NEO-6M GPS Module

The NEO-6M GPS module is a robust GPS receiver featuring a built-in ceramic antenna measuring 25 x 25 x 4mm, enhancing its satellite search capabilities. The power and signal indicators provide real-time module status monitoring[3].

FEATURES[3]

- 5Hz position update rate
- EEPROM to save configuration settings
- Rechargeable battery for Backup
- Support SBAS (WAAS, EGNOS, MSAS, GAGAN)
- Separated 18X18mm GPS antenna

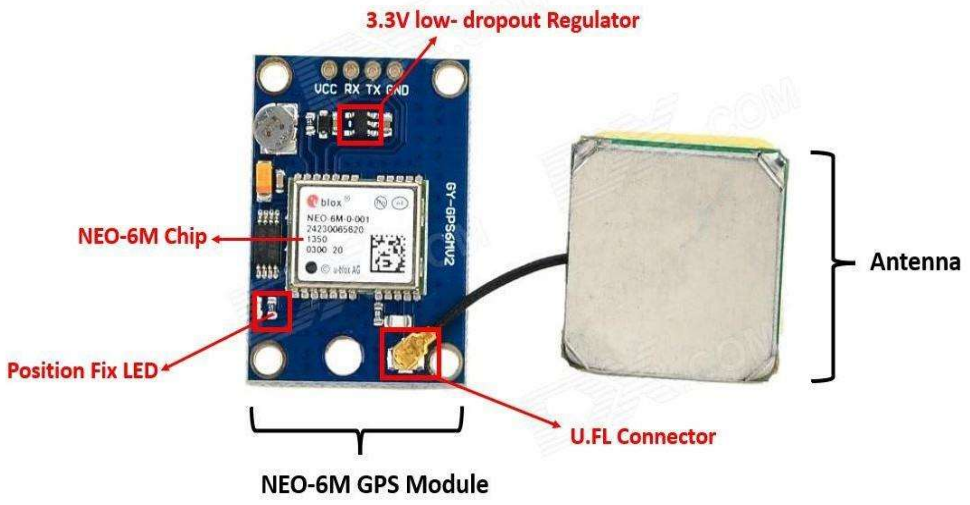


Fig.1:-NEO-6M GPS Module[3]

M20 PRESSURE SENSOR

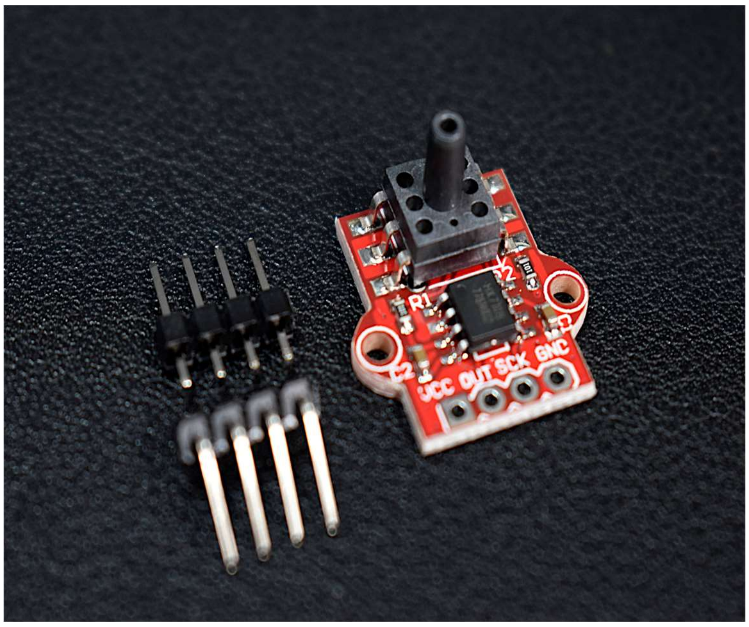


Fig.1:-MP20 Pressure Sensor

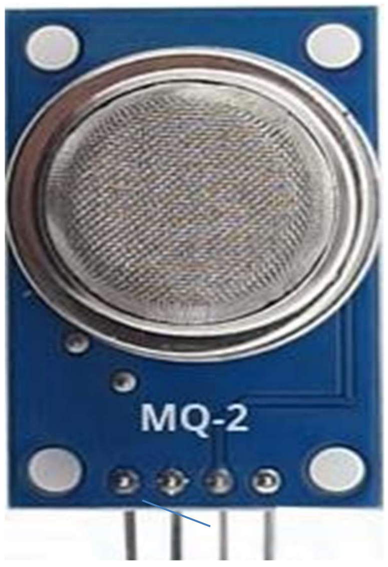
Pressure sensors like the MP20 are devices used to measure pressure in various applications, from industrial processes to automotive systems and medical devices. They're designed to detect and convert pressure into an electrical signal that can be interpreted and used for monitoring or control purposes.

The specific functionalities and technical details of the MP20 pressure sensor could vary based on the manufacturer and its intended use.

MQ2 Gas/Smoke Sensor

The MQ2 sensor is one of the most widely used in the MQ sensor series. It is a MOS (Metal Oxide Semiconductor) sensor. Metal oxide sensors are also known as **Chemiresistors** because sensing is based on the change in resistance of the sensing material when exposed to gasses[4].

The MQ2 gas sensor operates on 5V DC and consumes approximately 800mW.It can detect LPG**,** Smoke**,** Alcohol**,** Propane**,** Hydrogen**,** Methane.



*A*0 D0 GND VCC

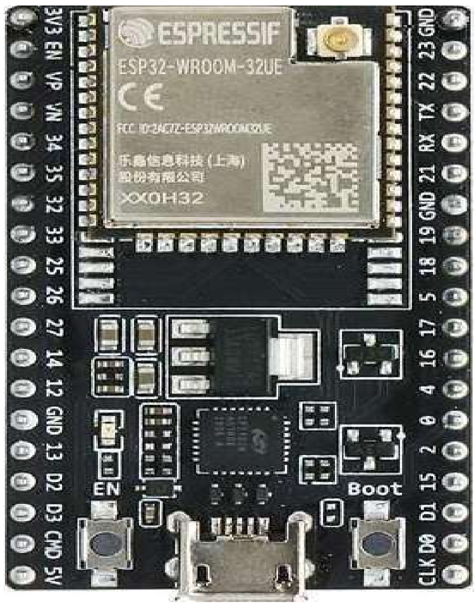
Fig.2:-MQ2 Gas Sensor[4]

ESP32 Fundamental

ESP32(Fig.1) is a powerful, low-cost, and highly integrated microcontroller developed by Espressif Systems. It also has built-in Wi- Fi and Bluetooth connectivity, making it ideal for IoT applications. The Wi-Fi connectivity supports both 2.4GHz and 5GHz frequency bands,while the Bluetooth connectivity supports classic Bluetooth and Bluetooth Low Energy (BLE) protocols [5].

The ESP32 microcontroller comes with a range of interfaces, including SPI, I2C, UART, I2S, PWM, and ADC. It also has a built-in hall sensor, temperature sensor, and touch sensor, making it suitable for a wide range of applications.

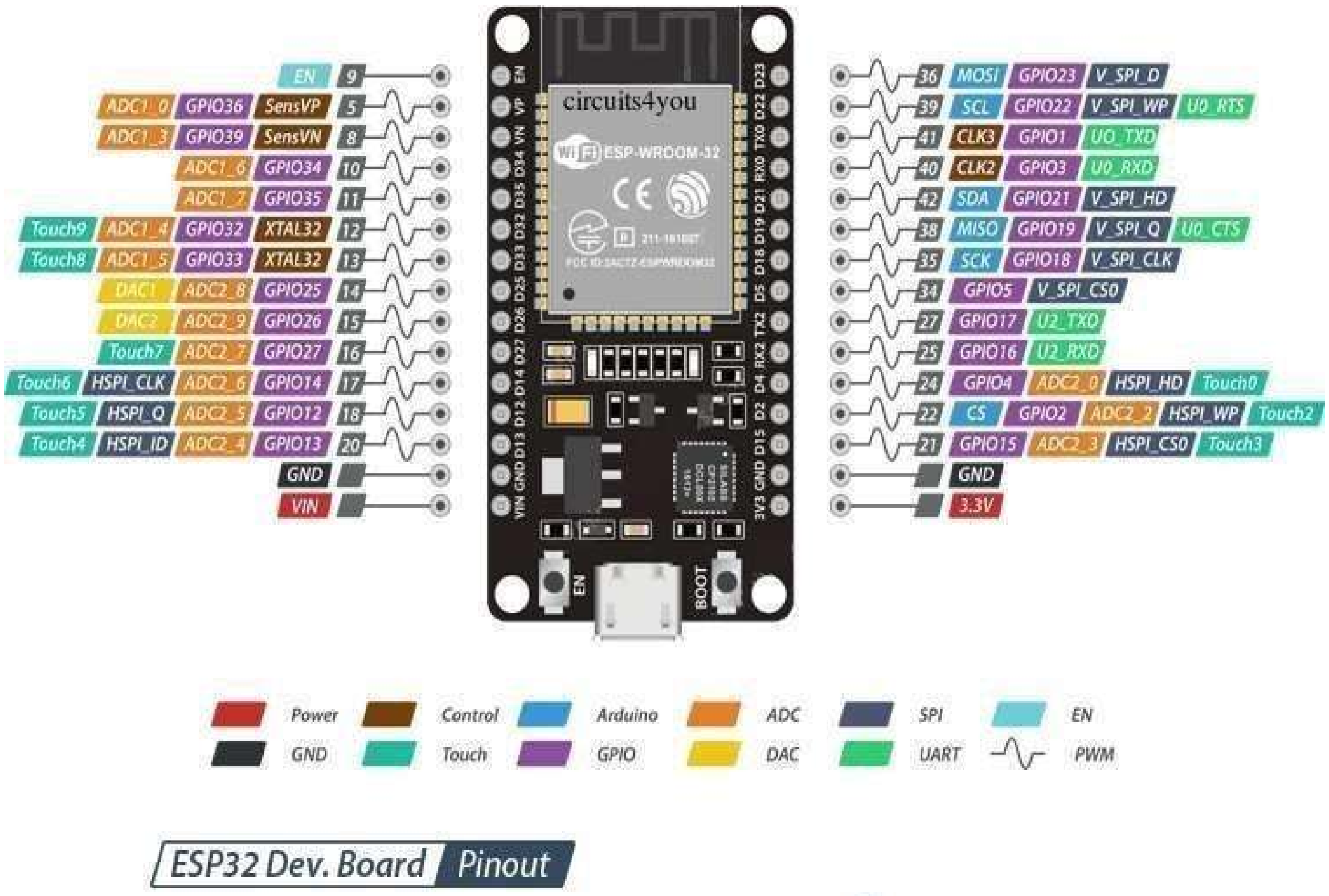
The ESP32 microcontroller supports several programming languages, including C, C++, and MicroPython. It also has an integrated development environment (IDE) called the ESP-IDF, which provides a set of software development tools for developing applications [5].



ESPRESSIF
ESP32-WROOM-32UE
3V3 EN VP VN 34 35 32 33 25 26 27 14 12 GND 13 D2 D3 CMD 5V
GND 23 22 TX RX 21 GND 19 18 5 17 16 4 0 2 15 D1 D0 CLK
EN
Boot

Fig.3:- ESP32 Board [5]

<u>ESP32 pin configuration is mentioned below:</u>- [5]



AskSensors

AskSensors is an IoT platform that enables users to collect, visualize, and analyze sensor data from various IoT devices. It provides tools to create dashboards, set up triggers and notifications, and monitor real-time data from sensors.

In the given code, AskSensors is used as the endpoint to which sensor data from the ESP32 board is sent. The MQTT protocol is employed to publish the sensor data to AskSensors.

Here's how AskSensors is utilized in the provided code:

Configuration: The code includes configuration details such as the MQTT server (mqtt.asksensors.com), MQTT port (typically 1883), and specific topics (pubTopic, pubTopic1) that represent the channels where the sensor data is published.

Data Publishing: The sensor data, including GPS coordinates, gas sensor readings, pressure, humidity, and temperature values, are formatted and published as MQTT messages to AskSensors. These messages contain the sensor data in specific payloads that are then sent to the topics configured in the code.

Visualization and Analysis: AskSensors receives the data published by the ESP32 and displays it on the AskSensors web application. Users can log in to their AskSensors account, configure their channels corresponding to the provided topics, and create visual representations (charts, graphs, etc.) of the incoming sensor data. This allows users to monitor and analyze the data in real-time.

In summary, AskSensors acts as the receiving endpoint for sensor data transmitted by the ESP32 using MQTT. It provides a platform for users to visualize, analyze, and manage the received sensor data through its web application.[10]

Chapter 2:-

Connection Diagram [7]

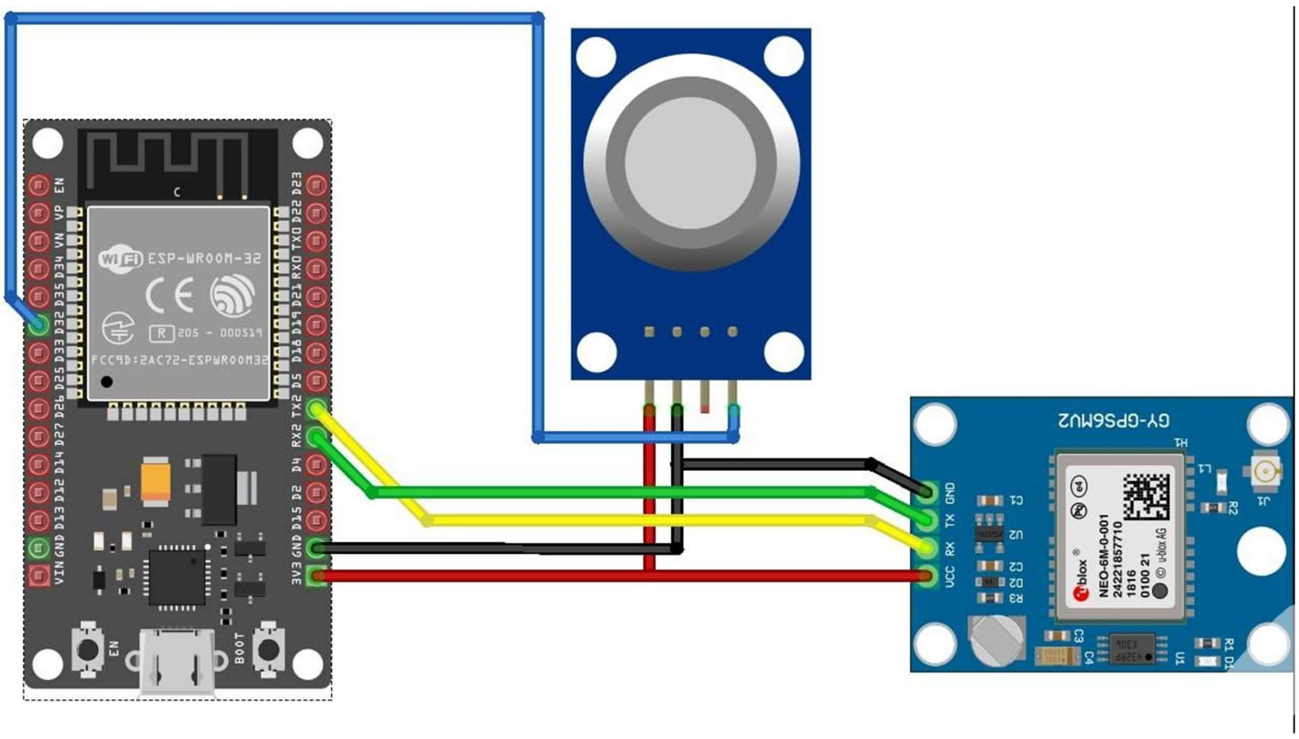
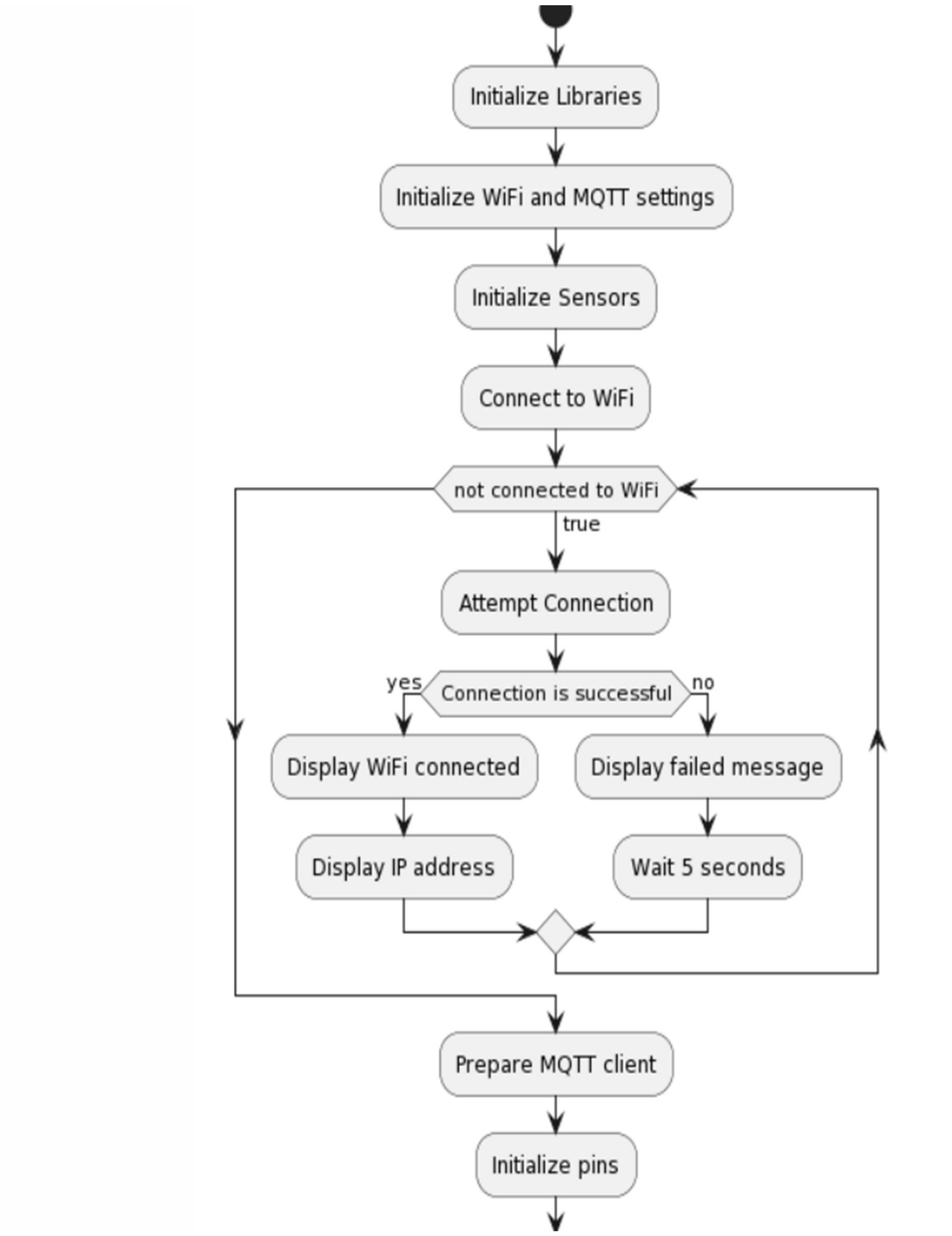
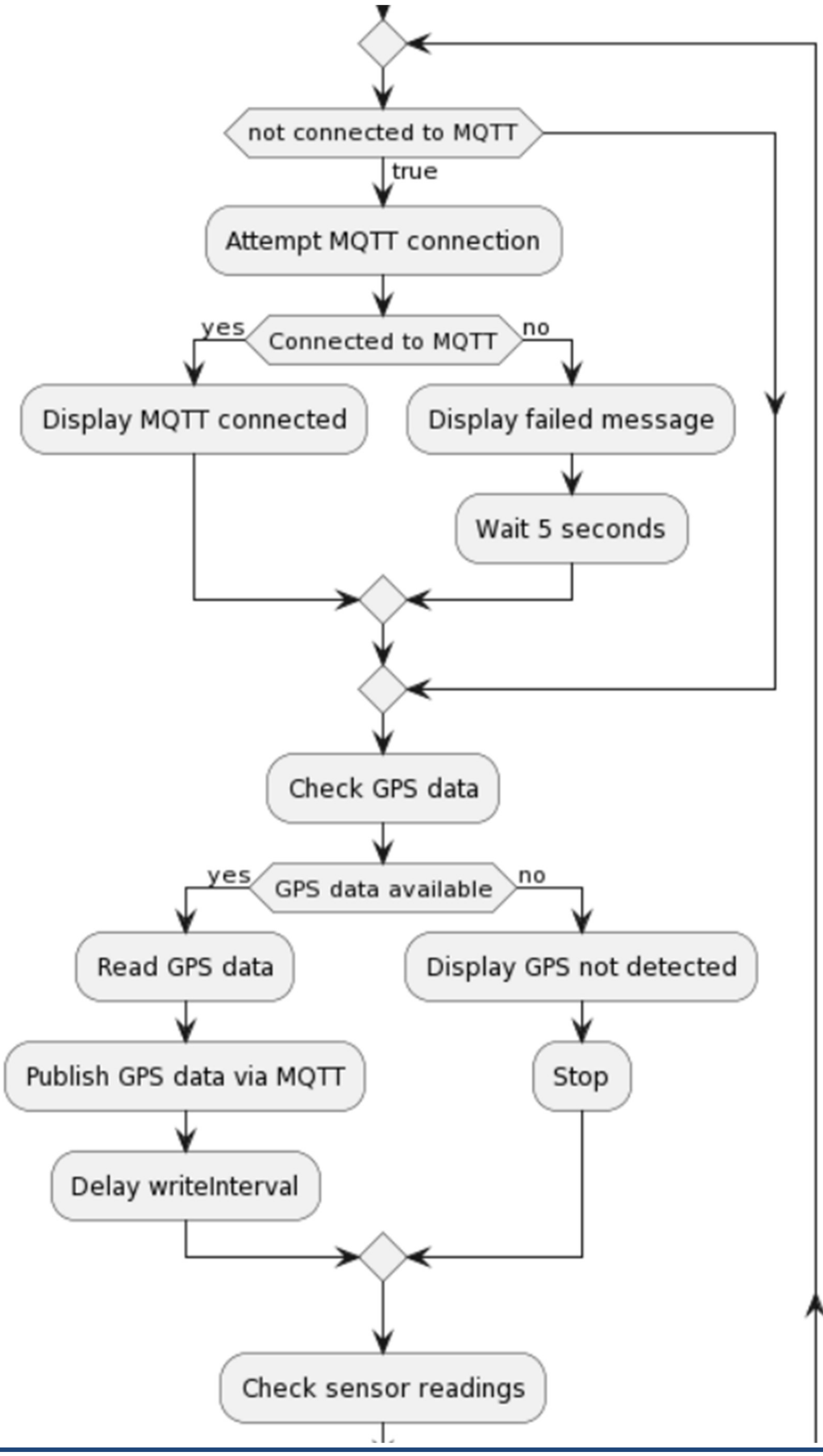


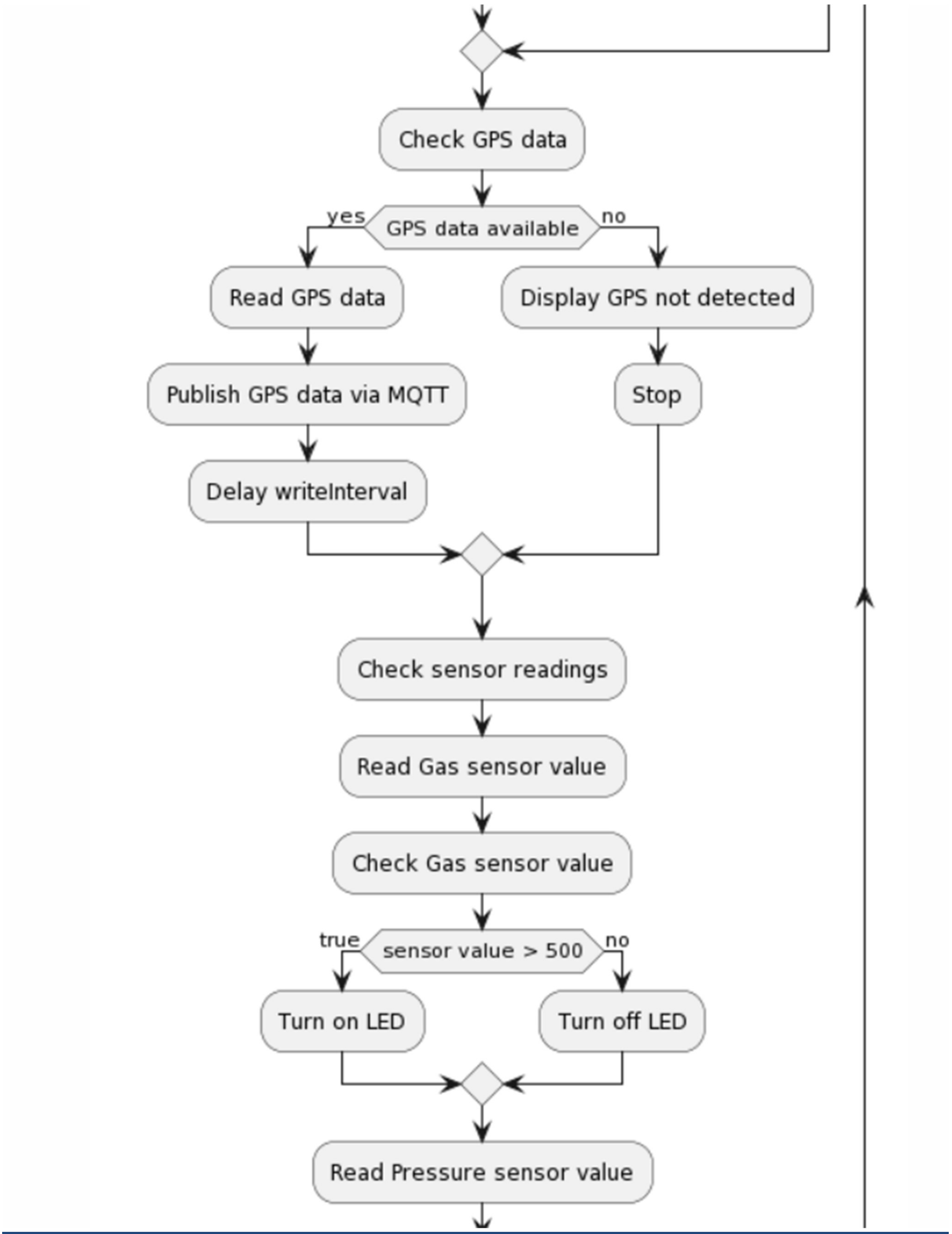
Fig .4:- Connection Diagram

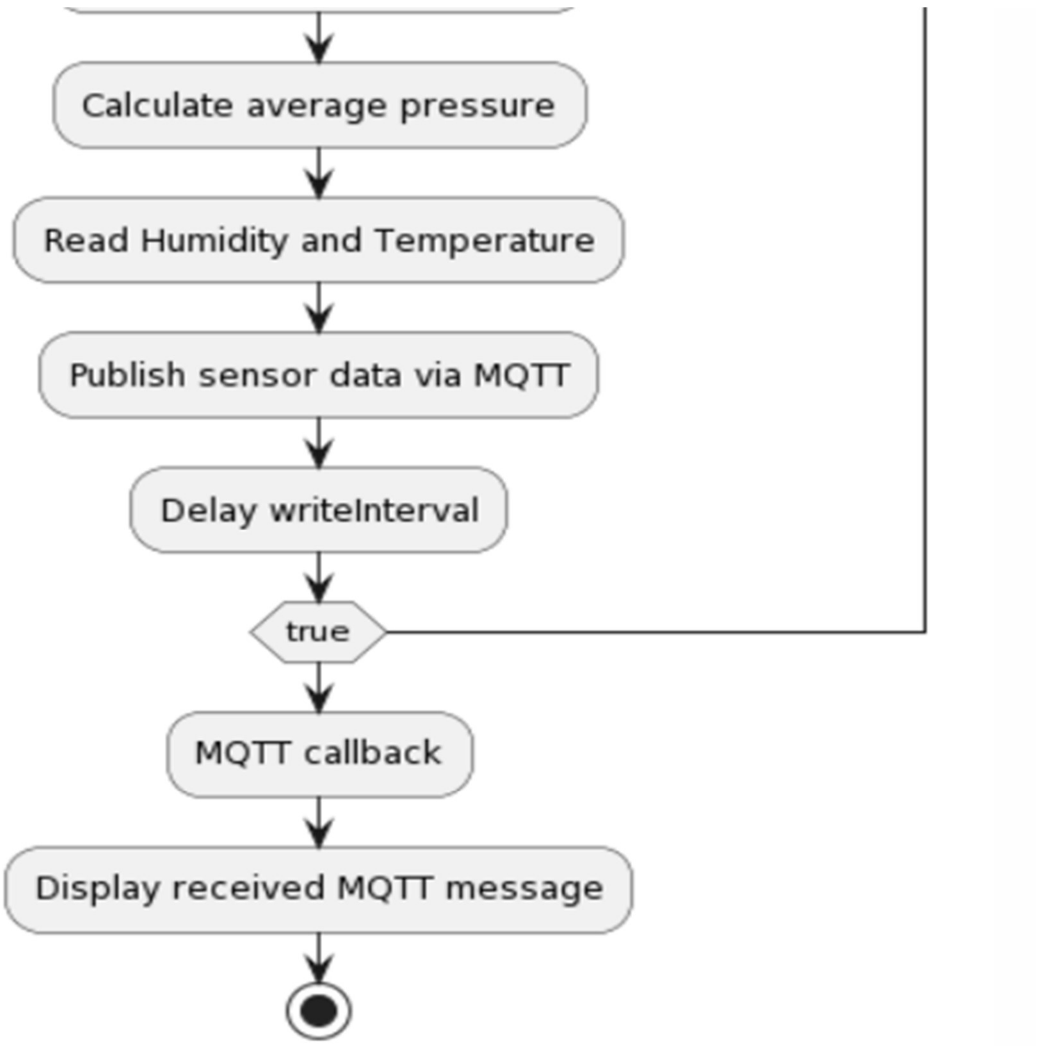
Chapter 3:-

Flowchart of the code [8]




not connected to MQTT
true
Attempt MQTT connection
Connected to MQTT
yes
no
Display MQTT connected
Display failed message
Wait 5 seconds
Check GPS data
GPS data available
yes
no
Read GPS data
Display GPS not detected
Publish GPS data via MQTT
Stop
Delay writeInterval
Check sensor readings


Check GPS data
GPS data available
yes
no
Read GPS data
Display GPS not detected
Publish GPS data via MQTT
Stop
Delay writeInterval
Check sensor readings
Read Gas sensor value
Check Gas sensor value
sensor value > 500
true
no
Turn on LED
Turn off LED
Read Pressure sensor value


Calculate average pressure
Read Humidity and Temperature
Publish sensor data via MQTT
Delay writeInterval
true
MQTT callback
Display received MQTT message

Chapter 4:-

Code [9]

```
#include <TinyGPSPlus.h>
#include<Q2HX711.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

#define DHTPIN 19
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

//TODO: ESP32 MQTT user config
const char* ssid = "OPPO A15s"; // Wifi SSID
const char* password = "hemantkumar"; // Wifi Password
const char* username = "hemant207"; // my AskSensors username
const char* pubTopic = "publish/hemant207/Dc0ZSWntrzrE92gRvsAVUQQu9leP0UCw"; //
publish/username/apiKeyIn
const char* pubTopic1 = "publish/hemant207/SxrMLATmXXljn2kO8To8uoct51NTre2F";
const unsigned int writeInterval = 2500; // write interval (in ms)
//AskSensors MQTT config
const char* mqtt_server = "mqtt.asksensors.com";
unsigned int mqtt_port = 1883;

// objects
WiFiClient askClient;
PubSubClient client(askClient);


TinyGPSPlus gps;

int  MQ2pin = 5;       //Gas Sensor
float sensorValue;      //Gas Sensor value


int ledpin = 18;

const byte MPS_OUT_pin = 2;
const byte MPS_SCK_pin = 4;
int avg_size = 10;
Q2HX711 MPS20N0040D(MPS_OUT_pin, MPS_SCK_pin);
```

```
void setup() {
  Serial.begin(9600);
  Serial2.begin(9600);
    Serial.println("*****************************************************");
    Serial.println("********** Program Start : ESP32 publishes NEO-6M GPS position to
AskSensors over MQTT");
    Serial.print("********** connecting to WIFI : ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("");
    Serial.println("->WiFi connected");
    Serial.println("->IP address: ");
    Serial.println(WiFi.localIP());

    client.setServer(mqtt_server, mqtt_port);
    client.setCallback(callback);
    Serial.println("MQ2 is ready");
    pinMode(ledpin, OUTPUT);
    digitalWrite(ledpin,LOW);
    dht.begin();
    delay(2000);
}


void loop() {
  if (!client.connected())
  reconnect();
  client.loop();

  //updateSerial();GPS
  while (Serial2.available() > 0)
    if (gps.encode(Serial2.read()))
      displayInfo();
  if (millis() > 5000 && gps.charsProcessed() < 10)
  {
    Serial.println(F("No GPS detected: check wiring."));
    while (true);
  }
  Serial.println("The sensrs readings are : ");
  displayInfo1();


  delay(200);
}

// GPS displayInfo
void displayInfo() {

if (gps.location.isValid()) {
double latitude = (gps.location.lat());
double longitude = (gps.location.lng());

Serial.println("********** Publish MQTT data to ASKSENSORS");
char mqtt_payload[50] = "";

snprintf (mqtt_payload, 50, "m1=%lf;%lf", latitude, longitude);

Serial.print("Publish message: ");
Serial.println(mqtt_payload);
client.publish(pubTopic, mqtt_payload);

Serial.println("> GPS data published");

Serial.println("*******************************************************");

delay(writeInterval);// delay
} else {
Serial.println(F("INVALID"));
}

}

void displayInfo1() {


  //Gas sensor readings
  sensorValue = analogRead(MQ2pin);
  Serial.print("Sensor Value : ");
  Serial.println(sensorValue);
  delay(700);

  if(sensorValue>500){
    digitalWrite(ledpin,HIGH);
  }else{
   digitalWrite(ledpin,LOW);
  }

  //pressure sensor
  float avg_val = 0.0;
  for (int ii=0;ii<avg_size;ii++){
   avg_val += MPS20N0040D.read();
   delay(50);
  }
  avg_val /= avg_size;
  Serial.println(avg_val,0);
  delay(200);

float h = dht.readHumidity();
// Read temperature as Celsius (the default)
float t = dht.readTemperature();
  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));

Serial.println("********** Publish MQTT data to ASKSENSORS");

char mqtt_payload1[50] = "";
char mqtt_payload2[50] = "";
char mqtt_payload3[50] = "";

snprintf (mqtt_payload1, 50, "m2=%d",avg_val );
snprintf (mqtt_payload2, 50, "m3=%lf", t);
snprintf (mqtt_payload3, 50, "m4=%lf", h);
Serial.print("Publish message: ");
Serial.println(mqtt_payload);

client.publish(pubTopic, mqtt_payload1);Serial.print(",");
client.publish(pubTopic, mqtt_payload2);Serial.print(",");
client.publishln(pubTopic, mqtt_payload3);
Serial.println("> Sensors data published");
Serial.println("********** End ");
Serial.println("*****************************************************");

delay(writeInterval);// delay
```

```
} else {
Serial.println(F("INVALID"));
}

}

void updateSerial()
{
  delay(500);
  while (Serial.available())
  {
    Serial2.write(Serial.read());//Forward what Serial received to Software Serial Port
  }
  while (Serial2.available())
  {
    Serial.write(Serial2.read());//Forward what Software Serial received to Serial Port
  }
}


//MQTT reconnect
void reconnect() {
// Loop until we're reconnected
while (!client.connected()) {
Serial.print("********** Attempting MQTT connection...");
// Attempt to connect
if (client.connect("ESP32Client", username, "")) {
Serial.println("-> MQTT client connected");
} else {
Serial.print("failed, rc=");
Serial.print(client.state());
Serial.println("-> try again in 5 seconds");
// Wait 5 seconds before retrying
delay(5000);
}
}
}

//MQTT callback
void callback(char* topic, byte* payload, unsigned int length) {
Serial.print("Message arrived [");
Serial.print(topic);
Serial.print("] ");
for (int i = 0; i < length; i++) {
```

```
Serial.print((char)payload[i]);
}
Serial.println();

}
```