AIR QUALITY MONITORING SYSTEM

Phase_4:

PROBLEM STATEMENT:

Air pollution poses a significant threat to public health and the environment, with growing concerns about its adverse effects on respiratory diseases, cardiovascular problems, and environmental degradation. In our urban area, there is a pressing need for an effective Air Quality Monitoring System (AQMS) to address the following critical issues like inadequate air quality data,lack of public awareness,ineffective emergency response,compilance with regulatory standards,limited research and policy support.

Description:

This project is designed to create a simple air quality monitoring and reporting system using Arduino and sensors. The system collects data on temperature, humidity, and gas levels and transmits this information to the ThingSpeak platform for remote monitoring and analysis. It is a practical solution for tracking environmental conditions and ensuring awareness of air quality.

Abstract:

We have opted for the Arduino Mega as the central microcontroller for our project. To quantify the presence of specific gases in the atmosphere, particularly measuring parts per million (ppm), we have incorporated the MQ-135 gas sensor into our setup. The decision to utilize the Arduino Mega stems from its robust capabilities and versatile features, making it well-suited for the intended application. Its ample processing power and numerous input/output pins provide the necessary foundation to support our gas sensor and other associated components. The MQ-135 gas sensor, a pivotal element in our system, serves the purpose of detecting and quantifying various gases, including but not limited to carbon dioxide (CO_2), ammonia (NH_3), and methane (CH_4). By employing this sensor, we aim to gain precise measurements of gas concentrations in the air, which is vital for various environmental monitoring and safety applications. This initial phase sets the stage for the subsequent stages of development, where we will delve deeper into the intricacies of code development, calibration, and integration of the sensor data into a comprehensive monitoring and control system.

Components:

- DHTesp sensor for temperature and humidity readings.
- Gas sensor (MQ-135) to measure air quality.
- ESP8266-based microcontroller for data processing and Wi-Fi connectivity.

- ThingSpeak, a cloud-based platform for data storage and visualization.

Features:

- **Real-time Data Collection:** The system continuously collects temperature, humidity, and gas level data.
- **Wireless Connectivity:** Utilizes Wi-Fi to connect to the local network and transmit data to the cloud.
- **Remote Monitoring:** Data is sent to ThingSpeak for remote monitoring and visualization.
- **Alerts:** The system can be extended to provide alerts or notifications when air quality exceeds a predefined threshold.
- **User-Friendly:** The project offers a simple and cost-effective way to monitor and report air quality in indoor or controlled environments.

PROJECT IMPLEMENTATION :

Code:

Gassensor.chip.c:

```c
// Wokwi Custom Chip - For docs and examples see:
// https://docs.wokwi.com/chips-api/getting-started
//
// SPDX-License-Identifier: MIT
// Copyright 2023 Pasupathy T

#include "wokwi-api.h"
#include <stdio.h>
#include <stdlib.h>

typedef struct {
  pin_t pin_out;
  uint32_t gas_level_attr;
} chip_state_t;

static void chip_timer_event(void *user_data);

void chip_init(void) {
  chip_state_t *chip = malloc(sizeof(chip_state_t));
  chip->pin_out = pin_init("OUT", ANALOG);
  chip->voltage_gas_level_attrattr = attr_init_float("GASLEVEL", 1.0);

  const timer_config_t timer_config = {
    .callback = chip_timer_event,
    .user_data = chip,
  };
  timer_t timer_id = timer_init(&timer_config);
  timer_start(timer_id, 1000, true);
}

void chip_timer_event(void *user_data) {
  chip_state_t *chip = (chip_state_t*)user_data;
  float voltage = attr_read_float(chip->gas_level_attr);
  pin_dac_write(chip->pin_out, voltage);
}
```

Sketch.ino:

```
#include <DHTesp.h>
#include<ThingSpeak.h>
```

```
#include <WiFi.h>

const int temp=15;
const int gas = 2;
DHTesp dhtSensor;
TempAndHumidity data;
char sid[] = "Wokwi-GUEST";
char pass[]="";
unsigned long cha = "your chanel id";
const char* key ="your key";

WiFiClient cli;
void setup() {

 Serial.begin(115200);
 pinMode(gas, INPUT);
 pinMode(temp,INPUT);

 WiFi.mode(WIFI_STA);
 ThingSpeak.begin(cli);
 dhtSensor.setup(temp,DHTesp::DHT22);
}

void loop() {
 getwifi();

 data=dhtSensor.getTempAndHumidity();

 Serial.print("Gas Level: ");
 Serial.println(analogRead(2));
```

```
  Serial.println(("temp"+string(data.temperature)));
  Serial.println("Humi:"+String(data.humidity));


  ThingSpeak.setField(1,data.temperature);
  ThingSpeak.setField(2,data.humidity);
  ThingSpeak.setField(3,distance);
  statuscode = ThingSpeak.writeFields(cha,key);
  if(statuscode == 200){
   Serial.println("updates");
  }
  else{
   Serial.println("error");
  }
  Serial.println("****************");
  delay(15000); // this speeds up the simulation
}


void getwifi(){
   if(WiFi.status()!= WL_CONNECTED){
      Serial.print("attempting");
      while(WiFi.status()!= WL_CONNECTED){
        WiFi.begin(sid,pass);
        Serial.print(".");
        delay(5000);
      }
   }
   Serial.println("connected");
```

```
}
```

aqm.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>AQM</title>
  <script src="script.js"></script>
  <style>
    *{
      text-align: center;
    }
      #data-container {
      font-size: 24px;
      margin: 20px;
      padding: 10px;
      border: 2px solid #333;
      background-color: #f0f0f0;

      text-align: center;
    }
    #result{
      font-size: 30px;
      text-align: center;
```

```
            margin: 20px;
            padding: 10px;
            border: 2px solid #333;
            background-color: #f0f0f0;

        }
    </style>
</head>
<body>
    <h1>Real Time Air Quality Monitoring</h1>
    <div id="data-container">

    </div>
    <div id="result">

    </div>
</body>
</html>
```

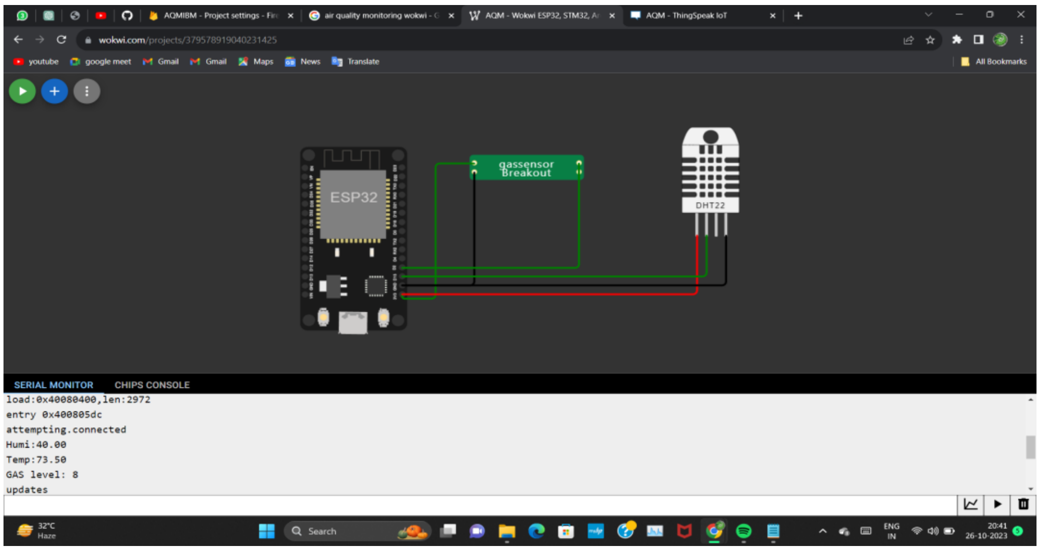
script.js:

```
 // Replace with your ThingSpeak channel ID and read API key
const channelID = '2320479';
const readAPIKey = 'NUWWKS8J4GUMC6SX';

// The field number you want to retrieve
```

```
const fieldNumber = 1;
var i ;
// ThingSpeak API URL
const apiUrl =
`https://api.thingspeak.com/channels/${channelID}/feeds.json?api_key=${read
APIKey}`;

// Perform an HTTP GET request to retrieve data
function latest(){
    {fetch(apiUrl)
        .then((response) => response.json())
        .then((data) =>{
           // i = data.feeds.length-4;
           i=0;

        })
    }
}
latest();
function fetchdata()
{fetch(apiUrl)
  .then((response) => response.json())
  .then((data) => {
    const f1 = data.channel.field1;
    const f2 = data.channel.field2;
    const f3 = data.channel.field3;

    if (data.feeds.length > 0 && i <data.feeds.length) {
      var value1 = data.feeds[i].field1;
      var value2 = data.feeds[i].field2;
      var value3 = data.feeds[i].field3; // Change field1 to match your field
number
      console.log(`Data from ThingSpeak:`);
      console.log(f1," ",value1);
      console.log(f2," ",value2);
      console.log(f3," ",value3);

      i=i+1;
      document.getElementById('data-container').innerHTML=`Temperature =
${value1} C<br> Humidity = ${value2} <br> GasLevel = ${value3}`;
      if(value1 > 50 && value2 > 50){
        document.getElementById('result').innerHTML = "<br>Very High
Temperature <br> Harmful gas is leaked";
      }
      else if(value1 > 50){
        document.getElementById('result').innerHTML = "<br>Very High
Temperature";
      }
      else if(value1 < 20){
        document.getElementById('result').innerHTML = "<br>Very Low
Temperature";
      }
      else if(value1 < 20 && value2>100){
        document.getElementById('result').innerHTML = "<br>Very Low
Temperature  <br> Harmful gas is leaked";
      }
      else if(value2>50){
        document.getElementById('result').innerHTML = " <br> Harmful gas is
leaked";
      }
      else{
        document.getElementById('result').innerHTML = "<br>Normal
Condition";
      }
    }
  })
  .catch((error) => {
    console.error('Error fetching data:', error);
  });
}

fetchdata();

setInterval(fetchdata,5000);
```

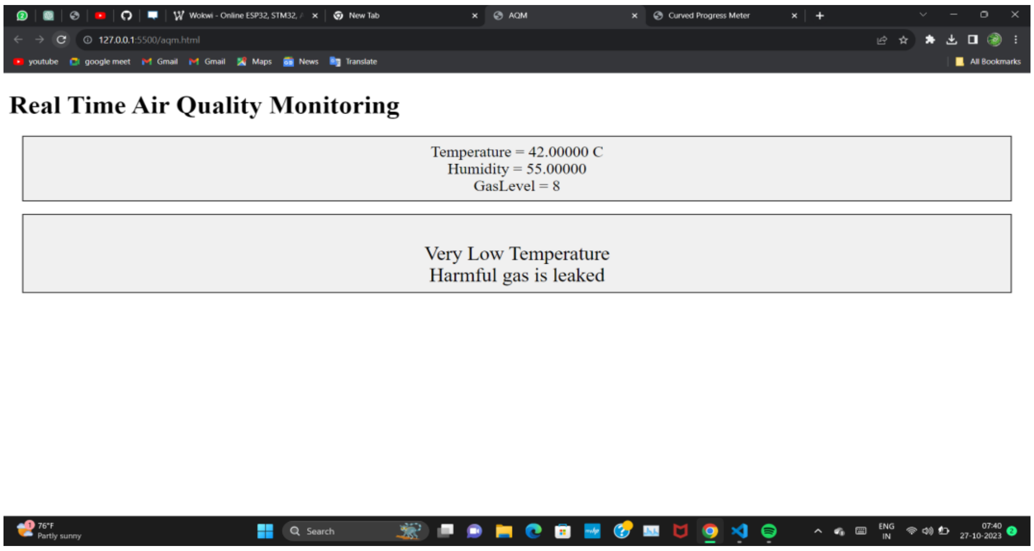
SAMPLE OUTPUT:

Wokwi output:



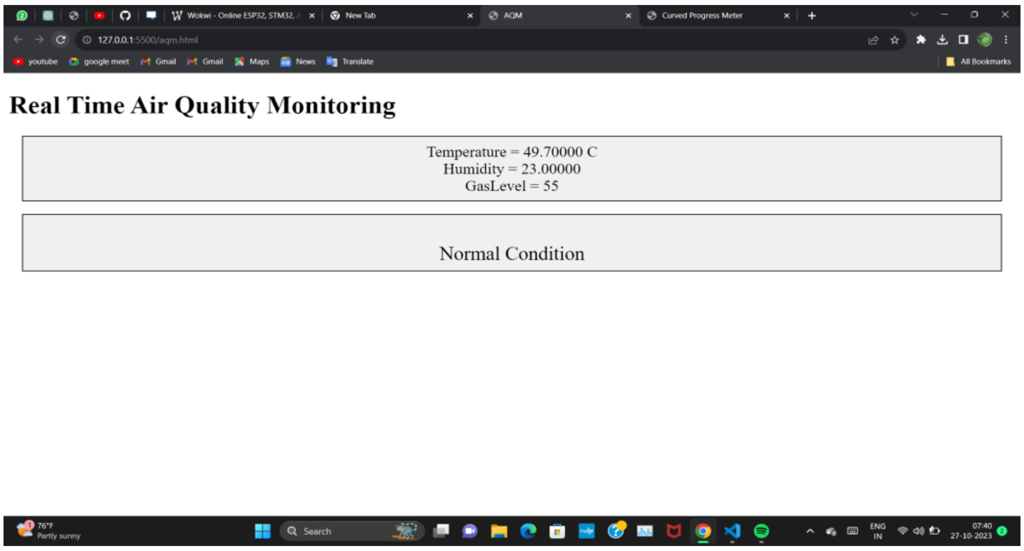
```
load:0x40080400,len:2972
entry 0x400805dc
attempting.connected
Humi:40.00
Temp:73.50
GAS level: 8
updates
```

Web:


Real Time Air Quality Monitoring

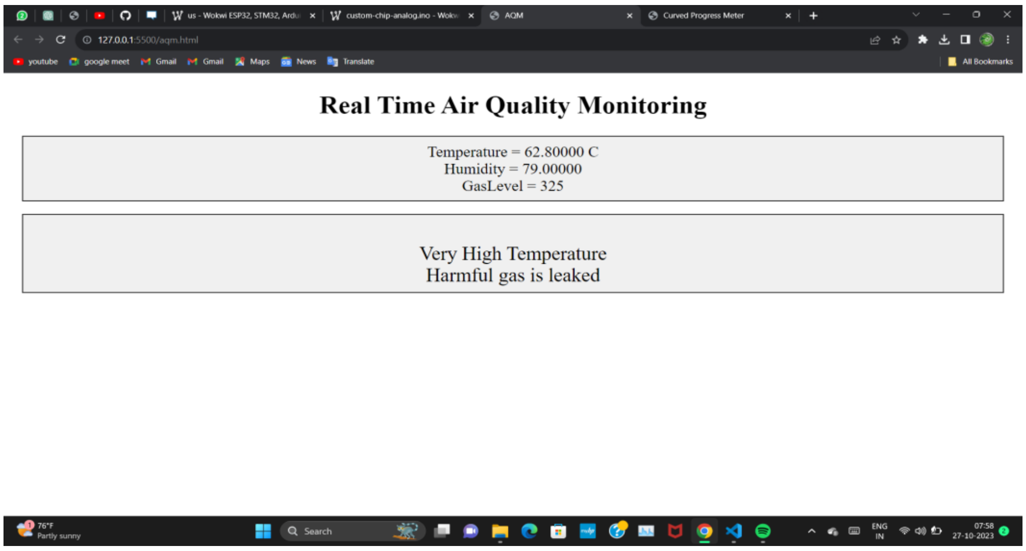
```
Temperature = 42.00000 C
Humidity = 55.00000
GasLevel = 8
```

```
Very Low Temperature
Harmful gas is leaked
```


Real Time Air Quality Monitoring

```
Temperature = 49.70000 C
Humidity = 23.00000
GasLevel = 55
```

```
Normal Condition
```

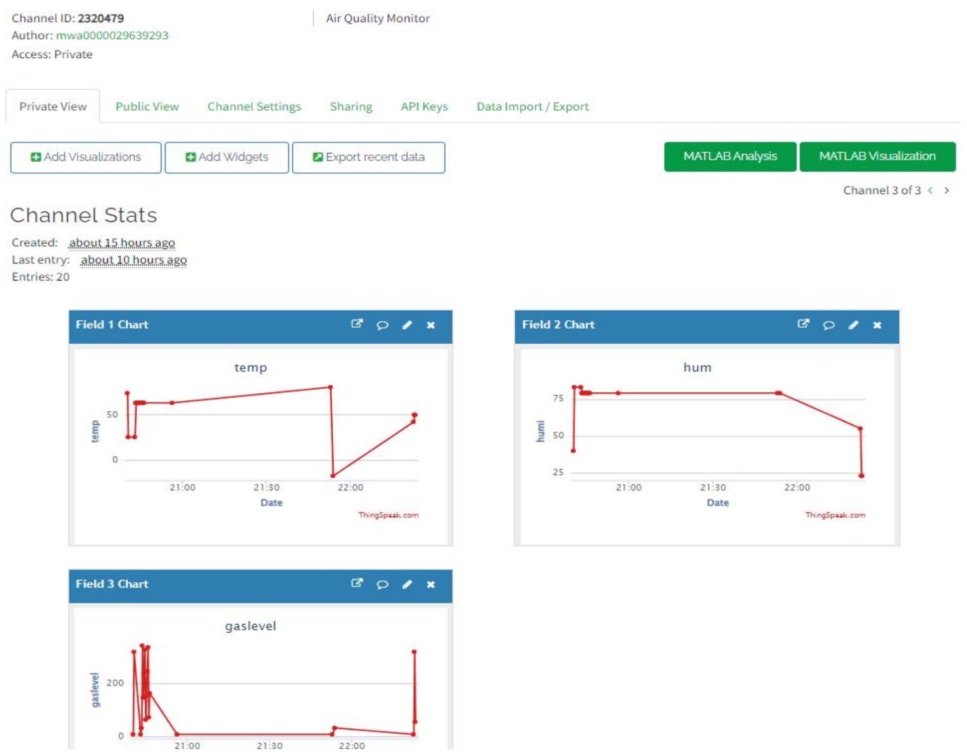


Real Time Air Quality Monitoring

Temperature = 62.80000 C
Humidity = 79.00000
GasLevel = 325

Very High Temperature
Harmful gas is leaked

ThingSpeak Cloud Data stats:



Channel ID: **2320479**
Author: mwa0000029639293
Access: Private

Air Quality Monitor

Channel Stats

Created: about 15 hours ago
Last entry: about 10 hours ago
Entries: 20