concentration using the model, a and b values set previously or from the setup
```

```
  Serial.print("\"CO\":"); Serial.print(CO); Serial.print(",");
  Serial.print("\"Alcohol\":"); Serial.print(Alcohol); Serial.print(",");
  // Note: 400 Offset for CO2 source: https://github.com/miguel5612/MQSensorsLib/issues/29
  /*****************************  MQ CAlibration
********************************************/
  // Explanation:
  // In this routine the sensor will measure the resistance of the sensor supposedly before being pre-
heated
  // and on clean air (Calibration conditions), setting up R0 value.
  // We recomend executing this routine only on setup in laboratory conditions.
  // This routine does not need to be executed on each restart, you can load your R0 value from
eeprom.
  // Acknowledgements: https://jayconsystems.com/blog/understanding-a-gas-sensor
  Serial.print("Calibrating please wait.");
  float calcR0 = 0;
  for(int i = 1; i<=10; i ++)
  {
    MQ135.update(); // Update data, the arduino will read the voltage from the analog pin
    calcR0 += MQ135.calibrate(RatioMQ135CleanAir);
    Serial.print(".");
  }
  MQ135.setR0(calcR0/10);
  MQ135.setRL(17.5);
  Serial.println("  done!.");

  if(isinf(calcR0)) {Serial.println("Warning: Conection issue, R0 is infinite (Open circuit detected)
please check your wiring and supply"); while(1);}
  if(calcR0 == 0){Serial.println("Warning: Conection issue found, R0 is zero (Analog pin shorts to
ground) please check your wiring and supply"); while(1);}
  /*****************************  MQ CAlibration
********************************************/
```