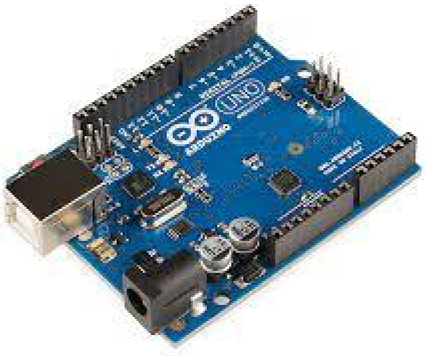
PHASE 5 - DOCUMENTATION

PROJECT OBJECTIVE:

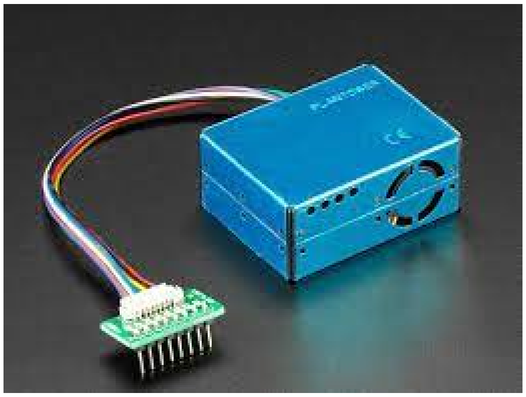
The Objective of this IOT project is to monitor the Air Quality by measuring the AQI(Air Quality Index) value based only on particulate matter.

DEVICE REQUIREMENT:

1. **ARDUINO BOARD - UNO**:

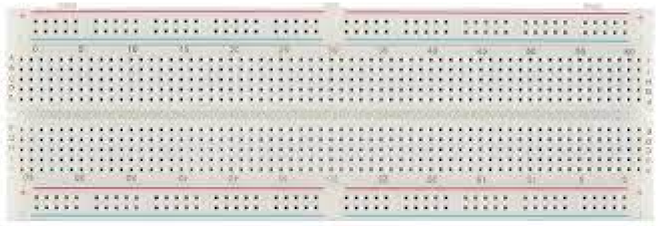


2. **PM2.5 SENSOR (Particulate Matter):**



3. **ESP8266 WIFI CHIP:**

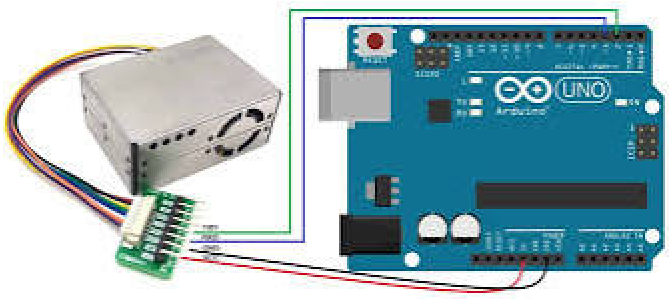
4. **BREAD BOARD:**



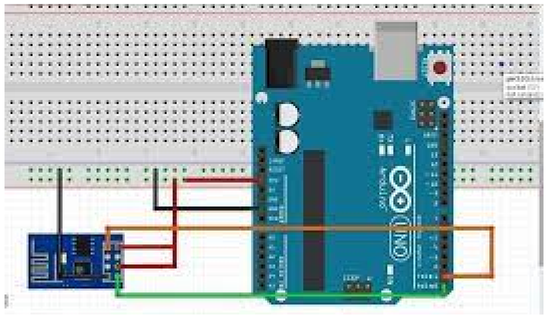
5. **ARDUINO UNO CABLE:**

IOT DEVICE SETUP:

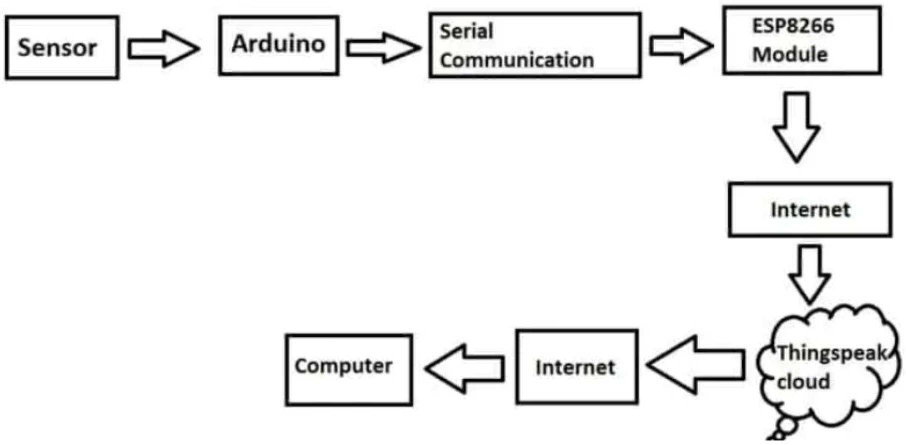
CONNECTING PM2.5 WITH ARDUINO UNO:



CONNECTING ESP8266 WITH ARDUINO UNO:



DATA TRANSMISSION FLOW:



CODE IMPLEMENTATION:

Integrating PM2.5 Sensor with Arduino UNO:

```
#include <SoftwareSerial.h>
SoftwareSerial pmsSerial(2, 3);
void setup() {
// our debugging output
Serial.begin(115200);
// sensor baud rate is 9600
pmsSerial.begin(9600);
}
struct pms5003data {
uint16_t framelen;
uint16_t pm10_standard, pm25_standard, pm100_standard;
uint16_t pm10_env, pm25_env, pm100_env;
uint16_t particles_03um, particles_05um, particles_10um, particles_25um,
particles_50um,
particles_100um;
uint16_t unused;
uint16_t checksum;
};
struct pms5003data data;
void loop() {
if (readPMSdata(&pmsSerial)) {
// reading data was successful!
Serial.println();
Serial.println("---------------------------------------");
Serial.println("Concentration Units (standard)");
Serial.print("PM 1.0: "); Serial.print(data.pm10_standard);
Serial.print("\t\tPM 2.5: "); Serial.print(data.pm25_standard);
Serial.print("\t\tPM 10: "); Serial.println(data.pm100_standard);
Serial.println("---------------------------------------");
Serial.println("Concentration Units (environmental)");
Serial.print("PM 1.0: "); Serial.print(data.pm10_env);
Serial.print("\t\tPM 2.5: "); Serial.print(data.pm25_env);
Serial.print("\t\tPM 10: "); Serial.println(data.pm100_env);
Serial.println("---------------------------------------");
Serial.print("Particles > 0.3um / 0.1L air:");
Serial.println(data.particles_03um);
Serial.print("Particles > 0.5um / 0.1L air:");
Serial.println(data.particles_05um);
Serial.print("Particles > 1.0um / 0.1L air:");
Serial.println(data.particles_10um);
Serial.print("Particles > 2.5um / 0.1L air:");
Serial.println(data.particles_25um);
Serial.print("Particles > 5.0um / 0.1L air:");
Serial.println(data.particles_50um);
Serial.print("Particles > 10.0 um / 0.1L air:");
Serial.println(data.particles_100um);
Serial.println("---------------------------------------");
}
}
boolean readPMSdata(Stream *s) {
```

```
if (! s->available()) {
return false;
}
// Read a byte at a time until we get to the special '0x42' start-byte
if (s->peek() != 0x42) {
s->read();
return false;
}
// Now read all 32 bytes
if (s->available() < 32) {
return false;
}
uint8_t buffer[32];
uint16_t sum = 0;
s->readBytes(buffer, 32);
// get checksum ready
for (uint8_t i=0; i<30; i++) {
sum += buffer[i];
}
/* debugging
for (uint8_t i=2; i<32; i++) {
Serial.print("0x"); Serial.print(buffer[i], HEX); Serial.print(", ");
}
Serial.println();
*/
// The data comes in endian'd, this solves it so it works on all platforms
uint16_t buffer_u16[15];
```

```
for (uint8_t i=0; i<15; i++) {
buffer_u16[i] = buffer[2 + i*2 + 1];
buffer_u16[i] += (buffer[2 + i*2] << 8);
}
// put it into a nice struct :)
memcpy((void *)&data, (void *)buffer_u16, 30);
if (sum != data.checksum) {
Serial.println("Checksum failure");
return false;
}
// success!
return true;
}
```

Connecting ESP8266 Sensor to ThinkSpeak:

```
#include “ThinkSpeak.h
#include <ESP8266WiFi.h>
char networkname[] = “”; // your network name
char passcode[] = “”; // your passcode
WiFiClient client;
unsigned long tsChannelID = ; // ThingSpeak Channel ID
const char * tsWriteAPIKey = “”; //ThingSpeak Write API Key
String airQuaility = “”;
const int fieldOne = 1;
void setup()
{
Serial.begin(115200);
```

```
WiFi.mode(WIFI_STA);
ThingSpeak.begin(client);
thingSpeak();
}
void loop()
{
thingSpeak();
if (Serial.available() > 0)
{
while (Serial.available() > 0)
{
int inChar = Serial.read();
airQuaility += (char)inChar;
}
}
pushData();
}
void thingSpeak()
{
if (WiFi.status() != WL_CONNECTED)
{
while (WiFi.status() != WL_CONNECTED)
{
WiFi.begin(networkname, passcode);
delay(5000);
}
}
```

```
}
void pushData()
{
int getData = ThingSpeak.writeField(tsChannelID, fieldOne, airQuaility,
tsWriteAPIKey);
if (getData != 200)
{
delay(15000);
pushData();
}
airQuaility = “”;
}
```

UI CODE:

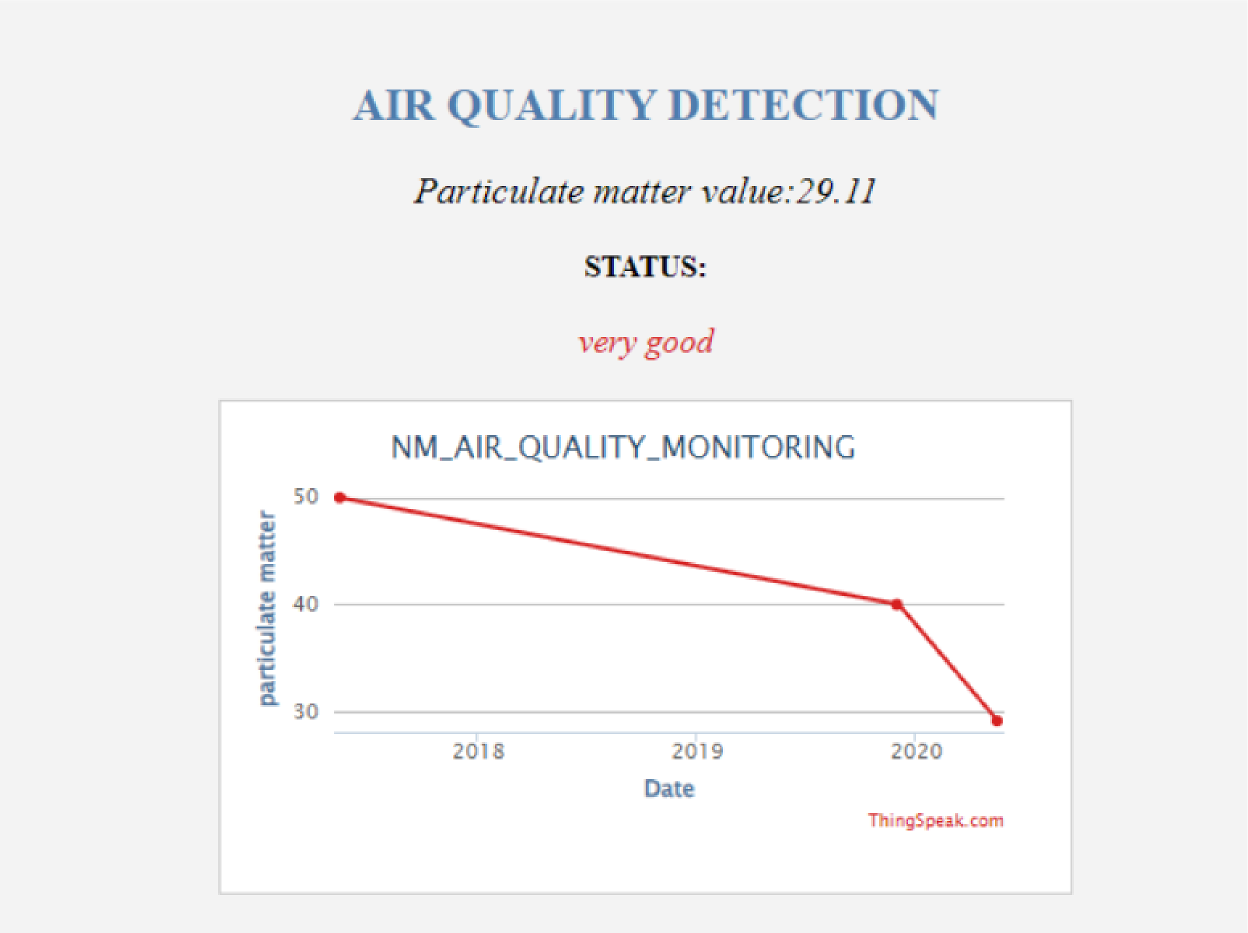
```
<!DOCTYPE html>
<html>
<head>
<title>ThingSpeak Data Validation</title>
<style>
.center {
margin: auto;
width: 80%;
padding: 20px;
background-color: #f3f3f3;
border-radius: 10px;
text-align: center;
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h2 {
color: #4e7cad;
}
#val1 {
font-style: italic;
font-size: 20px;
margin: 20px;
}
#data-validation {
font-style: italic;
font-size: 18px;
margin: 20px;
color: #d62020;
/* Red color for the status message */
}
iframe {
border: none;
}
</style>
</head>
<body>
<div class="center">
<h2>AIR QUALITY DETECTION</h2>
<div id="val1" style="font-style: italic;"></div>
```

```
<p><strong>STATUS:</strong></p>
<p id="data-validation" style="font-style: italic;"></p>
<iframe width="450" height="260" style="border: 1px solid #cccccc;"
src="https://thingspeak.com/channels/2320632/charts/1?bgcolor=%23ffffff&
color=%2
3d62020&dynamic=true&results=60&type=line&update=15"></iframe>
</div>
<script>
const channelId = '2320632'; // Replace with your ThingSpeak channel ID
const apiKey = 'Y44LFH2TF2O0W6UR'; // Replace with your ThingSpeak
read API key
const url =
`https://api.thingspeak.com/channels/${channelId}/feeds.json?api_key=${a
piKey}&resu
lts=1`;
fetch(url)
.then(response => response.json())
.then(data => {
const pm25Value = parseFloat(data.feeds[0].field1);
let Message, value;
if (pm25Value >= 0 && pm25Value <= 30) {
value = `Particulate matter value:${pm25Value}`
Message = `very good`;
} else if (pm25Value >= 31 && pm25Value <= 60) {
value = `Particulate matter value:${pm25Value}`
Message = `good`;
} else if (pm25Value >= 61 && pm25Value <= 90) {
```

```
value = `Particulate matter value:${pm25Value}`
Message = `fair`;
} else if (pm25Value >= 91 && pm25Value <= 120) {
value = `Particulate matter value:${pm25Value}`
Message = `poor`;
} else if (pm25Value >= 121 && pm25Value <= 250) {
value = `Particulate matter value:${pm25Value}`
Message = `very poor`;
} else {
value = `Particulate matter value:${pm25Value}`
Message = `harzardous`;
}
const val = document.getElementById('val1');
val.textContent = value;
const validationDiv = document.getElementById('data-validation');
validationDiv.textContent = Message;
})
.catch(error => console.error('Error fetching data:', error));
</script>
</body>
</html>
```

OUTPUT:

Output page of the Application developed



AIR QUALITY DETECTION

Particulate matter value:29.11

STATUS:

very good

PLATFORM DEVELOPMENT:

FRONTEND : HTML,CSS,JAVA SCRIPT

BACKEND AND DATA-TRANSMISSION: THINGSPEAK

FOR DATA RETRIEVAL:

We use arduino IDE to read the sensor and transfer it to the Thingspeak using wifi module.

IMPROVEMENT IN PUBLIC AWARENESS:

1. **Educational Campaigns:**

   - Develop informative brochures, pamphlets, or websites that explain what PM2.5 is, why it's harmful, and how it affects health. Make sure to use accessible language and visuals.

   - Organize workshops and seminars at schools, community centers, and workplaces to educate people about air quality and its impact on health.

2. **Social Media and Online Presence:**

   - Use social media platforms to share air quality information, health tips, and updates. Engage with the community through posts, videos, and live sessions.

   - Dedicated website or mobile app to provide real-time air quality data, health recommendations, and resources for further reading.

NOTE:

We have submitted the files regarding this Air quality Project within the repository as files:

1. **PM_with_Arduino.ino**
2. **ESP_with_Thinkspeak.ino**
3. **Index.html**