

Sistem Pemantauan Makanan Berbasis IoT



Penting untuk menjaga keamanan dan kebersihan makanan agar tetap segar dan dapat dimakan sehingga membantu **mengurangi pemborosan makanan**. Salah satu solusinya adalah dengan menjaga kondisi lingkungan yang sesuai untuk makanan yang disimpan guna mengendalikan laju pembusukan. Ada beberapa parameter berbeda yang menentukan penguraian makanan, parameter seperti kelembapan, bakteri, dan suhu merupakan faktor utama yang menentukan laju penguraian makanan. Jika suhu penyimpanan antara 40F hingga 140F, maka ini merupakan zona bahaya karena pada suhu tersebut bakteri tumbuh dengan cepat, jumlahnya berlipat ganda dalam waktu 20 menit. Demikian pula, kelembapan di ruang penyimpanan makanan harus berkisar 50-55% untuk menjaga kualitas makanan tetap tinggi selama mungkin.

Pada proyek kali ini, kita akan membangun **perangkat Food Monitoring menggunakan NodeMCU dan Arduino IDE**, untuk memantau suhu dan kelembaban lingkungan yang disimpan serta mengendalikannya. Untuk mengontrol suhu, kita akan menggunakan motor DC sebagai mekanisme pendinginan. Untuk mengetahui suhu, dan kelembaban digunakan modul **sensor DHT11**, dan untuk mengetahui status makanan digunakan **modul sensor gas MQ4**.

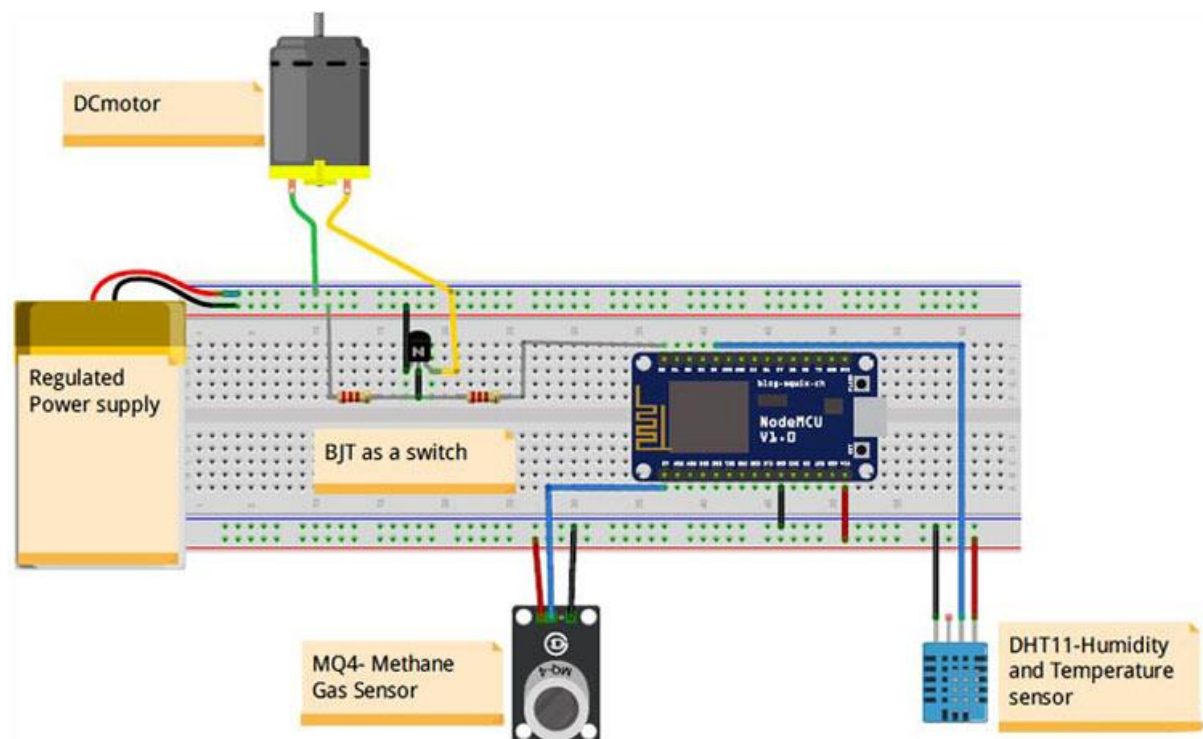
Nilai suhu, kelembapan, dan gas metana secara real-time akan diukur dan dikirim melalui web untuk ditampilkan di dalamnya. Jika suhu berada pada nilai kritis, kita akan menerima email peringatan, dan kipas juga akan dikontrol secara otomatis.

Alat dan Bahan

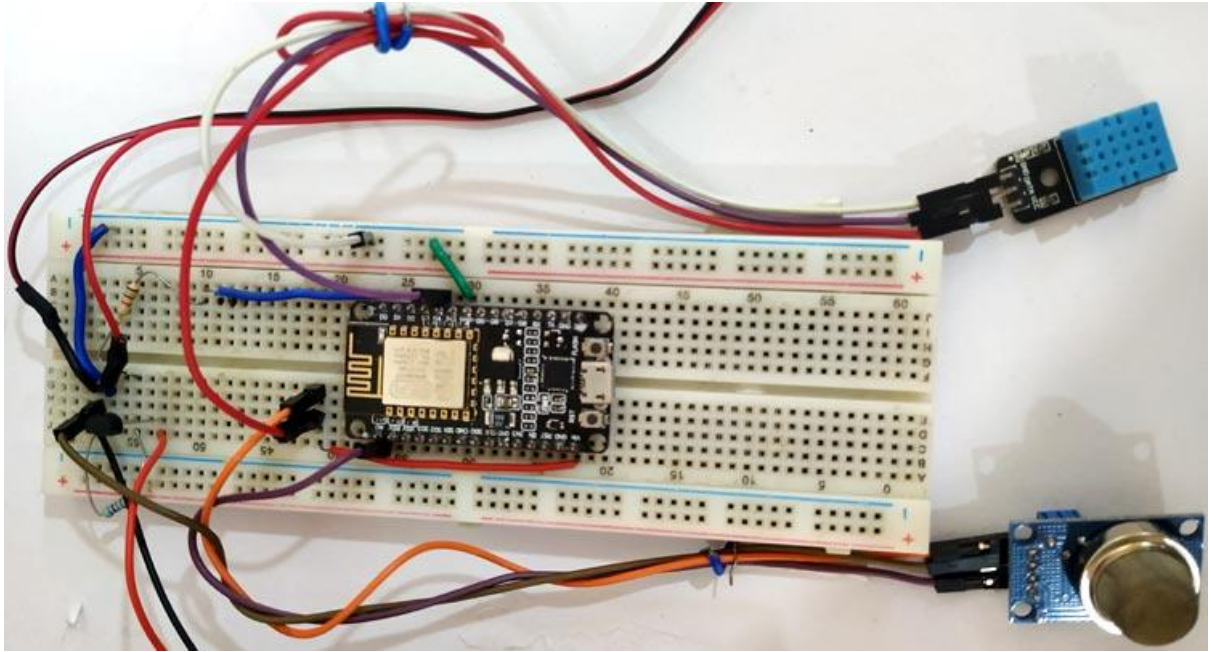
- NodeMCU ESP8266
- Modul Sensor MQ4
- Modul Sensor DHT11
- motor DC
- BC547-BJT
- Baterai
- Konektor
- RPS

Diagram Sirkuit

Diagram rangkaian lengkap untuk **proyek Pemantauan Makanan** ini diberikan pada gambar di bawah.



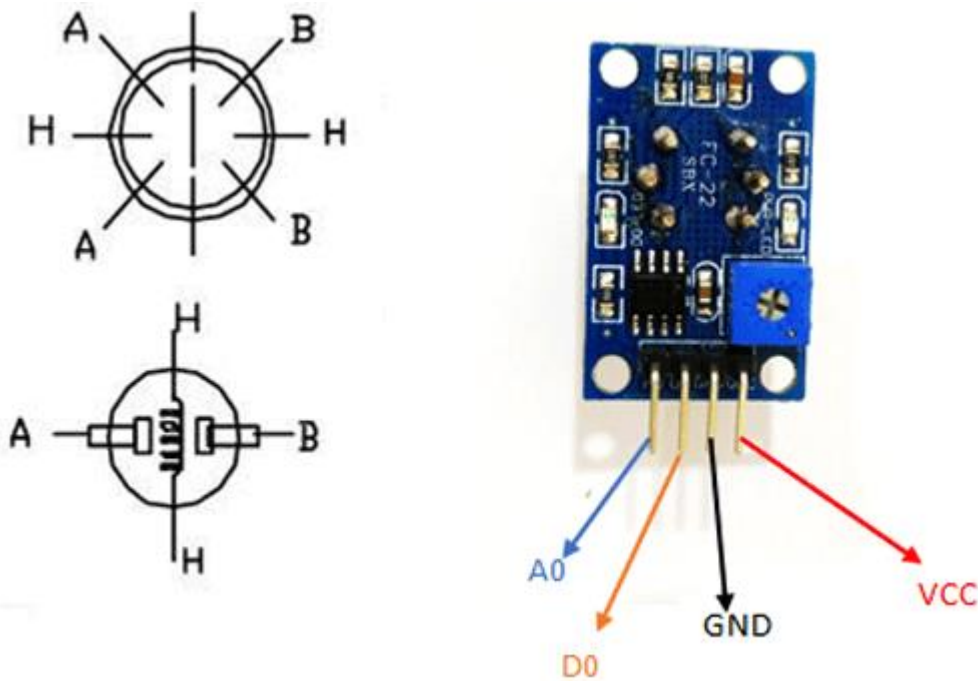
Tenaga ke motor diberikan oleh catu daya yang diatur. Terminal positif RPS dihubungkan ke terminal positif motor; terminal negatif motor dihubungkan ke terminal kolektor BJT. Terminal emitor BJT dibumikan, dan terminal basis BJT dihubungkan ke pin D0 MCU dengan resistor pembatas arus 1K. Terminal dasar BJT dihubungkan ke ground dengan resistor yang nilainya lebih besar dari resistor pembatas arus. Resistor ini bertindak sebagai resistor penekan ke BJT. VCC dan GND dari MCU terhubung ke satu sisi rel daya seperti yang ditunjukkan pada gambar di bawah. Terminal positif dan terminal GND dari kedua sensor dihubungkan ke power rail VCC dan GND seperti yang ditunjukkan pada gambar di atas.



Dua sensor penting utama di sini adalah Sensor Gas MQ-4 dan [Sensor Suhu dan Kelembapan DHT11](#).

Modul Sensor Gas MQ4

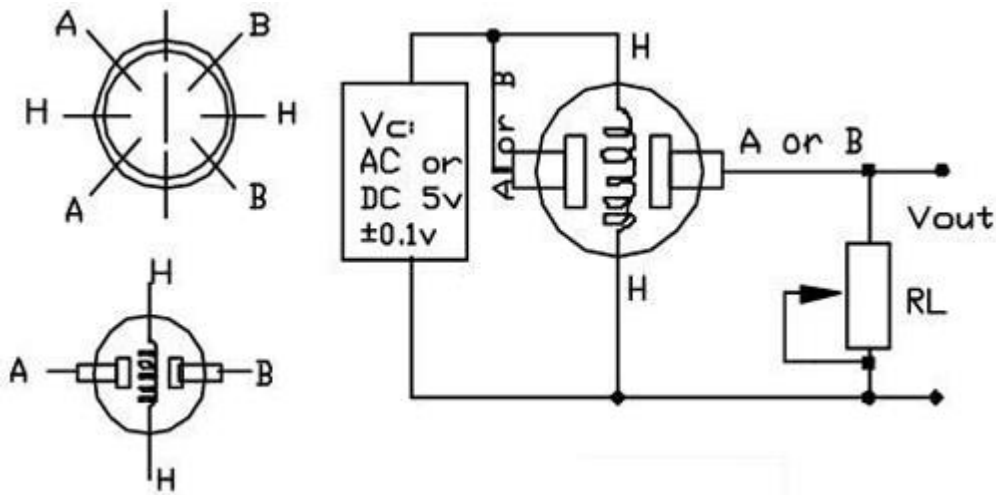
MQ4 adalah modul penginderaan gas yang digunakan untuk mengukur gas metana di atmosfer. Ini berisi lapisan penginderaan Gas, yang terdiri dari SnO_2 . SnO_2 sensitif terhadap gas seperti LPG, CH_4 , H_2 , CO, Alkohol, dan asap. Karena **makanan yang membusuk mengeluarkan gas metana (CH_4)**, sensor MQ4 dapat digunakan untuk mengukur gas ini guna **memantau kualitas makanan**. Anda juga dapat memeriksa proyek [antarmuka sensor MQ135](#) ini di mana kami menggunakan sensor gas serupa untuk **memantau kualitas udara dengan mengukur PPM**.



Selain SnO_2 , sensornya juga terdiri dari tabung keramik Al_2O_3 , elektroda pengukur, dan elemen pemanas. Elemen pemanas menyediakan kondisi kerja yang diperlukan agar sensor dapat beroperasi. Sensor MQ4 tersedia dalam dua format di pasaran, dalam format modul atau format sensor saja. Modul sensor memiliki 4 pin di mana kita hanya akan menggunakan 3 pin dalam proyek kita. Mereka adalah VCC, GND, dan A0. Kami meninggalkan pin D0, karena tidak berguna dalam **perhitungan ppm**. Cara **kerja sensor MQ4** mirip dengan LDR (light dependen resistor). Ketika konsentrasi gas metana tinggi, resistansi modul menurun, dan ketika konsentrasi rendah, resistansi meningkat.

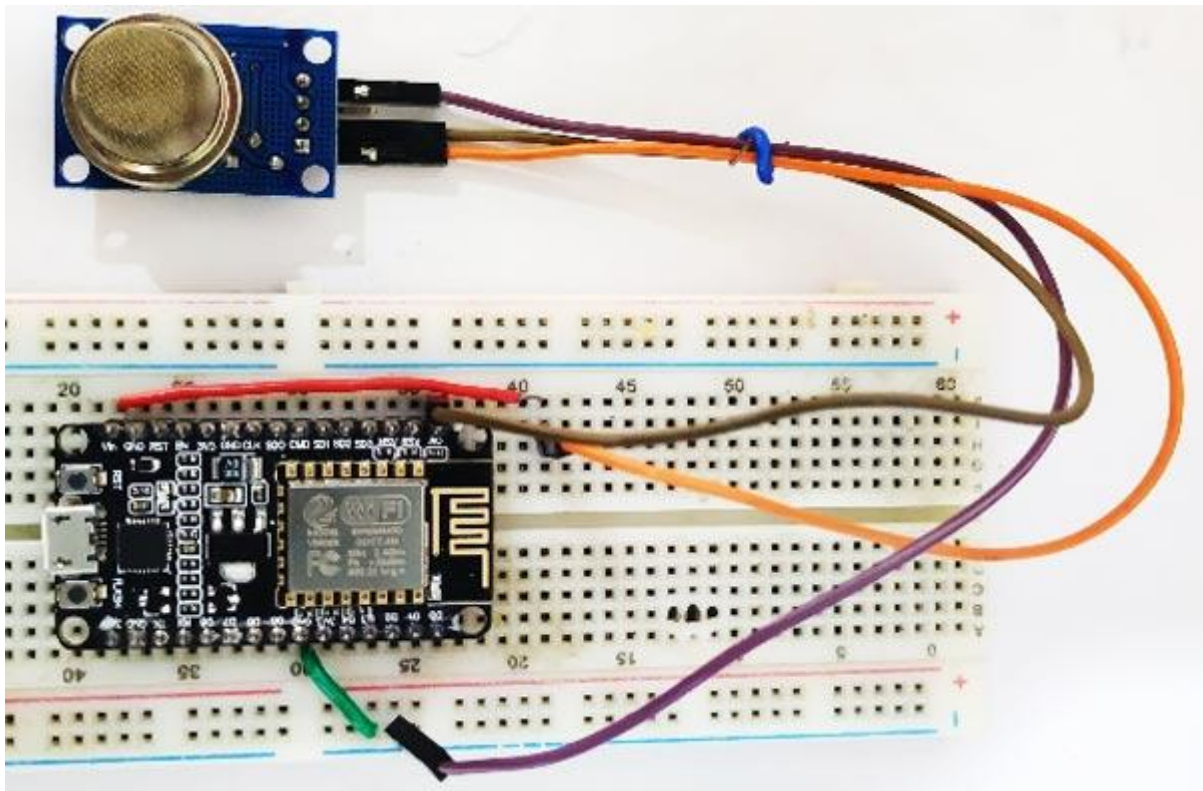
Sirkuit Internal Sensor Gas MQ4

Kedua terminal A mengalami korsleting, dan kedua terminal B mengalami korsleting, sehingga hanya menyisakan 4 sambungan untuk dihubungkan dalam rangkaian. Terminal H digunakan untuk menghubungkan tegangan suplai ke elemen pemanas, yang terbuat dari paduan Ni-Cr. Ini digunakan untuk menyediakan kondisi kerja yang diperlukan untuk komponen sensitif. Tegangan suplai diberikan ke salah satu terminal H bersama dengan terminal A atau B. RL adalah tahanan beban yang harus kita tambahkan ke sensor, seperti terlihat pada gambar. Jika Anda memiliki modul sensor, periksa jalur PCB dan temukan nilai RL dan ukur nilainya menggunakan multimeter.



Menghubungkan Sensor MQ4 dengan NodeMCU ESP8266

Sebelum kita masuk ke proyek utama, kita harus **melakukan kalibrasi pada sensor MQ4** agar kita dapat **mengukur nilai ppm dari sensor gas tersebut**. Hubungkan sensor gas Anda dengan NodeMCU seperti yang ditunjukkan pada gambar di bawah.

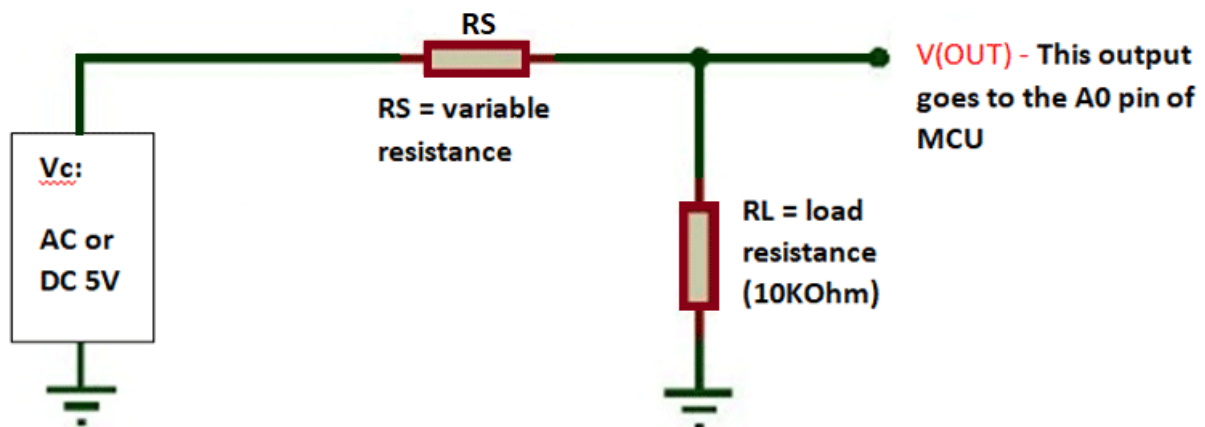


Pin VCC dan GND Modul Sensor terhubung ke Vin NodeMCU. Pin A0 modul sensor dihubungkan dengan pin A0 Mikrokontroler.

Kalibrasi Sensor Gas MQ4:

Kita perlu mencari rasio (R_S/R_0) untuk menentukan kandungan metana yang ada di udara. R_0 dan R_S adalah nilai resistansi internal udara segar.

Kita akan mencari nilai R_S dari V_{RL} (**tegangan pada resistor R_L**). Dengan menggunakan mikrokontroler kita dapat mengetahui nilai tegangan (V_{RL}) pada R_L (hambatan beban). Untuk mencari R_S dari V_{RL} , kita perlu menurunkan rumusnya. Rangkaian ekivalen pada **gambar 1** ditunjukkan di bawah ini, dan rangkaian ekivalen ini digunakan untuk menurunkan rumus yang digunakan untuk mencari nilai R_S dari V_{RL} .



Tegangan pada beban R_L adalah $V_L = I * R_L$, menggantikan I , kita mendapatkan $V_L = (V * R_L) / (R_S + R_L)$. Cukup persamaan untuk mendapatkan R_S .

Dari rangkaian ekivalen tersebut kita peroleh persamaannya

$$R_S = ((V - V_L) / V_L) * R_L$$

Dimana V adalah tegangan suplai.

Hitung Nilai R_0 di Udara Segar:

Sekarang, kita akan mencari nilai R_0 menggunakan nilai R_S dan grafik yang ada pada lembar data di bawah ini.

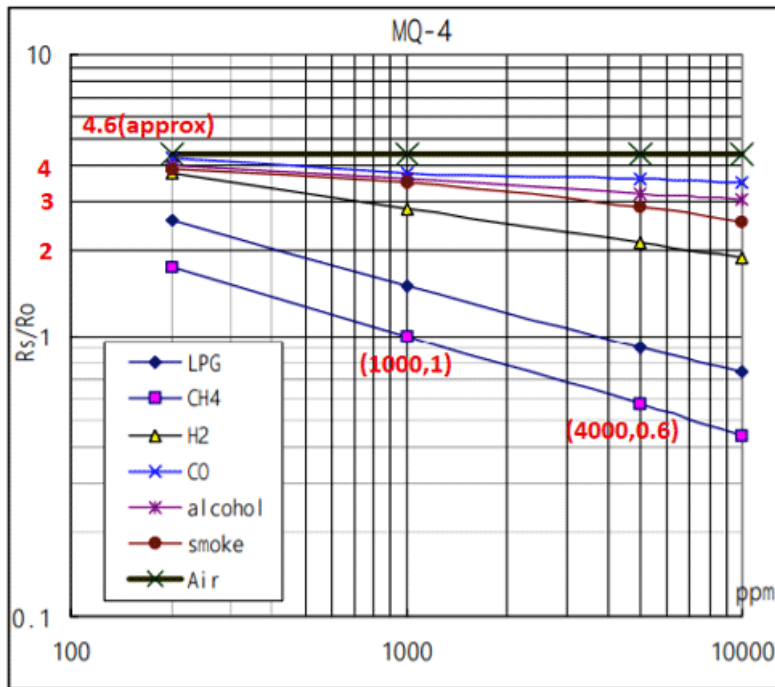


Fig.3 is shows the typical sensitivity characteristics of the MQ-4 for several gases.

in their: Temp: 20°C、

Humidity: 65%、

O₂ concentration 21%

RL=20k Ω

Ro: sensor resistance at 1000ppm of CH₄ in the clean air.

Rs:sensor resistance at various concentrations of gases.

Dengan melihat grafik tersebut kita dapat mengetahui bahwa sumbu X adalah ppm, dan sumbu Y adalah RS/RO. Telusuri nilai RO/RS udara segar. Dalam kasus kami, nilainya sekitar 4,6. Anda memiliki nilai RS (yang sebelumnya ditemukan menggunakan VRL), dan sekarang Anda dapat menemukan RO menggunakan relasi $RO/RS = 4.6$. Dalam kasus saya, saya mendapatkan nilai RO sebagai 1,9.

Hitung Metana (CH₄) dalam nilai PPM:

Setelah kita mengetahui nilai RO, kita perlu menurunkan persamaan yang dengannya kita dapat mencari ppm menggunakan RO yang diketahui dan **nilai RS (nilai yang dibaca oleh MCU)**. Untuk mendapatkan rumusnya, pertama-tama periksa grafik karakteristik sensitif yang ada di lembar data. Pilih kurva gas yang dibutuhkan dan ukur kemiringannya.

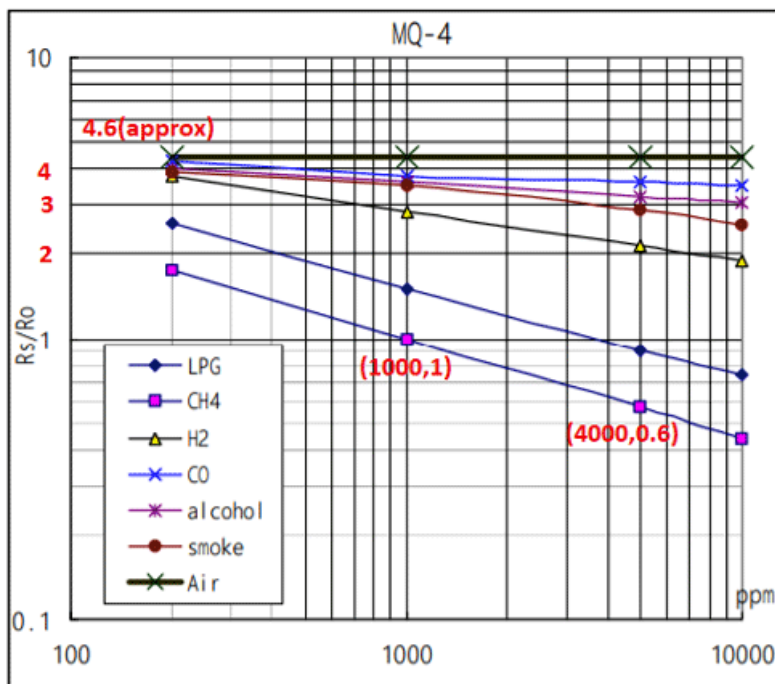


Fig.3 is shows the typical sensitivity characteristics of the MQ-4 for several gases.

in their: Temp: 20°C、

Humidity: 65%、

O₂ concentration 21%

RL=20k Ω

Ro: sensor resistance at 1000ppm of CH₄ in the clean air.

Rs:sensor resistance at various concentrations of gases.

Gunakan rumus $m = (y_2 - y_1) / (x_2 - x_1)$, untuk menghitung nilai kemiringan.

$$m = (\log(0,6) - \log(1)) / (\log(4000) - \log(1000))$$
$$m = -0,36848$$

Sekarang cari **nilai konstanta 'c'** menggunakan persamaan $y = mx + c$.

Ambil titik nilai yang diketahui untuk memudahkan penghitungan. $X = \log(1000)$, $y = (\log 1)$;

$$\log(1) = -0,36848 * \log(1000) + c. \text{ kita mendapatkan } c = 1,105$$

Sekarang cukup gunakan rumus $\log(y) = m * \log(x) + c$ untuk mendapatkan nilai x .

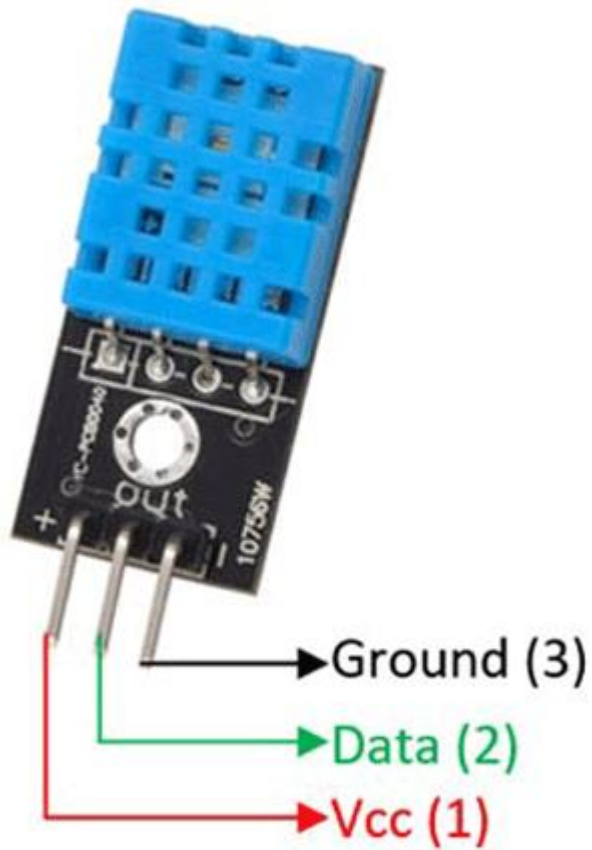
$$X = \text{pow}(10, ((\log_{10}(y) - c) / m));$$

Catatan: Y adalah rasio (RS/R0). R0 adalah resistansi yang diberikan oleh sensor, dan RS bergantung pada kandungan metana di atmosfer. X adalah kandungan gas yang diukur dalam nilai ppm.

Modul Sensor DHT11

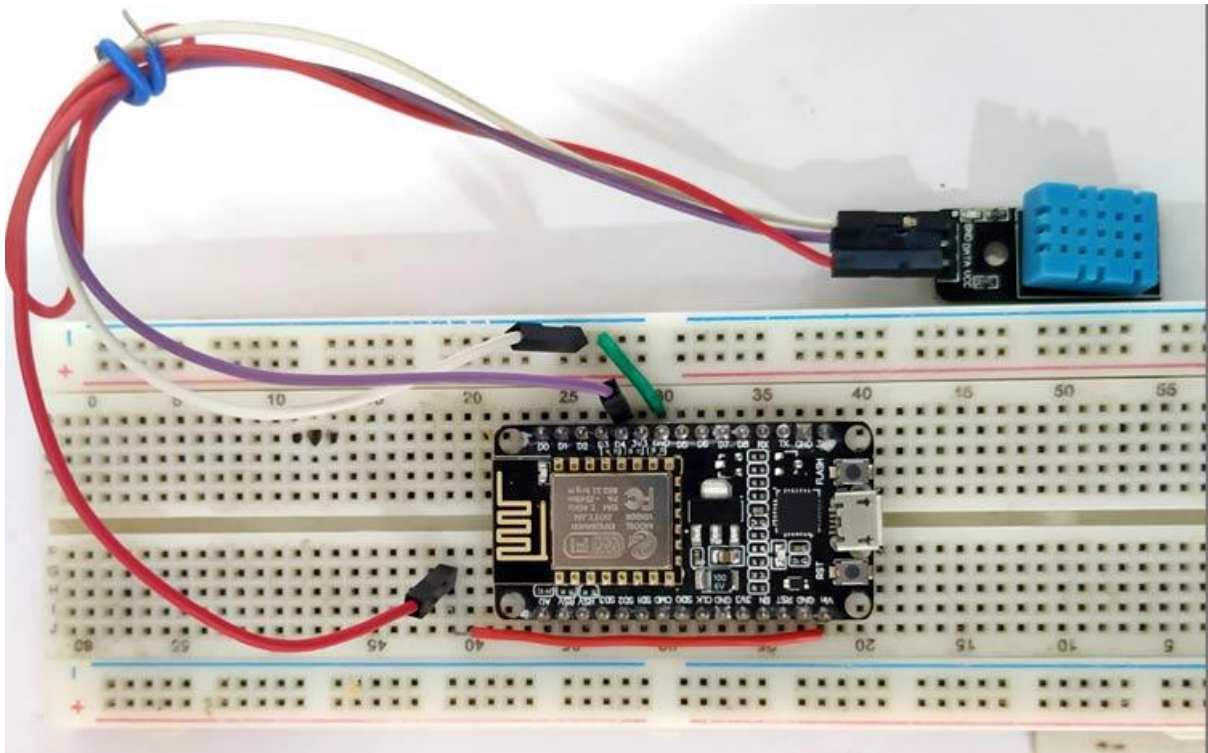
Modul [sensor DHT11](#) terdiri dari komponen pengukuran kelembaban tipe resistif dan komponen pengukuran suhu NTC serta mikrokontroler 8 bit. Ini memastikan kualitas, respons cepat, kemampuan anti-interferensi, dan efektivitas biaya. Modul sensor telah dikalibrasi sebelumnya di laboratorium sehingga pengguna akhir dapat langsung menggunakan sensor ini dalam proyek mereka. Data yang dikalibrasi disimpan dalam memori OTP, yang digunakan oleh proses pendeteksian sinyal internal sensor. Ini terdiri dari antarmuka serial kabel tunggal untuk mengirim data dari sensor ke mikrokontroler.

Sensor DHT11 tersedia sebagai sensor atau modul. Satu-satunya perbedaan di dalamnya adalah kita perlu menambahkan kapasitor penyaringan dan resistor pull-up ke sensor secara manual. Jika Anda memiliki modul sensor, tidak perlu menambahkan komponen tambahan karena modul sensor sudah memilikinya. Sensor dapat mengukur dari 0C hingga 50C dan kelembaban dari 20% hingga 90% dengan akurasi $\pm 1^\circ\text{C}$ dan $\pm 1\%$. Saat berkomunikasi antara sensor dan mikrokontroler, transmisi data lengkap adalah 40 bit. Sensor mengirimkan data yang lebih tinggi terlebih dahulu. Format data untuk pengiriman data adalah 8bit data RH integral + 8bit data RH + *bit T data + 8bit data T desimal + 8bit data checksum. Jika transmisi datanya benar, check-sumnya harus 8bit terakhir. Gambar di bawah menunjukkan pinout sensor DHT11. [Stasiun Cuaca Nirkabel Arduino](#), [pemantauan suhu dan kelembaban berbasis IoT](#), dll.



Menghubungkan DHT11 dengan NodeMCU ESP8266

Karena sensor DHT11 telah dikalibrasi sebelumnya di laboratorium, kita tidak perlu melakukan kalibrasi apa pun untuk sensor ini. Hubungkan sensor DHT11 Anda dengan NodeMCU seperti gambar di bawah ini.



Pin VCC, GND, dan Data dihubungkan ke pin Vin, GND, dan D4 mikrokontroler. Dengan menggunakan library Adafruit untuk DHT11, kita memanggil fungsi khusus, yang akan mengambil nilai suhu dan kelembaban. Format untuk menggunakan fungsi ini adalah `dht.readHumidity()` dan `dht.readTemperature`. Fungsi-fungsi ini mengembalikan nilai kelembapan dan suhu. Kami mendeklarasikan variabel `h` dan `t` sebagai float dan menggunakannya untuk menyimpan nilai suhu dan kelembapan.

Menggunakan ThingSpeak untuk Memantau Kualitas Makanan

Untuk mengirim data ke cloud dan menampilkannya di web, kita akan menggunakan ThingSpeak.

Ikuti langkah-langkah di bawah ini, untuk memulai, ThingSpeak

Langkah 1: Mendaftar ke ThingSpeak

Untuk membuat saluran di ThingSpeak, Anda harus mendaftar di ThingSpeak terlebih dahulu. Jika Anda sudah memiliki akun di ThingSpeak, cukup masuk menggunakan id dan kata sandi Anda.

Untuk membuat akun Anda, kunjungi www.thingspeak.com

Sign up for ThingSpeak

It is free to sign up for ThingSpeak. Free accounts offer a fully functional experience on ThingSpeak with limits on certain functionality. Commercial users may sign up for a time-limited free evaluation. To send data faster to ThingSpeak or to send more data, consider our [paid license options](#) for commercial, academic, home and student usage. To start using ThingSpeak you must create a new MathWorks account, or, click cancel and log in using an existing MathWorks account.

Create MathWorks Account

Email Address

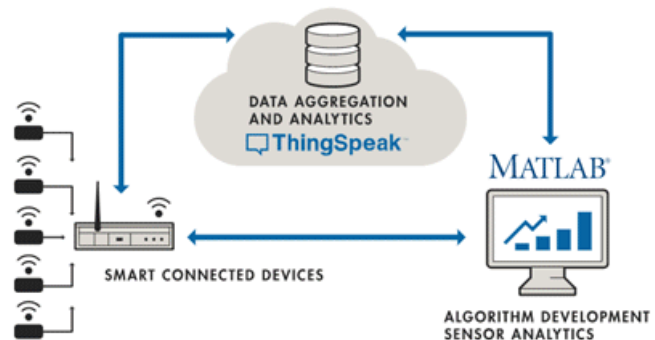
Missing required information

To access your organization's MATLAB license, use your school or work email.

Location

First Name

Last Name



Verifikasi ID email Anda dan lanjutkan

Langkah-2: Buat Channel Baru Anda

Untuk membuat channel baru Anda. Pilih channels>new channel.

New Channel

Name

Food monitoring

Description

Monitoring Food

Field 1

Temperature

Field 2

Humidity

Field 3

Methane content %

Field 4

Field 5

Field 6

Field 7

Field 8

Metadata

Tags

Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

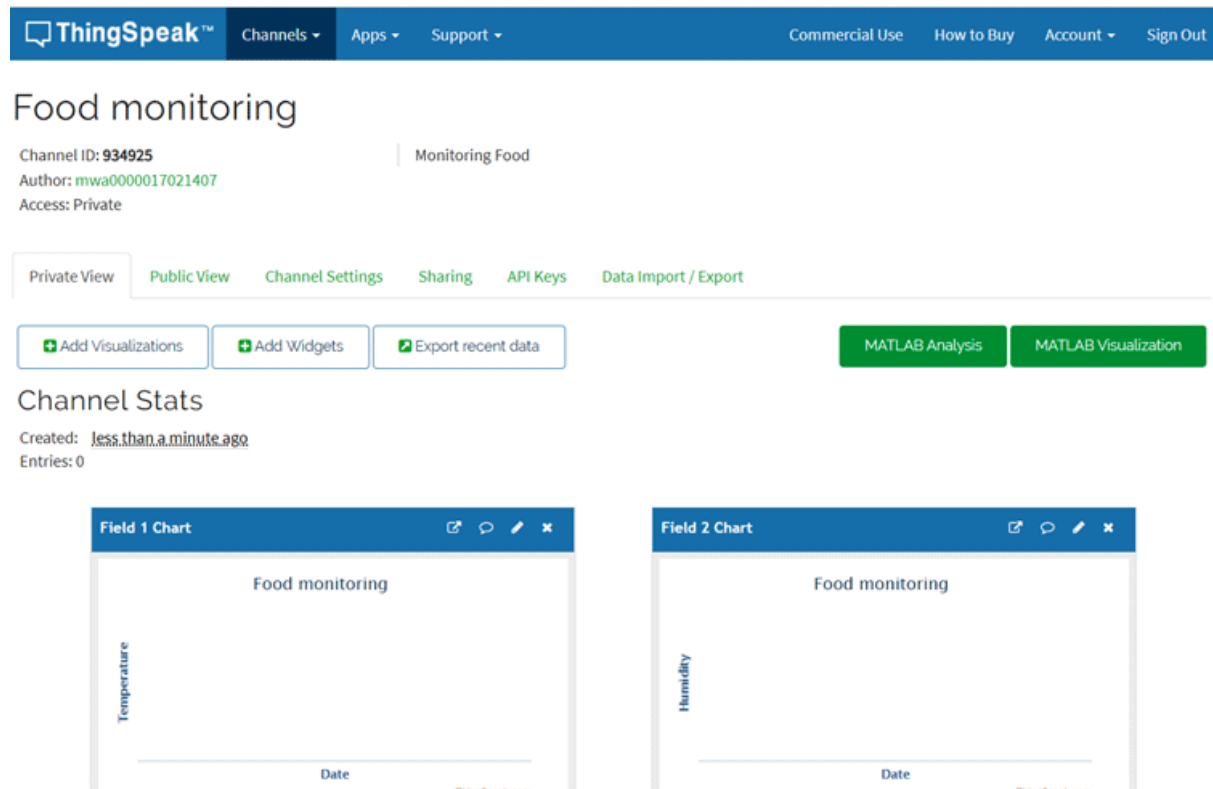
Channel Settings

- Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- Channel Name:** Enter a unique name for the ThingSpeak channel.
- Description:** Enter a description of the ThingSpeak channel.
- Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- Tags:** Enter keywords that identify the channel. Separate tags with commas.
- Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- Show Channel Location:**
 - Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
 - Longitude:** Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
 - Elevation:** Specify the elevation position meters. For example, the elevation of the city of London is 35.052.

Masukkan detailnya dan pilih **save channel**

Langkah-3:- Dapatkan kunci API Anda

Setelah membuat, buka saluran Anda dan pilih opsi API Keys



Catat API Keys yang Anda dapatkan . Kunci unik ini digunakan untuk mengirim data dari mikrokontroler ke web.

Private View Public View Channel Settings Sharing **API Keys**

Write API Key


Key

2BT21TRH1ULOMWPI

Generate New Write API Key

Kirim Email menggunakan IFTTT

Untuk memicu email, pertama-tama kita perlu mendaftar ke situs IFTTT <https://ifttt.com/> .

IFTTT  **Search**

Sign in

[Forgot your password?](#)

Sign in

[Continue with Google or Facebook](#)

Pilih tombol pencarian dan cari Webhook dan pilih Dokumentasi.

Explore

Connections

Services



Webhooks

< Back

Documentation

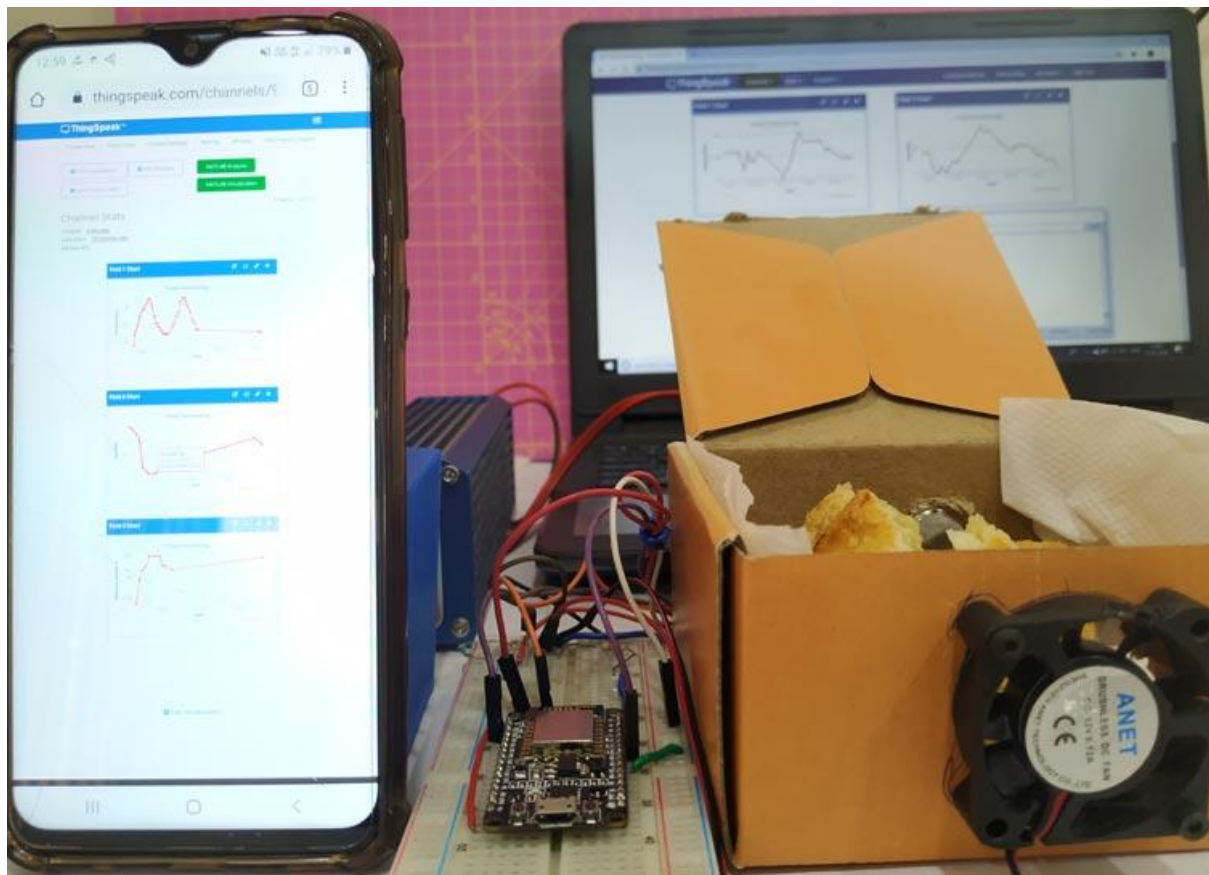
⚙ Settings



Webhooks

Integrate other services on IFTTT with your DIY projects. You can create Applets that work with any device or app that can make or receive a web request. If you'd like to build your own service and Applets, [check out the IFTTT platform](#).

Salin private key. Dengan menggunakan private key ini, Anda dapat memicu peringatan email.



Kode Program

```
#include <DHT.h>

#include <ESP8266WiFi.h>

#include <WiFiClient.h>;

#include <ThingSpeak.h>;

#define RL 10.0 // the value of RL is 10K

#define m -0.36848 // using formula y = mx+c and the graph in the datasheet

#define c 1.105 // btained by before calculation

#define R0 1.9

#define fan D0 // pin for fan

// replace with your channel's thingspeak API key,
const char * myWriteAPIKey = "YOUR_CHANNEL_API_CODE";

unsigned long myChannelNumber = YOUR_CHANNEL_ID; //Replace it with your channel ID

const char* ssid = "YOUR_SSID_POWER";
```

```
const char* password = "SSID_PASSWORD";
const char* server = "api.thingspeak.com";
const char *host = "maker.ifttt.com";
const char *privateKey = "YOUR_PRIVATE_KEY_FROM_IFTTT";
double gas;

#define Gas_Pin A0

#define DHTPIN D4 // CONNECT THE DHT11 SENSOR TO PIN D4 OF THE NODEMCU
DHT dht(DHTPIN, DHT11,15); //CHANGE DHT11 TO DHT22 IF YOU ARE USING DHT22
WiFiClient client;

void setup() {
  Serial.begin(9600);
  pinMode(fan,OUTPUT);
  delay(10);
  dht.begin();
  ThingSpeak.begin(client);
  WiFi.begin(ssid, password);
  pinMode(Gas_Pin,INPUT);
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  while (WiFi.status() != WL_CONNECTED) {
    delay(200);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
}
```

```

int ppm1() {
    float sensor_volt; //Define variable for sensor voltage
    float RS_gas; //Define variable for sensor resistance
    float ratio; //Define variable for ratio
    float sensorValue = analogRead(Gas_Pin); //Read analog values of sensor
    sensor_volt = sensorValue*(5.0/1023.0); //Convert analog values to voltage
    RS_gas = ((5.0*10.0)/sensor_volt)-10.0; //Get value of RS in a gas
    ratio = RS_gas/R0; // Get ratio RS_gas/RS_air
    double ppm_log = (log10(ratio)-c)/m; //Get ppm value in linear scale according to the the ratio
    value
    float ppm = pow(10, ppm_log); //Convert ppm value to log scale
    Serial.println(ppm);
    ThingSpeak.writeField(myChannelNumber, 3,ppm, myWriteAPIKey);
    delay(200);
    return(ppm);
}

void loop() {
    float t = dht.readTemperature();
    delay(200);
    ThingSpeak.writeField(myChannelNumber, 1,t, myWriteAPIKey);
    Serial.print("Temp = ");
    Serial.println(t);
    float h = dht.readHumidity();
    delay(2000);
    ThingSpeak.writeField(myChannelNumber, 2,h, myWriteAPIKey);
    Serial.print("Humidity = ");
    Serial.println(h);
    gas = ppm1();
    delay(2000);
    if (t > 22){
        digitalWrite(D0,HIGH);
    }
}

```

```

    send_event("temp_event");
    Serial.println("Fan On");
}
else{
    digitalWrite(fan,LOW);
}
if (isnan(h) || isnan(t) || isnan(gas)) {
    Serial.println("Failed to read from DHT sensor!");
}
}

```

```

void send_event(const char *event)
{
    Serial.print("Connecting to ");
    Serial.println(host);
    // Use WiFiClient class to create TCP connections
    WiFiClient client;
    const int httpPort = 80;
    if (!client.connect(host, httpPort)) {
        Serial.println("Connection failed");
        return;
    }
    // We now create a URI for the request
    String url = "/trigger/";
    url += event;
    url += "/with/key/";
    url += privateKey;
    Serial.print("Requesting URL: ");
    Serial.println(url);
    // This will send the request to the server
    client.print(String("GET ") + url + " HTTP/1.1\r\n" +

```



```
        "Host: " + host + "\r\n" +  
        "Connection: close\r\n\r\n");  
while(client.connected())  
{  
    if(client.available())  
    {  
        String line = client.readStringUntil('\r');  
        Serial.print(line);  
    } else {  
        // No data yet, wait a bit  
        delay(50);  
    };  
}  
Serial.println();  
Serial.println("closing connection");  
client.stop();  
}
```