BAB 5 PERANCANGAN DAN IMPLEMENTASI

5.1 Perancangan Perangkat Keras

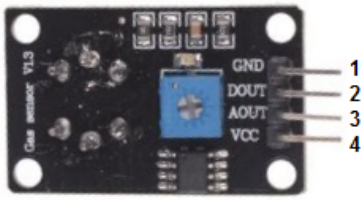
Secara umum, perancangan perangkat keras pada Sistem Monitoring Gas Berbahaya berdasarkan Amonia dan Metana pada Peternakan Ayam Broiler Menggunakan Protokol MQTT pada Real Time System ini dibagi menjadi 4 bagian, yakni bagian sensor, Arduino, ESP8266, dan MQTT yang outputnya ditampilkan dalam web server. Untuk lebih jelasnya dapat dilihat pada Gambar 5.1 berikut.

Gambar 5.1 Bagan Perancangan Umum Sistem

Nantinya, data pada sensor MQ-4 dan MQ-135 akan didapatkan dan dikirimkan ke Arduino melalui koneksi pin. Dari Arduino, data akan diolah dan diteruskan menuju modul wifi ESP8266. Dari ESP8266 inilah kemudian data akan dikirimkan secara *wireless* menuju ke web server berupa thingsboard.io dengan protokol MQTT.

5.1.1 Perancangan Koneksi Sensor ke Arduino

Pada bagian input dalam sistem, penulis menggunakan dua sensor berupa MQ-135 yang dapat mendeteksi gas amonia dan MQ-4 yang dapat mendeteksi gas metana. Kedua sensor akan berfungsi sebagai perasa dalam pengujian sistem. Dari kedua sensor tersebut akan didapatkan data mentah yang kemudian akan dikirimkan ke Arduino sebagai bagian proses dari sistem yang dibuat penulis melalui koneksi pada pin dalam Arduino. Secara keseluruhan, perancangan bagian input dapat dilihat pada Gambar 5.2 berikut.



1 = GND
2 = DOUT
3 = AOUT
4 = Vcc

(bottom view)

Gambar 5.2 Perancangan Pin pada Sensor MQ-4 menuju ke Arduino

Sebelum sensor dirangkai, terlebih dahulu akan dilakukan kalibrasi sensor untuk mengetahui tingkat keakuratan sensor dan membandingkan dengan akurasi sensor yang sudah ada. Namun dalam penelitian ini, kalibrasi sensor dilakukan dengan cara sebatas memberi rangkaian pada instalasi sensor menuju ke Arduino. Sebab, karena nilai output pada kedua sensor berupa resistansi (Ohm) dan nilai input pada Arduino berupa tegangan (Volt) sehingga nilai output dari sensor harus diubah menjadi bentuk tegangan agar dapat dibaca dengan baik oleh Arduino.

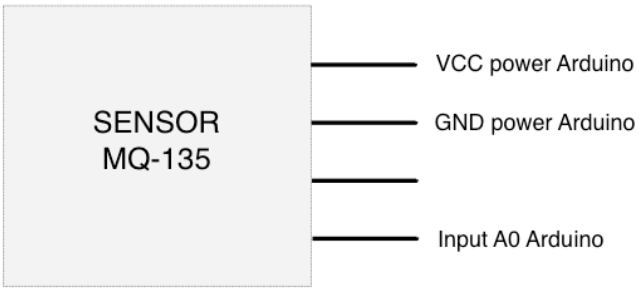
Gambar 5.3 Diagram Koneksi Sensor MQ-4 ke Arduino

Pada sensor MQ-4 terkoneksi dengan Arduino melalui pin A1 Arduino yang tersambung dengan pin A0 pada sensor seperti yang ditunjukkan pada Gambar 5.3. Sensor MQ-4 akan mendeteksi gas metana di sekitar dengan ambang batas kadar 300-10000 ppm. Data dari sensor MQ-4 adalah berupa data analog yang kemudian akan diteruskan ke Arduino sebagai bagian *processing unit* dari sistem. Untuk lebih jelasnya dapat dilihat dalam Tabel 5.1 berikut

Tabel 5-1 Perancangan Sensor MQ-4

Kaki	Koneksi
VCC	ICSP Arduino
GND	ICSP Arduino
AOUT	Input A1 Arduino
DOUT	-

Sama halnya dengan sensor MQ-4, pada sensor MQ-135 juga akan terkoneksi melalui Arduino, tetapi pada pin A0 yang tersambung dengan pin A0 pada sensor seperti yang ditunjukkan Gambar 5.4 di atas. Sensor MQ-135 mendeteksi gas amonia dengan ambang batas kadar 100-1000 ppm. Dari sensor MQ-135 data akan dikirimkan melalui koneksi pin pula ke Arduino.



Gambar 5.4 Diagram Koneksi Pin dari Sensor MQ-135 ke Arduino

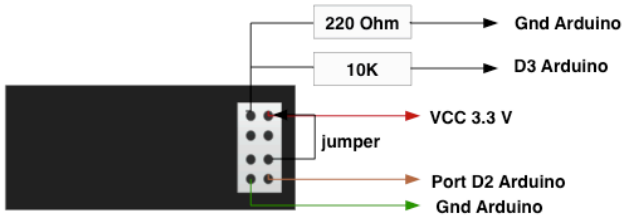
Tabel 5-2 Perancangan Sensor MQ-135

Kaki	Koneksi
VCC	VCC Power Arduino
GND	GND Arduino
AOUT	Input A0 Arduino
DOUT	-

Dalam perancangannya, antara pin output pada sensor dan pin input pada Arduino dihubungkan dengan kabel *jumper*. Hal ini bertujuan untuk efisiensi pada sistem sehingga sistem dapat berjalan dengan baik. Penggunaan kabel dimaksudkan agar sensor dapat mengirimkan data lebih cepat sehingga dengan kondisi sistem yang berjalan pada keadaan *real time*, sistem dapat mengetahui akurasi data secara pasti.

5.1.2 Perancangan Koneksi Modul Wifi ESP8266 ke Arduino

Dalam bagian pengiriman data akan dilakukan oleh modul wifi ESP8266 yang terkoneksi dengan *access point* di sekitarnya dan akan mengirimkan data dari Arduino yang dikirimkan dari koneksi pin dari pin D2 Arduino dengan format json, ke web server. Proses pengiriman data ini menggunakan protokol MQTT dengan QoS 0 hingga 1. Dari sini, data mentah berupa hasil proses dari Arduino akan dikirimkan menuju thingsboard secara terus menerus agar dapat tampil secara *real time* pada output. Untuk perancangan modul wifi ESP8266 lebih jelasnya ditunjukkan pada Gambar 5.5 berikut.



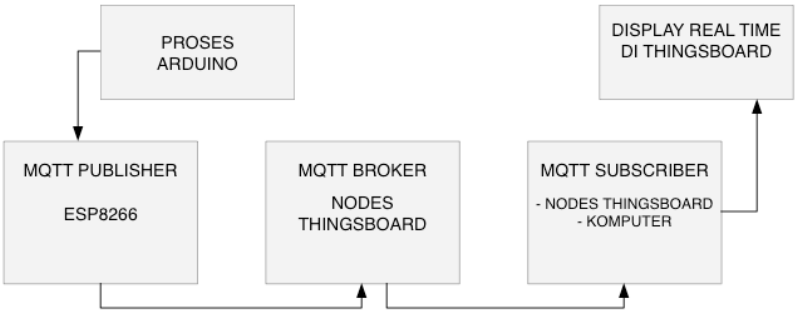
Gambar 5.5 Bagan Perancangan Modul Wifi ESP8266

Modul wifi ESP8266 dikoneksikan dengan sinyal wifi yang tersedia di sekitar lokasi sistem dengan terlebih dahulu mengatur nama dan kata kunci SSID pada perangkat lunaknya yang akan dibahas di subbab selanjutnya. Kemudian, data berupa json akan dikirimkan secara *wireless* menuju web server dengan protokol MQTT. *Nodes* pada thingsboard berfungsi menjadi MQTT broker, sedangkan modul wifi ESP8266 sebagai MQTT publisher yang akan mengirimkan data secara terus menerus dalam kondisi *real time*.

Sebagai pengiriman data digunakan fungsi *payload* untuk menyimpan data json dari sensor MQ-135 dan MQ-4 yang telah diolah di dalam Arduino. Data tersebut akan diambil oleh modul wifi ESP8266 untuk di-*publish* menuju web server.

5.1.3 Perancangan Pengiriman Data Menggunakan MQTT

Pada perancangan pengiriman data menggunakan MQTT dalam sistem monitoring ini, data akan dikirimkan ke web server berupa server dalam Thingsboard yang dapat diakses di *http://thingsboard.io*. Di bagian output ini, data dari Arduino yang dikirimkan oleh modul wifi ESP8266 secara *wireless* ke web server pada Thingsboard akan berperan sebagai MQTT Publisher, yang ditampilkan secara *real time*. Data yang ditampilkan berbentuk grafik dan chart yang akan memudahkan pengguna (dalam hal ini adalah peternak ayam broiler) untuk mengetahui kondisi kadar gas amonia dan metana pada kandang. Pada Gambar 5.6 digambarkan perancangan MQTT Publisher, MQTT Broker, dan MQTT Subscriber dalam sistem yang dibuat penulis,



Gambar 5.6 Perancangan MQTT

Pada perancangan MQTT di Thingsboard, *nodes* pada Thingsboard server berperan sebagai MQTT Broker yang menerapkan QoS MQTT level 0 (paling banyak satu data dapat terkirim) dan level 1 (paling tidak satu data dapat terkirim), sekaligus sebagai MQTT Subscriber. Kemudian, sistem akan dapat berkomunikasi dengan web server pada thingsboard dengan adanya token akses. Oleh karena itu, pada perancangan perangkat lunak akan diinisialisasikan komunikasi antara perangkat keras dengan web server dengan perintah $ACCESS_TOKEN. Inisialisasi ini untuk mengirimkan pesan MQTT CONNECT dengan pengguna yang menggunakan token yang sama. Untuk inisialisasi dan komunikasi antara thingsboard dengan perangkat akan dijelaskan dalam subbab selanjutnya.

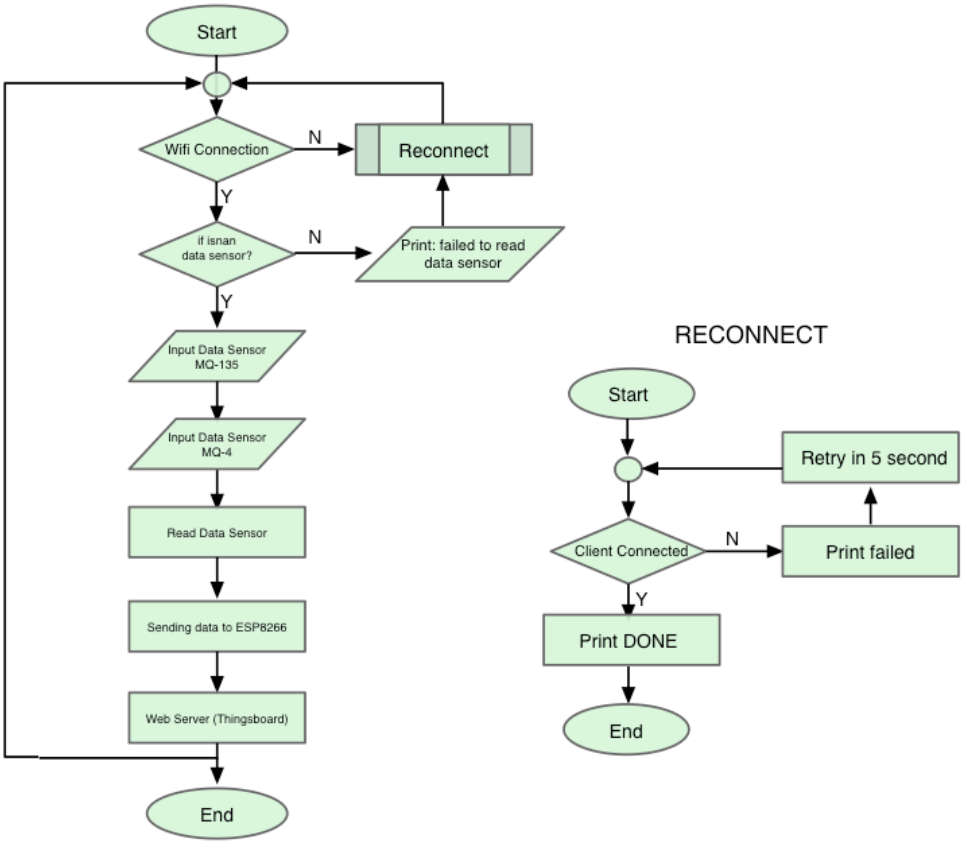
MQTT Publisher, dalam sistem ini adalah sensor, Arduino, dan modul wifi ESP8266 akan *publish* data yang didapatkan dari sensor melalui fungsi *payload* yang dapat mengirimkan data berbentuk byte. Nantinya, MQTT Broker yang ada di dalam *nodes* thingsboard akan mengatur proses pengiriman data tersebut. *Nodes* thingsboard yang juga berperan sebagari MQTT Subscriber akan menampilkan data ke thingsboard dalam bentuk grafik dan chart.

5.2 Perancangan Perangkat Lunak

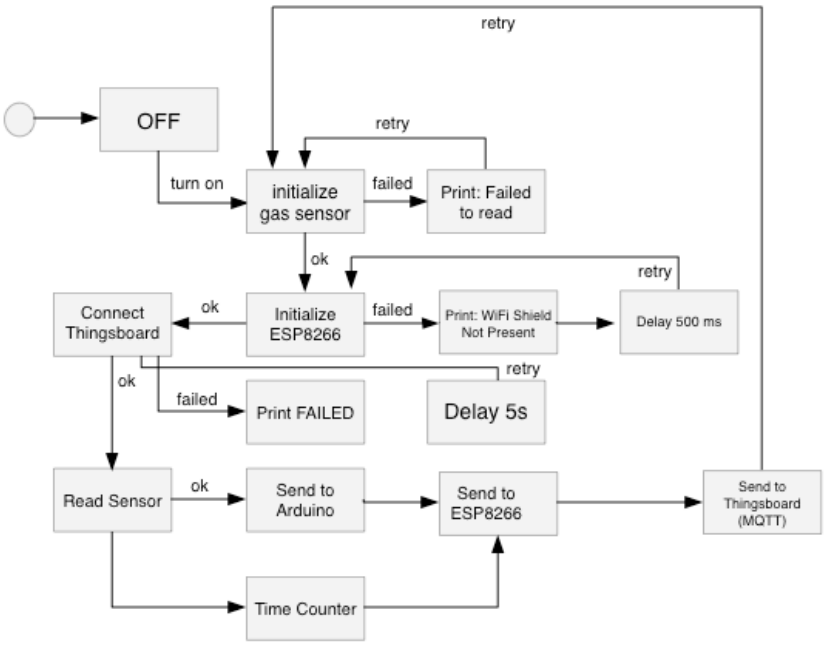
Perancangan perangkat lunak adalah perancangan kode program pada sistem yang dibuat penulis. Perancangan perangkat lunak terdiri dari beberapa bagian, antara lain bagaimana sistem berjalan, pembacaan data sensor, pengiriman data, MQTT Client, dan penghitungan waktu. Masing-masing bagian mempunyai fungsi tersendiri yang dapat memudahkan pengguna agar dapat memonitor kondisi gas berbahaya dalam peternakan ayam broiler secara *real time*.

5.2.1 Sistem *Running*

Dalam sistem yang dibuat penulis, sistem dikatakan berjalan apabila data pada sensor dapat dibaca dan dihasilkan keluaran berupa gambar dan grafik pada web server. Sensor menggunakan sensor MQ-135 sebagai sensor gas amonia dan sensor MQ-4 sebagai sensor gas metana, yang terhubung dengan Arduino. Dari Arduino, data sensor akan diolah dan dikirimkan ke modul wifi ESP8266. Dari modul wifi ESP8266 data akan dikirimkan dengan protokol MQTT menuju ke web server. Alur kerja sistem secara lebih jelas dapat dilihat di diagram alir pada Gambar 5.7 dan *state machine* pada Gambar 5.8 berikut.



Gambar 5.7 Flowchart Keseluruhan Sistem



Gambar 5.8 State Machine Diagram pada Sistem

5.2.2 Pembacaan Data pada Sensor

Dalam bagian pembacaan data sensor, terlebih dahulu harus diinisialisasikan melalui Arduino IDE. Proses yang terjadi adalah sensor MQ-135 dan MQ-4 mengirimkan data mentah ke Arduino lewat koneksi pin. Proses koneksi ini memerlukan inisialisasi, baik itu di sisi kedua sensor, web server sebagai output, modul wifi ESP8266 sebagai *ethernet client object*, dan output default pada kedua sensor.

Dalam proses inisialisasi pin untuk input Arduino dari kedua sensor gas, maka harus dideklarasikan inisialisasinya. Arduino akan mulai membaca data sensor dengan menginisialisasi masukan analog pada pin A0 dan A1 sebagai input dari sensor MQ-135 dan MQ-4. Pada bagian ini akan dijelaskan lebih lanjut dalam subbab selanjutnya.

5.2.3 Pengiriman Data

Dalam bagian pengiriman data akan dilakukan oleh modul wifi ESP8266 yang mengirimkan data dari Arduino yang dikirimkan dari koneksi pin melalui pin Tx yang terhubung dengan pin D2 Arduino, dengan format json ke web server. Proses pengiriman data menggunakan fungsi *payload* untuk membungkus paket yang dikirimkan ke web server. Fungsi *payload* dalam sistem yang dibuat penulis juga berguna untuk menyimpan data hasil pembacaan sensor agar dapat dikirim dalam bentuk *byte* ke dalam Thingsboard.

Proses pengiriman menggunakan topik GAS_MQ_4 dan GAS_MQ_135 agar dapat berjalan dengan cepat karena hanya membutuhkan paket data yang kecil, mengingat sistem berada dalam kondisi *realtime* yang akan mengirimkan data

secara terus menerus. Topik-topik ini disimpan dalam fungsi *payload* dan kemudian dikirimkan menuju ke Thingsboard.

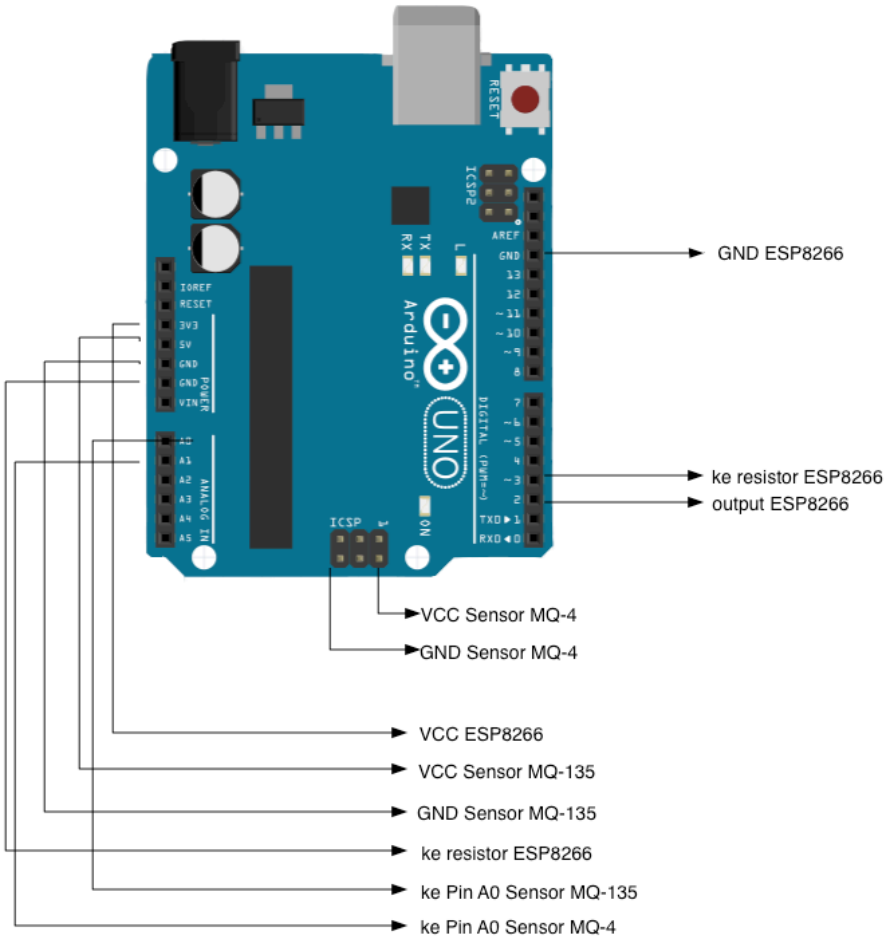
5.2.4 MQTT Client

Dalam bagian pengiriman data akan dilakukan oleh modul wifi ESP8266 yang terkoneksi dengan *access point* di sekitarnya dan akan mengirimkan data dari Arduino yang didapatkan dari pin Rx yang terhubung dengan pin D3 Arduino. Data tersebut kemudian dikirimkan dari koneksi pin melalui pin Tx yang terhubung dengan pin D2 Arduino, dengan format json, ke web server.

5.3 Implementasi Sistem

5.3.1 Implementasi Hardware

Pada bagian proses dalam sistem monitoring ini terdapat sebuah mikrokontroler berupa Arduino Uno dan sebuah modul wifi ESP8266. Arduino akan mengambil inputan data dari kedua sensor, baik MQ-135 maupun MQ-4, untuk kemudian diolah dan data dikirimkan lagi ke web server menggunakan ESP8266 dengan protokol MQTT. Sedangkan modul wifi ESP8266 akan mengirimkan data yang dikirimkan dari Arduino berupa data beformat json ke web server. Pada Gambar 5.9 dijabarkan bagaimana perancangan Arduino dalam sistem monitoring gas berbahaya yang dibuat penulis.

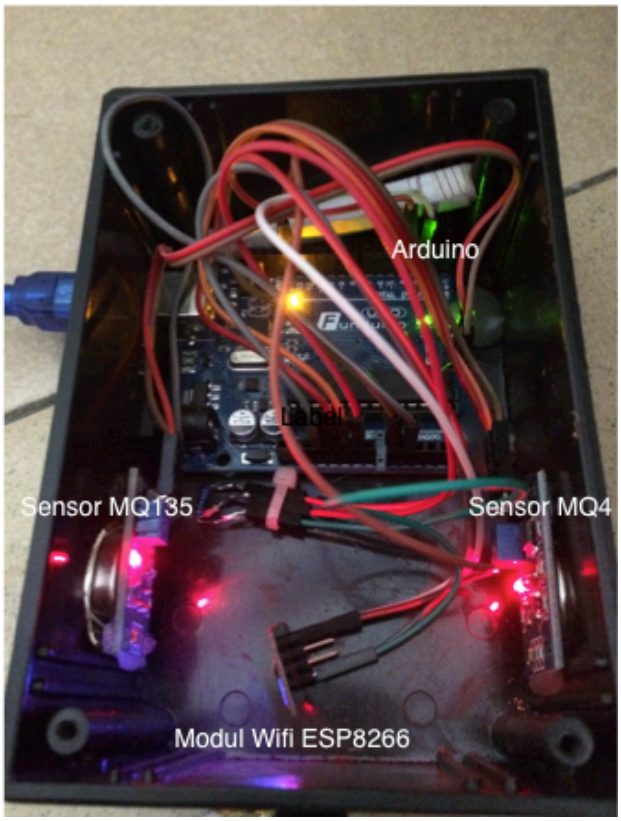


Gambar 5.9 Bagan Koneksi Pin Arduino pada Sistem

Arduino akan mengambil data dari sensor yang dikirimkan melalui pin A0 dan pin A1 pada *board* Arduino. Data dari sensor tersebut kemudian diolah untuk ditampilkan ke web server. Dalam pengolahan data sensor, Arduino hanya mengubah data analog dari sensor untuk kemudian dihasilkan data berformat .json yang kemudian dikirimkan ke web server berupa Thingsboard. Data berupa json yang dikirimkan Arduino diteruskan ke modul wifi ESP8266 melalui pin D3 yang terhubung dengan resistor 220 ohm dan 10Kohm yang dirangkai paralel, yang kemudian dikirimkan ke web server sehingga dapat diketahui berapa kadar gas di lokasi pengukuran secara *real time*.

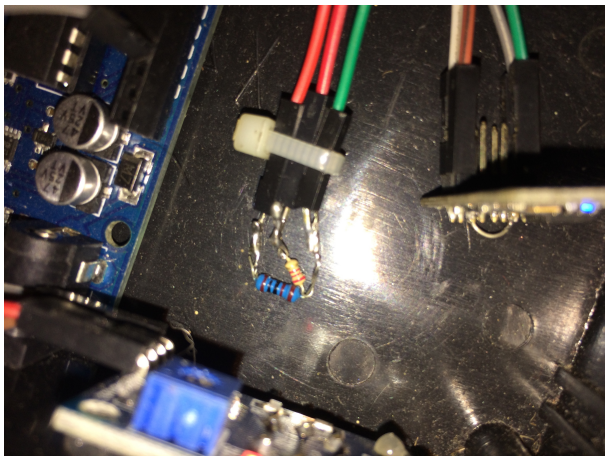
Dalam perancangan bagian proses pada sistem, karena sistem berjalan dalam kondisi *real time*, maka penulis menggunakan fungsi *time* yang berguna untuk mengatur waktu eksekusi proses. Dalam fungsi *time* pada sistem, penulis menggunakan *time interval* 1 detik. Ini berarti setiap proses dalam sistem monitoring yang dibuat penulis, interval waktunya 1 detik. Jika proses yang dilakukan lebih dari 1 detik, maka proses akan dianggap *failed* (gagal) dan harus kembali ke proses pengambilan data pada sensor. Tetapi, jika proses yang dilakukan total waktunya kurang dari atau sama dengan 1 detik, maka proses akan berlanjut ke proses selanjutnya.

Dalam perancangan bagian proses dan waktu interval, penulis menggunakan fungsi *micros* dalam Arduino. Fungsi *micros* adalah fungsi untuk menghitung dan mengatur waktu eksekusi proses dalam Arduino berdasarkan interval waktu yang telah dibuat. Penggunaan fungsi micros ini juga untuk mengkalkulasi waktu yang diperlukan sistem untuk mengeksekusi beberapa proses yang ada agar dapat dihitung waktu yang diperlukan saat implementasi sistem. Selain itu, dalam kondisi *real time*, fungsi micros juga meringankan kerja Arduino karena dapat melakukan proses lanjutan apabila proses sebelumnya belum selesai ditampilkan ke output sistem.



Gambar 5.10 Implementasi Hardware

Pada bagian perancangan ESP8266, terdapat dua resistor yang berfungsi sebagai resistor *pull down* untuk menstabilkan arus listrik dan menurunkan voltase dari data yang dikirimkan dari Arduino. Kedua resistor bernilai 220 Ohm dan 10 KOhm yang dirangkai secara paralel. Implementasi rangkaian resistor *pull down* dapat dilihat pada Gambar 5.11 berikut.



Gambar 5.11 Rangkaian Resistor Paralel 10K & 220 Ohm

Di dalam modul wifi ESP8266, data dari Arduino berupa data input sensor dengan format json dikirimkan ke web server berupa Thingsboard. Komunikasi dari Arduino ke ESP8266 menggunakan kabel dari pin D2 di Arduino menuju pin Tx di ESP8266. Penggunaan power berupa VCC dari Arduino sangat penting sebab sistem yang dibuat penulis akan bekerja dalam kondisi *real time* sehingga lebih efisien jika power dari ESP8266 berasal dari sumber yang sama dengan Arduino.

5.3.2 Implementasi Software

Sebelum program bagian pertama dieksekusi, terlebih dahulu inisialisasi program harus dilakukan. Pada Tabel 5-3 ditampilkan potongan kode program untuk inisialisasi dan penggunaan *library* pada sistem.

Tabel 5-3 Inisialisasi Library

```
#include <WiFiEspClient.h>
#include <WiFiEsp.h>
#include <WiFiEspUdp.h>
#include <PubSubClient.h>
#include "SoftwareSerial.h"
```

Pada Tabel 5-3 ditampilkan *library* Arduino yang digunakan dalam sistem untuk inisialisasi sistem. *Library* digunakan untuk memudahkan pengguna Arduino membuat sebuah program. Library Arduino berbeda dari satu hardware ke hardware yang lain karena berisi subrutin yang mempunyai fungsi tertentu, yang harus ditambahkan *developer* agar hardware dapat berjalan dengan maksimal (Ajie, 2015). Library yang digunakan dalam sistem yang dibuat penulis adalah sebagai berikut:

- Library dari modul wifi ESP8266 yang ditambahkan ke Arduino berupa *master default* library ESP8266, library untuk *client* ESP8266, dan library untuk UDP.
- Library untuk protokol komunikasi menggunakan MQTT berupa library PubSubClient. Library PubSubClient adalah library di Arduino untuk menjalankan MQTT sebagai MQTT Publisher dengan QoS 0 hingga QoS 1. Di dalam library ini juga dikonfigurasikan beberapa hal antara lain

besar paket maksimal sebesar 128 byte dalam MQTT_MAX_PACKET_SIZE, interval sistem *default* sebesar 15 detik, dan penggunaan *default* MQTT versi 3.1.1.

- Library SoftwareSerial untuk komunikasi antar serial dalam papan Arduino. Dalam library ini, Arduino diberikan akses untuk komunikasi serial dari pin selain 0 dan 1 yang berguna untuk banyak pin yang akan berkomunikasi sehingga menghemat penggunaan sumber daya.

Setelah inisialisasi library yang akan digunakan dalam program, kemudian untuk akses pengiriman data secara *wireless* menggunakan modul wifi ESP8266 juga harus diinisialisasi. Selain agar koneksi antara perangkat dengan web server dapat berjalan dengan lancar, juga agar keamanan data terjaga karena web server pada thingsboard dilengkapi dengan token.

Tabel 5-4 Mendefinisikan Koneksi dan Token

```
#define WIFI_AP "iPhone SE"
#define WIFI_PASSWORD "12345789"
#define TOKEN "FHW7qCTGfFv6fBMC47jM"
```

Pada Tabel 5-4 adalah potongan kode program yang berguna untuk mendefinisikan koneksi internet nirkabel yang akan digunakan oleh modul wifi ESP8266 dalam melakukan koneksi ke web server thingsboard. Pendefinisian ini bertujuan sebagai penegasan bahwa modul wifi ESP8266 akan melakukan koneksi dengan Access Point (AP) bernama iPhone SE dengan password 12345789. Apabila akan melakukan koneksi dengan AP yang lain, maka kode program untuk WIFI_AP dan WIFI_PASSWORD harus diubah sesuai dengan nama access point dan password yang akan digunakan.

Token yang terkandung dalam potongan program pada Tabel 5-4 adalah kode unik untuk akses koneksi perangkat ke web server thingsboard. Token berfungsi untuk menentukan perangkat yang akan terkoneksi ke thingsboard, karena web server thingsboard dapat menerima lebih dari satu perangkat yang terkoneksi secara bersama.

5.3.2.1 Pembacaan Data Sensor

Pada bagian pembacaan data sensor, proses yang terjadi adalah sensor MQ-135 dan MQ-4 mengirimkan data mentah ke Arduino lewat koneksi pin. Proses koneksi ini memerlukan inisialisasi, baik itu di sisi kedua sensor, web server sebagai output, modul wifi ESP8266 sebagai *ethernet client object*, dan output default pada kedua sensor.

Dalam proses inisialisasi pin untuk input Arduino dari kedua sensor gas, maka harus dideklarasikan inisialisasinya seperti pada Tabel 5-5. Potongan kode program pada Tabel 5-5 menunjukkan bahwa Arduino akan mulai membaca data sensor dengan menginisialisasi masukan analog pada pin A0 dan A1 sebagai input dari sensor MQ-135 dan MQ-4.

Tabel 5-5 Inisialisasi Pin Input pada Arduino

```
const int analogInPin  = A0;
const int analogInPin2 = A1;
```

Kemudian, agar dapat terkoneksi dengan web server di Thingsboard, maka digunakan perintah seperti yang ditunjukkan dalam potongan kode program pada Tabel 5-6. Pada Tabel 5-6 di bawah, proses koneksi ke server thingsboard menuju kepada alamat web *demo.thingsboard.io*.

Tabel 5-6 Inisialisasi Server Thingsboard

```
char thingsboardServer[] = "demo.thingsboard.io";
```

Setelah itu, agar dapat terkoneksi dengan server thingsboard melalui modul wifi ESP8266, maka harus diinisialisasikan pula dalam Arduino agar Arduino mengenali bahwa ESP8266 sebagai media pengirim data ke web server. Untuk menginisialisasi ESP8266 sebagai ethernet client object adalah dengan memasukkan perintah *WifiEspClient espClient;* dalam kode program. Penggunaan perintah tersebut akan mengarahkan Arduino untuk mengeksekusi sintaks pada library modul wifi ESP8266. Setelah selesai diinisialisasi, maka dideklarasikan pula untuk client publisher pada ESP8266 dengan perintah *PubSubClient client(espClient);* sehingga ESP8266 siap menerima data sekaligus mengirim data menuju ke web server thingsboard.

Selanjutnya, kedua sensor akan diinisialisasi. Pada proses ini, semua nilai pada sensor dibuat menjadi 0 (nol) terlebih dahulu agar pembacaan data yang dikirim oleh sensor akurasinya terjaga dan dimulai dari angka 0. Tabel 5-7 menunjukkan potongan kode program untuk inisialisasi sensor. Pada Tabel 5-7 tersebut, digunakan 2 nilai pada inisialisasi sensor karena pada sistem monitoring ini menggunakan dua sensor, yakni sensor MQ-135 dan sensor MQ-4 sehingga perlu untuk menginisialisasi keduanya agar komunikasi dengan Arduino dapat berjalan dengan baik.

Tabel 5-7 Inisialisasi Kedua Sensor Gas pada Sistem

```
int sensorValue   = 0;
int sensorValue2  = 0;
int outputValue   = 0;
int outputValue2  = 0;
```

Setelah inisialisasi sensor selesai, maka kemudian dalam Arduino akan diatur port dengan serial manakah yang akan menjadi *transmitter* dan *receiver* untuk koneksi dengan sensor. Dalam sistem ini, pada software serial nomor 2 akan menjadi penerima atau receiver, sedangkan serial nomor 3 akan menjadi transmitter atau pengirim. Maka perlu ditambahkan kode program *SoftwareSerial soft(2,3);* yang berarti serial komunikasi nomor 2 sebagai receiver dan nomor 3 sebagai transmitter.

Proses pembacaaan data dari sensor dimulai dengan cara membaca data inputan pin pada Arduino dari sensor MQ-135 dan sensor MQ-4. Pada Tabel 5-8 adalah potongan kode program awal untuk pembacaan data yang dikirimkan oleh sensor ke dalam Arduino.

Tabel 5-8 Fungsi Loop Pembacaan Data Sensor

```
void loop()
{
  sensorValue = analogRead(analogInPin);
  sensorValue2 = analogRead(analogInPin2);
```

Pada potongan kode program dalam Tabel 5-8 di atas, pembacaan data sensor dimulai dari fungsi *void loop ()* yang berarti bahwa pembacaan data akan berlangsung secara terus-menerus (looping), dengan memerintahkan Arduino untuk membaca data analog dari pin 1 dan pin 2 yang telah diinisialisasikan sebagai pin yang menerima data dari sensor. Data dari sensor yang telah diterima ini kemudian disimpan dalam variabel sensorValue untuk data dari sensor MQ-4 dan sensorValue2 untuk data dari sensor MQ-135.

Perancangan pembacaan sensor selanjutnya adalah untuk memasukkan data dari sensor ke dalam Arduino untuk dikumpulkan dan diolah berdasarkan rentang dari masing-masing sensor. Untuk lebih jelasnya akan dijelaskan sesuai Tabel 5-9 berikut.

Tabel 5-9 Proses Pengambilan Data Berdasarkan *Sensor Range*

```
  outputValue = map(sensorValue, 0, 1023, 200, 10000); //MQ4
  outputValue2 = map(sensorValue2, 0, 1023, 10, 1000);//MQ135
```

Potongan kode program di atas menjelaskan bahwa variabel *sensorValue* dan *sensorValue2* yang berisi data mentah dari pin analog pada Arduino akan dimasukkan ke dalam variabel baru bernama *outputValue* dan *outputValue2* yang telah dideklarasikan di bagian sebelumnya dengan dimulai dari angka 0. Untuk memasukkan ke dalam variabel *outputValue* dan *outputValue2*, maka digunakanlah fungsi *map* pada kode program sistem. Fungsi *map* dalam Arduino berguna untuk memetakan nilai suatu variabel berdasarkan rentang yang telah ditentukan.

Pada sensor MQ-135 dan MQ-4 terdapat rentang minimal dan maksimal untuk pengukuran gas yang berbeda. Sensor MQ-135 yang mengukur kadar gas amonia di udara memiliki rentang antara 10 hingga 1000 ppm sehingga dapat bekerja dengan baik di antara rentang tersebut. Sedangkan MQ-4 memiliki rentang antara 200 hingga 10000 ppm. Kedua rentang tersebut dijadikan acuan untuk diimplementasikan ke dalam fungsi *map* yang harus mencakup parameter sebagai berikut:

- Nilai dari fungsi *map*, dalam sistem ini berupa nilai data sensor yang disimpan dalam variabel *sensorValue* dan *sensorValue2*,
- Nilai terendah dari variabel yang dituju, dalam sistem ini karena variabel *sensorValue* dan *sensorValue2* adalah sebuah variabel dengan tipe data integer, maka nilai terendahnya adalah 0,
- Nilai tertinggi dari variabel yang dituju, dalam sistem ini karena variabel *sensorValue* dan *sensorValue2* adalah sebuah variabel dengan tipe data integer, maka nilai tertingginya adalah 1023,
- Rentang terendah dari variabel yang dituju, dalam sistem ini rentang terendah dari sensor MQ-135 adalah 10 ppm dan rentang terendah dari sensor MQ-4 adalah 200 ppm,
- Rentang tertinggi dari variabel yang dituju, dalam sistem ini rentang tertinggi dari sensor MQ-135 adalah 1000 ppm dan rentang terendah dari sensor MQ-4 adalah 10000 ppm.

Parameter di atas diimplementasikan ke dalam fungsi *map* yang berguna untuk memasukkan data ke dalam variabel *outputValue* maupun *outputValue2* yang pada bagian selanjutnya akan diproses dan dikumpulkan dengan sesuai rentang yang ditentukan. Data dari sensor yang telah masuk ke variabel *outputValue* dan *outputValue2* akan dapat terbaca oleh Arduino.

5.3.2.2 Pemrosesan Data

Pada bagian pemrosesan data, data dari sensor MQ-135 dan MQ-4 akan dikumpulkan untuk diolah, dan kemudian dikirim melalui koneksi serial ke modul wifi ESP8266. Pada proses ini memerlukan beberapa kebutuhan yang harus dilakukan terlebih dahulu, terutama mengkoneksikan modul wifi ESP8266 dengan access point terdekat. Selain itu, karena waktu eksekusi program akan juga diteliti dan diimplementasikan, maka untuk waktu pemrosesan data juga akan ditampilkan.

Tabel 5-10 Koneksi Modul Wifi ESP8266 ke Jaringan

```
status = WiFi.status();
if ( status != WL_CONNECTED)
{
  while ( status != WL_CONNECTED)
  {
    Serial.print("Attempting to connect to WPA SSID: ");
    Serial.println(WIFI_AP);
    // Connect to WPA/WPA2 network
    status = WiFi.begin(WIFI_AP, WIFI_PASSWORD);
    delay(500);
  }
  Serial.println("Connected to AP");
}

if ( !client.connected() )
{
  reconnect();
}
```

Pada Tabel 5-10 adalah potongan kode program untuk koneksi modul wifi ESP8266 dengan jaringan nirkabel terdekat, yang diawali dengan memeriksa apakah modul wifi ESP8266 sudah terkoneksi dengan fungsi *status = WiFi.status()*. Proses ini akan berulang karena mengikuti fungsi void loop () pada perintah sebelumnya.

Jika terjadi koneksi, maka sistem akan melanjutkan koneksi tersebut dengan mencocokkan nama SSID dan password pada SSID, dan menampilkannya jika kita mengecek pada serial monitor Arduino. Variabel WIFI_AP dan WIFI_PASSWORD yang telah diinisialisasi pada langkah sebelumnya akan dicocokkan dengan jaringan nirkabel di sekitar modul wifi ESP8266. Jika nama SSID dan password sudah benar, maka koneksi dimulai, ditandai dengan perintah *status = WiFi.begin* dan akan tampil di layar serial monitor dengan teks “Connected to AP”

Delay proses berarti setiap proses pada koneksi jaringan nirkabel di sekitar modul wifi ESP8266 pada sistem monitoring ini diberikan waktu 500 ms atau 0,5 detik untuk melakukan koneksi.

Tetapi jika modul wifi ESP8266 tidak bisa terkoneksi dengan jaringan nirkabel di sekelilingnya, maka sistem akan terus melakukan koneksi ulang dengan fungsi *reconnect* yang akan dijelaskan kemudian hingga modul wifi ESP8266 benar-benar terkoneksi dengan jaringan. Pada layar serial monitor akan terus ditampilkan tulisan “Attempting to connect to WPA SSID: “ hingga terkoneksi dengan jaringan dan cocok dengan yang telah diinisialisasi nama SSID dan passwordnya.

Tabel 5-11 Pemantauan Waktu Eksekusi pada Serial Monitor

```
  if ( micros() - lastSend > 1000000 )
  { // Update and send only after 1 seconds
    getAndSendGASData();
    lastSend = micros();

    Selesai = micros();
Serial.println(Selesai-Mulai);
  client.loop();
  }
```

Pada Tabel 5-11 di atas adalah proses *background* untuk melihat dan menampilkan waktu eksekusi sejak Arduino menjalankan baris pertama pada program sistem monitoring yang dibuat penulis. Untuk mengakses waktu eksekusi ada dua perintah yang dapat digunakan, yaitu *millis* dan *micros*. Penulis menggunakan perintah fungsi *micros* karena proses monitoring waktu eksekusi akan sangat cepat jika menggunakan fungsi *millis* mengingat bahwa fungsi *millis* mempunyai hitungan dengan satuan milisekon, sedangkan *micros* mempunyai hitungan dengan satuan mikrosekon yang lebih rendah.

Sebelum menggunakan *micros* maka harus diinisialisasikan dulu beberapa variabel yang akan digunakan. Variabel *Mulai* diinisialisasikan terlebih dahulu sebagai awal waktu yang dihitung pada saat Arduino mulai mengeksekusi perintah pada program sistem ini dengan tipe data *unsigned long* yang dapat menyimpan jumlah data yang besar untuk ukuran variabel. Inisialisasi variabel *Mulai* dapat

diletakkan di bagian atas awal program, yang dalam sistem ini adalah saat Arduino mulai membaca data sensor dengan perintah di bawah ini.

Mulai = micros();

Dengan adanya perintah fungsi micros di atas pembacaan data pada sensor, maka proses pertama ini akan dihitung waktu eksekusinya hingga perintah selesai diinisialisasikan dengan variabel *Selesai* yang akan diletakkan di bagian terakhir proses eksekusi program. Pada sistem ini, bagian akhir eksekusi program adalah ketika Arduino mengirimkan data ke modul wifi ESP8266. Variabel Selesai juga diinisialisasikan sebagai variabel *unsigned* long sehingga dapat menyimpan data berupa waktu lebih besar. Perintah untuk menghentikan monitor waktu dan menyimpan datanya adalah *Selesai = micros();*.

Jika variabel mulai dan selesai sudah diinisialisasi dan sudah terdefinisikan dengan baik pada sistem, maka fungsi *micros* dijalankan. Pada Tabel 5-11 di atas, fungsi akan berjalan jika ada interval. Pada subbab sebelumnya, interval dalam sistem adalah 1 detik atau 1 sekon yang dikonversikan menjadi 1.000.000 milisekon. Jika program berjalan lebih dari 1 detik, maka program akan kembali ke proses sebelumnya.

Penghitungan waktu eksekusi program didapatkan dengan cara mencari nilai selisih antara waktu selesai dengan waktu mulai. Pada potongan program di atas, selisih tersebut ditampilkan pada layar serial monitor dengan operasi pengurangan variabel *Selesai* dikurangi variabel *Mulai*. Proses ini akan terus berjalan selama Arduino masih menyala dan menerima data dari kedua sensor dalam sistem.

5.3.2.3 Pengiriman Data

Pada bagian pengiriman data ini, data dari sensor MQ-135 dan sensor MQ-4 yang telah dikirimkan ke Arduino melalui koneksi pin dengan variabel *outputValue* dan *outputValue2* diinisialisasikan lagi dengan variabel baru. Untuk lebih jelasnya ada di Tabel 5-12 berikut.

Tabel 5-12 Pengambilan Data dan Pengalihan Variabel

```
void getAndSendGASData()
{
  Serial.println("Collecting GAS data.");


  float h = outputValue;
  float t = outputValue2;
```

Pada Tabel 5-12 menjelaskan bahwa setelah mencetak tulisan “Collecting GAS data” pada layar serial monitor, maka proses pengambilan data dan pengalihan variabel ke variabel *h* dan variabel *t* dimulai. Kedua variabel ini memiliki tipe data *float* yang mendukung adanya angka desimal. Variabel *h* didapatkan dari nilai *outputValue* sedangkan variabel *t* didapatkan dari nilai *outputValue2*. Keduanya adalah hasil dari proses yang telah dijelaskan sebelumnya. Setelah dialihkan ke

variabel yang mendukung angka desimal, maka akan didapatkan nilai pembacaan sensor yang sesuai dan di antara rentang yang telah didefinisikan sebelumnya.

Selanjutnya, agar kehandalan sistem dapat berjalan dengan baik, maka sistem juga akan mengecek apakah pembacaan kedua sensor terjadi error atau gagal baca. Jika terdapat kegagalan dalam membaca data sensor, maka proses akan otomatis berhenti dan sistem akan mengulang untuk membaca data sensor kembali, sesuai dengan Tabel 5-13.

Tabel 5-13 Fungsi IsNan dalam Arduino

```
if (isnan(h) || isnan(t))
  {
    Serial.println("Failed to read from GAS sensor!");
    return;
  }
```

Pada Tabel 5-13 Arduino akan mengecek apakah ada kegagalan dalam membaca data sensor melalui fungsi IsNan. Dalam sistem ini, pada variabel *h* dan variabel *t* akan dicek apakah pada saat data diambil nilainya kosong atau *false*, sesuai dengan kaidah fungsi IsNan. Jika kosong atau false, maka akan dikembalikan dengan tampilan layar pada serial monitor yang bertuliskan “Failed to read from GAS sensor!”

Jika keadaan kedua sensor dapat dibaca dengan baik, maka akan berlanjut ke proses selanjutnya untuk menampilkan data sensor menuju ke serial monitor yang potongan kode programnya ditunjukkan di Tabel 5-14 berikut

Tabel 5-14 Menampilkan Kadar Gas yang telah Diukur

```
  Serial.print("GAS_MQ-4: ");
  Serial.print(h);
  Serial.print("%");
  Serial.print("GAS_MQ_135 ");
  Serial.print(t);
  Serial.print("%");

  String GAS_MQ_4   = String(h);
  String GAS_MQ_135 = String(t);
```

Pada Tabel 5-14 di atas, sistem akan diperintah untuk menampilkan data dari kedua sensor ke dalam serial monitor Arduino. Variabel *h* yang merupakan hasil dari pengukuran sensor MQ-4 yang telah bertipe data float yang mendukung angka desimal akan ditampilkan di serial monitor dengan format “GAS_MQ-4: *h* %”. Hal yang sama juga berlaku di variabel *t* yang merupakan hasil dari nilai output pengukuran sensor MQ-135.

Kemudian, data akan dikirim ke modul wifi ESP8266 dengan menyiapkan data berformat json-nya. Untuk menyiapkan data json, maka akan digunakan fungsi

payload karena dalam sistem ini pengiriman data menggunakan protokol MQTT. Fungsi *payload* adalah untuk menyimpan data dalam format byte sebagai topik MQTT yang akan digunakan oleh MQTT Broker yang dalam sistem ini adalah server thingsboard, untuk meneruskan pesan ke klien. Potongan kode program untuk menyiapkan data berformat json tersebut dapat dilihat pada Tabel 5-15 berikut.

Tabel 5-15 Menyiapkan String Payload yang Berformat Json

```
String payload = "{";
  payload += "\"GAS_MQ_4\":"; payload += GAS_MQ_4; payload += 
",";
  payload += "\"GAS_MQ_135\":"; payload += GAS_MQ_135;
  payload += "}";
```

Untuk publish topik menggunakan fungsi *payload* tersebut menggunakan perintah client.publish dengan memasukkan topik telemetry di dalam perintah tersebut sehingga data atribut dapat dikirimkan. Dalam perintah publish ini, fungsi *payload* berguna untuk menyimpan data yang didapatkan dari kedua sensor dalam bentuk paket sebelum dikirimkan dalam topik telemetry tersebut. Nantinya, pada Thingsboard, topik yang sesuai dengan token yang digunakan sistem untuk berkomunikasi akan diterima dan ditampilkan dalam bentuk *chart* dan grafik.

Tabel 5-16 Mengirimkan Payload Agar Dapat Diterima Thingsboard

```
char attributes[100];
  payload.toCharArray( attributes, 100 );
  client.publish( "v1/devices/me/telemetry", attributes );
  Serial.println( attributes );
}
```

Pada proses inisialisasi modul wifi ESP8266, program akan menginisialisasi apakah modul wifi ESP8266 tersambung dalam sistem dengan perintah WiFi.init(&soft);

Jika modul wifi ESP8266 terdeteksi tidak ada di dalam sistem, maka akan dimunculkan perintah bahwa modul wifi ESP8266 tidak ada. Dalam serial monitor akan muncul pesan “WiFi shield not present”. Tetapi, jika modul wifi ESP8266 terdeteksi, maka akan berlanjut ke proses berikutnya.

Dalam Tabel 5-17 di bawah, terlihat bahwa proses selanjutnya ketika modul wifi ESP8266 terdeteksi adalah melakukan koneksi nirkabel ke SSID terdekat. Pada serial monitor ditampilkan pesan bahwa sistem melakukan koneksi ke *access point* yang terdapat dalam perintah Serial.println(“Connecting to AP …”);. Ketika telah terkoneksi, maka sistem akan otomatis mencocokkan nama dan password SSID dengan *access point* pada perintah status = WiFi.begin (WIFI_AP, WIFI_PASSWORD);. Pesan bahwa sistem sudah terkoneksi dengan jaringan nirkabel akan muncul pada serial monitor dengan tulisan “Connected to AP” ketika koneksi berhasil.

Tabel 5-17 Inisialisasi pada Modul Wifi ESP8266

```
void InitWiFi()
{
  // initialize serial for ESP module
  soft.begin(9600);
  // initialize ESP module
  WiFi.init(&soft);
  // check for the presence of the shield
  if (WiFi.status() == WL_NO_SHIELD)
  {
    Serial.println("WiFi shield not present");
    // don't continue
    while (true);
  }

  Serial.println("Connecting to AP ...");
  // attempt to connect to WiFi network
  while ( status != WL_CONNECTED)
  {
    Serial.print("Attempting to connect to WPA SSID: ");
    Serial.println(WIFI_AP);
    // Connect to WPA/WPA2 network
    status = WiFi.begin(WIFI_AP, WIFI_PASSWORD);
    delay(500);
  }
  Serial.println("Connected to AP");
}
```

Pada Tabel 5-18, proses selanjutnya adalah sistem akan melakukan koneksi ulang modul wifi ESP8266 jika tidak dapat terhubung ke AP. Fungsi pada Tabel 5-18 di bawah akan dijalankan secara terus menerus hingga sistem berhasil terkoneksi dengan jaringan nirkabel.

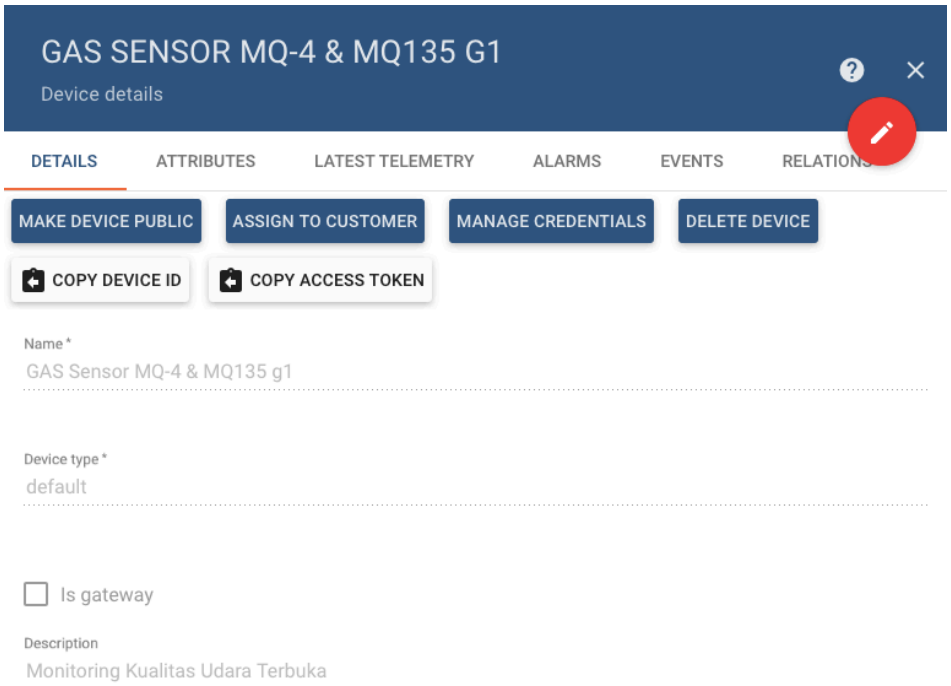
Jika sistem sudah berhasil terkoneksi dengan jaringan nirkabel, maka selanjutnya adalah koneksi ke *node* Thingsboard sebagai web server output pada sistem monitoring yang dibuat penulis. Proses koneksi ke Thingsboard dimulai dari munculnya pesan “Connecting to Thingsboard node …”) pada serial monitor Arduino. Kemudian, sistem akan mengecek token pada program untuk mengetahui apakah token tersebut sama dengan token yang ada di Thingsboard. Jika token tersebut sama, program siap dieksekusi dengan kembali ke bagian pembacaan data sensor setelah sebelumnya menampilkan pesan “[DONE]” di serial monitor. Namun, jika token tidak sama atau sistem tidak bisa menjangkau server Thingsboard, maka akan muncul pesan “[FAILED]” dan sistem akan kembali mengulang koneksi 5 detik kemudian.

Tabel 5-18 Proses Koneksi Ulang

```
void reconnect()
{
  while (!client.connected())
  {
    Serial.print("Connecting to ThingsBoard node ...");
    if ( client.connect("Arduino Uno Device", TOKEN, NULL) )
    {
      Serial.println( "[DONE]" );
    }
    else
    {
      Serial.print( "[FAILED] [ rc = " );
      Serial.print( client.state() );
      Serial.println( " : retrying in 5 seconds]" );
      delay( 5000 );
    }
  }
}
```

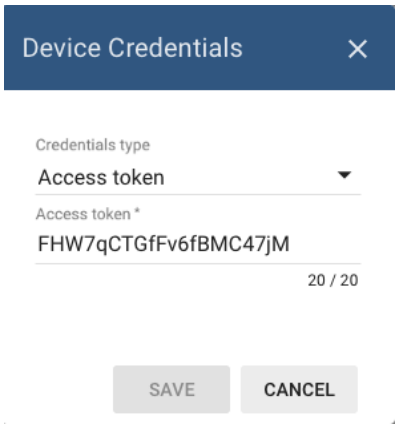
5.3.3 Output Data ke Web Server

Untuk menambahkan hasil pembacaan data ke dalam Thingsboard, terlebih dahulu setelah login pada *tenant* Thingsboard, perlu untuk menambahkan Device dalam Thingsboard Demo yang diakses di alamat http://demo.thingsboard.io. Device yang akan ditambahkan adalah device Gas Sensor MQ-4 dan MQ-135 dengan rincian yang ada dalam *library* Thingsboard. Setelah ditambahkan device, maka akan muncul tampilan seperti pada Gambar 5.12 berikut.



Gambar 5.12 Tampilan Device yang Tersambung di Thingsboard

Untuk melakukan koneksi pada sistem, diperlukan token yang didapatkan dari inisialisasi dan penambahan data sensor dalam server Thingsboard. Token didapatkan dari *credentials* yang ada pada Thingsboard untuk berhubungan dengan device yang akan menampilkan datanya pada Thingsboard. Dalam sistem monitoring ini, token yang disediakan Thingsboard untuk melakukan koneksi adalah ***FHW7qCTGfFv6fBMC47jM*** yang didapatkan seperti pada Gambar 5.13 berikut.

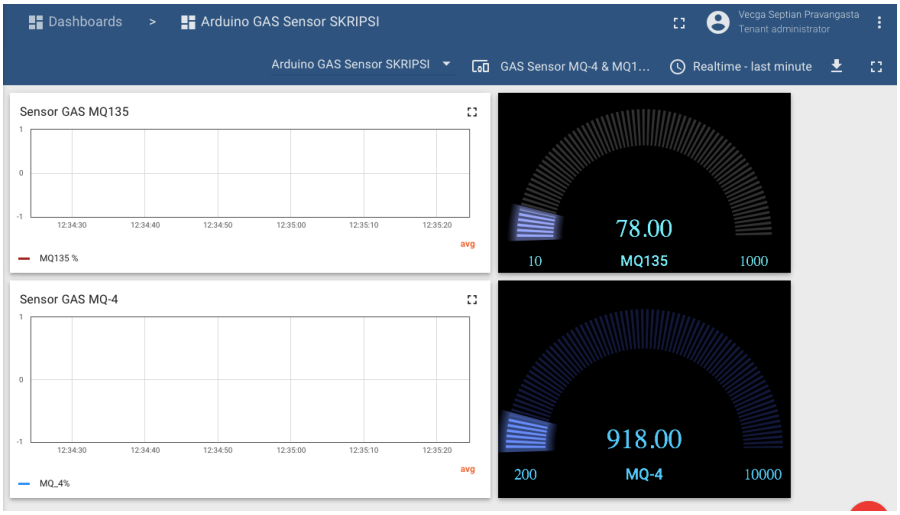


Gambar 5.13 Token Akses pada Thingsboard

Setelah device ditambahkan, selanjutnya adalah menambahkan *widgets* untuk mengetahui perubahan data yang dideteksi oleh sensor MQ-4 maupun MQ-135 dalam bentuk *chart* ataupun grafik. *Widgets* tersebut ditambahkan melalui *dashboard* Thingsboard sehingga data yang akan ditampilkan dapat dikelompokkan dengan rapi dan memudahkan pembacaan. Pada penelitian ini,

penulis menggunakan *chart* berupa *timeseries chart* dan grafik berupa *digital gauges* yang ada dalam *widgets library* pada Thingsboard.

Secara keseluruhan, pada Gambar 5.14 adalah cuplikan layar *dashboard* pada Thingsboard yang diberi nama Arduino GAS Sensor SKRIPSI untuk mengonversikan data yang didapatkan dari sensor menjadi format yang berbentuk *chart* dan grafik.



Gambar 5.14 Tampilan *Dashboard* pada Thingsboard