PERCANCANGAN SISTEM PEMANTAUAN KUALITAS UDARA INTERNET OF THING MENGGUNAKAN MQ-135 DITAMPILKAN MELALUI WEB

SKRIPSI

Untuk Memenuhi Sebagian Persyaratan
Mencapai Gelar Sarjana Komputer

Program Studi : Sistem Komputer

Jejang Pendidikan : Strata 1 (S1)

Diajukan Oleh :

TAUFIQURAHMAN

18101152620034

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PUTRA INDONESIA "YPTK"

PADANG

2021

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	:	Taufiqurahman
NoBP	:	18101152620034
Fakultas	:	ILMU KOMPUTER
Jurusan	:	SISTEM KOMPUTER

Menyatakan Bahwa :

1. Sesungguhnya skripsi yang saya susun ini merupakan hasil karya tulis saya sendiri. Adapun bahagian-bahagian tertentu dalam skripsi yang saya peroleh dan hasil karya tulis orang lain, telah saya tuliskan sumbernya dengan jelas, sesuai dengan kaidah penulisan ilmiah.
2. Jika dalam pembuatan skripsi baik pembuatan program maupun skripsi secara keseluruhan terbukti dibuatkan oleh orang lain, maka saya bersedia menerima sanksi yang diberikan akademik, berupa pembatalan skripsi dan mengulang penelitian serta mengajukan judul baru

Demikian surat pernyataan ini saya buat dengan sesungguhnya tanpa ada paksaan dari pihak mana pun.

Padang, 03 Maret 2022

(Taufiqurahman)

18101152620034

HALAMAN PENGESAHAN SKRIPSI

PERCANCANGAN SISTEM PEMANTAUAN KUALITAS UDARA INTERNET OF THING MENGGUNAKAN MQ-135 DITAMPILKAN MELALUI WEB

Yang dipersiapkan dan disusun oleh

TAUFIQURAHMAN

18101152620034

Telah memenuhi persyaratan untuk dipertahankan di depan dewan penguji pada ujian tahap akhir

Padang, 03 Maret 2022

Pembimbing I	Pembimbing II
(Hadi Syahputra, S.Kom, M.Kom)	Ondra Eka Putra, S.Kom, M.Kom
NIDN : 1011108502	NIDN : 1006068701

HALAMAN PENGESAHAN PENGUJI SIDANG SKRIPSI

PERCANCANGAN SISTEM PEMANTAUAN KUALITAS UDARA INTERNET OF THING MENGGUNAKAN MQ-135 DITAMPILKAN MELALUI WEB

OLEH

TAUFIQURAHMAN

18101152620034

PROGRAM STUDI SISTEM KOMPUTER

Skripsi Ini Telah Dinyatakan LULUS Oleh Penguji Materi pada Sidang Skripsi Program Studi Strata 1 Ilmu Komputer Program Studi Sistem Komputer Universitas Putra Indonesia "YPTK" Padang

Pada Hari/Tanggal: Rabu, 03 Maret 2022

TIM PENGUJI

1. **Ondra Eka Putra, S.Kom, M.Kom** .....

   NIDN : 1006068701

2. **Hasri Awal, S.Kom, M.Kom** .....

   NIDN : 1020099101

Padang, 03 Maret 2021

Dekan Fakultas Ilmu Komputer

Universitas Putra Indonesia "YPTK" Padang

Dr. Yuhandri, S.Kom, M.Kom

NIDN : 1015057301

HALAMAN PENGESAHAN LULUS SIDANG SKRIPSI

PERCANCANGAN SISTEM PEMANTAUAN KUALITAS UDARA INTERNET OF THING MENGGUNAKAN MQ-135 DITAMPILKAN MELALUI WEB

Yang dipersiapkan dan disusun oleh

TAUFIQURAHMAN

18101152620034

Telah dipertahankan di depan dewan penguji

Pada tanggal : 03 Maret 2021

dan dinyatakan telah lulus memenuhi syarat.

Pembimbing I	Pembimbing II
(Hadi Syahputra, S.Kom, M.Kom)	Ondra Eka Putra, S.Kom, M.Kom
NIDN : 1011108502	NIDN : 1006068701

Padang, 03 Maret 2021

Dekan Fakultas Ilmu Komputer

Universitas Putra Indonesia "YPTK" Padang

Dr. Yuhandri, S.Kom, M.Kom

NIDN : 1015057301

ABSTRACT

Thesis Title	**:**	**THE ESTABLISHMENT OF INTERNET OF THING AIR QUALITY MONITORING SYSTEM USING MQ-135 DISPLAYED VIA WEB**
Name	**:**	**Taufiqurahman**
No Bp	**:**	**18101152620034**
Study Program	**:**	**Computer System**
Level of Education	**:**	**Strata 1 (S1)**
Guide	**:**	**1. Hadi Syahputra, S.Kom, M.Kom**
		2. Ondra Eka Putra, S.Kom, M.Kom

The study intends to design and build an air quality monitoring device both outdoors and indoor. This system can be used as a parameter for decision making in the suppression of air pollution. The tool consists of servers and sensor nodes. Servers are built in NodeJS that can run any devices and sensor node is a tool equipped with MQ-135 sensors controlled by NodeMCU for air quality detection, then the data will be sent to the server to saved at database and display over the web.

Keywords: Air Quality Monitoring System, Air Quality, NodeMCU, MQ-135

ABSTRAK

Judul Skripsi	**:**	**PERCANCANGAN SISTEM PEMANTAUAN KUALITAS UDARA INTERNET OF THING MENGGUNAKAN MQ-135 DITAMPILKAN MELALUI WEB**
Nama	**:**	**Taufiqurahman**
No Bp	**:**	**18101152620034**
Program Studi	**:**	**Sistem Komputer**
Jenjang Pendidikan	**:**	**Strata 1 (S1)**
Pembimbing	**:**	**1. Hadi Syahputra, S.Kom, M.Kom**
		2. Ondra Eka Putra, S.Kom, M.Kom

Penelitian ini bermaksud untuk merancang dan membangun suatu perangkat pemantauan kualitas udara baik di outdoor maupun di indoor. Sistem ini dapat digunakan sebagai parameter untuk pengambilan keputusan dalam penekanan terhadap polusi udara. Alat ini terdiri dari server dan node sensor. Server dibuat dalam NodeJS yang dapat dijalankan banyak perangkat dan node sensor merupakan alat yang dilengkapi dengan sensor MQ-135 yang dikendalikan oleh NodeMCU untuk pendeteksian kualitas udara, kemudian data tersebut dikirim ke server untuk disimpan di *database* dan ditampilkan melalui web.

Kata Kunci: Sistem Pemantauan Kualitas Udara, Kualitas Udara, NodeMCU, MQ-135

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah memberikan rahmat dan karunia-Nya kepada peneliti sehingga akhirnya peneliti dapat juga menyelesaikan skripsi ini dengan judul **" PERCANCANGAN SISTEM PEMANTAUAN KUALITAS UDARA INTERNET OF THING MENGGUNAKAN MQ-135 DITAMPILKAN MELALUI WEB".**

Salawat serta salam peneliti ucapkan kepada junjungan segala umat yakni Nabi Besar Kita Muhammad SAW yang telah memberikan hidayah, petunjuk kepada umatnya di permukaan bumi.

Peneliti menyadari akan keterbatasan kemampuan dan pengetahuan yang ada, sehingga proses penelitian skripsi ini tidak terlepas dari bantuan serta bimbingan dari berbagai pihak. Untuk itu pada kesempatan ini perkenankanlah peneliti mengucapkan terima kasih yang sebesar-besarnya kepada :

1. **Bapak Alm. H. Herman Nawas**, selaku Ketua Yayasan PerguruanTinggi Komputer "YPTK" Padang.
2. **Bapak Prof. Dr. H. Sarjon Defit, S.Kom, M.Sc** selaku RektorUniversitas Putra Indonesia "YPTK" Padang.
3. **Dr. Yuhandri, S.Kom, M.Kom** selaku Dekan Fakultas Ilmu KomputerUniversitas Putra Indonesia "YPTK" Padang.
4. **Ibu Retno Devita, S.Kom, M.Kom** selaku Ketua Jurusan SistemKomputer Universitas Putra Indonesia "YPTK" Padang.
5. **Hadi Syahputra, S.Kom, M.Kom** selaku Dosen

pembimbing I yang telah meluangkan waktu memberikan bimbingan, arahan serta saran-saran peneliti sehingga skripsi ini dapat selesaikan dengan baik.

6. **Ondra Eka Putra, S.Kom, M.Kom** selaku dosen pembimbing II yang telah memberikan bimbingan dan petunjuk serta meluangkan waktunya selama penyusunan skripsi ini.
7. Seluruh staf dosen, karyawan dan karyawati Universitas Putra Indonesia "YPTK" Padang.
8. Orang tua, saudara-saudara, dan teman-teman yang telah mendukung dalam proses pengerjaan skripsi.

Peneliti menyadari bahwa skripsi ini masih jauh dari kesempurnaan, untuk itu peneliti sangat mengharapkan saran-saran dan kritikan yang dapat membangun guna penyempurnaan skripsi ini.

Akhir kata peneliti berharap semoga skripsi ini bermanfaat bagi kita semuaterutama bagi instansi yang terkait dan berhubungan dengan pembahasan skripsi ini.

Padang, 03 Maret 2022

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Polusi udara bukanlah hal baru di kalangan masyarakat, yang mana polusi udara memiliki efek serius pada kesehatan manusia. Menurut Adrian menjelaskan bahwa “Bahaya polusi udara terhadap paru-paru bukanlah sesuatu yang dapat dianggap sepele. Paparan polusi udara yang berlebihan diketahui bisa meningkatkan risiko terjadinya berbagai penyakit pada paru-paru, mulai dari infeksi saluran pernapasan, pneumonia, bronkitis, hingga kanker”. Adrian juga menjelaskan data dari WHO menyebutkan bahwa terdapat sekitar 7 juta orang di dunia yang meninggal setiap tahunnya akibat paparan polusi udara, baik polusi udara yang berasal dari luar ruangan maupun dalam ruangan. Sementara itu di Indonesia sendiri, angka kematian akibat polusi udara diperkirakan mencapai lebih dari 60.000 kasus setiap tahunnya.

Kota Padang sebagai salah satu kota besar di Indonesia mengalami peningkatan jumlah kendaraan setiap tahunnya. Data Badan Pusat Statistik Kota Padang tahun 2011 menunjukkan bahwa pada tahun 2010 jumlah kendaraan sebanyak 417.068 unit dengan persentase peningkatan jumlah kendaraan sebesar 23,4%. Hal ini diperkirakan berdampak terhadap kualitas udara di kawasan jalan padat lalu lintas (Gunawan dkk., 2015). Untuk mengetahui seberapa besar pencemaran udara telah mempengaruhi kualitas udara ambien di Kota Padang,

perlu dilakukan pemantauan kualitas udara ambien secara kontinu. Namun karena belum adanya stasiun *monitoring* kualitas udara ambien otomatis seperti kota-kota besar lainnya yaitu Jakarta, Bandung, Denpasar, Pekanbaru, maka pemantauan kualitas udara dilakukan hanya dari penelitian *(grab sampling)* yang jumlahnya masih sedikit (Ruslinda, 2014).

Sangat penting untuk menyadari bahwa tingkat polusi di sekitar kita. Tingkat polusi telah meningkat seiring waktu oleh banyak faktor seperti peningkatan populasi, peningkatan penggunaan kendaraan, industrialisasi, dan urbanisasi yang menghasilkan efek berbahaya pada kesejahteraan manusia dengan secara langsung mempengaruhi kesehatan populasi yang terpapar penyakit pernafasan.

Perkembangan teknologi mengalami perkembangan yang pesat setiap hari. Puluhan ide dan inovasi dimunculkan, melahirkan teknologi baru yang menawarkan kemudahan teknologi informasi lebih spesifiknya komputer. Sama-sama diketahui internet sudah menjadi hal yang umum dan kebutuhan dalam masyarakat. Sebuah istilah yang muncul dengan pengertian sebuah akses perangkat elektronik melalui media internet yang disebut *Internet of Things* (Handayani, 2019). Akses perangkat tersebut terjadi karena keinginan untuk berbagi data, berbagi akses, dan juga mempertimbangkan keamanan dalam aksesnya (Wasista dkk., 2019).

Untuk dapat melakukan pengontrolan kualitas udara dilakukan secara cepat dan tepat, penulis tertarik untuk mengangkat masalah tersebut dalam tugas

akhir yang berjudul **"PERANCANGAN SISTEM PEMANTAUAN KUALITAS UDARA INTERNET OF THINGS MENGGUNAKAN MQ-135 DITAMPILKAN MELALUI WEB".**

MQ-135 Sensor adalah polusi udara merupakan perangkat yang mendeteksi dan menyaring polusi udara di area terdekat. Masih bisa digunakan untuk lingkungan luar dan dalam (Sharma dkk., 2020). Sensor ini memiliki harga yang terjangkau di pasaran menjadi salah satu alasan penulis untuk memilih sensor ini. Konten MQ135 adalah Sn0, dan ini adalah bahan khusus. Sensor sensor gas yang dapat dideteksi MQ-135 adalah nitrogen amonia, oksigen, alkohol, senyawa vertikal, belerang, dan asap 117 (Sharma dkk., 2020).

Website adalah sebuah kumpulan halaman yang berisi informasi tertentu dan dapat diakses oleh banyak orang melalui internet. Website dapat dibuka dengan menuliskan URL atau alamat website di browser (Nur Wijayanti, 2021). Web merupakan media cross-platform yang bisa langsung diakses perangkat mana yang memiliki browser. Dengan kemudahan akses tersebut penulis tertarik untuk menampilkan data pembacaan sensor tersebut melalui web.

1.2 Rumusan Masalah

Berdasarkan latar belakang, perumusan masalah penelitian ini dapat diuraikan sebagai berikut:

1. Bagaimana NodeMCU dapat mengendalikan sistem sebagai node sensor dengan baik?

2. Bagaimana sensor MQ-135 dapat mendeteksi kualitas udara pada ruangan dengan baik?
3. Bagaimana database dapat menyimpan data-data pengukuran kualitas udara dengan baik?
4. Bagaimana Raspberry Pi dapat menjadi pengontrol database dan web service dengan baik?
5. Bagaimana Web Service menjadi antar muka pengguna untuk menampilkan data yang direkam dengan baik?
6. Bagaimana Push Button menjadi tombol reset konfigurasi dari NodeMCU dengan baik?
7. Bagaimana LED dapat menjadi indikator untuk NodeMCU dengan baik?
8. Bagaimana dapat menjadi alarm jika polutan udara tinggi dengan baik?

1.3 Batasan Masalah

Berdasarkan rumusan masalah, batasan masalah dalam penelitian ini dapat disimpulkan sebagai berikut.

Berdasarkan rumusan masalah, batasan masalah penelitian ini adalah:

1. Alat ini adalah bentuk dasar atau purwa rupa yang nantinya bisa dikembangkan lagi.
2. NodeMCU akan menjadi node sensor yang akan mengirimkan data pembacaan sensor ke server.

3. Sensor yang digunakan yaitu sensor MQ-135 sebagai pengukur kualitas udara ambien.
4. Data-data pengukuran kualitas udara akan disimpan ke Database.
5. Raspberry Pi menjadi host database dan web service.
6. Antar muka pengguna untuk menampilkan data yang direkam adalah Web.
7. Push button digunakan sebagai tombol reset konfigurasi dari NodeMCU.
8. LED digunakan sebagai indikator untuk NodeMCU.
9. Buzzer digunakan sebagai alarm jika kualitas udara terlalu rendah.

1.4 Hipotesis

Berdasarkan rumusan masalah di atas, penulis dapat mengambil beberapa hipotesis sebagai berikut.

1. Diharapkan Sensor MQ2 dapat mendeteksi kualitas udara pada ruangan dengan baik.
2. Diharapkan NodeMCU dapat mengukur dan mengirim data tersebut ke Raspberry Pi.
3. Diharapkan *Database* dapat menyimpan data-data pengukuran kualitas udara.
4. Diharapkan Raspberry Pi dapat menjadi pengontrol *database* dan *web service.*
5. Diharapkan *Web service* menjadi antar muka pengguna untuk menampilkan data yang diukur dan data yang disimpan dari database.

6. Diharapkan Push button menjadi tombol reset konfigurasi dari NodeMCU.

7. Diharapkan LED dapat menjadi indikator untuk NodeMCU.

8. Diharapkan Buzzer dapat menjadi alarm jika kualitas udara terlalu rendah.

1.5 Tujuan Penelitian

Adapun tujuan yang diinginkan dalam pembuatan alat ini yaitu sebagai berikut.

1. Dapat merancang sistem pemantauan kualitas udara yang ditampilkan melalui web dan disimpan data pemantauan tersebut ke database.
2. Dapat mengimplementasikan interoperabilitas antara NodeMCU dengan Raspberry Pi menggunakan protokol MQTT.
3. Sebagai syarat bagi penulis untuk mendapatkan gelar di jenjang pendidikan Strata 1 (S1).
4. Sebagai portofolio bagi penulis untuk melamar pekerjaan.
5. Dapat menerapkan ilmu yang telah dari mata kuliah yang telah diajarkan.

1.6. Manfaat Penelitian

Manfaat penelitian dari penelitian ini adalah sebagai berikut.

A. Bagi Penulis

1. Untuk dapat menambah wawasan penulis di bidang penerapan pada pembuatan alat berbasis *Internet of Things*.

2. Untuk dapat melatih kemampuan penulis dalam menganalisa dan memecahkan masalah.

3. Untuk dapat mengetahui dan memahami bagaimana cara kerja dari alat yang berbasis *Internet of Things.*

B. Bagi Program Studi

1. Penelitian ini diharapkan menjadi motivasi mahasiswa Sistem Komputer untuk memanfaatkan teknologi terbaru dan terus *up-to-date* ilmunya agar bermanfaat bagi lingkungan kerja maupun masyarakat.

2. Menambah referensi dalam literatur bagi mahasiswa yang berhubungan dengan NodeMCU.

3. Hasil akhir penelitian dapat dijadikan pedoman penelitian mahasiswa selanjutnya di jurusan Sistem Komputer.

C. Bagi Masyarakat

1. Memberikan wawasan kepada masyarakat bahwa pentingnya udara bersih dan mencegah peningkatan polutan di udara.

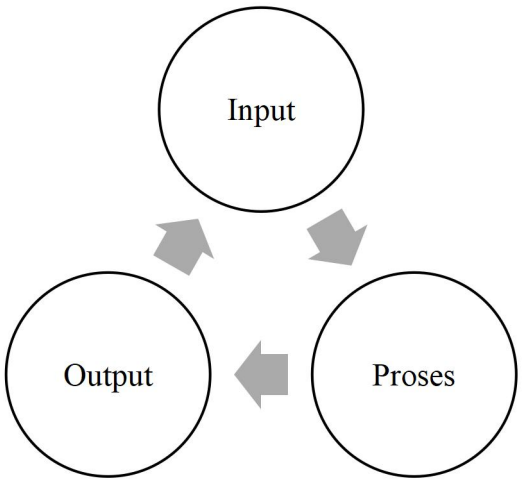
1. Dapat membantu instansi atau pemerintah untuk mengambil keputusan untuk menekan polusi udara.

BAB II

TINJAUAN PUSTAKA

2.1 Konsep Dasar Sistem

Sistem dapat didefinisikan sebagai kumpulan dari elemen-elemen berupa data, jaringan kerja dari prosedur-prosedur yang saling berhubungan, sumber daya manusia, teknologi baik *hardware* maupun *software* yang saling berinteraksi sebagai satu kesatuan mencapai tujuan/sasaran tertentu yang sama (Maniah & Hamidin, 2017). Menurut Maniah dan Hamidin juga menjelaskan sebuah sistem harus memenuhi syarat minimum yaitu, memiliki 3 unsur pembentuk sistem yang terdiri dari *input*, proses, dan *output*. Berikut adalah diagram bentuk sistem yang paling sederhana:



Sumber: Maniah & Hamidin (2017)

Gambar 2.1 *System Life Cycle*

Input adalah data atau informasi yang dibutuhkan oleh sebuah sistem untuk selanjutnya diproses sesuai dengan ketentuan proses-proses yang telah ditentukan. Pada akhirnya akan menghasilkan keluaran (*output*) yang bila diperlukan lagi maka hasil *output* tersebut akan kembali menjadi sebuah *input*, begitu seterusnya, ini yang disebut dengan *system life cycle* (siklus hidup sistem) (Maniah & Hamidin, 2017).

2.1.1 Karakteristik Sistem

Untuk memahami atau mengembangkan suatu sistem, maka kita perlu mengetahui karakteristik sistem tersebut. Berikut adalah uraian karakteristik sistem menurut Fatta:

1. Batasan (*baundary*), penggambaran dari suatu elemen atau unsur mana yang termasuk di dalam sistem dan mana yang di luar sistem.
2. Lingkungan (*environment*), segala sesuatu di luar sistem, lingkungan yang menyediakan asumsi, kendala, dan *input* terhadap suatu sistem.
3. Masukan (*input*), sumber daya (data, bahan baku, peralatan, energi) dari lingkungan yang dikonsumsi dan dimanipulasi oleh suatu sistem.
4. Keluaran (*output*), sumber daya atau produk (informasi, laporan, dokumen, tampilan layar komputer, barang jadi) yang disediakan untuk lingkungan sistem oleh kegiatan dalam suatu sistem.
5. Komponen (*component*), kegiatan-kegiatan atau proses dalam suatu sistem yang mentransformasikan *input* menjadi bentuk setengah jadi (*output*). Komponen ini bisa merupakan sub sistem dari sebuah sistem.

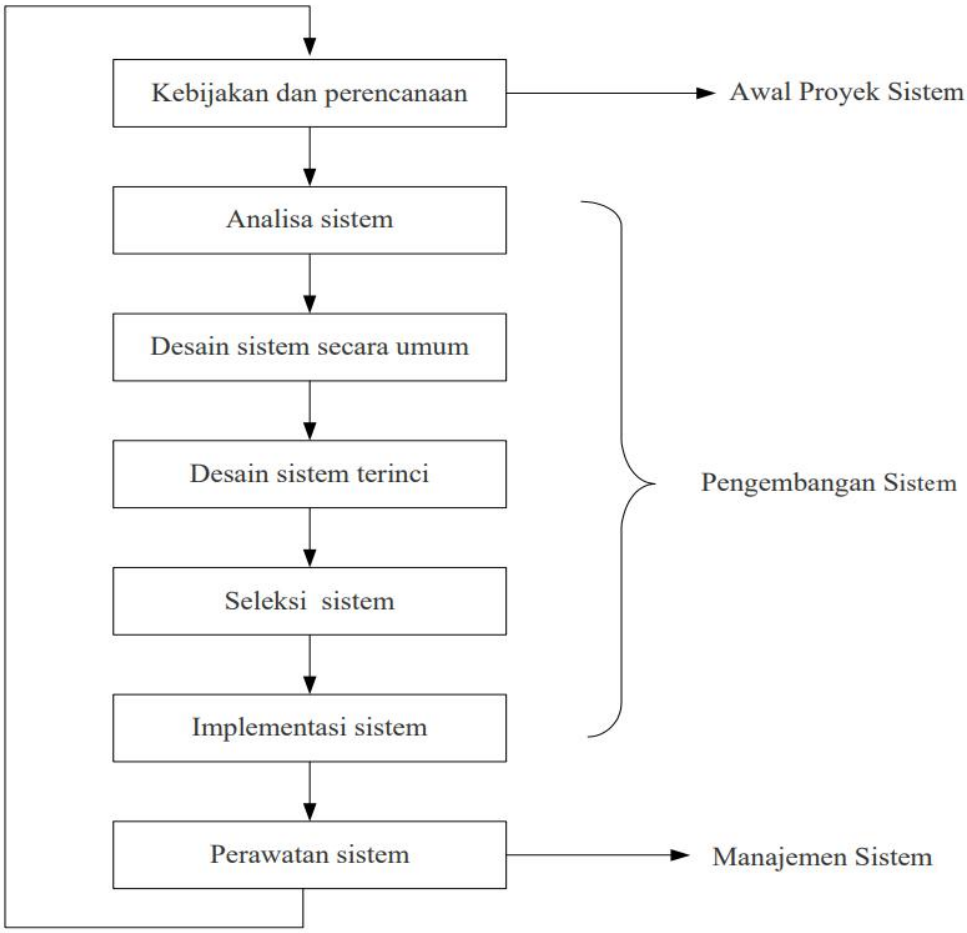
6. Penghubung (*interface*), tempat komponen atau sistem dan lingkungannya bertemu atau berinteraksi.
7. Penyimpanan (*storage*), area yang dikuasai dan digunakan untuk penyimpanan sementara dan tetap dari informasi, energi, bahan baku, dan sebagainya. Penyimpanan merupakan suatu media penyangga di antara komponen tersebut bekerja dengan berbagai tingkatan yang ada dan memungkinkan komponen yang berbeda dari berbagai data yang sama.

2.1.2 Siklus Pengembangan Sistem

Siklus pengembangan sistem adalah bentuk yang digunakan untuk menggambarkan tahapan utama dan langkah-langkah dalam tahapan tersebut dalam proses pengembangan. Siklus hidup dari pengembangan sistem atau yang dikenal juga dengan *system development life cycle (SDLC)* dapat *member* panduan dan prosedur bagi semua yang terlibat dalam proyek pembangunan sistem dengan beberapa manfaat seperti alokasi waktu yang terencana, mengurangi risiko kegagalan proyek, memastikan bahwa semua kebutuhan tercakup dalam proyek, mengidentifikasi masalah teknikal dan manajerial yang mungkin muncul, mengukur kemajuan jalannya proyek, dan mempermudah pengaturan sumber daya serta anggaran. (Alhamidi, 2016).

Tahapan utama dari siklus hidup pengembangan sistem dapat terdiri dari: perencanaan sistem (systems planning), analisis sistem (systems analysis), desain sistem (systems design), seleksi sistem (systems selection), implementasi sistem

(systems implementation), dan perawatan sistem (systems maintenance). Berikut ini pada gambar 2.2 merupakan siklus pengembangan sistem.



Sumber: Ahmadi, 2016

Gambar 2.2 Siklus Pengembangan Sistem

2.2 Alat Bantu Perancangan Sistem

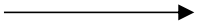
Untuk melakukan penganalisaan terhadap sebuah sistem-sistem, maka peneliti harus melakukan pendefinisian secara keseluruhan sistem yang akan

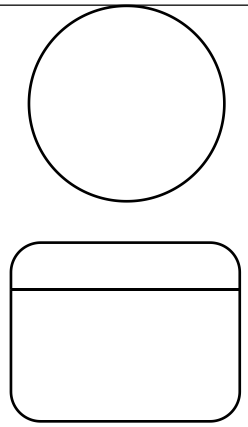
dirancang. Media yang digunakan untuk menggambarkan sistem tersebut adalah *Context Diagram* dan *Data Flow Diagram* (DFD).

2.2.1 *Context Diagram*

Context Diagram merupakan gambaran umum sebuah sistem, dari diagram konteks tersebut maka akan diketahui ke mana dan dari mana informasi yang ada pada sistem berjalan (Nurcholish, 2018). Berikut ini adalah simbol-simbol yang digunakan pada *Context Diagram*.

Tabel 2.1 Simbol-Simbol *Context Diagram* dan Keterangannya

No.	Simbol	Keterangan
1.		Kesatuan Luar (*Eksternal Entity*), merupakan kesatuan luar sistem yang dapat berupa orang, organisasi atau sistem lainya yang berada di luar lingkungan luarnya yang akan memberikan *input* atau menerima *output* sistem.
2.		Arus Data (*Data Flow*), arus data mengalir di antara proses, simpanan data dan kesatuan. Arus data ini menunjukkan arus data dari yang masuk ke dalam proses sistem.
3.		Proses (*Process*), kegiatan atau kerja yang dilakukan oleh, mesin atau komputer dari

		suatu arus data yang masuk ke dalam proses untuk dihasilkan arus data yang akan keluar dari proses.

Sumber: Tanjung & Sukrianto, 2017

2.2.2 *Data Flow Diagram*

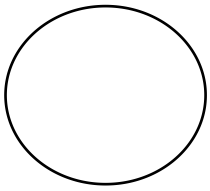
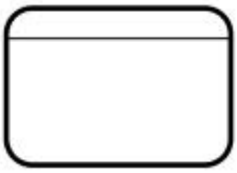
Data Flow Diagram (DFD) adalah penjabaran dari sebuah konteks diagram (Ramadhanti, 2021). Berikut pengertian DFD menurut para ahli yang dikutip oleh Ramadhanti :

1. Menurut Andri Kristanto, model proses data yang dibuat atau dirancang untuk menggambarkan aliran data, dari mana ia masuk dan ke mana tujuannya.
2. Menurut Tata Sutabri, suatu *network* yang menggambarkan sistem secara otomatis atau komputerisasi, manol serta gabungan dari keduanya. Penggambarannya disusun dalam bentuk komponen dengan aturan tertentu.
3. Menurut Jogiyanto Hartono, suatu diagram yang menggunakan simbol atau notasi dalam menggambarkan sebuah arus sistem.

4. Menurut Wijaya, sebuah gambaran grafis yang menggambarkan serta memperlihatkan aliran data dari sumbernya dalam suatu objek kemudian ditransformasikan ke tujuan lain dalam objek lain.

Di dalam pembuatan *Data Flow Diagram* terdapat aturan penulisan, notasi-notasi dan simbol-simbol standar yang digunakan. Notasi-notasi dan simbol-simbol DFD dijabarkan pada tabel 2.2

Tabel 2.2 Notasi-notasi dan simbol-simbol DFD

Notasi Yurdon DeMarco	Notasi Gane & Sarson	Deskripsi
		Simbol entitas eksternal/*terminator*, menggambarkan asal atau tujuan data di luar sistem
		Simbol lingkaran, menggambarkan entitas atau proses aliran data masuk ditransformasikan ke aliran data keluar.
		Simbol aliran data, menggambarkan aliran data.

		Simbol file, menggambarkan tempat data disimpan

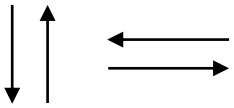
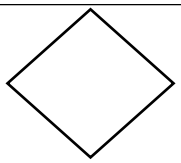
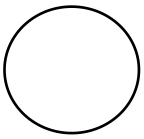
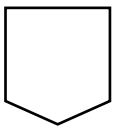
Sumber: Ramadhanti, 2021

2.2.3 *Flowchart* Program

Flowchart adalah teknik penyajian algoritma yang menggunakan gambar atau simbol-simbol yang menggambarkan urutan proses dalam penyelesaian masalah pada pembuatan suatu program (Yuniansyah, 2020). *Flowchart* program memiliki aturan penulisan yang digambarkan dari simbol-simbol yang dihubungkan dengan simbol flow alur proses. Simbol-simbol *flowchart* beserta keterangannya dapat dilihat pada tabel 2.2.

Tabel 2.3 Simbol-Simbol Flowchart dan Keterangannya

Simbol	Nama dan Keterangan
	*Terminator*, untuk menyatakan awal dan akhir *flowchart* atau suatu kegiatan.
	*Input/Output*, untuk menyatakan proses *input* dan *output* tanpa tergantung jenis peralatannya.
	Proses, untuk menyatakan suatu tindakan atau proses pada komputer.

	*Preparation*, untuk menyiapkan suatu variabel atau tempat penyimpanan suatu pengolahan data atau pemberian nilai awal
	Arus proses, merupakan simbol *flowchart* yang berfungsi menghubungkan antara simbol satu dengan simbol lainnya atau menyatakan jalannya arus dalam suatu proses. Simbol arus ini sering disebut juga dengan *connecting line*.
	*Decision*, untuk menyatakan pengambilan keputusan.
	*Predefine proses*, simbol yang dinyatakan suatu proses berada di salam sub bagian/sub program/*procedure*
	*Connector*, untuk menyatakan sambungan dari satu proses ke proses yang lain di dalam halaman yang sama.
	*Offline Connector*, untuk menyatakan sambungan dari satu proses ke proses yang lain pada halaman yang berbeda.

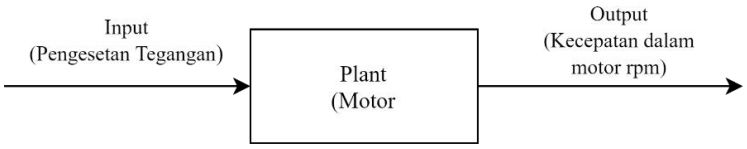
Sumber: Yuniansyah, 2020

2.3 Sistem Kontrol

Sistem kendali atau sistem kontrol (*control system*) adalah suatu alat (kumpulan alat) untuk mengendalikan, memerintah, dan mengatur keadaan dari suatu alat (kumpulan alat) untuk mengendalikan, memerintah dan mengatur keadaan dari suatu sistem (Kontributor Wikipedia, 2008). Istilah sistem kendali ini dapat dipraktikkan secara manual untuk mengendalikan setir mobil pada saat kita mengendarai/menyetir mobil kita misalnya, dengan menggunakan prinsip umpan balik. Dalam sistem yang otomatis, alat semacam ini sering dipakai untuk peluru kendali sehingga peluru akan mencapai sasaran yang diinginkan (Kontributor Wikipedia, 2008).

2.3.1 Sistem *Loop* Terbuka

Sistem *loop* terbuka biasa dikenal dengan sistem *open loop* memakai peralatan penggerak atau aktuator yang mengatur proses secara langsung tanpa memakai umpan balik (Yudaningtyas, 2017). Harga keluaran dari keluaran sistem ini tidak dapat dibandingkan dengan harga masukannya yang berarti keluaran tidak akan memberikan pengaruh terhadap harga masukan atau variabel yang dikontrol tidak bisa dibandingkan dengan harga yang diharapkan (Yudaningtyas, 2017). Berikut adalah diagram *loop* terbuka:



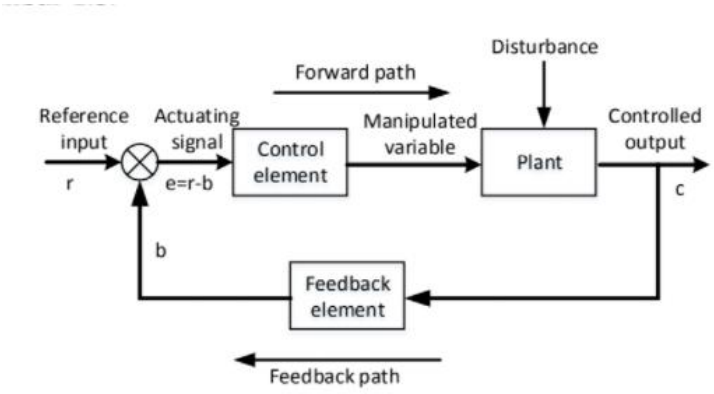
Sumber: Yudaningtyas, 2017

Gambar 2.3 Diagram *Loop* Terbuka

2.3.2 Sistem *Loop* Tertutup

Sistem *loop* tertutup (*closed loop*) memakai pengukuran *output* (keluaran) lalu mengumpan balikan sinyal tersebut agar dibandingkan dengan harga yang diharapkan (*input* atau referensi) (Yudaningtyas, 2017). Hal tersebut berarti keluaran dari sistem dapat memberikan pengaruh pada besaran referensi atau besaran yang akan dikontrol dan dapat dibandingkan terhadap harga yang diharapkan (Yudaningtyas, 2017).

Berikut adalah diagram balok dari sistem *loop* tertutup :



Sumber: Yudaningtyas, 2017

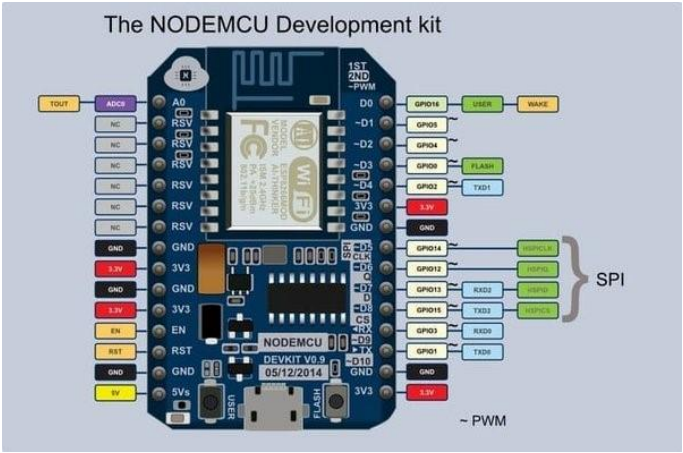
Gambar 2.4 Diagram Balok Sistem *Loop* Tertutup.

2.4 Komponen Utama

Pada bagian ini akan diuraikan komponen-komponen utama yang digunakan dalam penelitian perancangan sistem ini akan dijelaskan sebagai berikut.

2.4.1 NodeMCU

NodeMCU adalah lingkungan pengembangan perangkat lunak dan perangkat keras open source yang dibangun di sekitar *System-on-a-Chip* yang sangat murah yang disebut ESP8266. ESP8266, dirancang dan diproduksi oleh Espressif Systems, berisi semua elemen penting dari komputer modern: CPU, RAM, jaringan, dan bahkan sistem operasi modern dan SDK. Itu menjadikannya pilihan yang sangat baik untuk semua jenis proyek IoT (Dinata, 2017). Namun, sebagai sebuah chip, ESP8266 juga sulit diakses dan digunakan. Anda harus menyolder kabel, dengan voltase analog yang sesuai, ke PIN-nya untuk tugas paling sederhana seperti menyalakannya atau mengirim penekanan tombol ke "komputer" pada chip (Dinata, 2017). Berikut adalah gambar dan pin diagram yang akan bisa digunakan dalam NodeMCU.



Sumber:

http://reslab.sk.fti.unand.ac.id/index.php?option=com_k2&view=item&id=246:nodemcu

Gambar 2.5 NodeMCU dan Pin Diagramnya

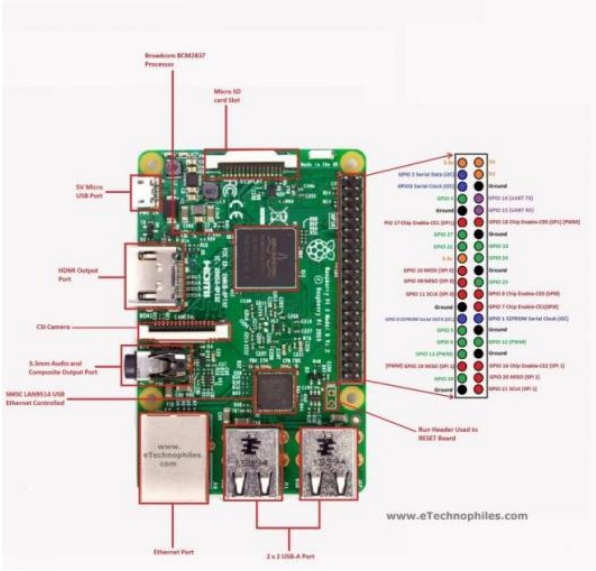
2.4.2 *Raspberry Pi*

Raspberry Pi, sering disingkat dengan nama *Raspi*, adalah komputer board tunggal (single-board circuit; SBC) yang seukuran dengan kartu kredit yang dapat digunakan untuk menjalankan program perkantoran, permainan komputer, dan sebagai pemutar media hingga video beresolusi tinggi. *Raspberry Pi* dikembangkan oleh yayasan nirlaba, *Raspberry Pi Foundation*, yang digawangi sejumlah pengembang dan ahli komputer dari Universitas Cambridge, Inggris (Kontributor Wikipedia, 2019).

Ide dibalik *Raspberry Pi* diawali dari keinginan untuk mencetak pemrogram generasi baru. Seperti disebutkan dalam situs resmi *Raspberry Pi Foundation*, waktu itu Eben Upton, Rob Mullins, Jack Lang, dan Alan Mycroft,

dari Laboratorium Komputer Universitas Cambridge memiliki kekhawatiran melihat kian turunnya keahlian dan jumlah siswa yang hendak belajar ilmu komputer. Mereka lantas mendirikan yayasan *Raspberry Pi* bersama dengan Pete Lomas dan David Braben pada 2009. Tiga tahun kemudian, *Raspberry Pi* Model B memasuki produksi massal. Dalam peluncuran pertamanya pada akhir Februari 2012 dalam beberapa jam saja sudah terjual 100.000 unit. Pada bulan Februari 2016, *Raspberry Pi* Foundation mengumumkan bahwa mereka telah menjual 8 juta perangkat *Raspi*, sehingga menjadikannya sebagai perangkat paling laris di Inggris (Kontributor Wikipedia, 2019).

Menurut Dinata, untuk memenuhi kebutuhan di banyak bidang, maka pembuat *Raspberry Pi* memproduksinya dalam berbagai macam model untuk menyesuaikannya dengan kebutuhan penggunanya. Jika kita mencari *Raspberry* di Internet, kita akan menemui model A dan model B. Model 1A dan 1B adalah model yang pertama kali dikembangkan. Awalnya mereka merencanakan akan ada model 2A, 2B, 3A, 3B, dan seterusnya. Tetapi, tidak pernah dibuat model A selanjutnya, hanya model 2B dan 3B. Model A digunakan untuk aplikasi yang membutuhkan daya kecil dengan memori yang kecil juga. Ada kemungkinan jika *Raspberry* akan mengembangkan model A yang baru lagi. Model B adalah penyempurnaan dari Model A baik dan sisi kemampuan komputasi maupun kelengkapannya. Model B sudah dikembangkan menjadi *Raspberry* Pi 2B, Pi 3B, dan Pi Zero. Agar menyingkat penyebutan, dalam pembahasan selanjutnya, atribut "B" dihilangkan. Berikut adalah gambar *Raspberry Pi* 3 dan pin diagramnya:



Sumber: https://www.etechnophiles.com/raspberry-pi-3-gpio-pinout-pin-diagram-and-specs-in-detail-model-b/

Gambar 2.6 Raspberry Pi 3 dan Pin diagramnya

2.4.3 Sensor MQ-135

MQ-135 udara adalah perangkat yang mendeteksi dan menyaring polusi udara di area terdekat. Masih bisa digunakan untuk lingkungan luar dan dalam. Sensor ini masih dapat diselesaikan di rumah atau diterima dari beberapa barang jadi. Sensor sangat eksklusif di masa lalu, tetapi dengan perkembangan teknis, populasi lebih terjangkau dan tumbuh lebih banyak di semua sensor ini. Sensor ini dapat membantu melayani banyak tujuan dan membantu fokus pada masalah lingkungan di luar lingkaran mata manusia. Konten MQ-135 adalah Sn02, dan ini adalah bahan khusus. Sensor gas MQ-135 dapat mendeteksi amonia nitrogen, oksigen, alkohol, senyawa vertikal, belerang, dan asap 1171 (Sharma dkk., 2020). Berikut adalah gambar dari sensor MQ-135:

Sumber: https://www.tokopedia.com/pro-instrumen/sensor-mq-135-mq135-air-quality-hazardous-gas-sensor-kualitas-udara

Gambar 2.7 Sensor MQ-135

Menurut jurnal yang berjudul "*Low Cost IoT Based Air Quality Monitoring Setup Using Arduino and MQ Series Sensors With Dataset Analysis*" yang dikemukakan oleh KinneraBharath Kumar Salt, Subhaditya Mulcherjee, dan Parveen Sultana H menerangkan bahwa indeks kualitas udara yang bisa dipakai oleh sensor MQ-135 sebagai tabel berikut.

Tabel 2.4 Indeks Kualitas Udara

Rentang (PPM)	Status
0 – 50	Bagus
51 – 100	Sedang
100 – 150	Tidak sehat untuk orang-orang sensitif
151 – 200	Tidak Sehat
200 – 300	Sangat tidak sehat
301 – 500	Berbahaya

Sumber : Kumar Sai dkk. (2019)

2.4.4 *Push-button*

Push-button (tombol tekan) atau tombol sederhana adalah mekanisme sakelar sederhana untuk mengontrol beberapa aspek mesin atau proses. Kancing biasanya terbuat dari bahan keras yaitu plastik atau logam (Kontributor Wikipedia, 2021). Permukaannya biasanya datar atau berbentuk untuk menampung jari atau tangan manusia, sehingga mudah ditekan atau didorong. Tombol paling sering adalah sakelar yang bias, meskipun banyak tombol yang tidak bias (karena sifat fisiknya) masih memerlukan pegas untuk kembali ke keadaan tidak ditekan (Kontributor Wikipedia, 2021). Berikut adalah gambar dari wujud *push-button*.

Sumber: https://www.tokopedia.com/ctm/push-button-kecil

Gambar 2.8 Push Button

2.4.5 *Buzzer*

Buzzer atau pager adalah perangkat sinyal audio, yang mungkin mekanis, elektromekanis, atau piezoelektrik (Kontributor Wikipedia, 2021). Penggunaan

khas *buzzer* dan *pager* termasuk perangkat alarm, *timer*, dan konfirmasi *input* pengguna seperti klik mouse atau *keystroke* (Kontributor Wikipedia, 2021). Berikut adalah gambar dari wujud *buzzer*:

Sumber: https://timur.ilearning.me/2019/04/30/buzzer/

Gambar 2.9 *Buzzer*

2.4.6 Situs Web

Situs web adalah sekumpulan halaman web yang saling berhubungan yang umumnya berada pada server yang sama berisikan kumpulan informasi yang disediakan secara perorangan, kelompok, atau organisasi. Sebuah situs web biasanya ditempatkan setidaknya pada sebuah server web yang dapat diakses melalui jaringan seperti Internet, ataupun jaringan area lokal (LAN) melalui alamat Internet yang dikenali sebagai URL (Kontributor, 2021). Berikut adalah contoh situs web yang diambil dari salah satu halaman wikipedia:

Sumber: https://id.wikipedia.org/wiki/Situs_web#/media/Berkas:Situs_Web.PNG

Gambar 2.10 Contoh Halaman Situs Web

Dalam Wikipedia juga dijelaskan macam-macam web terdiri dari dua macam yaitu sebagai berikut.

1. Situs Web statis

Situs web dinamis merupakan situs web yang secara spesifik didesain agar isi yang terdapat dalam situs tersebut dapat diperbarui secara berkala dengan mudah. Sesuai dengan namanya, isi yang terkandung dalam situs web ini umumnya akan berubah setelah melewati satu periode tertentu. Situs berita adalah salah satu contoh jenis situs yang umumnya mengimplementasikan situs web dinamis.

2. Situs Web dinamis

Situs web statis merupakan situs web yang memiliki isi tidak dimaksudkan untuk diperbarui secara berkala sehingga pengaturan ataupun perubahan isi atas situs web tersebut dilakukan secara manual.

2.4.7 *Database*

Database atau basis data adalah kumpulan data yang dikelola sedemikian rupa berdasarkan ketentuan tertentu yang saling berhubungan sehingga mudah dalam pengelolaannya. Melalui pengelolaan tersebut pengguna dapat memperoleh kemudahan dalam mencari informasi, menyimpan informasi dan membuang informasi (Dicoding Intern, 2020). Dicoding Intern juga menjelaskan macam-macam *database* sebagai berikut:

a) Operational Database

Operational Database atau biasa disebut dengan *database* OLTP (*On Line Transaction Processing*), berguna untuk mengelola data yang dinamis secara langsung atau real-time. Jenis ini memungkinkan para pengguna dapat melakukan, melihat, dan memodifikasi data. Modifikasi tersebut bisa berupa mengubah, menambah, menghapus data secara langsung melalui perangkat keras yang digunakan.

b) Warehouse Database

Database Warehouse adalah sistem basis data yang biasa digunakan untuk pelaporan dan analisis data. Sistem ini dianggap sebagai komponen inti dari

business intelligence. *Database Warehouse* merupakan repositori sentral data yang terpadu dari satu atau lebih sumber yang berbeda. *Database* tersebut juga menyimpan data terkini dan historis dengan satu tempat yang digunakan untuk membuat laporan analisis.

2.5 Komponen Pendukung

Pada bagian ini akan diuraikan komponen-komponen pendukung yang digunakan dalam penelitian perancangan sistem ini akan dijelaskan sebagai berikut.

2.5.1 LED

Light-emitting diode (LED) adalah sumber cahaya semikonduktor yang memancarkan cahaya ketika arus mengalir melaluinya. Elektron dalam semikonduktor bergabung kembali dengan lubang elektron, melepaskan energi dalam bentuk foton. Warna cahaya (sesuai dengan energi foton) ditentukan oleh energi yang dibutuhkan elektron untuk melintasi celah pita semikonduktor. Cahaya putih diperoleh dengan menggunakan beberapa semikonduktor atau lapisan fosfor pemancar cahaya pada perangkat semikonduktor (Kontributor Wikipedia, 2021). Berikut adalah gambar dari bentuk LED.

Sumber:

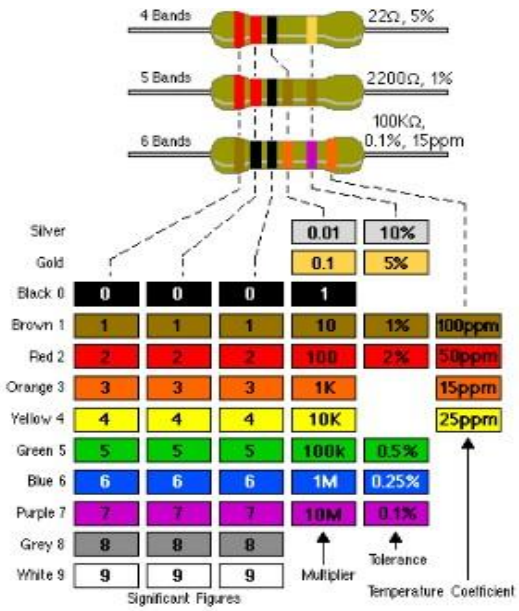
https://id.wikipedia.org/wiki/Diode_pancaran_cahaya#/media/Berkas:Ledmrp.jpg

Gambar 2.11 LED

2.5.2 Resistor

Resistor adalah komponen elektronika yang berfungsi untuk menghambat atau membatasi aliran listrik yang mengalir dalam suatu rangkaian elektronika. Resistor termasuk komponen pasif pada rangkaian elektronika. Sebagaimana fungsi resistor yang sesuai namanya bersifat resistif dan termasuk salah satu komponen elektronika dalam kategori komponen pasif. Satuan atau nilai resistansi suatu resistor di sebut Ohm dan dilambangkan dengan simbol Omega (Ω). Hukum Ohm menyatakan bahwa resistansi berbanding terbalik dengan jumlah arus yang mengalir melaluinya. Selain nilai resistansi (Ohm), resistor juga memiliki nilai yang lain seperti nilai toleransi dan kapasitas daya yang mampu dilewatkannya. Semua nilai yang berkaitan dengan resistor tersebut penting untuk diketahui dalam perancangan suatu rangkaian elektronika oleh karena itu pabrikan resistor selalu mencantumkan dalam kemasan resistor tersebut (Yulia Basri & Irfan, 2018).

Berikut ini akan dijelaskan cara mengetahui nilai resistor tetap berdasarkan kode warna. Resistor ini mempunyai bentuk seperti tabung dengan dua kaki di kiri dan kanan. Pada badannya terdapat lingkaran membentuk cincin kode warna, kode ini untuk mengetahui besar resistansi tanpa harus mengukur besarnya dengan ohm meter. Kode warna tersebut adalah standar manufaktur yang dikeluarkan oleh EIA (*Electronic Industries Association*) (Yulia Basri & Irfan, 2018).



Sumber : Yulia Basri & Irfan, 2018

Gambar 2.12 Ketentuan Warna Resistor

Besaran resistansi suatu resistor dibaca dari posisi cincin yang paling depan ke arah cincin toleransi. Biasanya posisi cincin toleransi ini berada pada badan resistor yang paling pojok atau juga dengan lebar yang lebih menonjol, sedangkan posisi cincin yang pertama agak sedikit ke dalam. Dengan demikian

pemakai sudah langsung mengetahui berapa toleransi dari resistor tersebut (Yulia Basri & Irfan, 2018).

2.5.3 Kapasitor

Kapasitor (*Capacitor*) atau disebut juga dengan Kondensator (*Condensator*) adalah komponen elektronika pasif yang dapat menyimpan muatan listrik dalam waktu sementara dengan satuan kapasitansinya adalah Farad. Satuan kapasitor tersebut diambil dari nama penemunya yaitu Michael Faraday (1791 ~ 1867) yang berasal dari Inggris (Yulia Basri & Irfan, 2018).

Sumber: https://upload.wikimedia.org/wikipedia/commons/3/3f/Kondensator-Al-Elko-Wiki-07-02-11.jpg

Gambar 2.13 Bentuk dari Kapasitor

Konversi Satuan Farad adalah sebagai berikut yang dijelaskan oleh Yulia Basri. Satuan-satuan yang sering dipakai untuk kapasitor adalah : 1 Farad = 1.000.000 µF (mikro Farad) = 10^6µF (mikro Farad)

1 Farad = 1.000.000.000 nF (nano Farad)= 10^9 nF (nano Farad)

1 Farad = 1.000.000.000.000 pF (piko Farad) = 10^{12} pF (piko Farad)

1 μFarad = 1.000 nF (nano Farad) = 10^3 nF (nano Farad)

1 μFarad = 1.000.000 pF (piko Farad) = 10^6 pF (piko Farad)

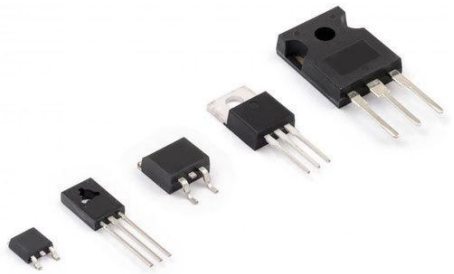
1 nFarad = 1.000 pF (piko Farad) = 10^3 nF (nano Farad)

Kapasitor merupakan komponen elektronika yang terdiri dari 2 pelat konduktor yang pada umumnya adalah terbuat dari logam dan sebuah isolator di antara pelat tersebut sebagai pemisah. Isolator tersebut disebut juga dengan di elektronika (Yulia Basri & Irfan, 2018).

Bahan dielektrik tersebut dapat mempengaruhi nilai dari kapasitansi kapasitor tersebut. Adapun bahan dielektrik yang paling sering dipakai adalah keramik, kertas, udara, metal film dan lain-lain. Kapasitor sering juga disebut sebagai kondensator. Kapasitor memiliki berbagai macam bentuk dan ukuran, tergantung dari kapasitas, tegangan kerja, dan lain sebagainya (Yulia Basri & Irfan, 2018).

2.5.4 Transistor

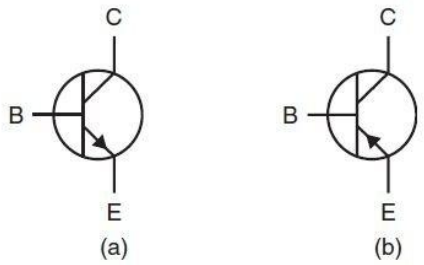
Transistor adalah alat semikonduktor yang biasanya dipakai untuk penguat, pemutus dan penyambung (swtiching), stabilisasi tegangan, modulasi sinyal atau sebagai fungsi lainnya. Transistor memiliki tiga terminal, yaitu basis (B), emiter (E), dan kolektor (C). Transistor dibagi menjadi dua jenis, yaitu transistor bipolar dan uni polar (Darmawan & Suprihatin, 2020).



Sumber: Darmawan & Suprihatin, 2020

Gambar 2.14 Transistor

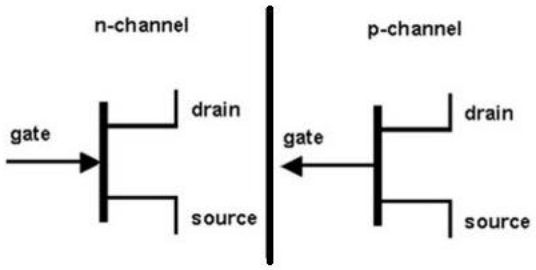
Transistor bipolar adalah transistor yang memiliki dua penyambung kutub. Terdapat dua jenis transistor bipolar, yaitu NPN BJT transistor dan PNP BJT transistor (Darmawan & Suprihatin, 2020). Transistor bipolar dilambangkan seperti di bawah berikut ini.



Sumber: https://www.researchgate.net/profile/Mohamed-Obaia/publication/320987534/figure/fig3/AS:559144662990848@1510321884444/a-npn-bipolar-transistor-b-pnp-bipolar-transistor.png

Gambar 2.15 Simbol Kapasitor Bipolar

Transistor Uni polar adalah transistor yang hanya memiliki satu buat penyambung kutub. Transistor Uni polar terbuat dari bahas semikonduktor dan mempunyai kaki yang disebut dengan gerbang (gate), sumber (source), dan cerat (drain). Transistor Uni polar terbagi jadi dua jenis yaitu FET (Field Effect Transistor) dan MOSFET (metal-oxide semiconductor FET) (Darmawan & Suprihatin, 2020).



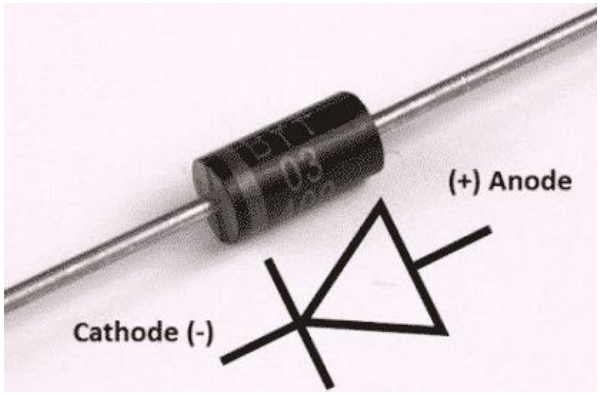
Sumber: https://1.bp.blogspot.com/-aYoAfIPvrwc/YRZARk9eeTI/AAAAAAAADoo/6G4iAOuEmX4CWGv919jUMB3C0HOE78dPgCLcBGAsYHQ/s540/perbedaan-transistor-bjt-dan-FET.png

Gambar 2.16 Simbol Kapasitor Uni Polar

2.5.5 Diode

Sebuah diode dibuat dari silikon, Silikon adalah bahan yang tidak bersifat sebagai penghantar (konduktor), tetapi tidak pula sebagai penyekat (isolator). Silikon adalah bahan semikonduktor. Hal ini berarti bahwa sifat-sifat silikon berbeda dengan bahan-bahan konduktor biasa, seperti tembaga. Diode dikemas di dalam sebuah kapsul kecil yang terbuat dari kaca atau plastik. Kemasan ini

memiliki dua kawat terminal, yaitu anode dan katode (Darmawan & Suprihatin, 2020).



Sumber: Darmawan & Suprihatin, 2020

Gambar 2.17 Diode

Sebuah diode hanya akan menghantarkan arus listrik jika diberi bias maju. Ketika kaki anode disambung dengan kutub positif baterai, kita dapat mengatakan bahwa diode diberikan bias maju. Namun, sebaliknya, jika katode disambungkan ke kutub positif, dapat dikatakan bahwa diode diberikan bias mundur. Sebuah diode yang diberi bias maju memiliki penurunan tegangan sekitar 0,7 V (Darmawan & Suprihatin, 2020).

2.6 *Framework Arduino*

Arduino Framework adalah *wiring-based Framework* memungkinkan penulisan perangkat lunak lintas platform untuk mengontrol perangkat yang terpasang pada berbagai *board* Arduino untuk membuat semua jenis programmer kreatif, objek interaktif, ruang, atau pengalaman fisik (PlatformIO, 2022). Menurut definisi di atas bahwa tidak hanya untuk framework Arduino ini lintas

platform, framework ini mendukung 166 *board* mikrokontroler yang tersedia pada PlatformIO.

2.7 *Nodejs*

Node.js adalah *runtime environment* untuk JavaScript yang bersifat *open-source* dan *cross-platform*. Dengan Node.js kita dapat menjalankan kode JavaScript di mana pun, tidak hanya terbatas pada lingkungan browser (Dicoding Indonesia, 2021). Node.js menjalankan V8 JavaScript engine (yang juga merupakan inti dari Google Chrome) di luar browser. Ini memungkinkan Node.js memiliki performa yang tinggi (Dicoding Indonesia, 2021). Banyaknya *library/module* nodejs akan membantu menyederhanakan pengembangan aplikasi.

2.8 Bahasa Pemrograman Arduino

Bahasa pemrograman Arduino yang dipakai adalah bahasa C/C++ yang merupakan bahasa yang lazim dipakai sejak awal komputer diciptakan dan sangat berperan dalam perkembangan software. Karena Arduino menggunakan bahasa C yang multi-platfrom, software Arduino pin bisa dijalankan pada semua mesin operasi yang umum, misalnya windows, Linux dan MacOS.

2.8.1 Struktur

Di dalam library Arduino terdapat fungsi yang disediakan oleh Arduino yaitu fungsi loop dan setup. Fungsi ini dijelaskan dalam dokumentasi Arduino sebagai berikut.

1. Fungsi setup()

   Fungsi setup() dipanggil saat sketsa dimulai. Digunakan untuk menginisialisasi variabel, mode pin, mulai menggunakan perpustakaan, dll. Fungsi setup() hanya akan berjalan sekali, setelah setiap *powerup* atau reset *board* Arduino.

2. Fungsi loop()

   Setelah membuat fungsi setup(), yang menginisialisasi dan menyetel nilai awal, fungsi loop() melakukan persis seperti yang disarankan namanya, dan mengulang secara berurutan, memungkinkan program Anda untuk berubah dan merespon. Gunakan untuk secara aktif mengontrol *board* Arduino.

2.8.2 Syntax Program

Berikut ini adalah elemen bahasa C/C++ yang dibutuhkan untuk format penulisan :

1. // (Komentar Satu Baris)

   Teks apa pun antara // dan akhir baris diabaikan oleh *compiler* (tidak akan dieksekusi).

2. /* */ (Komentar Banyak Baris)

   Setiap teks antara /* dan */ akan diabaikan oleh *compiler*.

3. {} (Curly Bracket)

   Kurung Kurawal digunakan untuk menandai blok program. Kode apa pun di dalam tanda kurung kurawalnya {} akan dieksekusi.

4. ; (Semicolon)

Setiap instruksi dalam C/C++ diakhiri dengan titik koma.

2.8.3 Variabel

Variabel adalah representasi alamat memori yang akan disimpan sebuah nilai atau data. Setiap variabel mempunyai tipe data yang terdapat pada Tabel 2.5.

Tabel 2.5 Tipe Data Pada C/C++

Tipe Data	Ukuran	Deskripsi
int	4 bytes	Menyimpan bilangan bulat, tanpa desimal
float	4 bytes	Menyimpan bilangan pecahan, berisi satu atau lebih desimal. Cukup untuk menyimpan 7 digit desimal
double	8 bytes	Menyimpan bilangan pecahan, berisi satu atau lebih desimal. Cukup untuk menyimpan 15 digit desimal
boolean	1 byte	Menyimpan nilai benar atau salah
char	1 byte	Menyimpan satu karakter/huruf/angka, atau nilai ASCII

Sumber : W3School, 2022

2.8.6 Operator

Terdapat dua jenis operator pada C/C++, yaitu operator matematik yang digunakan untuk operasi matematik dan operator pembanding yang biasanya

digunakan pada operasi percabangan untuk membandingkan dua nilai atau variabel. Berikut Operator C++ yang dijelaskan Programiz.

1. Operator Aritmatika

Operator aritmatika digunakan untuk melakukan operasi aritmatika pada variabel dan data. Sebagai contoh, a + b. Di sini, + operator digunakan untuk menambahkan dua variabel sebuah dan B. Demikian pula ada berbagai operator aritmatika lainnya di C/C++.

Tabel 2.6 Operator Aritmatika

Operator	Operasi
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Divisi
%	Operasi Modulo (Sisa setelah pembagian)

Sumber: Programiz, 2021

2. Operator Perbandingan

Operator perbandingan digunakan untuk memeriksa hubungan antara dua operan. Sebagai contoh, a > b. Di sini, > adalah operator perbandingan. Ini memeriksa apakah sebuah lebih besar dari B atau tidak. Jika relasinya benar, ia mengembalikan 1 sedangkan jika hubungan itu salah , ia mengembalikan 0.

Tabel 2.7 Operator Perbandingan

Operator	Ketengan
==	Operator sama dengan
!=	Tidak sama dengan
>	Lebih besar dari
<	Kurang dari
>=	Lebih dari atau sama dengan
<=	Kurang dari atau Sama Dengan
&&	ekspresi1 && ekspresi2
||	ekspresi1 || ekspresi2
!	! ekspresi

Sumber: Programiz, 2022

2.8.6 Struktur Kondisi

Struktur kondisi biasa digunakan untuk menyeleksi apa yang akan dieksekusi selanjutnya oleh program berdasarkan kondisi yang terpenuhi pada ekspresi yang diberikan. Gunakan pernyataan if untuk menentukan blok kode

C/C++ yang akan dieksekusi jika suatu kondisi benar. Berikut *syntax* pengkondisian:

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

Berikut struktur pengkondisian yang dijelaskan oleh (W3School, 2022).

a. C++ mendukung kondisi logis yang biasa dari matematika:
   - Kurang dari: a < b.
   - Kurang dari atau sama dengan: a <= b.
   - Lebih besar dari: a > b.
   - Lebih besar dari atau sama dengan: a >= b.
   - Sama dengan: a == b.
   - Tidak Sama dengan: a != b.

b. C++ memiliki pernyataan kondisional berikut:
   - Gunakan *if* untuk menentukan blok kode yang akan dieksekusi, jika kondisi yang ditentukan benar.
   - Gunakan *else* untuk menentukan blok kode yang akan dieksekusi, jika kondisi yang sama salah.
   - Gunakan *else if* untuk menentukan kondisi baru yang akan diuji, jika kondisi pertama salah.
   - Gunakan *switch* untuk menentukan banyak blok kode alternatif yang akan dieksekusi.

2.8.6 Perulangan For

Perulangan adalah instruksi-instruksi yang eksekusi secara berulang-ulang. Macam-macam perulangan dijelaskan oleh sesuai persyaratan yang ditetapkan. Contoh *syntaxnya* sebagai berikut:

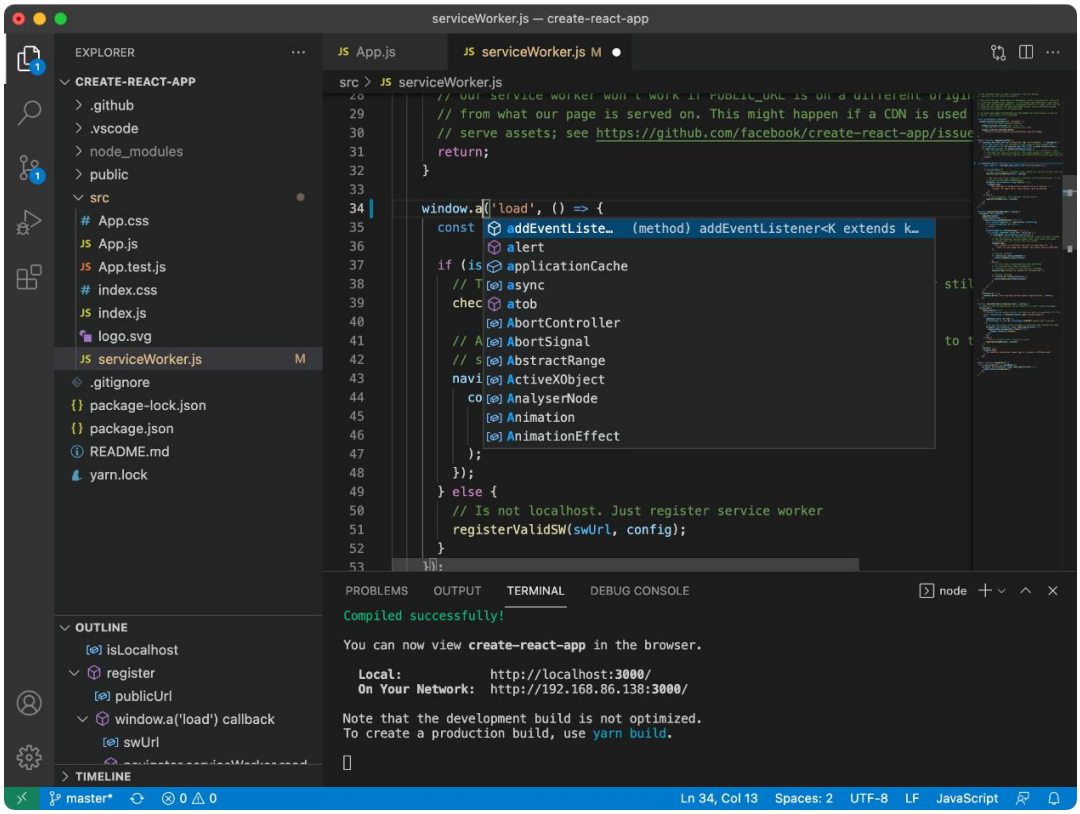
for (*statement 1*; *statement 2*; *statement 3*) {

// code block to be executed

}

2.8 Visual Studio Code

Visual Studio Code adalah distribusi repositori dengan kustomisasi khusus Microsoft yang dirilis di bawah lisensi produk Microsoft tradisional (Microsoft, 2021). Visual Studio Code menggabungkan kesederhanaan editor kode dengan apa yang dibutuhkan pengembang untuk siklus *edit-build-debug* inti mereka (Microsoft, 2021). Ini memberikan pengeditan kode yang komprehensif, navigasi, dan dukungan pemahaman bersama dengan *debugging* ringan, model ekstensibilitas yang kaya, dan integrasi ringan dengan alat yang ada (Microsoft, 2021). Anda dapat mengunduhnya untuk Windows, MacOS, dan Linux di situs web Visual Studio Code (Microsoft, 2021). Berikut adalah gambar dari tampilan editor kode Visual Studio Code:



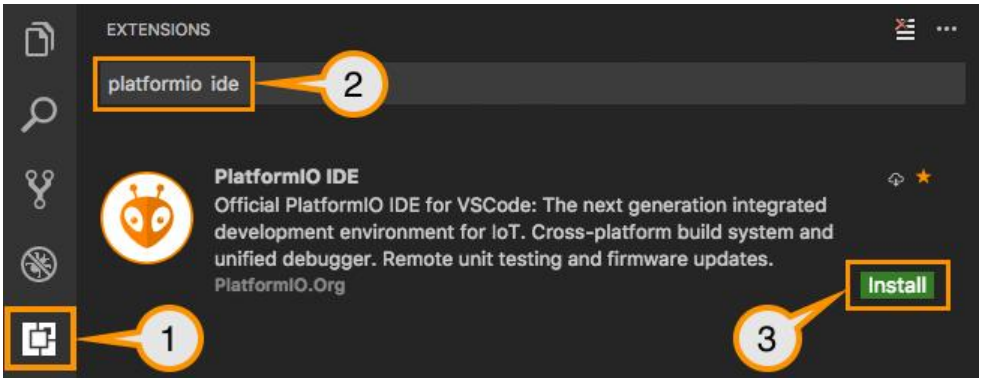
Sumber: https://user-images.githubusercontent.com/35271042/118224532-3842c400-b438-11eb-923d-a5f66fa6785a.png

Gambar 2.18 Tampilan Visual Studio Code

2.9 Platform IO IDE di Visual Studio Code

PlatformIO adalah platform kolaboratif profesional untuk pengembangan tertanam (PlatformIO, 2017). PlatformIO menyediakan berbagai platform seperti: Atmel AVR, Atmel SAM, Espressif 32, Espressif 8266, Freescale Kinetis, Infineon XMC, Intel ARC32, Intel MCS-51 (8051), Kendryte K210, Lattice iCE40, Maxim 32, Microchip PIC32, Nordic nRF51, Nordic nRF52, NXP LPC, RISC-V, Samsung ARTIK, Silicon Labs EFM32, ST STM32, ST STM8, Teensy, TI MSP430, TI Tiva, WIZNet W7500.

Visual Studio Code adalah editor kode sumber yang ringan namun kuat yang berjalan di desktop Anda dan tersedia untuk Windows, MacOS, dan Linux (PlatformIO, 2020). Muncul dengan dukungan bawaan untuk JavaScript, TypeScript dan Node.js dan memiliki ekosistem ekstensi yang kaya untuk bahasa lain (seperti C++, C#, Python, PHP, Go) dan runtime (seperti .NET dan Unity) (Platform IO, 2020). Cara penginstalan PlatformIO IDE di VSCode cukup mudah dengan cara membuka Market Extension, lalu cari "platform ide". Lalu install.

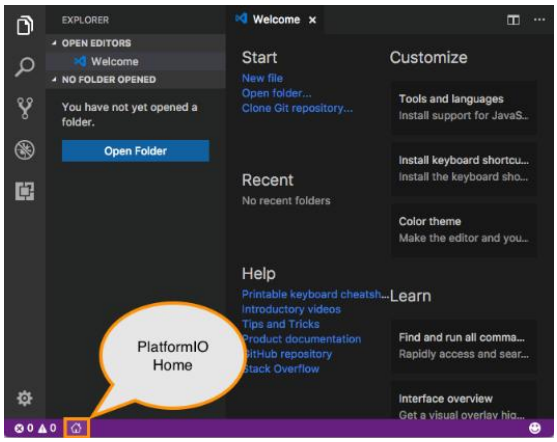


Sumber: *https://docs.platformio.org/en/latest/_images/platformio-ide-vscode-pkg-installer.png*

Gambar 2.19 Pencarian Extension PlatformIO IDE

Adapun cara pembuatan *project* Platform IO sebagai berikut yang dijelasakan Platform IO pada web resminya.

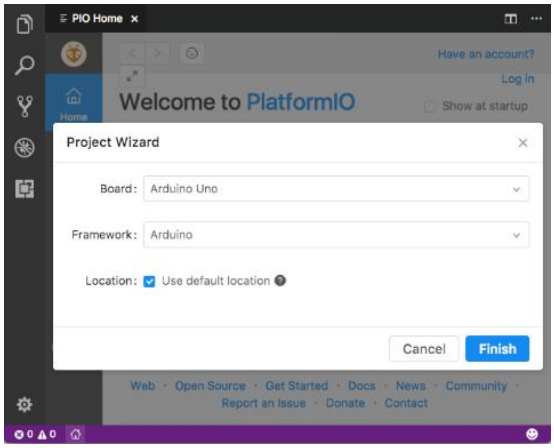
1. Klik tombol “PlatformIO Home” di bagian bawah PlatformIO Toolbar.



Sumber: https://docs.platformio.org/en/latest/_images/platformio-ide-vscode-welcome.png

Gambar 2.20 Tombol *PlaftormIO Home*

2. Klik "New Project", pilih *board* dan buat Proyek PlatformIO baru.



Sumber: https://docs.platformio.org/en/latest/_images/platformio-ide-vscode-new-project.png

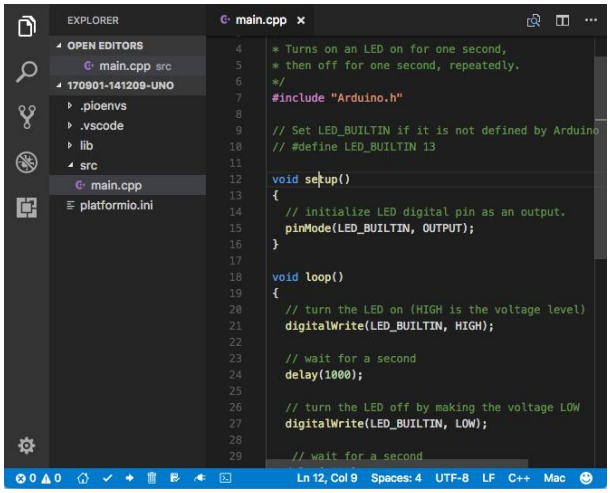
Gambar 2.21 Formulir Pembuatan Project PlatformIO IDE

3. Buka file main.cpp dari folder src dan ganti isinya dengan yang berikut:

```
/**
 * Blink
 *
 * Turns on an LED on for one second,
 * then off for one second, repeatedly.
 */
#include "Arduino.h"

// Set LED_BUILTIN if it is not defined by Arduino framework
// #define LED_BUILTIN 13

void setup()
{
  // initialize LED digital pin as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  // turn the LED on (HIGH is the voltage level)
  digitalWrite(LED_BUILTIN, HIGH);

  // wait for a second
  delay(1000);

  // turn the LED off by making the voltage LOW
  digitalWrite(LED_BUILTIN, LOW);

   // wait for a second
  delay(1000);
}
```

Sumber: Platform IO, 2020

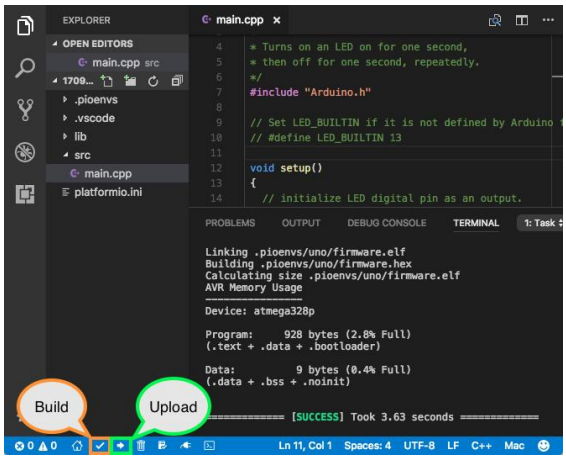
Gambar 2.22 Contoh Program Arduino di PlatformIO IDE



Sumber: https://docs.platformio.org/en/latest/_images/platformio-ide-vscode-blink-project.png

Gambar 2.23 Pencarian *Extension* PlatformIO IDE

4. Bangun proyek Anda dengan hotkey ctrl+alt+b (lihat semua Ikatan Kunci di bagian "*User Guide*" di bawah) atau menggunakan tombol "*Build*" di *PlatformIO Toolbar*.



Sumber: https://docs.platformio.org/en/latest/_images/platformio-ide-vscode-build-project.png

Gambar 2.24 Tombol *Build* dan *Upload*

Toolbar PlatformIO IDE terletak di VSCode Status Bar (pojok kiri) dan berisi tombol akses cepat untuk perintah populer. Setiap tombol berisi petunjuk (tunda mouse di atasnya).

Sumber: https://docs.platformio.org/en/latest/_images/platformio-ide-vscode-toolbar.png

Gambar 2.25 *Toolbar* PlatformIO IDE

Keterangan :

1. PlatformIO Home
2. PlatformIO: Build
3. PlatformIO: Upload
4. PlatformIO: Clean
5. Serial Port Monitor
6. PlatformIO Core (CLI)
7. Project environment switcher (jika tersedia lebih dari satu lingkungan). Lihat Bagian [env] dari “platformio.ini” (File Konfigurasi Proyek) .

2.10 Interoperabilitas dalam *Internet of Things*

Interoperabilitas adalah karakteristik dari suatu produk atau sistem, antarmuka sepenuhnya dipahami, untuk bekerja dengan produk sistem lain, saat ini atau masa depan, baik dalam implementasi dalam akses, tanpa batasan apa pun. Sementara istilah awalnya didefinisikan sebagai layanan teknologi informasi atau

rekayasa untuk memungkinkan pertukaran informasi, definisi yang lebih luas memperhitungkan faktor-faktor sosial, politik dan organisasi yang mempengaruhi kinerja sistem-ke-sistem (Kontributor Wikipedia, 2021). Menurut kutipan di atas "antarmuka yang sepenuhnya dipahami" dapat juga diartikan protokol apa yang digunakan dalam komunikasi tersebut dalam penelitian ini penulis menggunakan protokol MQTT.

MQTT adalah protokol M2M (*machine to machine*). MQTT didesain sebagai protokol transpor yang sangat ringan dengan konsep publish/subscribe. Sangat bermanfaat untuk membangun aplikasi yang digunakan pada lokasi kurang terjangkau jaringan Internet dan harga *bandwidth* yang mahal. MQTT juga ideal digunakan untuk aplikasi mobile karena ukuran datanya yang kecil dan penggunaan daya yang rendah (Rachmad, 2019).

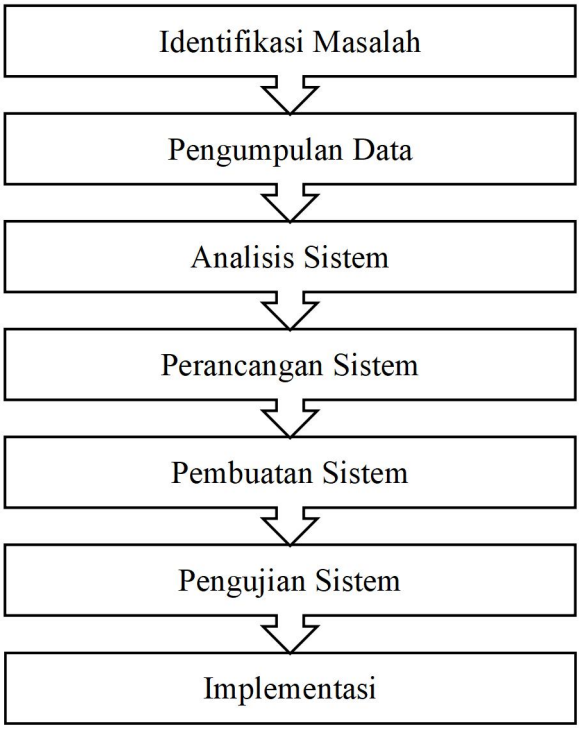
Rachmad juga menjelaskan protokol MQTT menggunakan konsep publish/subscribe yang sangat berbeda dengan konsep protokol yang lain seperti HTTP yang menggunakan paradigma request/response. Titik pusat komunikasi adalah MQTT broker yang bertanggung jawab untuk mengirim semua pesan di antara pengirim dan penerima yang sah. Setiap client yang mengirim pagan ke broker harus menyisipkan informasi nama topik yang dituju. Topik adalah informasi routing untuk broker yang bertugas mengarahkan pesan yang masuk ke dalam broker. Client yang ingin menerima pesan harus melakukan Subscribe ke topik yang dituju dan selanjutnya broker akan mengirimkan semua pesan yang diarahkan pada topik tersebut. Oleh karena itu client tidak perlu saling mengenal, mereka dapat berkomunikasi melalui topik ini.

BAB III

METODE PENELITIAN

3.1 Kerangka Kerja

Kerangka kerja penelitian merupakan tahap awal dari sebuah proses penelitian yang akan kita lakukan yang bertujuan menggambarkan proses dari sistem penelitian agar tidak melenceng dari konsep yang ingin dibuat. Kerangka penelitian yang akan dilakukan berdasarkan metode *Research and Development* seperti Gambar 3.2 di bawah ini.



Gambar 3.2 Kerangka Penelitian

3.2 Uraian Kerangka Kerja Penelitian

Adapun beberapa tahap-tahap yang dilakukan secara terstruktur di dalam penelitian dan dapat dijabarkan sebagai berikut:

3.2.1 Identifikasi Masalah

Identifikasi masalah merupakan tahapan awal dalam perancangan dan pengembangan dari suatu sistem, karena di tahap ini akan diukur dan dievaluasi kinerja sistem yang akan dirancang dan dibuat. Dalam pengidentifikasian masalah, tentu harus mengetahui dan memahami sistem yang akan dibuat terlebih dahulu. Untuk penganalisannya diperlukan data yang berkenaan dengan sistem yang akan dirancang dan dibuat.

Di dalam penelitian ini, permasalahan yang dijadikan diangkat yaitu meningkatnya polusi di udara yang berpotensi menimbulkan masalah kesehatan di masyarakat. Perlu adanya usaha untuk melakukan penekanan terhadap polutan di udara. Diperlukan alat untuk melakukan pemantauan kualitas udara agar dapat membuat keputusan-keputusan yang tepat dan efektif. Sayangnya, stasiun pemantauan kualitas udara ambien otomatis dinilai belum memadai dan pemantauan kualitas udara dilakukan hanya dari penelitian (*grab sampiling*) yang jumlahnya masih sedikit.

3.2.2 Pengumpulan Data

Metode yang dilakukan pada saat melakukan proses penelitian untuk memperoleh data yang akurat dan juga dalam pembuatan alat dan pengerjaan laporan ini peneliti menggunakan metodologi penelitian sebagai berikut:

3.2.2.1 Waktu Penelitian

Penelitian dilakukan dengan memproses data-data yang didapat oleh penulis, lalu pengambilan data dilakukan dari bulan September 2021 sampai Februari 2022.

Tabel 3.1 Waktu Penelitian

Kegiatan	2021																2021					
	September				Oktober				November				Desember				Januari				Februari	
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2
Potensi dan Masalah																						
Pengumpulan Data																						
Desain Produk																						
Validasi Sistem																						
Perancangan Sistem																						
Pembuatan Sistem																						
Uji Coba Tahap 1																						
Uji Coba Tahap 2																						
Implementasi Sistem																						
Pembuatan Laporan																						

3.2.2.2 Metode Penelitian

Adapun metode penelitian yang dilakukan yaitu:

1. Penelitian Laboratorium (*laboratory research*)

   Merupakan metode yang langsung dilakukan dengan menguji konsep-konsep yang ada menggunakan peralatan-peralatan (*tools*) yang sesuai dan digunakan untuk melakukan pengujian-pengujian terhadap modul-modul maupun komponen-komponen yang digunakan pada sistem yang akan dirancang dan dibuat. Adapun *hardware, software* dan *tools* yang digunakan, yaitu :

Tabel 3.2 Hardware dan Software yang Digunakan

Hardware	***Software***
1. Laptop ASUS TUF FX505DT Memory 8GB RAM, SSD 1TB. 2. Mouse Logitech G102 3. Mikrokontroler Arduino NodeMCU sebagai node sensor dan Raspberry Pi 3B+ sebagai server. 4. Sensor MQ-132 5. Push Button	1. Sistem Operasi Windows 10 Home SL 64 bit 2. Microsoft Office 365 3. Visual Studio Code 4. Ubuntu Server 5. NodeJS Runtime 6. Browser

2. Penelitian Perpustakaan (*library research*)

Penelitian ini di lakukan untuk melengkapi perbendaharaan kaidah, konsep, teori dan lain-lain, sehingga menjadi suatu yang mempunyai landasan dan keilmuan yang mantap. Selain itu penelitian ini juga melakukan penelitian pada beberapa material yang sudah ada, baik itu buku-buku, jurnal-jurnal serta yang ada hubungannya dengan penelitian ini maupun catatan yang di lakukan selama perkuliahan. Penelitian ini ditujukan untuk mengumpulkan semua data yang sangat dibutuhkan dalam penelitian ini.

3. Penelitian Lapangan (*field research)*

Field research adalah bentuk penelitian yang bertujuan mengungkapkan makna yang diberikan oleh anggota masyarakat pada perilakunya dan kenyataan sekitar. Metode *field research* digunakan ketika metode survei ataupun eksperimen dirasakan tidak praktis, atau ketika lapangan penelitian masih terbentang dengan demikian luasnya. *Field research* dapat pula diposisikan sebagai pembuka jalan kepada metode survei dan eksperimen.

3.2.3 Analisis Sistem

Tahap analisa merupakan tahap yang paling penting dalam pengembangan sebuah sistem, karena pada tahap inilah nantinya dilakukan evaluasi dari sistem, serta rancangan sistem dan langkah-langkah yang dibutuhkan untuk perancangan sistem dari alat yang akan dibuat sampai mendapatkan analisis yang diharapkan.

Berdasarkan identifikasi masalah di atas, peneliti melakukan analisa data. Hal ini bertujuan agar pemecahan masalah dapat menghasilkan sebuah solusi yang baru sebelum melakukan sebuah perancangan alat.

3.2.4 Perancangan Sistem

Perancangan dalam Penelitian berupa gambaran dan bentuk awal dari alat secara keseluruhan, dengan adanya desain ini maka prinsip kerja dari alat serta komponen-komponen dari sistem yang digunakan akan dapat dilihat dengan jelas sebagai berikut :

1. *Context Diagram*

   Context diagram menggambarkan rancangan keseluruhan, komponen luar harus digambarkan sedemikian rupa, sehingga terlihat data yang mengalir pada input-proses-output.

2. Data Flow Diagram (DFD)

   Data Flow Diagram (DFD) merupakan suatu diagram yang menggambarkan aliran data dalam suatu entitas ke sistem atau sistem ke entitas. DFD juga dapat diartikan sebagai teknik grafis yang menggambarkan aliran data dan transformasi yang digunakan sebagai perjalanan data dari input atau masukan menuju keluaran output

.

3. Blok Diagram

Diagram Blok yang dibuat untuk memetakan proses kerja pada suatu alat, hal ini bertujuan untuk memudahkan seseorang dalam mengenal komponen-komponen elektronika dari alat dan memahami alur kerja di dalamnya.

4. *Flowchart*

Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urutan-urutan prosedur dari suatu program. Flowchart menolong programmer untuk memecahkan masalah dalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif lain dalam pengoperasian. Flowchart biasanya mempermudah penyelesaian suatu masalah, khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut.

5. *Software Design*

Perancangan alat dibuat berupa gambaran umum secara keseluruhan. Dengan perancangan ini, prinsip kerja alat dan komponen sistem yang digunakan dapat terlihat dengan jelas. Perangkat lunak yang digunakan untuk merancang alat ini adalah Google SketchUp.

6. Perakitan Hardware

Setelah desain dibuat, langkah selanjutnya adalah menghubungkan komponen-komponen yang ada agar sistem dapat bekerja sesuai dengan yang diharapkan.

7. *Coding*

Coding merupakan tahapan yang dilakukan setelah semua komponen dihubungkan. NodeMCU yang digunakan akan diprogram sehingga dapat mengontrol *input* dan *output* dari sistem yang sedang dibuat. Raspberry Pi yang digunakan akan diprogram sehingga dapat mengontrol server broker MQTT, *database* dan *web service*,

3.2.5 Pembuatan Sistem

Pembuatan sistem didasarkan pada beberapa pertimbangan yang telah disebutkan pada perancangan sistem sebelumnya seperti *context diagram*, *data flow diagram*, blok diagram, *flowchart*, dan *design*.

3.2.6 Pengujian Sistem

Pada tahapan ini sistem yang dirancang sudah dibuat. Kemudian sistem tersebut diuji dan dianalisis, apakah sistem tersebut sudah berjalan sebagaimana mestinya sesuai dengan rancangan, sebelum sistem diimplementasikan. Pengujian ini dilakukan pada bagian *software*, *hardware* dan sistem secara keseluruhan. Adapun tahan pengujian sistem adalah sebagai berikut.

1. Menyiapkan alat dan komponen pembantu yang digunakan dalam proses pengujian.
2. Menyalakan alat dengan menghubungkannya ke *power supplay*.
3. Menjalankan server yang dibuat di Raspberry Pi
4. Menghubungkan *node sensor* (NodeMCU) ke Raspberry Pi

5. Lalu buka *web browser* dan akses sesuai IP yang terdapat di Rasbperry Pi. Data-data yang dibaca dari *node sensor* dan *database* akan ditampilkan di web.

3.2.7 Implementasi

Tahapan implementasi merupakan tahapan sistem yang telah dibuat diterapkan dan siap dioperasikan. Pada tahapan ini sistem yang dibuat harus sudah selesai. Untuk implementasi alat harus diimplementasikan sesuai dengan tujuan penelitian pada awalnya, maksudnya sistem akan diimplementasikan pada sasaran yang tepat.

Alat ini dapat diimplementasikan di dalam ruangan maupun di luar ruangan. Pemerintah, instansi maupun masyarakat umum dapat menggunakan alat ini sebagai parameter dalam pengambilan keputusan untuk penekanan polusi udara.

BAB IV

ANALISA DAN HASIL

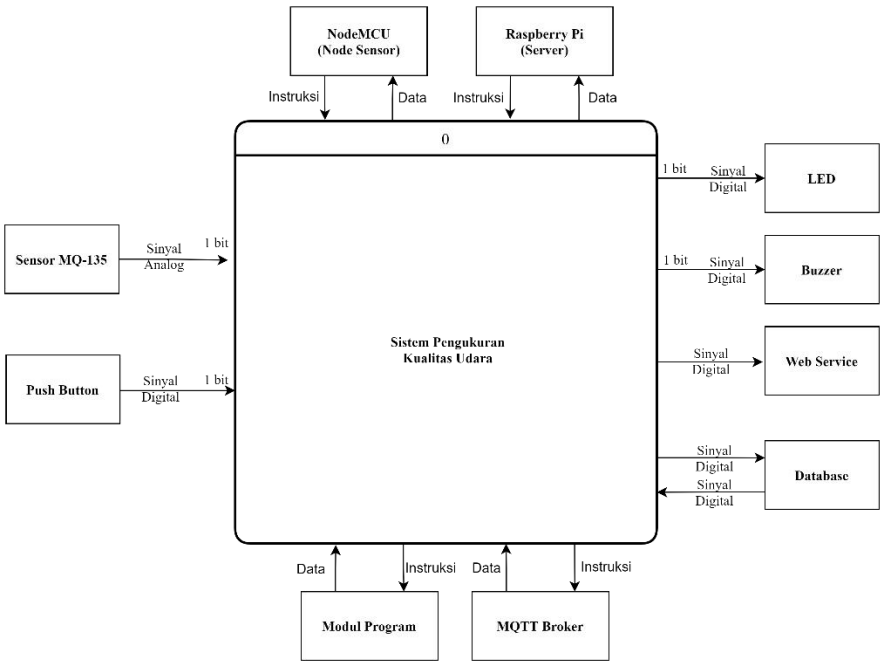
4.1 Desain Sistem Secara Umum

Desain dari sistem yang dibuat merupakan gambaran dari sistem secara keseluruhan. Dengan adanya desain ini maka prinsip kerja dari sistem serta komponen-komponen dari sistem digunakan akan dilihat dengan jelas. Sebagai medianya terdiri dari *context diagram*, *data flow diagram*, blok diagram serta *flowchart*. Untuk lebih jelasnya akan diuraikan sebagai berikut.

4.1.1 *Context Diagram*

Context diagram merupakan pendefinisian terhadap sistem yang akan dirancang yang bersifat menyeluruh. *Context diagram* ini digunakan untuk memudahkan dalam proses penganalisan sistem yang dirancang secara keseluruhan.

Context diagram berfungsi sebagai media, yang terdiri dari suatu proses dan beberapa buah external entity. *Context diagram* yang dimaksud dapat dilihat pada Gambar 4.1.



Gambar 4.1 Context Diagram

Sistem ini berinteraksi dengan beberapa *entity* yaity NodeMCU, Sensor MQ-135, *Push Button*, LED, *Buzzer*, *Raspberry Pi*, dan Modul Program. Selanjutnya *entity-entity* tersebut akan dibahas sebagai berikut.

1. NodeMCU

   NodeMCU ini berfungsi sebagai *node* sensor. Pembacaan kualitas udara akan dilakukan melalui NodeMCU dan data tersebut akan dikirimkan ke *Server Broker Raspberry Pi*.

2. Sensor MQ-135

   Sensor MQ-134 berfungsi sebagai pengukur kualitas udara pada *node server* NodeMCU.

3. *Push Button*

   Push Button ini berfungsi sebagai tombol reset konfigurasi dalam *node*

server. *Push Button* akan diprogram untuk menghapus semua data-data pengaturan yang tersimpan di EEPROM sehingga *node* server dapat diatur ulang kembali.

4. LED

   LED ini berfungsi sebagai indikator. LED akan diprogram mengindikasikan indikator-indikator status pada *node* server saat dia aktif.

5. *Buzzer*

   Buzzer ini berfungsi sebagai Alarm. *Buzzer* akan diprogram untuk menjadi alarm saat polusi udara di atas angka wajar.

6. Rasbperry Pi

   Raspberry Pi ini berfungsi sebagai penyimpanan data-data yang dikirim oleh node sensor yaitu NodeMCU. Raspberry Pi juga sebagai *webserver* untuk menampilkan data-data tersebut ke *end-user*.

7. MQTT Broker

   MQTT broker berfungsi protokol sebagai penghubung komunikasi antara node sensor NodeMCU dan server Raspberry Pi.

8. *Database*

   Database berfungsi sebagai penyimpanan data-data yang direkam oleh node sensor NodeMCU.

9. *Web Service*

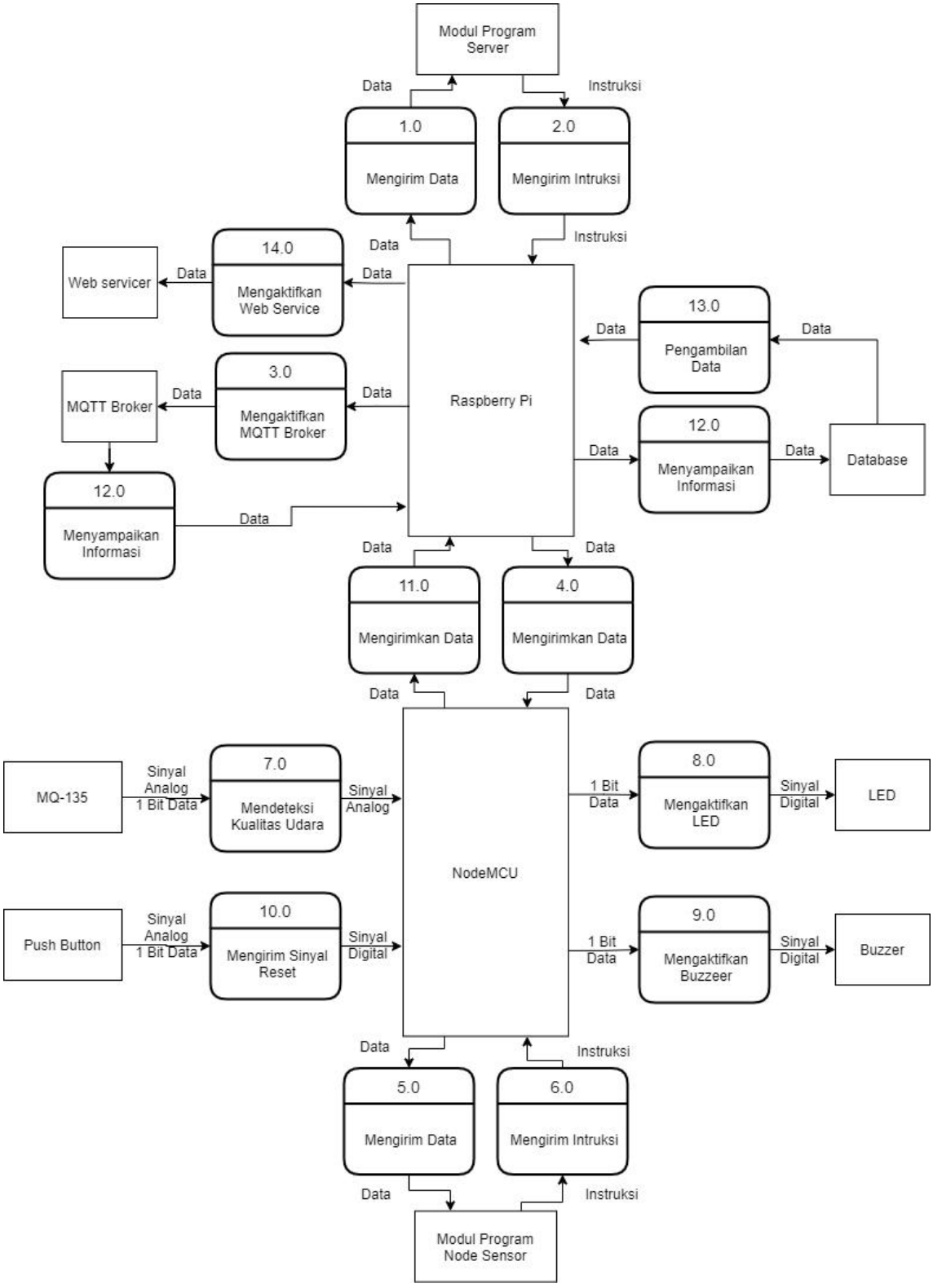
   Web Service berfungsi sebagai media untuk menampilkan data-data pembacaan secara *realtime* dan data-data yang sudah tersimpan di *database*.

10. Modul Program

    Modul program akan melakukan pembacaan terhadap pin-pin NodeMCU, pembacaan terhadap sinyal *input*, memberikan instruksi-instruksi untuk mengaktifkan *pin output*. Modul program mengontrol semua proses yang akan terjadi pada sistem dan program yang digunakan adalah bahasa pemrograman C++ dengan framework Arduino pada PlaformIO di Visual Studio Code.

4.1.2 *Data Flow Diagram* (DFD)

Data Flow Diagram adalah gambaran yang lebih rinci dari alat yang dirancang. *Data Flow Diagram* diuraikan berdasarkan *context diagram* yang telah dijabarkan sebelumnya. Gambar *data flow diagram* dapat dilihat pada Gambar 4.2 berikut ini.



Gambar 4.2 ***Data Flow Diagram***

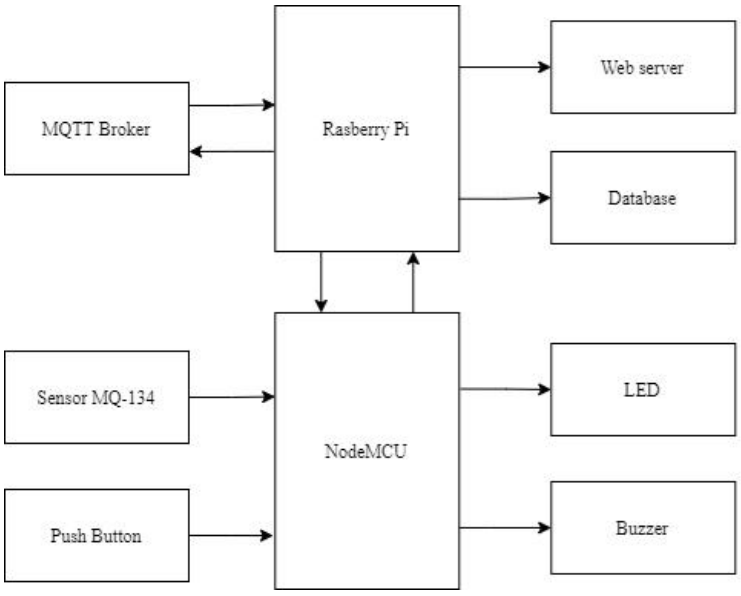
Dari gambar di atas dapat dijelaskan bahwa *prototype* pemantauan kualitas udara ini memiliki beberapa urutan proses instruksi yaitu :

1. Dari *data flow diagram* di atas dapat dilihat bahwa pada sistem proses pertama yaitu mengirimkan data ke modul program. (1.0)
2. Setelah itu, modul program akan memberikan instruksi ke Raspberry Pi. (2.0)
3. Raspberry pi akan memerintahkan MQTT broker untuk mengirimkan sinyal request ke node server NodeMCU. (3.0)
4. Mengirim data paket data melalui TCP/IP yaitu perintah *request* data pada MQTT Broker. (4.0)
5. Data yang diterima NodeMCU akan diproses dan dikirim ke modul program *node sensor* untuk dilakukan pengiriman data dan membandingkan data tersebut dengan *threshold-threshold* yang ditentukan. (5.0)
6. Hasil pengolahan data atau instruksi pada modul program akan dikirim kembali ke modul NodeMCU sebagai *data instruction* yang akan dikirimkan ke *output* sistem yang akan dikontrol. (6.0)
7. Sensor MQ-135 akan memberikan informasi berupa kualitas udara ambien yang nantinya akan diteruskan ke NodeMCU. (7.0)
8. NodeMCU akan mengaktifkan LED untuk menandakan data akan dikirim sesuai dengan instruksi modul program *node sensor*. (8.0)
9. Jika mencapai *threshold* tertentu *buzzer* akan berbunyi sebagai alarm tanda bahaya. (9.0)

10. *Push Button* mengirimkan data ke modul program, diolah untuk keputusan proses penghapusan data konfigurasi yang terdapat pada NodeMCU. (10.0)
11. Kemudian mengirimkan data kembalian berupa data pembacaan kualitas udara ambien pada sensor MQ-135. (11.0)
12. MQTT broker menerima data pembacaan node sensor tersebut. (12.0)
13. Lalu, data pembacaan dari NodeMCU pada MQ-135 akan di simpan ke *database*. (13.0)
14. Mengaktifkan *web service* untuk penampilan data-data *realtime* dan data-data yang berada di *database*. (14.0)

4.1.3 Blok Diagram

Dengan mengacu pada *Data Flow Diagram* di atas, untuk mengetahui komponen-komponen sistem ini dapat dilihat dalam blok diagram pada Gambar 4.3

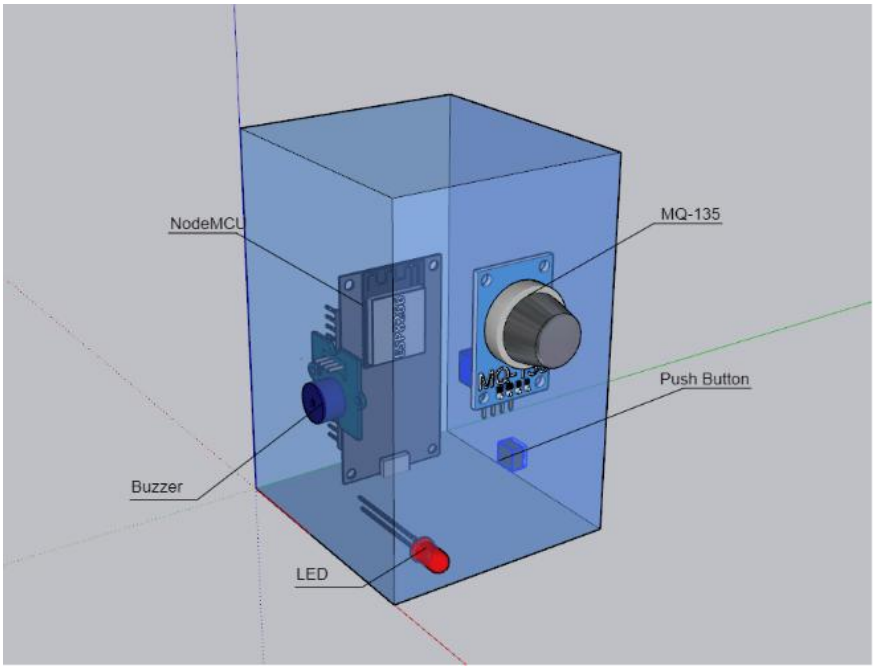


Gambar 4.3 Blok Diagram

Dari blok diagram di atas dapat dilihat bahwa sistem terdiri dari *input, process, output*. Raspberry Pi berperan sebagai pusat pemrosesan dan pengontrolan NodeMCU dll. Sensor MQ-134, *Push Button*, sebagai *input*, *Output* dari sistem adalah LED, *Buzzer, Raspberry Pi, Web Service*, dan *Database*.

4.2 Rancangan Fisik Alat

Perancangan alat ini merupakan tahap awal dari pemasangan dan menganalisa permasalahan yang dihadapi berdasarkan literatur yang menunjang perancangan alat. Rancangan fisik alat yang dibuat dapat dilihat pada gambar 4.4.



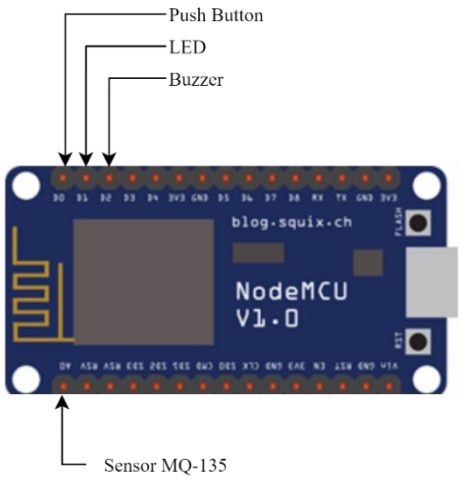
Gambar 4.4 Rancangan Alat Fisik

4.3 Desain Secara Terinci

Desain dari sistem yang dibuat merupakan gambaran dari sistem secara keseluruhan. Dengan adanya desain ini maka prinsip kerja dari sistem serta komponen-komponen dari sistem yang digunakan akan dapat dilihat dengan jelas.

4.3.1 Modul NodeMCU

Rangkaian sistem minimum berfungsi untuk NodeMCU agar dapat bekerja sesuai dengan yang dibutuhkan perancangannya bertujuan untuk mempermudah penggunaan alat.



Gambar 4.5 Modul NodeMCU

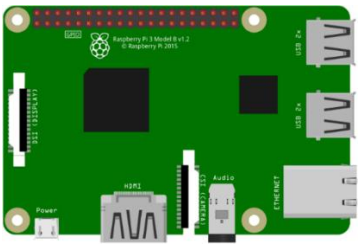
Berdasarkan Gambar 4.5 konfigurasi pin yang digunakan pada modul NodeMCU adalah sebagai berikut.

1. Pin A0 pada NodeMCU akan terhubung ke MQ-135
2. Pin D0 pada NodeMCU akan terhubung dengan *Push Button*
3. Pin D1 pada NodeMCU akan terhubung dengan LED

4. Pin D2 pada NodeMCU akan terhubung dengan *Buzzer*

4.3.2 Rangkaian Raspberry Pi 3 B+

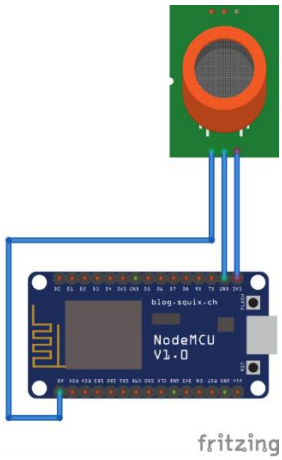
Pada perancangan alat ini digunakan menggunakan Raspberry Pi yang digunakan untuk sebagai Web Service dan MQTT Broker. Rangkaian Raspberry Pi ditujukan pada gambar di bawah.



Gambar 4.6 Rangkaian Raspberry Pi 3B+

4.3.3 Rangkaian MQ-135

Pada perancangan alat ini menggunakan sensor MQ-135 sebagai *input* yang digunakan untuk mengukur kualitas udara. Pada modul sensor MQ-135 terdapat 3 pin yaitu: A0, GND, VCC.

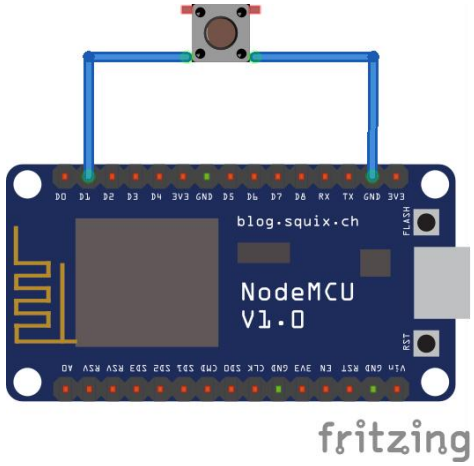


Gambar 4.7 Rangkaian MQ-135

Penjelasan pin yang digunakan pada modul sensor MQ-135 yaitu sebagai berikut.

1. Pin VCC terhubung ke pin 3V pada NodeMCU.
2. GND terhubung ke pin GND pada NodeMCU
3. A0 terhubung ke pin A0 pada NodeMCU

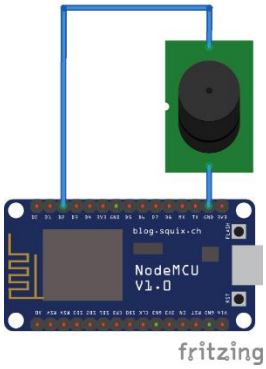
4.3.4 Rangkaian *Push Button*



Gambar 4.8 Rangkaian *Push Button*

Pada perancangan alat ini menggunakan *Push Button* sebagai *input* yang digunakan untuk melakukan perintah reset data pada NodeMCU. Pin *push button* akan terhubung dengan pin D1 dan GND pada NodeMCU.

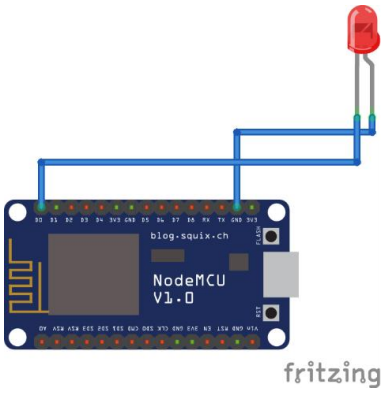
4.3.5 Rangkaian *Buzzer*



Gambar 4.9 Rangkaian Buzzer

Pada perancangan alat ini menggunakan *Buzzer* sebagai *output* alarm juga jika tingkat polusi mencapai nilai yang ditentukan. Pada pin daya positif *buzzer* dihubungkan ke pin D2 pada NodeMCU dan pin daya negatif *buzzer* akan dihubungkan ke pin GND pada NodeMCU.

4.3.6 Rangkaian LED

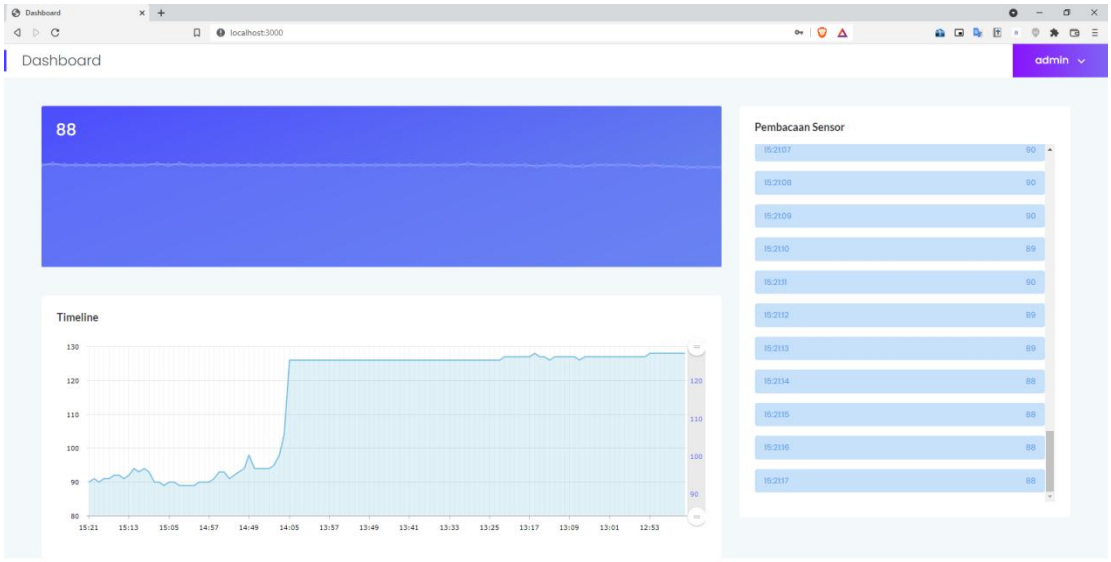


Gambar 4.10 Rangkaian LED

Pada perancangan alat ini menggunakan LED sebagai *output* sebagai notifikasi status alat dalam *runtime*. Pada pin daya positif LED akan dihubungkan

ke pin D0 pada NodeMCU dan pin daya negatif akan dihubungkan ke pin GND pada NodeMCU

4.4 Rancangan Tampilan WEB *Dashboard*



Gambar 4.11 Tampilan WEB *Dashboard*

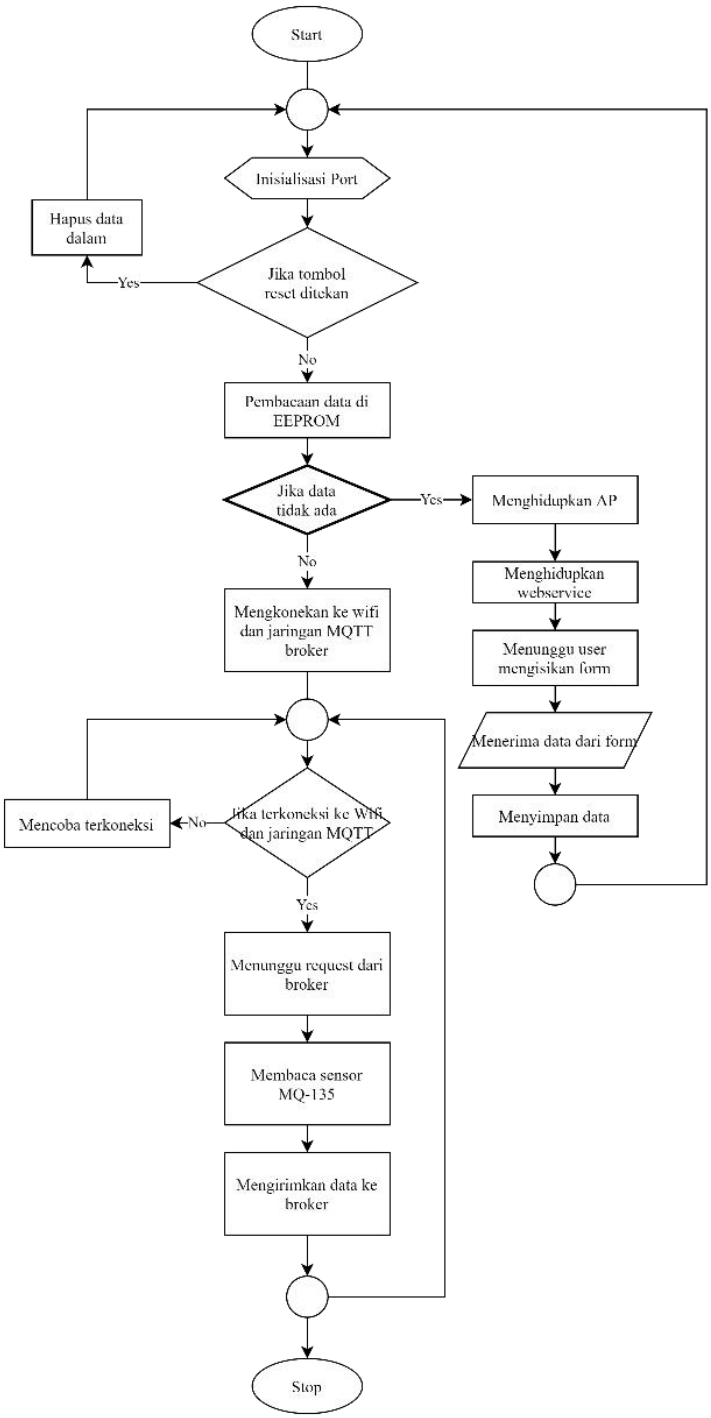
4.5 Rancangan Modul Program

Sub bab ini menjelaskan tentang Modul Program yang digunakan untuk mengontrol kerja dari sistem yang dirancang. Untuk lebih mudah dimengerti rancangan modul dapat menjadi dua bagian yaitu: Flowchart dan *Listing Program*.

4.6.1 *Flowchart*

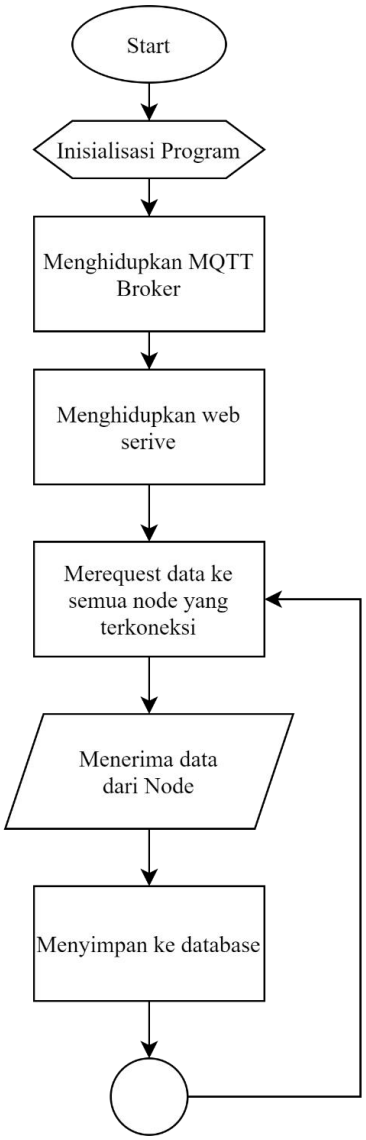
Modul program dirancang memiliki struktur dengan kualitas yang baik dan mudah dimengerti, maka sebelum pembuatan *listing program* perlu diawali

dengan penentuan logika program. Logika dasar gambaran pada penulisan ini adalah dengan menggunakan *flowchart* seperti gambar 4. berikut:



Gambar 4.12 *Flowchart*

4.6.2 *Flowchart* Server



Gambar 4.13 Flowchart Server

4.6.3 *Listing Program*

Pada sub bab ini diuraikan mengenai modul program untuk menunjang kemampuan sistem dalam menggunakan bahasa pemrograman dengan framework Arduino.

Program MQ-135

```
#include <Arduino.h>

void setup()
{
  //Serial Port Begin
  Serial.begin(9600);
  //Define input
  pinMode(A0, INPUT);
}

void loop()
{
  //Membaca sensor MQ-135 dan dikeluarkan ke Serial
  Serial.println(analogRead(A0));
  delay(1000);
}
```

Program LED

```
#include <Arduino.h>

void setup()
{
  //Define output
  pinMode(D0, OUTPUT);
}

void loop()
{
  //LED Blinking Program
  digitalWrite(D0, HIGH);
  delay(1000);
  digitalWrite(D0, LOW);
  delay(1000);
}
```

Program Buzzer

```
#include <Arduino.h>

void setup()
{
 //Define output
 pinMode(D3, OUTPUT);
}

void loop()
{
 //LED Blinking Program
 digitalWrite(D3, HIGH);
 delay(1000);
 digitalWrite(D3, LOW);
 delay(1000);
}
```

Program Push Button

```
#include <Arduino.h>

const int pinButton = D1;

void setup()
{
 //Inisialisasi port serial
 Serial.begin(9600);
 //Inisialisasi pin Tombol
 pinMode(pinButton, INPUT_PULLUP);
}

void loop()
{
 //Mencetak Status Tombol
 if (digitalRead(pinButton) == LOW)
 {
  Serial.println("Button Pressed");
 }
 else
 {
```

```
    Serial.println("Button Released");
  }
  delay(100);
}
```

Program NodeMCU

```
#include <Arduino.h>

void setup()
{
  //Inisialisasi port serial
  Serial.begin(9600);
}

void loop()
{
  //Mencetak Text ke Serial
  Serial.println("Hello World");
  delay(1000);
}
```

Program Web Service dengan Express.js

```
const express = require("express");
const app = express();
const port = 3000;

app.get("/", (req, res) => {
  res.send("Hello World!");
});

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`);
});
```

Program MQTT Broker

```
const aedes = require('aedes')()
const server = require('net').createServer(aedes.handle)
const port = 1883

let clients = new Map();

server.listen(port, function () {
  console.log('server started and listening on port ', port)
})

broker.on("clientReady", (client) => {
  console.log(`BROKER : Client (${client.id}) connected.`);
  clients.set(client.id, client);
});


setInterval(() => {
  for (var client of clients.entries()) {
    var packet = {
      cmd: "publish",
      messageId: 42,
      qos: 2,
      dup: false,
      topic: "command",
      payload: Buffer.from("Hello World"),
      retain: false,
    };
    client[1].publish(packet, (err) => {
      //console.log(`BROKER : Request client (${client[0]}).`);
    });
  }
}, 1000);
```

Program MQTT Publisher

```
#include <Arduino.h>
#include <PubSubClient.h>
#include <ESP8266WiFi.h>

IPAddress server(172, 16, 0, 2);

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();
}

WiFiClient wifiClient;
PubSubClient client(wifiClient);

void reconnect()
{
    // Loop until we're reconnected
    while (!client.connected())
    {
        Serial.print("Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect("arduinoClient"))
        {
            Serial.println("connected");
            // Once connected, publish an announcement...
            client.publish("outTopic", "hello world");
            // ... and resubscribe
            client.subscribe("inTopic");
        }
        else
        {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
}

void setup()
{
    Serial.begin(57600);

    client.setServer(server, 1883);
    client.setCallback(callback);
    delay(1500);
}
```

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

BAB V

PENGUJIAN SISTEM

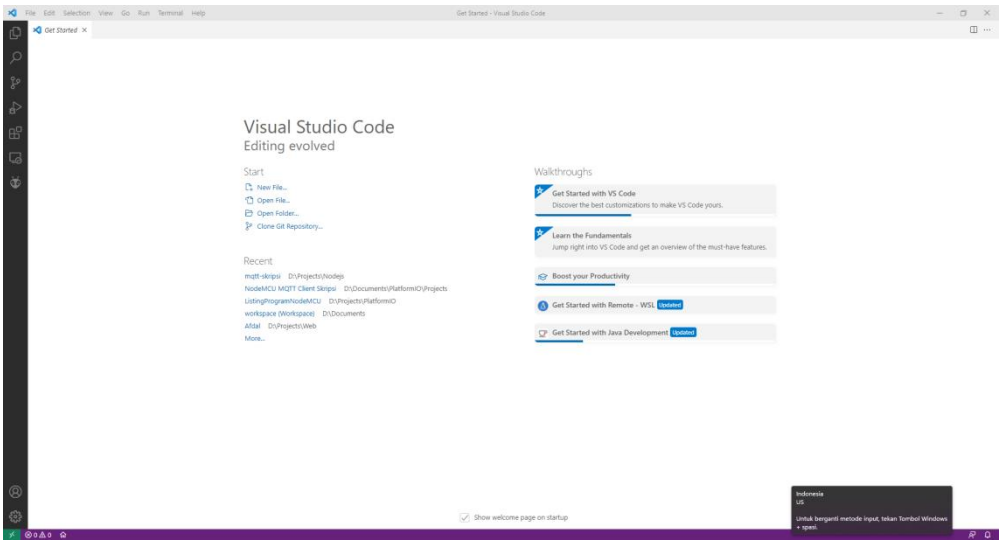
5.1 Pengujian Sistem Permodul

Adapun langkah yang dapat dilakukan dalam menjalankan aplikasi pemrograman aplikasi NodeMCU dan Raspberry Pi adalah sebagai berikut.

5.1.1 Pengujian Rangkaian Minimum

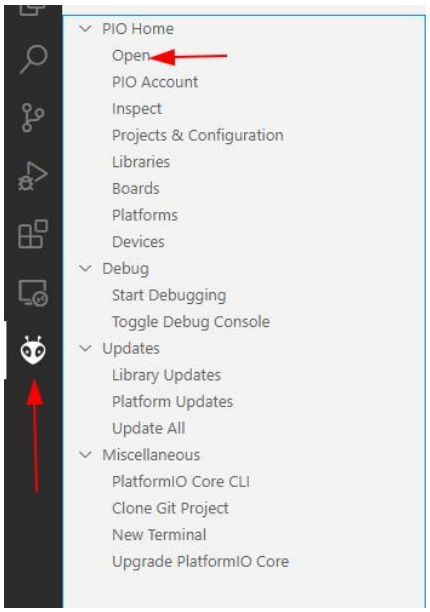
Adapun langkah yang harus dilalui dalam pemrograman mikrokontroler adalah yaitu :

1. Buka Visual Studio Code dengan mengklik ganda ikon aplikasi sehingga aplikasi akan terbuka dan menampilkan seperti gambar 5.1.



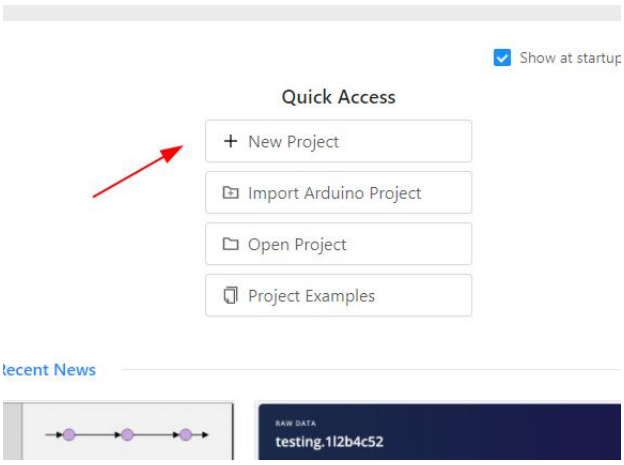
Gambar 5.1 Tampilan Awal Visual Studio Code

2. Lalu perhatikan Activity Bar pada terdapat ikon PlatformIO yang ditunjukan pada gambar 5.2. Klik ikon tersebut untuk membuka menu PlatformIO. Setelah itu klik “Open” pada menu untuk membuka halaman Home.



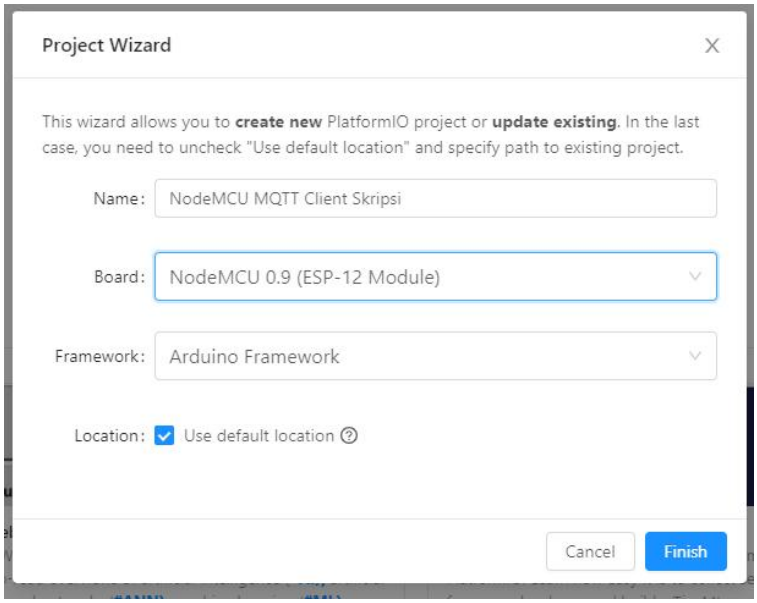
Gambar 5.2 *Activity Bar*

3. Pada halaman home, klik “New Project” pada menu “Quick Access.



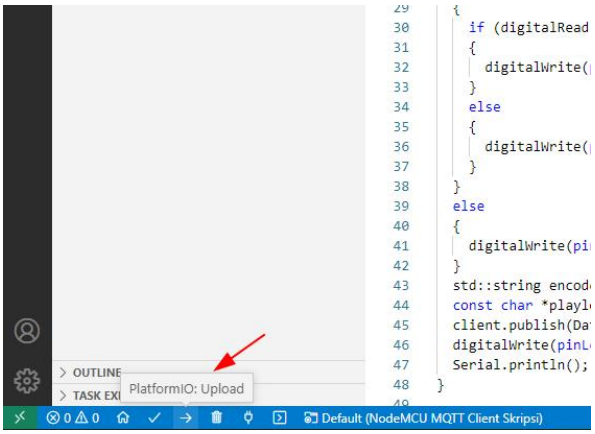
Gambar 5.3 *Quick Access Menu*

4. Pada Project Wizard akan isi seperti gambar dibawah, lalu klik finish.



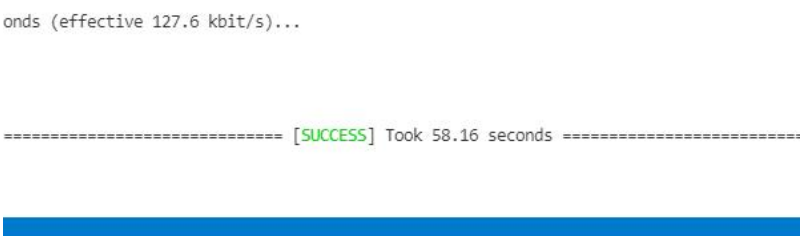
Gambar 5.4 *Project Wizard* PlatformIO

5. Lalukan pengetikan program, kemudian simpan dengan cara menekan tombol CTRL+S. Lalu pada pojok kiri bawah terdapat tombol untuk peng*apload* program ke NodeMCU.



Gambar 5.5 Tombol Upload

6. Jika program sukses di upload ke dalam NodeMCU, maka akan muncul tulisan “Success”, seperti Gambar 5. NodeMCU siap digunakan.



Gambar 5.6 Notifikasi Sukses pada Peng*aploada*n

5.1.2 Pengujian Sensor MQ-135

Sensor MQ-135 berfungsi untuk pendeteksi polutan pada udara, hasil dari pengujian dapat dilihat pada tabel berikut :

Tabel 5.1 Hasil Pembacaan Sensor MQ-135

No.	Sensor MQ-135	*Buzzer*
1.	Dijauhkan dari asap	Berbunyi
2.	Didekatkan dari asap	Diam

5.1.2 Pengujian *Push Button*

Push Button berfungsi sebagai pemanggilan program reset data. Program ini akan me*reset setting* yang telah disimpan di dalam EEPROM. Pemanggilan program reset dilakukan saat menghidupkan alat pertama kali. Hasil dari pengujian dapat dilihat pada tabel berikut:

Tabel 5.2 Hasil Pembacaan *Push Button*

No.	*Push Button*	LED	*Buzzer*	*Hotspot*
1.	Ditekan	Mati	Hidup	Hidup
2.	Tidak Ditekan	Hidup	Mati	Mati

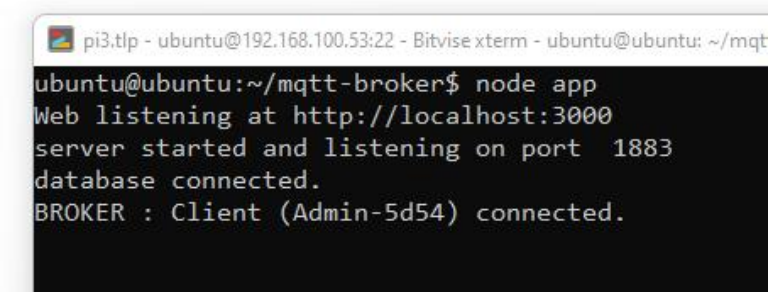
5.2 Pengujian Sistem Keseluruhan

Secara elektronis rangkaian telak bekerja dengan baik, yaitu sistem minimum, Sensor MQ-135, *Push Button*, *Buzzer*, LED, dan NodeMCU. Tahap-tahap dalam pengujian rangkaian keseluruhan adalah sebagai berikut:

1. Hubungkan Server (Raspberry Pi) dan *Node Sensor* (NodeMCU) ke sumber tegangan agar mendapatkan daya. Masing-masing dari perangkat dihubungkan ke *power supplay* menggunakan kabel mikro USB dapat dilihat pada gambar di bawah.

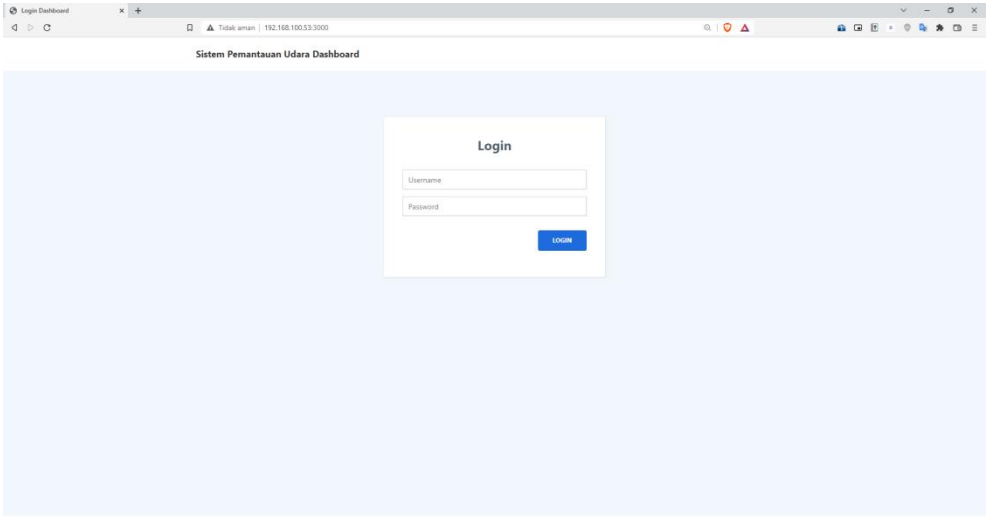
Gambar 5.7 Alat Dihubungkan dengan *Power Supplay*

2. Hidupkan program server pada Raspberry Pi melalui *remote* SSH dan alat akan otomatis tersambung ke server, seperti yang terlihat pada Gambar 5.8



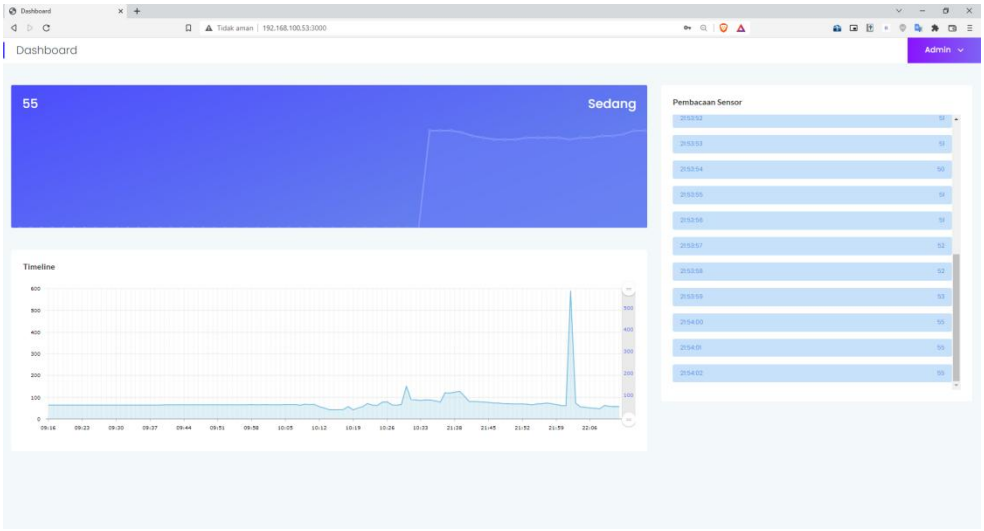
Gambar 5.8 Program Server

3. Lalu buka browser dan masukkan IP yang sesuai pada server dalam kasus ini 192.168.100.54:3000 dan akan muncul halaman *login* seperti pada Gambar 5.9



Gambar 5.9 Tampilan Halaman Login

4. Lalu masukan username dan password yang telah ditentukan dalam test ini yaitu “Admin” dan “admin” lalu tekan tombol login. Kita akan memasuki *dashboard* sistem seperti pada Gambar 5.10



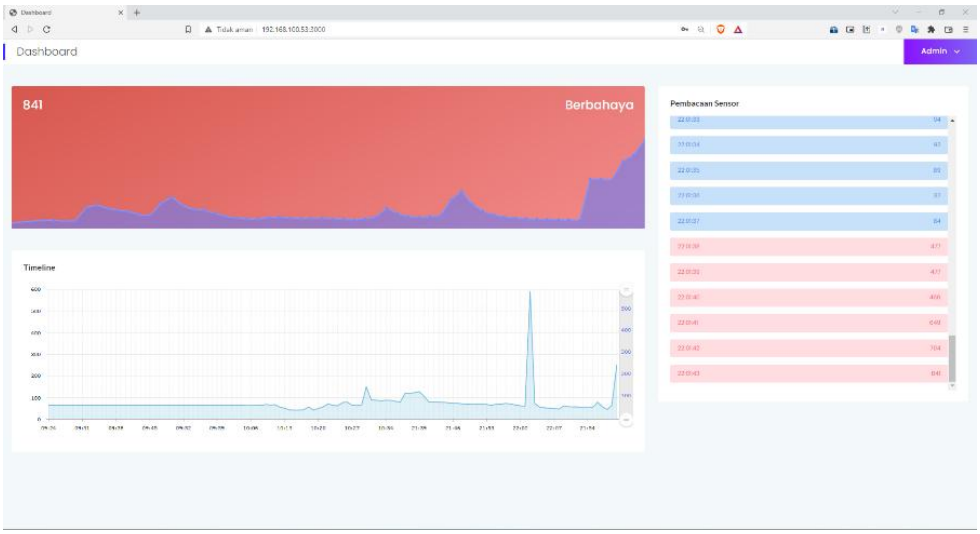
Gambar 5.10 Tampilan *Dashboard Sistem*

Terdapat pada kiri atas merupakan bagan pembacaan *realtime*, Lalu pada bagian kiri terdapat catatan log aktivitas pembacaan sensor secara *realtime* dan pada kiri bawah merupakan grafik pembacaan sensor yang disimpan di *database*. Grafik yang diperlihatkan dari *database* adalah grafik dari pembacaan sensor selama dua jam terakhir.

5. Untuk memicu alarm berbahaya sensor didekatkan dengan Asap yang dapat dilihat pada Gambar 5.11

Gambar 5.11 Alat Didekatkan dengan Asap

6. Jika polusi udara di atas ambang yang ditentukan yaitu 300 PPM, maka *buzzer* akan berbunyi dan lonjakan grafik garis pada web dan log pada web juga berubah menjadi merah seperti pada Gambar 5.12

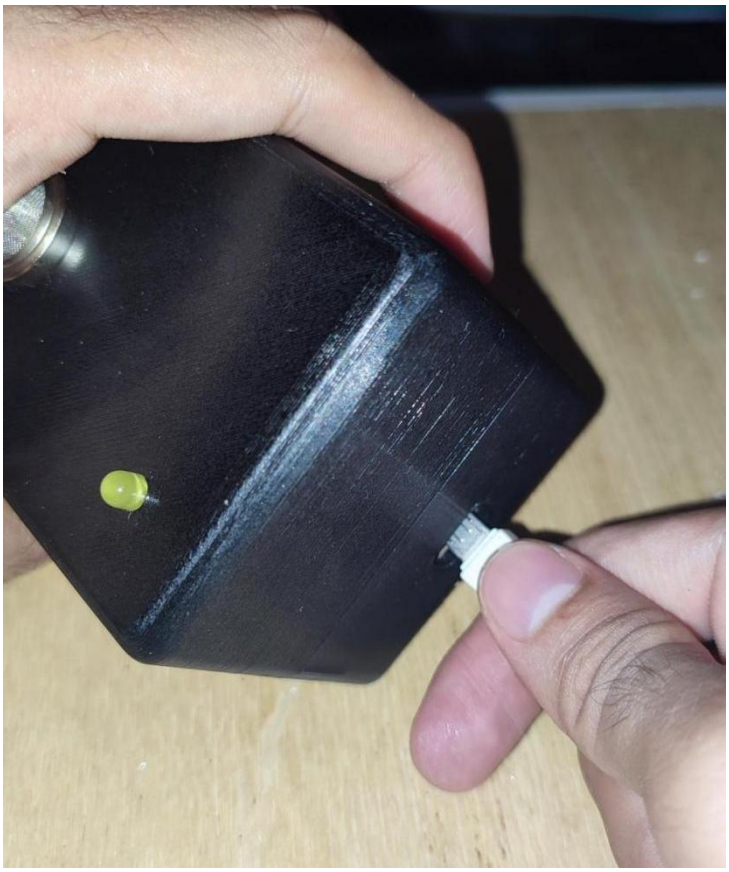


Gambar 5.12 Tampilan *Dashboard* Saat Alat Didekatkan dengan Asap

Dapat dilihat pada Gambar 5.12 terdapat lonjakan grafik dari pembacaan sensor pada bagan grafik *realtime* di pojok kiri atas dan *background* dari bagan tersebut berubah menjadi merah dan status dari pembacaan sensor ditampilkan "Berbahaya". Lalu juga terdapat perubahan warna pada catatan aktivitas pembacaan sensor.

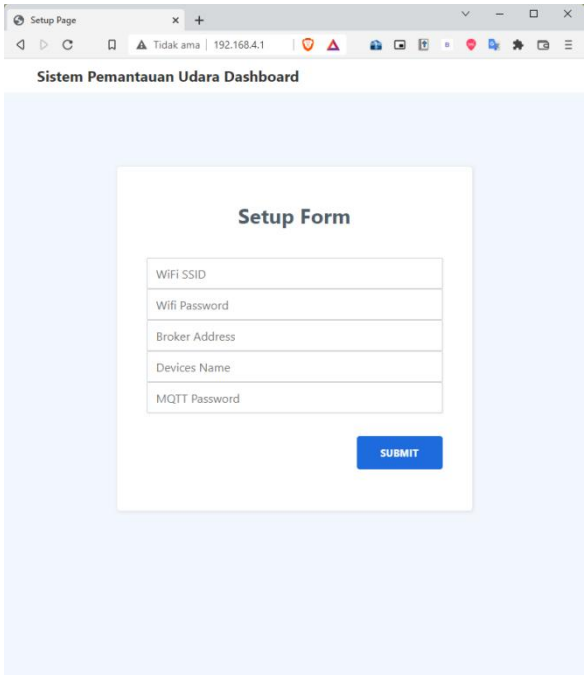
Untuk antisipasi pada jaringan dan ip yang berbeda maka perlu adanya program untuk me*reset setting* yang terdapat di alat. Untuk melakukan peng*resetan setting* tersebut dari alat dapat dilakukan sebagai berikut.

1. Tekan dan tahan tombol reset pada alat. Lalu hubungkan alat ke *power supplay*. Di saat *push button* ditekan LED Indikator akan tetap mati sampai *buzzer* berbunyi seperti pada Gambar 5. 13



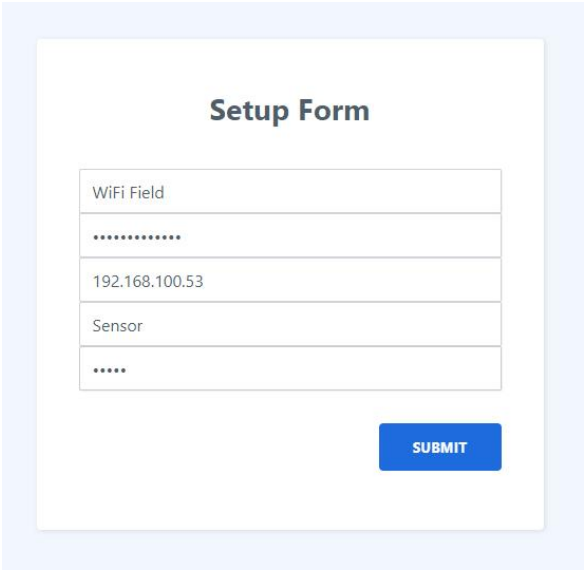
Gambar 5.13 Tombol Reset Alat Ditekan Saat Disambung ke *Power Supplay*

Maka *buzzer* akan berbunyi menandakan program reset berhasil di panggil. Lalu alat akan memasuki mode setup tertanda saat alat hidup LED akan berkedip 2x. Pada mode setup ini alat akan memancarkan *hotspot* untuk dapat menyunting alat melalu form yang disiapkan yang dapat diakses melalui ip 192.168.4.1 seperti pada Gambar 5.14



Gambar 5.14 Tampilan Formulir Setup

2. Lalu isi formulir setup yang sesuai dengan jaringan agar terhubung dengan server seperti pada Gambar 5.15

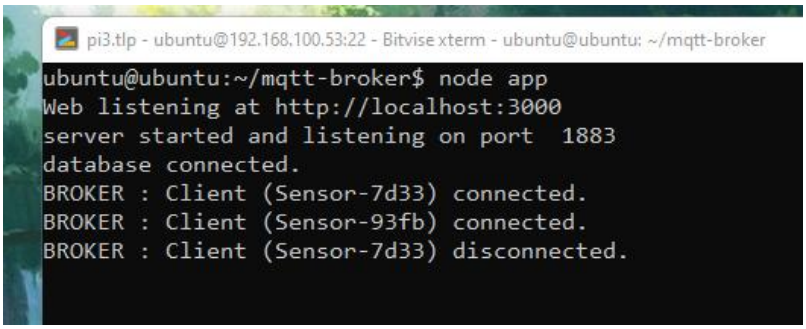


Gambar 5.15 Tampilan Formulir Alat Diisi dengan Data

3. Setelah itu pencet submit untuk menyimpan konfigurasi setting untuk alat. Kesuksesan penyimpanan setting ditandai dengan kata sukses yang direspon oleh alat seperti pada gambar 5.16.

Gambar 5.16 Tampilan Respons Sukses dari Penyimpanan *Setting*

4. Jika data yang dimasukkan benar, maka alat akan terhubung ke server seperti pada Gambar 5.17



Gambar 5.17 Program Server Saat Alat Terhubung Kembali

BAB VI

PENUTUP

Berdasarkan hasil perancangan sistem dan pembuatan alat ini, maka dapat diambil kesimpulan, saran-saran serta keterbatasan sistem yang nantinya akan bermanfaat bagi pengembangan sistem berikutnya.

6.1 Kesimpulan

Dari semua penjelasan yang tertera dari bab-bab sebelumnya dapat diambil kesimpulan seperti di bawah ini:

1. Sistem *Monitoring* Kualitas Udara ini mampu bekerja dengan baik untuk menyimpan dan menampilkan nilai pembacaan dari sensor MQ-135.
2. Penggunaan Sensor MQ2 dapat mendeteksi kualitas udara pada ruangan dengan baik.
3. Penggunaan NodeMCU dapat mengukur dan mengirim data tersebut ke Raspberry Pi.
4. *Database* dapat dengan baik menyimpan data-data pengukuran kualitas udara dengan baik.
5. *Web service* dapat dengan baik menjadi antar muka pengguna untuk menampilkan data yang diukur dan data yang disimpan dari *database*.

6. *Push button* dapat dengan baik menjadi tombol reset konfigurasi dari NodeMCU.
7. LED dapat dengan baik menjadi indikator untuk NodeMCU.
8. *Buzzer* dapat dengan baik menjadi alarm jika kualitas udara terlalu rendah.
9. Sistem ini dapat gunakan untuk melalukan pemantauan kualitas udara secara kontinu untuk mengambil keputusan dalam menekan polusi udara

6.2 Saran-Saran

Berdasarkan pengalaman yang diperoleh selama perancangan dan pembuatan alat ini, ada beberapa kendala yang dihadapi dan di sini akan disampaikan beberapa saran yang bermanfaat untuk mengembangkan dan menyempurnakan hasil karya berikutnya:

1. Sistem ini hanya dapat melakukan penyimpanan dan penampilan data dari satu *node sensor* saja. Peneliti berikutnya diharapkan dapat melakukan pengembangan untuk melakukan penanganan penyimpanan dan penampilan data dari beberapa *node sensor*.
2. Untuk pengembangan berikutnya, diharapkan adanya pengembangan untuk manajemen *user* agar dapat mendukung banyak *user*.
3. Agar lebih pembacaan sensor MQ-135 lebih akurat, sistem sebaiknya disisipkan sensor suhu.
4. Diharapkan adanya kreativitas pada peneliti berikutnya untuk menambah fitur terhadap alat ini agar memiliki daya guna lebih.

Berdasarkan kesimpulan dan saran di atas, diharapkan tugas akhir ini bisa dipahami dan sistem yang dibuat dapat dikembangkan menjadi lebih sempurna dari penelitian sebelumnya dan dapat menjadikan sumber informasi yang cukup untuk menyusun tugas akhir maupun penelitian berikutnya yang lebih baik.

DAFTAR PUSTAKA

Adrian, K. (2021, March 2). *Waspadai Bahaya Polusi Udara terhadap Kesehatan Paru-paru*. Alodokter; Alodokter. https://www.alodokter.com/lindungi-paru-paru-anda-dari-polusi-udara

Alhamidi. (2016). Membangun Sistem Aplikasi Untuk Seleksi Calon Mahasiswa Undangan Pada Tingkat Sekolah Menengah Atas. *Jurnal J-Click*, *3*(2). http://ejurnal.jayanusa.ac.id/index.php/J-Click/article/view/26

Arduino. (2022a). *loop() - Arduino Reference*. Arduino.cc. https://www.arduino.cc/reference/en/language/structure/sketch/loop/

Arduino. (2022b). *setup() - Arduino Reference*. Arduino.cc. https://www.arduino.cc/reference/en/language/structure/sketch/setup/

Atmoko, R. A. (2019). Dasar Implementasi Protokol MQTT Menggunakan Python dan NodeMCU. Mokosoft Media.

Darmawan, M., & Suprihatin. (2020). Sistem Komputer untuk SMK/MAK Kelas X. Gramedia Widiasarana Indonesia.

Dicoding Indonesia. (2021, May 14). *Tutorial Node.js: Apa itu Node.js dan dasar-dasar Node.js - Dicoding Blog*. Dicoding Blog. https://www.dicoding.com/blog/apa-itu-node-js/

Dicoding Intern. (2020, September 15). *Apa itu Database? Contoh Produk dan Fungsinya - Dicoding Blog*. Dicoding Blog. https://www.dicoding.com/blog/apa-itu-database/

Dinata, A. (2017). Physical Computing dengan Raspberry Pi. Elex Media Komputindo.

Fatta, H. A., & Amikom, U. (2001). Analisis dan Perancangan Sistem Informasi untuk Keunggulan Bersaing Perusahaan dan Organisasi Modern. Penerbit Andi.

Gunawan, H., Ruslinda, Y., & Putri, D. (2015). Pengaruh Karakteristik Lalu Lintas Terhadap Konsentrasi Gas No2 Di Udara Ambien Roadside Jaringan Jalan Sekunder Kota Padang. *Prosiding 2nd Andalas Civil Engineering National Conference*, 93–99. https://conference.ft.unand.ac.id/index.php/ace/ace2015/paper/download/365/156

Handayani, F. (2019). Tren Masif Internet of Things (IOT) Di Perpustakaan. *Jurnal Ilmu Perpustakaan Dan Informasi*, *4*(2), 194–209. http://jurnal.uinsu.ac.id/index.php/jipi/article/download/4381/3230

Jenifer, & Aravindhar, D. J. (n.d.). Iot Based Air Pollution Monitoring System Using Esp8266-12 With Google Firebase. *Journal of Physics: Conference Series*, *1362*(1), 012072. Retrieved December 20, 2021, from https://www.academia.edu/35085518/IOT_BASED_AIR_POLLUTION_MONITORING_SYSTEM_USING_ARDUINO

Kontributor Wikipedia. (2004, March 9). *komponen*. Wikipedia.org; Wikimedia Foundation, Inc. https://id.wikipedia.org/wiki/Resistor

Kontributor Wikipedia. (2005, December 12). *kumpulan halaman web terkait yang disediakan dari sebuah domain web tunggal*. Wikipedia.org; Wikimedia Foundation, Inc. https://id.wikipedia.org/wiki/Situs_web

Kontributor Wikipedia. (2008, February). *Sistem kendali*. Wikipedia.org; Wikimedia Foundation, Inc. https://id.wikipedia.org/wiki/Sistem_kendali

Kontributor Wikipedia. (2014, July 18). *Raspberry Pi*. Wikipedia.org; Wikimedia Foundation, Inc. https://id.wikipedia.org/wiki/Raspberry_Pi

Kontributor Wikipedia. (2019, February 5). *Interoperability*. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Interoperability

Kontributor Wikipedia. (2021a, September 20). *Push-button*. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Push-button

Kontributor Wikipedia. (2021b, November 17). *Buzzer*. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Buzzer

Kontributor Wikipedia. (2021c, December 11). *Light-emitting diode*. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Light-emitting_diode

Kumar Sai, K. B., Mukherjee, S., & Parveen Sultana, H. (2019). Low Cost IoT Based Air Quality Monitoring Setup Using Arduino and MQ Series Sensors With Dataset Analysis. *Procedia Computer Science*, *165*, 322–327. https://doi.org/10.1016/j.procs.2020.01.043

Maniah, & Hamidin, D. (2017). *Analisis dan Perancangan Sistem Informasi*. Deepublish.

Microsoft. (2021, December 11). *microsoft/vscode: Visual Studio Code*. GitHub. https://github.com/microsoft/vscode

Nur Wijayanti, N. (2021, December 31). *Apa Itu Website? Pengertian, Jenis, dan Manfaatnya!* Niagahoster Blog. https://www.niagahoster.co.id/blog/pengertian-website/

Nurcholish, A. (2018). *Membangun database arsip persuratan menggunakan pemrograman PHP dan Mysql: studi kasus pada Kantor Balai Taman Nasional Gunung Rinjani*. CV Jejak (Jejak Publisher).

Platform IO. (2020). *VSCode — PlatformIO latest documentation*. Platformio.org. https://docs.platformio.org/en/latest/integration/ide/vscode.html

PlatformIO. (2017). *PlatformIO IDE - Visual Studio Marketplace*. @Code. https://marketplace.visualstudio.com/items?itemName=platformio.platformio-ide

PlatformIO. (2022). *Arduino — PlatformIO v5.2 documentation*. Platformio.org. https://docs.platformio.org/en/stable/frameworks/arduino.html

Programiz. (2022). *C++ Operators*. Programiz.com. https://www.programiz.com/cpp-programming/operators

Ramadhanti. (2021, June 12). *Contoh DFD (Data Flow Diagram) & Diagram Konteks - Guratgarut*. Pinhome. https://www.pinhome.id/blog/contoh-dfd/#:~:text=Apabila%20diartikan%2C%20diagram%20konteks%20ini,dapat%20menggambarkan%20lingkup%20sebuah%20sistem.&text=Diagram%20konteks%20menggambarkan%20sistem%20secara%20keseluruhan%2C%20sedangkan%20DFD%20lebih%20mendetail,telah%20digambarkan%20oleh%20diagram%20konteks.

Ruslinda, Y. (2014). Analisis Kualitas Udara Ambien Kota Padang akibat Pencemar Particulate Matter. *TeknikA*, *21*(2).

Sharma, D. K., Balas, V. E., Son, L. H., Sharma, R., & Cengiz, K. (2020). Micro-Electronics and Telecommunication Engineering: Proceedings of 3rd ICMETE 2019. Springer Nature.

Sugiyono. (2013). *Cara mudah menyusun skripsi, tesis, dan disertasi (STD)*. Alfabeta.

Tanjung, I., & Sukrianto, D. (2017). Perancangan Sistem Informasi Rekam Medis Terpadu Dalam Upaya Meningkatkan Pelayanan Rumah Sakit Jiwa Tampan Prov. Riau. *Jurnal Intra Tech*, *1*(1), 43–54. https://www.journal.amikmahaputra.ac.id/index.php/JIT/article/view/3

W3School. (2022a). *C++ For Loop*. W3schools.com. https://www.w3schools.com/cpp/cpp_for_loop.asp

W3School. (2022b). *C++ If … Else*. W3schools.com. https://www.w3schools.com/cpp/cpp_conditions.asp

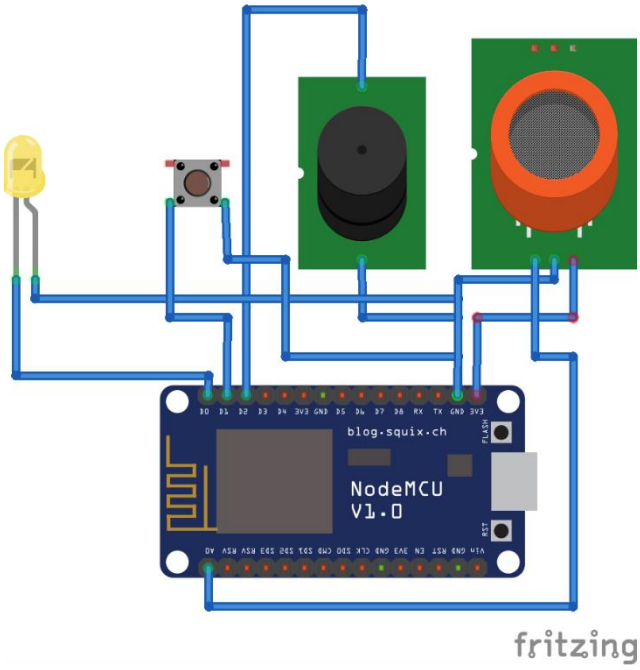
Wasista, S., Setiawardhana, Saraswati, D. A., & Susanto, E. (2019). Aplikasi Internet Of Things (IOT) Dengan Arduino Dan Android “Membangun Smart Home Dan Smart Robot Berbasis Arduino Dan Android.”. Deepublish.

Yudaningtyas, E. (2017). Belajar Sistem Kontrol: Soal dan Pembahasan. In *Google Books*. Universitas Brawijaya Press.

Yulia Basri, I., & Irfan, D. (2018, August). *Komponen Elektronika*. Padang; SUKABINA Press.

Yuniansyah. (2020). *Algoritma dan Pemrograman Menggunakan Bahasa Pemrograman Java (Teori dan Aplikasinya)*. Penerbit Lindan Bestari.

LAMPIRAN RANGKAIAN KESELURUHAN



LAMPIRAN LISTING PROGRAM KESELURUHAN

Program Server

```
require("dotenv").config();
const express = require("express");
const path = require("path");
const fs = require("fs");
const app = express();
const port = process.env.WEB_PORT;
const session = require("express-session");
const bcrypt = require("bcrypt");
const { MongoClient } = require("mongodb");
const { resolve } = require("path");
const { reject } = require("bcrypt/promises");

const dburl = "mongodb://127.0.0.1:27017";
const dbname = "skripsi-air-monitoring";
const dbclient = new MongoClient(dburl);

dbclient.connect((error, client) => {
  if (error) {
    console.log("cannot connect to database");
    process.exit();
  }
  console.log("database connected.");
});

const db = dbclient.db(dbname);
const users = db.collection("users");
const timeline = db.collection("timeline");

const salt = bcrypt.genSaltSync(10);

let dir = "./data";

app.use(
  session({
    secret: "keyboard cat",
    resave: false,
    saveUninitialized: true,
  })
);

app.use(express.json());
app.use(express.urlencoded({ extended: false }));

let sensordata = {
  time: 0,
```

```
  value: 0,
  timeline: 0,
};

function renderMessage(res, _title, _message) {
  var message = {
    title: _title,
    message: _message,
  };
  res.render("message", message);
}

app.set("view engine", "ejs");

function renderRegister(res, _warning) {
  res.render("form", {
    form: {
      name: "Registration",
      warning: _warning,
      action: "/register",
      button: "register",
      format: [
        { id: "username", description: "Username", type: "text" },
        { id: "password", description: "Password", type:
"password" },
        {
          id: "verify-password",
          description: "Verify Passsword",
          type: "password",
        },
      ],
    },
  });
}

app.post("/register", (req, res) => {
  let lenght = 0;
  users.count({}, (error, _length) => {
    lenght = _length;
  });

  if (lenght != 0) {
    renderMessage(res, "You are already registered.", "Go back to
login");
    return;
  }

  if (
    (req.body.user == "") |
    (req.body.password == "") |
    (req.body["verify-password"] == "")
  ) {
```

```
    renderRegister(res, "Do not let form blank!");
  } else if (req.body.password != req.body["verify-password"]) {
    renderRegister(res, "Your password you're inputting is not
same.");
  } else {
    users.insertOne(
      {
        username: req.body.username,
        password: bcrypt.hashSync(req.body.password, salt),
      },
      (error, result) => {
        if (!error) {
          res.redirect("/");
        }
      }
    );

    return;
  }
});

function renderLogin(res, _warning) {
  return res.render("form", {
    form: {
      name: "Login",
      warning: _warning,
      action: "/login",
      button: "login",
      format: [
        { id: "username", description: "Username", type: "text" },
        { id: "password", description: "Password", type:
"password" },
      ],
    },
  });
}

app.post("/login", async (req, res) => {
  try {
    const user = await (
      await users.find({ username: req.body.username }).toArray()
    )[0];
    if (user == undefined) {
      return renderLogin(res, "Username and password is wrong");
    }
    if ((req.body.username == "") | (req.body.password == "")) {
      return renderLogin(res, "Do not let form blank!");
    } else if (
      (req.body.username != user.username) |
      !(await bcrypt.compare(req.body.password, user.password))
    ) {
      return renderLogin(res, "Username and password is wrong");
```

```
    } else {
      req.session.isAuth = true;
      req.session.user = user.username;
      res.redirect("/");
    }
  } catch (err) {
    return renderLogin(res, "Username and password is wrong");
  }
});

app.get("/logout", (req, res) => {
  req.session.destroy();
  res.redirect("/");
});

app.get("/", (req, res) => {
  if (req.session.isAuth) {
    res.render("dashboard", {
      user: {
        name: req.session.user,
      },
    });
  } else {
    db.collection("users").count({}, (error, length) => {
      if (length == 0) {
        renderRegister(res, "");
      } else {
        renderLogin(res, "");
      }
    });
  }
});

app.get("/api", async (req, res) => {
  /*
  if (!req.session.isAuth) {
    res.status(403);
    res.end();
    return;
  }
  */

  if (req.query.get == "now") {
    res.json(sensordata);
  } else if (req.query.get == "timeline") {
    let limit = Number(req.query.limit);
    limit = isNaN(limit) ? 10 : limit;
    let data = await timeline.find().sort({ _id: -
1 }).limit(limit).toArray();

    res.json(data);
  }
```

```
  res.status(200);
  res.end();
});

app.use("/assets", express.static(path.join(__dirname, "assets")));

app.listen(port, () => {
  console.log(`Web listening at http://localhost:${port}`);
});

/*
########################################################################
################################# */
const broker = require("aedes")();
const broker_server = require("net").createServer(broker.handle);
const broker_port = process.env.MQTT_PORT;

let clients = new Map();

broker.authenticate = (client, username, password, callback) => {
  password = Buffer.from(password, "base64").toString();
  if (password == process.env.MQTT_PASSWORD) {
    return callback(null, true);
  } else {
    return callback(new Error(`AEDES : ${client.id} is blocked`));
  }
};

broker_server.listen(broker_port, function () {
  console.log("server started and listening on port ", broker_port);
});

broker.on("clientReady", (client) => {
  console.log(`BROKER : Client (${client.id}) connected.`);
  clients.set(client.id, client);
});

broker.on("clientDisconnect", (client) => {
  console.log(`BROKER : Client (${client.id}) disconnected.`);
  clients.delete(client.id);
});

async function storeData(data) {
  const query = { timeline: data.timeline };

  if ((await (await timeline.find(query).toArray()).length) > 0) {
    let timelinedata = await (await
timeline.find(query).toArray())[0].value;
    let round = Math.round((timelinedata + data.value) / 2);
    timeline.updateOne(
      query,
```

```
      {
        $set: { timeline: data.timeline, value: round },
      },
      (error, result) => {
        if (error) {
          console.log("Error Updating Data");
        } else {
          //console.log("Data Updated");
        }
      }
    );
  } else {
    timeline.insertOne(
      { timeline: data.timeline, value: data.value },
      (error, result) => {
        if (error) {
          console.log("Error Inserting Data");
        } else {
          //console.log("Data Inserted");
        }
      }
    );
  }
}

broker.on("publish", (_packet, client) => {
  if (client == null) {
    return;
  }
  let now = Date.now();
  sensordata.time = now;
  let timeline = new
Date(now).toISOString().split(".")[0].split(":");
  timeline = `${timeline[0]}:${timeline[1]}`;
  sensordata.timeline = timeline;
  sensordata.value =
    Number(_packet.payload) == null ? 0 : Number(_packet.payload);
  storeData(sensordata).then(
    (resolve) => {},
    (reject) => {
      console.log("Data saving failed.");
    }
  );
  console.log(`BROKER : ${client.id} published message :
(${_packet.payload})`);
});

setInterval(() => {
  for (var client of clients.entries()) {
    var packet = {
      cmd: "publish",
      messageId: 42,
```

```
      qos: 2,
      dup: false,
      topic: "command",
      payload: Buffer.from("request"),
      retain: false,
    };
    client[1].publish(packet, (err) => {
      //console.log(`BROKER : Request client (${client[0]}).`);
    });
  }
}, 1000);

/*
#########################################################################
################################ */
```

Program Node Sensor

```
#include <Arduino.h>
#include <EEPROM.h>
#define SSD_ADDRESS 0
#define WIFI_PASSWORD_ADDRESS 33
#define HOST_ADDRESS 97
#define NAME_ADDRESS 351
#define MQTT_PASSWORD 384

#define pinButton D1
#define pinLed D0
#define pinBuzzer D2
#define MQ135 A0

struct Config
{
    char ssid[32];
    char wifi_password[63];
    char host[253];
    char name[32];
    char mqtt_password[63];
};

namespace config
{
    void init();
    void reset();
    bool setData(Config dataConfig);
}

extern Config DataConfig;

extern void (*resetFunc)(void);
```

```
void (*resetFunc)(void) = 0;

Config DataConfig = {
    "",
    "",
    "",
    "",
    ""};

void config::init()
{
    EEPROM.begin(500);
    EEPROM.get(0, DataConfig);
}

bool config::setData(Config dataConfig)
{
    EEPROM.put(0, dataConfig);
    return EEPROM.commit();
}

void config::reset()
{
    digitalWrite(pinLed, HIGH);
    Serial.println("Resetting...");
    Config resetConfig = {"",
                          "",
                          "",
                          "",
                          ""};
    config::setData(resetConfig);
    digitalWrite(pinBuzzer, LOW);
    resetFunc();
}


#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

extern ESP8266WebServer http;
void HTTPServer_init();

String homepage =
R"(<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1.0, shrink-to-
fit=no"
```

```
/>
<title>Setup Page</title>
<style>
  body {
    margin: 0;
    font-family: var(--bs-body-font-family);
    font-size: var(--bs-body-font-size);
    font-weight: var(--bs-body-font-weight);
    line-height: var(--bs-body-line-height);
    color: var(--bs-body-color);
    text-align: var(--bs-body-text-align);
    background-color: var(--bs-body-bg);
    -webkit-text-size-adjust: 100%;
    -webkit-tap-highlight-color: transparent;
  }
  *,
  ::after,
  ::before {
    box-sizing: border-box;
  }
  user agent stylesheet body {
    display: block;
    margin: 8px;
  }
  .navigation-clean {
    background: #fff;
    padding-top: 0.75rem;
    padding-bottom: 0.75rem;
    color: #333;
    border-radius: 0;
    box-shadow: none;
    border: none;
    margin-bottom: 0;
  }
  @media (min-width: 768px) {
    .navigation-clean {
      padding-top: 1rem;
      padding-bottom: 1rem;
    }
  }
  .navigation-clean .navbar-brand {
    font-weight: 700;
    color: inherit;
  }
  .navigation-clean .navbar-brand:hover {
    color: #222;
  }
  .navbar {
    position: relative;
    display: flex;
    flex-wrap: wrap;
    align-items: center;
```

```
  justify-content: space-between;
  padding-top: 0.5rem;
  padding-bottom: 0.5rem;
}
*,
::after,
::before {
  box-sizing: border-box;
}
@media (max-width: 767px) {
  .contact-clean {
    padding: 20px 0;
  }
  .contact-clean form {
    padding: 30px;
  }
}
.contact-clean h2 {
  text-align: center;
  margin-top: 5px;
  font-weight: 700;
  font-size: 28px;
  margin-bottom: 36px;
  color: inherit;
}
.contact-clean form .form-control {
  background: #fff;
  border-radius: 2px;
  box-shadow: 1px 1px 1px rgba(0, 0, 0, 0.05);
  outline: 0;
  color: inherit;
  padding-left: 12px;
  height: 42px;
}
.contact-clean form .form-control:focus {
  border: 1px solid #b2b2b2;
}
.contact-clean form textarea.form-control {
  min-height: 100px;
  max-height: 260px;
  padding-top: 10px;
  resize: vertical;
}
.contact-clean form .btn {
  padding: 16px 32px;
  border: none;
  background: 0 0;
  box-shadow: none;
  text-shadow: none;
  opacity: 0.9;
  text-transform: uppercase;
  font-weight: 700;
```

```
  font-size: 13px;
  letter-spacing: 0.4px;
  line-height: 1;
  outline: 0 !important;
}
.contact-clean form .btn:hover {
  opacity: 1;
}
.contact-clean form .btn:active {
  transform: translateY(1px);
}
.contact-clean form .btn-primary {
  background-color: #055ada !important;
  margin-top: 15px;
  color: #fff;
}
.contact-clean p {
  text-align: center;
  font-size: 25px;
}
.contact-clean {
  background: #f1f7fc;
  padding: 100px 0;
  height: 90%;
}
.contact-clean form {
  max-width: 480px;
  max-height: 486px;
  width: 90%;
  margin: auto;
  background-color: #fff;
  padding: 40px;
  border-radius: 4px;
  color: #505e6c;
  box-shadow: 1px 1px 5px rgba(0, 0, 0, 0.1);
}
.container,
.container-fluid,
.container-lg,
.container-md,
.container-sm,
.container-xl,
.container-xxl {
  width: 100%;
  padding-right: var(--bs-gutter-x, 0.75rem);
  padding-left: var(--bs-gutter-x, 0.75rem);
  margin-right: auto;
  margin-left: auto;
}
@media (min-width: 576px) {
  .container,
  .container-sm {
```

```
          max-width: 540px;
        }
      }
      @media (min-width: 768px) {
        .container,
        .container-md,
        .container-sm {
          max-width: 720px;
        }
      }
      @media (min-width: 992px) {
        .container,
        .container-lg,
        .container-md,
        .container-sm {
          max-width: 960px;
        }
      }
      @media (min-width: 1200px) {
        .container,
        .container-lg,
        .container-md,
        .container-sm,
        .container-xl {
          max-width: 1140px;
        }
      }
      @media (min-width: 1400px) {
        .container,
        .container-lg,
        .container-md,
        .container-sm,
        .container-xl,
        .container-xxl {
          max-width: 1320px;
        }
      }
      *,
      ::after,
      ::before {
        box-sizing: border-box;
      }
      :root {
        --bs-font-sans-serif: system-ui, -apple-system, "Segoe UI",
Roboto,
          "Helvetica Neue", Arial, "Noto Sans", "Liberation Sans",
sans-serif,
          "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol",
          "Noto Color Emoji";
        --bs-font-monospace: SFMono-Regular, Menlo, Monaco, Consolas,
          "Liberation Mono", "Courier New", monospace;
        --bs-gradient: linear-gradient(
```

```
            180deg,
            rgba(255, 255, 255, 0.15),
            rgba(255, 255, 255, 0)
          );
          --bs-body-font-family: var(--bs-font-sans-serif);
          --bs-body-font-size: 1rem;
          --bs-body-font-weight: 400;
          --bs-body-line-height: 1.5;
          --bs-body-color: #212529;
          --bs-body-bg: #fff;
        }
        .navigation-clean .navbar-brand {
          font-weight: 700;
          color: inherit;
        }
        .navbar-brand {
          padding-top: 0.3125rem;
          padding-bottom: 0.3125rem;
          margin-right: 1rem;
          font-size: 1.25rem;
          text-decoration: none;
          white-space: nowrap;
        }
        .contact-clean form .form-control {
          background: #fff;
          border-radius: 2px;
          box-shadow: 1px 1px 1px rgb(0 0 0 / 5%);
          outline: 0;
          color: inherit;
          padding-left: 12px;
          height: 42px;
        }
        .form-control {
          display: block;
          width: 100%;
          padding: 0.375rem 0.75rem;
          font-size: 1rem;
          font-weight: 400;
          line-height: 1.5;
          color: #212529;
          background-color: #fff;
          background-clip: padding-box;
          border: 1px solid #ced4da;
          -webkit-appearance: none;
          -moz-appearance: none;
          appearance: none;
          border-radius: 0.25rem;
          transition: border-color 0.15s ease-in-out, box-shadow 0.15s
ease-in-out;
        }
        .btn-primary {
          color: #fff;
```

```
        background-color: #0d6efd;
        border-color: #0d6efd;
      }
      .btn {
        display: inline-block;
        font-weight: 400;
        line-height: 1.5;
        color: #212529;
        text-align: center;
        text-decoration: none;
        vertical-align: middle;
        cursor: pointer;
        -webkit-user-select: none;
        -moz-user-select: none;
        user-select: none;
        background-color: transparent;
        border: 1px solid transparent;
        padding: 0.375rem 0.75rem;
        font-size: 1rem;
        border-radius: 0.25rem;
        transition: color 0.15s ease-in-out, background-color 0.15s
ease-in-out,
          border-color 0.15s ease-in-out, box-shadow 0.15s ease-in-
out;
      }
      [type="button"],
      [type="reset"],
      [type="submit"],
      button {
        -webkit-appearance: button;
      }
      button,
      select {
        text-transform: none;
      }
      button,
      input,
      optgroup,
      select,
      textarea {
        margin: 0;
        font-family: inherit;
        font-size: inherit;
        line-height: inherit;
      }
      *,
      ::after,
      ::before {
        box-sizing: border-box;
      }
      .mb-3 {
        margin-bottom: 1rem !important;
```

```
      }
      .text-end {
        text-align: right !important;
      }
    </style>
  </head>

  <body style="height: 100vh">
    <nav class="navbar navbar-light navbar-expand-lg navigation-
clean">
      <div class="container">
        <a class="navbar-brand" href="/">Sistem Pemantauan Udara
Dashboard</a>
        <div class="collapse navbar-collapse" id="navcol-2"></div>
      </div>
    </nav>
    <section class="contact-clean">
      <form method="post" action="/setup">
        <h2 class="text-center">Setup Form</h2>
        <div class="mb-3">
          <input class="form-control" type="text" placeholder="WiFi
SSID"
          name="ap"
        </div>
        <div class="mb-3">
          <input class="form-control" type="password" name="ap-
password"
          placeholder="Wifi Password"
        </div>
        <div class="mb-3">
          <input class="form-control" type="text"
placeholder="Broker Address"
          name="ip"
        </div>
        <div class="mb-3">
          <input class="form-control" type="text" name="name"
          placeholder="Devices Name"
        </div>
        <div class="mb-3">
          <input class="form-control" type="password"
placeholder="MQTT Password"
          name="mqtt-password"
        </div>

        <div class="mb-3">
          <small class="form-text text-danger"></small>
        </div>
        <div class="mb-3"></div>
        <div class="text-end mb-3">
          <button class="btn btn-primary"
type="submit">Submit</button>
        </div>
```

```
      </form>
    </section>
    <script src="/assets/bootstrap/js/bootstrap.min.js"></script>
  </body>
</html>
)"
    ;

#include <string.h>

ESP8266WebServer http(80);

void HTTPServer_init()
{
    http.on("/setup", []()
            {
                Config data = {
                    "",
                    "",
                    "",
                    "",
                    ""};

                strcpy(data.ssid, http.arg("ap").c_str());
                strcpy(data.wifi_password, http.arg("ap-
password").c_str());
                strcpy(data.host, http.arg("ip").c_str());
                strcpy(data.name, http.arg("name").c_str());
                strcpy(data.mqtt_password, http.arg("mqtt-
password").c_str());

                Serial.println(data.ssid);
                Serial.println(data.wifi_password);
                Serial.println(data.host);
                Serial.println(data.name);
                Serial.println(data.mqtt_password);

                if (config::setData(data))
                {
                    http.send(200, "text/plain", "SUCCESS!");
                }
                else
                {
                    http.send(200, "text/plain", "Failed!");
                }
                delay(1000);
                resetFunc();
            });

    http.on("/", []()
            { http.send(200, "text/html", homepage); });
```

```
    http.begin();
}


#include <PubSubClient.h>
#include <string>

const int threshold = 250;
bool setupmode = false;

WiFiClient wifiClient;
PubSubClient client(wifiClient);

void callback(char *topic, byte *payload, unsigned int length)
{
  digitalWrite(pinLed, HIGH);
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 2; i < length; i++)
  {
    Serial.print((char)payload[i]);
  }
  int readSensor = analogRead(MQ135);

  if (readSensor > threshold)
  {
    if (digitalRead(pinBuzzer) == LOW)
    {
      digitalWrite(pinBuzzer, HIGH);
    }
    else
    {
      digitalWrite(pinBuzzer, LOW);
    }
  }
  else
  {
    digitalWrite(pinBuzzer, LOW);
  }
  std::string encoded = std::to_string(readSensor);
  const char *playload = encoded.c_str();
  client.publish(DataConfig.name, playload);
  digitalWrite(pinLed, LOW);
  Serial.println();
}

void setup()
{
  Serial.begin(9600);
  Serial.println("Program Started!");
  pinMode(MQ135, INPUT);
```

```
pinMode(pinLed, OUTPUT);
pinMode(pinBuzzer, OUTPUT);
digitalWrite(pinLed, LOW);
digitalWrite(pinBuzzer, LOW);
pinMode(pinButton, INPUT_PULLUP);
config::init();
setupmode = strcmp(DataConfig.name, "") == 0;
delay(3000);
int resetcounter = 0;

while (digitalRead(pinButton) == 0)
{
  if (resetcounter > 3)
  {
    digitalWrite(pinBuzzer, HIGH);
    delay(500);
    config::reset();
  }
  Serial.printf("\nReset! %d", resetcounter);
  resetcounter = resetcounter + 1;
  delay(1000);
}
Serial.println();
if (setupmode)
{
  if (WiFi.softAP("Sensor Node"))
  {
    Serial.println("Hostpot created.");
    digitalWrite(pinLed, LOW);
    delay(500);
    digitalWrite(pinLed, HIGH);
    delay(500);
    digitalWrite(pinLed, LOW);
    delay(500);
    digitalWrite(pinLed, HIGH);
    delay(500);
    Serial.println("Menjalankan webserver...");
    HTTPServer_init();
  }
  else
  {
    Serial.println("Hostpot creating is failed.");
  }
}
else
{
  digitalWrite(pinLed, LOW);
  delay(1000);
  digitalWrite(pinLed, HIGH);

  Serial.printf("Device name : %s\n", DataConfig.name);
  WiFi.mode(WIFI_STA);
```

```
    WiFi.begin(DataConfig.ssid, DataConfig.wifi_password);

    Serial.print("WiFi connecting");
    while (WiFi.status() != WL_CONNECTED)
    {
      Serial.print(".");
      delay(500);
    }

    Serial.print("\nIP address: ");
    Serial.println(WiFi.localIP());
    Serial.printf("Connecting to %s...\n", DataConfig.host);
    client.setServer(DataConfig.host, 1883);
    client.setCallback(callback);
  }
}

void reconnect()
{ // Loop until we're reconnected
  digitalWrite(pinLed, HIGH);
  digitalWrite(pinBuzzer, LOW);
  while (!client.connected())
  {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = DataConfig.name;
    clientId += String("-");
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str(), DataConfig.name,
DataConfig.mqtt_password))
    {
      Serial.println("reconnected");
    }
    else
    {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(1000);
    }
  }
}

void loop()
{
  if (setupmode)
  {
    http.handleClient();
  }
  else
```

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

KARTU BIMBINGAN

SISTEM KOMPUTER ILMU KOMPUTER UPI YPTK

PADANG

Nama	:	Taufiqurahman
No Bp	:	18101152620034
Judul Skripsi	:	**PERCANCANGAN SISTEM PEMANTAUAN KUALITAS UDARA INTERNET OF THING MENGGUNAKAN MQ-135 DITAMPILKAN MELALUI WEB**
Pembimbing	:	1. Hadi Syahpuutra, S.Kom, M.Kom 2. Ondra Eka Putra, S.Kom, M.Kom

HARI / TANGGAL	KOMENTAR	TANDA TANGAN
12 November 2021	Pengajuan BAB I	
30 November 2021	ACC BAB I Revisi BAB 2	
13 Desember 2021	Pengajuan BAB 3 ACC BAB 2	
20 Desember 2021	Perbaiki BAB 3 (kerangka penelitian buat dari atas kebawah)	
22 Desember 2021	Pengajuan BAB 4	
14 Januari 2021	Lanjut Alat	
20 Januari 2022	Revisi BAB 4 (semua deskripsi gambar dituliskan gambar berapanya)	

24 Januari 2021	Pengajuan BAB 4 dan video alat	
4 Februari 2022	Revisi BAB 4 (perbaiki modul program,flowchart,listing program lampirkan)	
11 Februari 2022	Revisi BAB 5 perbaiki bab 5 & acc alat	
14 Februari 2022	Perbaiki bab 5 dan acc alat	
17 Februari	ACC SKRIPSI	