Code

```
#include <WiFi.h>
#include <DHT.h>
#include <ThingSpeak.h>
#include <MQUnifiedsensor.h>

#define Pin_DHT 12
#define Type_DHT DHT11
DHT dht(Pin_DHT, Type_DHT);

#define Pin_MQ2 32
#define Pin_MQ135 33

const char* ssid = "Jayanth.S.B";
const char* password = "Jayrocks";
const char* apiKey = "BGQJV4IXZTA656D9";
const long channelID = 2682650;
WiFiClient client;

#define Voltage_Resolution 3.3
#define ADC_Bit_Resolution 12
#define RatioMQ2CleanAir 9.83
MQUnifiedsensor MQ2("ESP-32", Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ2, "MQ-2");

const float R0_MQ135 = 51.632;
MQUnifiedsensor MQ135("ESP-32", Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ135, "MQ-135");

void setup() {
  Serial.begin(115200);
```

```
 WiFi.begin(ssid, password);
 Serial.print("Establishing WiFi Connection");
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(" ... ");
 }
 Serial.println("\nConnection Established");
 Serial.println(WiFi.localIP());
 dht.begin();
 MQ2.setRegressionMethod(1);
 MQ2.setA(605.18); MQ2.setB(-3.937);
 MQ2.init();
 MQ135.setRegressionMethod(1);
 MQ135.setA(110.47); MQ135.setB(-2.862);
 MQ135.init();
 MQ135.setR0(R0_MQ135);
 ThingSpeak.begin(client);
}

void Reconnect() {
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(" ... ");
 }
 Serial.println("Reconnected");
}

void loop() {
 if (WiFi.status() != WL_CONNECTED) {
  Reconnect();
```

```
 }

 float humidsense = dht.readHumidity();
 float tempsense = dht.readTemperature();

 MQ2.update();
 float methane_ppm = MQ2.readSensor();
 float smoke_ppm = MQ2.readSensor();

 MQ135.update();
 float co_ppm = MQ135.readSensor();
 float ammonia_ppm = MQ135.readSensor();

 int aqi_methane = calculateAQI(methane_ppm, 0, 200, 1, 100);
 int aqi_smoke = calculateAQI(smoke_ppm, 0, 300, 1, 150);
 int aqi_co = calculateAQI(co_ppm, 0, 50, 1, 100);
 int aqi_ammonia = calculateAQI(ammonia_ppm, 0, 300, 1, 150);

 thingcompute(methane_ppm, smoke_ppm, co_ppm, ammonia_ppm, humidsense,
tempsense, aqi_methane, aqi_smoke, aqi_co, aqi_ammonia);
 ThingSpeak.writeFields(channelID, apiKey);

 delay(2500);
}

void thingcompute(float methane, float smoke, float co, float ammonia, float humidsense, float
tempsense, int aqi_methane, int aqi_smoke, int aqi_co, int aqi_ammonia) {
 ThingSpeak.setField(6, tempsense);
 ThingSpeak.setField(5, humidsense);
 ThingSpeak.setField(1, methane);
 ThingSpeak.setField(2, smoke);
 ThingSpeak.setField(3, co);
```

```
  ThingSpeak.setField(4, ammonia);


  Serial.print("Temperature = " + String(tempsense) + " °C; ");
  Serial.println("Humidity = " + String(humidsense) + " %; ");
  Serial.print("Methane = " + String(methane) + " ppm, AQI = " + String(aqi_methane) + "; ");
  Serial.print("Smoke = " + String(smoke) + " ppm, AQI = " + String(aqi_smoke) + "; ");
  Serial.print("CO = " + String(co) + " ppm, AQI = " + String(aqi_co) + "; ");
  Serial.println("Ammonia = " + String(ammonia) + " ppm, AQI = " + String(aqi_ammonia) +
";\n\n");


  Serial.print(String(tempsense) + "\t");
  Serial.print(String(humidsense) + "\t");
  Serial.print(String(methane) + "\t");
  Serial.print(String(smoke) + "\t");
  Serial.print(String(co) + "\t");
  Serial.println(String(ammonia));
  Serial.println("\n\n");
}

int calculateAQI(float concentration, int minPPM, int maxPPM, int minAQI, int maxAQI) {
  if (concentration < minPPM) return minAQI;
  if (concentration > maxPPM) return maxAQI;
  return ((concentration - minPPM) * (maxAQI - minAQI)) / (maxPPM - minPPM) + minAQI;
}
```

Results

Code.ino

```
/*************************Globals for MQ135 sensor************************************/
const float R0_MQ135 = 51.632; // Pre-calculated R0 value for MQ135 (from calibration)
MQUnifiedsensor MQ135("ESP-32", Voltage_Resolution, ADC_Bit_Resolution, Pin_MQ135, "MQ-135");

void setup() {
  Serial.begin(115200);

  // WiFi connection
  WiFi.begin(ssid, password);
  Serial.print("Establishing WiFi Connection");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(" ... ");
  }
  Serial.println("\nConnection Established");
  Serial.println(WiFi.localIP());

  // Sensor initialization
  dht.begin();
  MQ2.setRegressionMethod(1); // 1 = Exponential, 0 = linear for MQ2
  MQ2.setA(605.18); MQ2.setB(-3.937); // Constants for MQ2 gas calculation
  MQ2.init();

  MQ135.setRegressionMethod(1);
  MQ135.setA(110.47); MQ135.setB(-2.862); // Constants for MQ135 gas calculation
```

Output Serial Monitor ×

Message (Enter to send message to 'ESP32-WROOM-DA Module' on 'COM4') | New Line | 115200 baud

```
23:25:50.421 ->
23:25:50.421 ->
23:25:50.421 -> 28.50   69.00   32.40   48.60   2.25    3.15
23:25:50.421 ->
23:25:50.421 ->
23:25:50.421 ->
```

Output Serial Monitor ×

Message (Enter to send message to 'ESP32-WROOM-DA Module' on 'COM4') | New Line | 115200 baud

```
23:24:37.927 ->
23:24:42.398 -> Temperature = 28.00 °C; Humidity = 69.00 %;
23:24:42.431 -> Methane = 74.00 ppm; Smoke = 111.00 ppm;
23:24:42.431 -> CO = 8.25 ppm; Ammonia = 11.55 ppm;
23:24:42.431 ->
23:24:42.431 ->
23:24:42.431 -> 28.00   69.00   74.00   111.00  8.25    11.55
23:24:42.431 ->
23:24:42.431 ->
```

ST_Project - Air Quality Monitoring

Channel ID: **2682650**
Author: mwa0000027892153
Access: Public

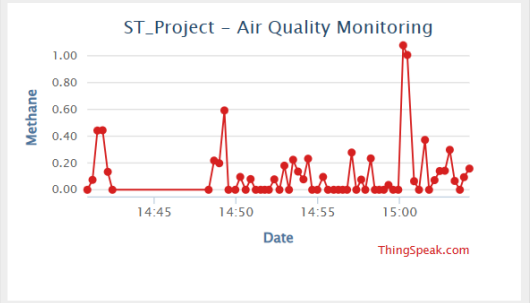
MQ2, MQ135, DHT11 Data Collection through ESP32

Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export

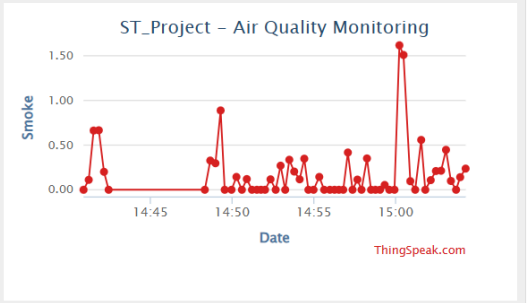
Add Visualizations | Add Widgets | Export recent data | MATLAB Analysis | MATLAB Visualization

Channel 2 of 3 < >

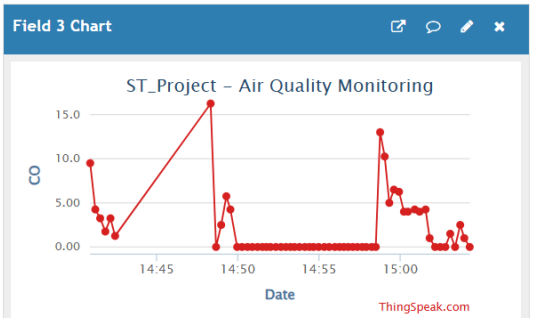
Field 1 Chart



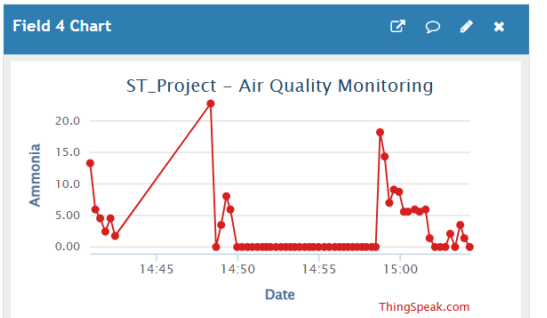
Field 2 Chart



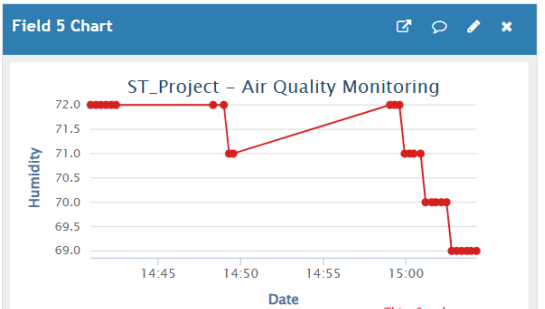
Field 3 Chart



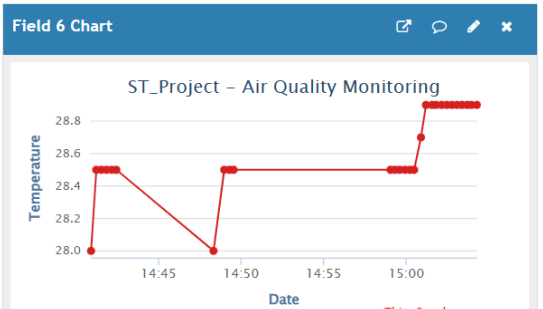
Field 4 Chart



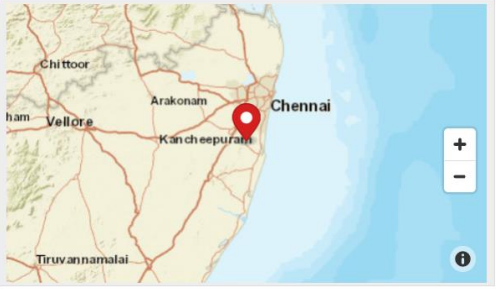
Field 5 Chart



Field 6 Chart



Channel Location



Channel Status Updates