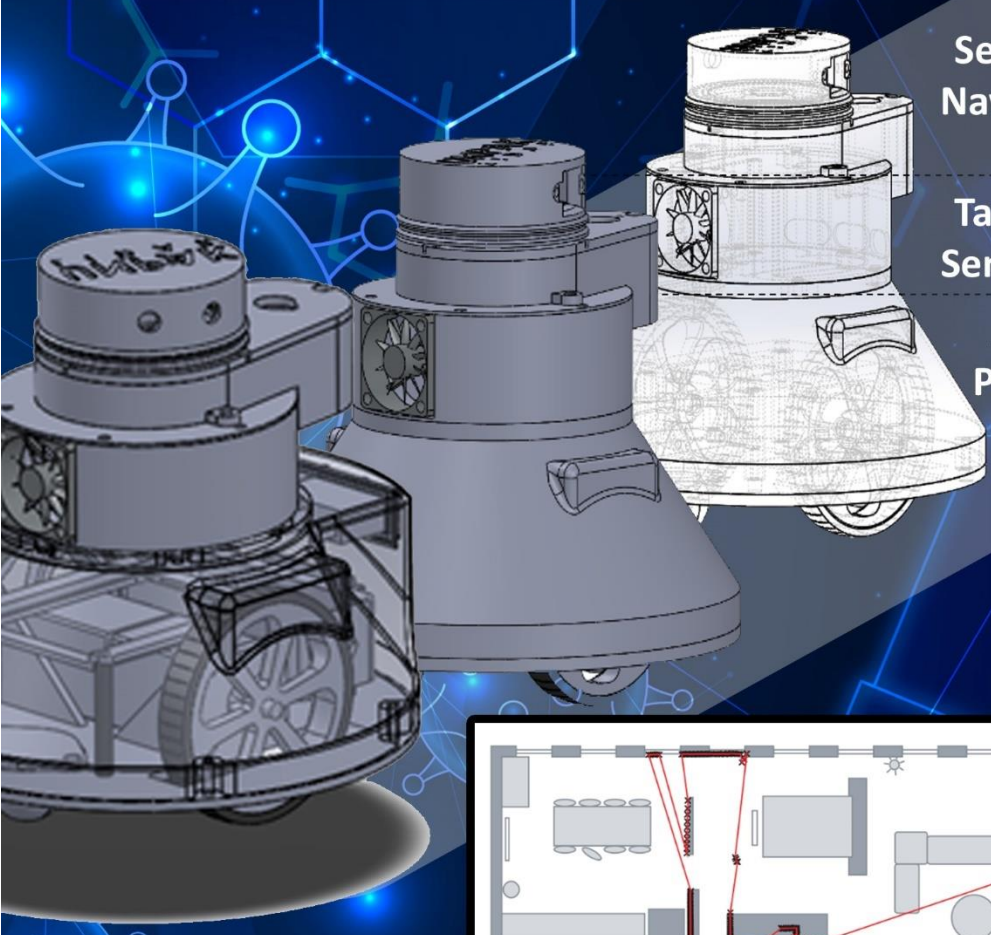
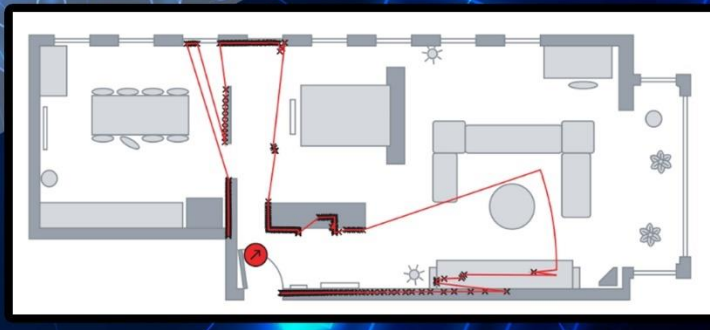
Asep Denih, S.Kom., M.Sc., Ph.D.





Reference Book

ROBOT PINTAR PENDETEKSI KUALITAS UDARA DI DALAM RUANGAN

BERBASIS INTERNET OF THINGS (IOT)

ROBOT PINTAR PENDETEKSI KUALITAS UDARA DI DALAM RUANGAN BERBASIS INTERNET OF THINGS (IOT)

Asep Denih, S.Kom., M.Sc., Ph.D.

ROBOT PINTAR PENDETEKSI KUALITAS UDARA DI DALAM RUANGAN BERBASIS INTERNET OF THINGS (IOT)

Asep Denih, S.Kom., M.Sc., Ph.D.
Editor: Ema Kurnia, S.Kom., M.Sc.

Printed,
ISBN: 978-623-6961-97-1

Publisher:
Komojoyo Press
Jl. Komojoyo 21A, RT11, RW4, Mrican
Caturtunggal, Depok Sleman 55281
Anggota IKAPI No. 125/DIY/2020

KATA PENGANTAR

Puji dan syukur atas berkah rahmat Tuhan Yang Maha Esa. Alhamdulillah, Buku Robot Pintar Pendeteksi Kualitas Udara di dalam Ruangan Berbasis *Internet of Things (IoT)* dapat diselesaikan dengan baik.

Penyusunan buku ini disusun dalam beberapa bagian, yaitu Kualitas udara dalam ruangan, *robot mapping*, *Internet of Things*, *Microcontroller* ESP32, sensor gas SGP30, sensor gas MQ-2. sensor *Time of Flight* VL53L0X, sensor MPU6050, motor stepper, sensor suhu dan kelembaban DHT11, metode *fuzzy logic*, trigonometri, regresi linear serta perancangan robot seperti desain sistem mekanik, sistem listrik, perangkat lunak, *hardware*, kelistrikan dan *software*, untuk pengoperasian robot disajikan secara terpisah dalam Buku "Pengoperasian Robot Pendeteksi Kualitas Udara Dalam Ruangan Berbasis IoT". Besar harapan kami buku ini dapat memberikan manfaat dalam proses kegiatan belajar mengajar mahasiswa dan sebagai referensi dalam melakukan penelitian. Akhir kata, kami menyadari buku ini masih perlu dilakukan penyempurnaan secara berkala, dan semoga buku ini dapat memberikan manfaat dalam menunjang proses pembelajaran.

Bogor, November 2022

Penyusun

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

Latar Belakang

Udara merupakan salah satu komponen lingkungan yang perlu dipelihara agar dapat memberikan daya dukungan bagi mahluk hidup secara optimal. Udara dapat dikelompokkan menjadi udara luar ruangan *(outdoor air)* dan udara dalam ruangan *(indoor air)*. Kualitas udara dalam ruangan sangat berperan penting pada kesehatan manusia, karena hampir 90% hidup manusia berada didalam ruangan. Sebanyak 400 sampai 500 juta orang sedang menghadapi masalah polusi udara dalam ruangan khususnya di negara berkembang. Terdapat beberapa komponen kualitas kimia udara dalam ruangan meliputi Volatile Organic Compound (VOC), karbon dioksida (CO_2) dan asap rokok (Prabowo, 2018). Sementara itu, NIOSH dalam penelitiannya menyebutkan ada enam sumber pencemaran di dalam ruangan yaitu, kurangnya ventilasi udara (52%), adanya kontaminasi di dalam ruangan (16%), kontaminasi dari luar ruangan (10%), mikroba (5%), bahan material bangunan (4%), lain-lain (13%). Dalam hal ini gangguan ventilasi menjadi masalah utama kualitas udara dalam ruangan. (NIOSH, 2017).

Gangguan ventilasi udara menyebabkan perbedaan distribusi dan tekanan udara pada ruangan sehingga gas berbahaya dapat menyebar dengan konsentrasi yang berbeda tergantung jarak dari sumbernya. Mengingat sifat udara yang bergerak dari satu titik ke titik lainnya

oleh perbedaan tekanan (adveksi). Selain itu partikel gas akan berpindah dari konsentrasi tinggi ke konsentrasi yang lebih rendah (difusi) (Widyantara, 2018). Dari perbedaan distribusi dan konsentrasi tersebut, pengukuran udara pada satu titik akan mengurangi ketepatan informasi.

Tahap awal pengumpulan data tentang parameter pencemaran udara perlu dilakukan pemetaan lokasi, dan hasil pemetaan ini merupakan masukan untuk menentukan titik titik pengambilan dan pengukuran udara (Prabowo, 2018). Untuk mempermudah pengumpulan data, teknologi robotika dapat dimanfaatkan sebagai alat untuk pemetaan sekaligus mengukur kualitas udara salah satunya adalah robot *mapping*. Robot *mapping* merupakan salah satu jenis robot yang memiliki beberapa modul terintegrasi sehingga robot dapat berfungsi secara efektif sebagai pembuatan peta *(mapping)*, lokalisasi, eksplorasi, perencanaan dan penghindaran tabrakan pada lingkungan yang tidak diketahui (Xingdong, 2018).

Pada referensi sebelumnya dengan judul “Rancang Bangun Robot SAR Sebagai Pendeteksi Gas Beracun Pra Evakuasi” oleh Bima Renaldi (2020), yang memanfaatkan teknologi robotika sebagai pendeteksi gas berbahaya dan dapat dikontrol dari jarak jauh yaitu koneksi internet. Namun sistem navigasi robot masih menggunakan kontrol manual serta robot tidak dapat menentukan titik koordinat gas yang terdeteksi.

Internet of Things atau yang disingkat IoT adalah suatu konsep dimana objek tertentu yang mempunyai kemampuan untuk mentransfer data melalui jaringan internet tanpa memerlukan adanya interaksi dari manusia ke manusia ataupun dari manusia ke perangkat komputer. IoT sendiri memiliki beberapa unsur pembentuk yaitu kecerdasan buatan, konektifitas, sensor, active engangement dan perangkat berukuran kecil (Denih, 2020).

Buku ini berisi bab 1 pendahuluan yang menjelaskan tentang latar belakang rumusan masalah, tujuan, ruang lingkup dan manfaat, robot, robot *mapping*, *internet of things*, *microcontroller* ESP32, sensor gas SGP30, sensor gas MQ-2, sensor *time-of-flight* VL53L0X, sensor MPU6050, motor stepper, sensor suhu dan kelembaban DHT11, metode fuzzy logic, metode trigonometri dan metode regresi linear, bab 2 metodologi yang menjelaskan tentang kualitas udara dalam ruangan, tentang perancanaan rancangan penelitian, pengetesan komponen, desain sistem mekanik, desain sistem listrik, desain perangkat lunak, tes fungsional, integrasi atau perakitan, tes fungsional keseluruhan sistem, application, proses implementasi dan penerapan metode pada robot, bab 3 pembahasan menjelaskan tentang hasil tes fungsional sistem, pengujian, struktural, pengujian fungsional, uji coba validasi, bab 4 kesimpulan menjelaskan kesimpulan. Ruang lingkup pembahasan buku ini meliputi; perancangan dan implementasi model robot pintar, menggunakan sensor SGP30, sensor MQ-2, sensor VL53L0X dan sensor MPU6050, mikrokontroller yang digunakan adalah ESP32,

perancangan aplikasi website yang berfungsi sebagai media monitoring dan *control* robot, implementasi *Internet of Things* (IoT), robot dapat menelusuri ruangan mendeteksi dinding dan benda, robot dapat mendeteksi kualitas udara meliputi CO2, VOC dan Asap rokok, robot dapat mengirimkan informasi melalui koneksi internet.

Buku ini menyajikan sebuah solusi untuk merancang sebuah model robot yang dapat memetakan sebuah ruangan dan mendeteksi kualitas udara. Konsepnya dengan menggunakan beberapa sensor sederhana yaitu sensor jarak dan *sensor accelerometer* untuk pemetaan pada robot. Selain itu robot dapat mendeteksi gas menggunakan beberapa sensor gas sederhana. *Output* pada robot akan ditampilkan pada sebuah web untuk memudahkan pengguna dalam monitoring dan kontrol robot. Selain itu model robot ini bermanfaat untuk dapat memetakan sebuah ruangan dan mengetahui titik – titik gas yang berbahaya pada ruangan. Tujuannya adalah memodelkan sebuah robot yang dapat memetakan sebuah ruangan dan mendeteksi kualitas udara dengan konsep menggunakan beberapa sensor sederhana yaitu sensor jarak dan sensor accelerometer untuk pemetaan pada robot serta mendeteksi gas menggunakan beberapa sensor gas sederhana yang outputnya akan ditampilkan pada sebuah web untuk memudahkan pengguna dalam monitoring dan kontrol robot.

Kualitas Udara dalam Ruangan

Sumber polusi udara dalam ruang dapat berasal dari bahan-bahan sintetis dan beberapa bahan alamiah antara lain CO2, asap rokok dan *VOC* (*Volatile Organic Compound*). Beberapa jenis *VOC* dikenal bersifat racun *(toxic)*, menimbulkan perubahan sel dan kanker. Salah satu jenis *VOC* yang penting adalah formaldehid. Dalam konsentrasi normal dan waktu yang relatif pendek, pada umumnya *VOC* kurang serius bagi kesehatan manusia. Contoh senyawa *VOC* adalah senyawa BTEX atau benzena, toulena, etilbenzena, dan xyelena (Kuat Prabowo, 2018). Berikut adalah persyaratan minimum kualitas kimia udara dalam ruangan menurut peraturan menteri kesehatan Republik Indonesia nomor 1077/MENKES/PER/V/2011 tentang pedoman penyehatan udara dalam ruang rumah dapat dilihat pada Tabel 1. Persyaratan Kualitas Kimia Udara dalam Ruangan.

Tabel 1. Persyaratan Kualitas Kimia Udara dalam Ruangan

No	Jenis Parameter	Satuan	Kadar Maksimal	Keterangan
1	Karbondioksida (CO2)	ppm	1000	8 jam
2	Volatile Organic Compound (VOC)	ppm	3	8 jam
3	Environmental Tobaco Smoke (ETS) / Asap Rokok	$\mu g/m^3$	35	24 jam
4	Suhu	^{O}C	10 – 30	-
5	Kelembaban	%Rh	40 – 60	-

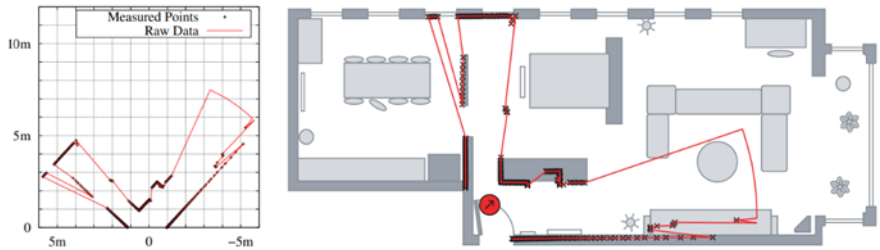
Dalam menentukan kualitas udara, metode fuzzy merupakan suatu cara yang tepat untuk memetakan ruang input ke dalam suatu ruang output (Pratama & Fitriani, 2017). Metode fuzzy yang digunakan adalah metode fuzzy Mamdani yang telah dilakukan dalam beberapa penelitian salah satunya "Analysis of Air Pollution Levels in DKI Jakarta Province Using the Mamdani Fuzzy Inference System Method" pada penelitian tersebut metode fuzzy mamdani dapat digunakan sebagai penganalisa tingkat polusi udara.

Robot

Robot berasal dari bahasa Czech, Robota, yang berarti pekerja. Kata robot diperkenalkan dalam bahasa Inggris pada tahun 1921 oleh Wright Karel Capek pada suatu drama, Rossum's Universal Robots. Robot adalah mesin hasil rakitan karya manusia yang mampu bekerja terus- menerus tanpa mengenal lelah. Pada awalnya robot diciptakan sebagai pengganti pekerja manusia, akan tetapi untuk jangka waktu ke depan robot akan mampu mengambil alih posisi manusia sepenuhnya dan bahkan mengganti ras manusia dengan beragam jenisrobot. saat ini hampir semua industri manufaktur menggunakan robot. Hal itu karena biaya operasional per jam untuk robot jauh lebih murah dibandingkan menggunakan tenaga manusia. Robot pada awalnya hanya digunakan untuk melakukan fungsi tertentu saja, seperti pengocoran, penyolderan, atau lainnya, namun sekarang sudah banyak robot yang dapat melakukan banyak fungsi.

Robot Mapping

Salah satu jenis robot yang memiliki beberapa modul terintegrasi sehingga robot dapat berfungsi secara efektif sebagai pembuatan peta *(mapping)*, lokalisasi, eksplorasi, perencanaan jalur pada lingkungan yang tidak diketahui. Pembuatan peta dan lokalisasi sebuah robot biasanya dilakukan secara bersamaan sehingga menghasilkan beragam literatur diantaranya algoritma lokalisasi Monte Carlo, algoritma perencanaan jalur Dijkstra, dan yang paling sering digunakan yaitu *SLAM (Simultaneous Localization and Mapping)* (Xingdong, 2018).



Gambar 1. Robot *Mapping*

Internet of Things (IoT)

Internet of Things (IoT) adalah perangkat apapun yang terhubung ke internet sehingga dapat berkomunikasi dengan perangkat lain. *IoT* juga sebagai *network* perangkat yang terhubung dan setiap perangkat memiliki alamat IP yang berbeda. Teknologi internet memberikan kemudahan salah satunya dalam bidang *monitoring* polusi udaraa (Suriansyah, 2018). Internet of Things atau yang disingkat IoT adalah suatu konsep dimana objek tertentu yang mempunyai

kemampuan untuk mentransfer data melalui jaringan internet tanpa memerlukan adanya interaksi dari manusia ke manusia ataupun dari manusia ke perangkat komputer (Denih *et al*, 2020). Iot sendiri memiliki beberapa unsur pembentuk yaitu sebagai berikut:

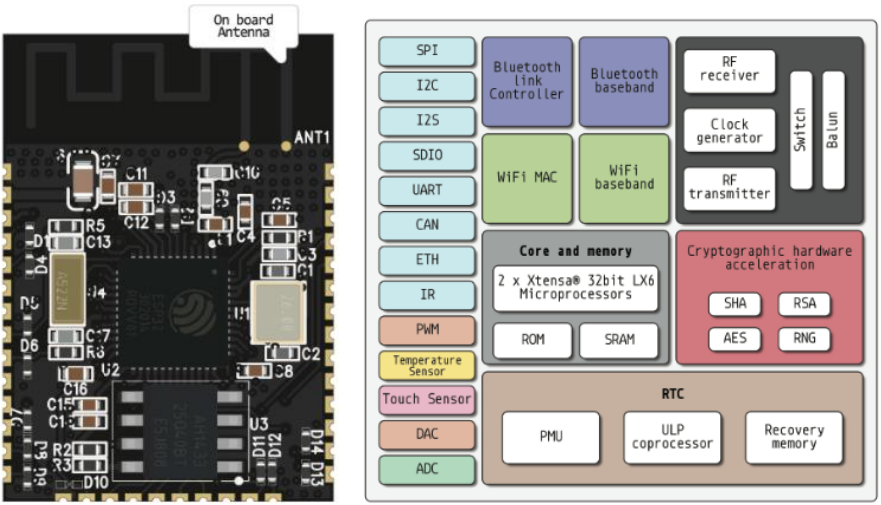
- Kecerdasan buatan (*AI*) - *IoT* membuat hampir semua mesin yang tersedia menjadi "*Smart*", jadi dalam proses pengembangan teknologi yang sudah ada dilakukan dengan cara pengumpulan data, artificial intelligence algorithm, dan jaringan yang tersedia. Contoh perkembangan dari Iot yang menggunakan kecerdasan buatan adalah perkembangan teknologi pada kulkas yang mungkin terjadi di masa depan, yakni kulkas atau lemari es yang dapat mendeteksi bahan makanan yang habis misalnya susu, selain dapat mendeteksi makanan yang hampir habis, lemari es ini juga dapat membuat sebuah pesanan ke supermarket secara otomatis.
- Konektivitas-konektivitas disini berfungsi sebagai penghubung dan pertukaran informasi yang terjadi pada *Internet of Things (IoT)*. Konektivitas ini biasanya yang dibutuhkan harus stabil namun tidak perlu dalam bentuk yang besar juga.
- Sensor-sensor merupakan perangkat yang sangat canggih dimana alat ini bisa menangkap atau mendapatkan informasi terkait dari hal hal tertentu seperti sensor gerak, suhu, udara, panas, dan lainnya.
- Keterlibatan aktif (*Active Engangement*) - *Engangement* yang sering diterapkan teknologi umumnya yang termasuk pasif. IoT

ini mengenalkan paradigma yang baru bagi konten aktif, produk, maupun keterlibatan layanan.

Perangkat berukuran kecil - Perangkat, seperti yang diperkirakan para pakar teknologi, memang menjadi semakin kecil, makin murah, dan lebih kuat dari masa ke masa. IoT memanfaatkan perangkat-perangkat kecil yang dibuat khusus ini agar menghasilkan ketepatan, skalabilitas, dan fleksibilitas yang baik.

Microcontroller ESP32

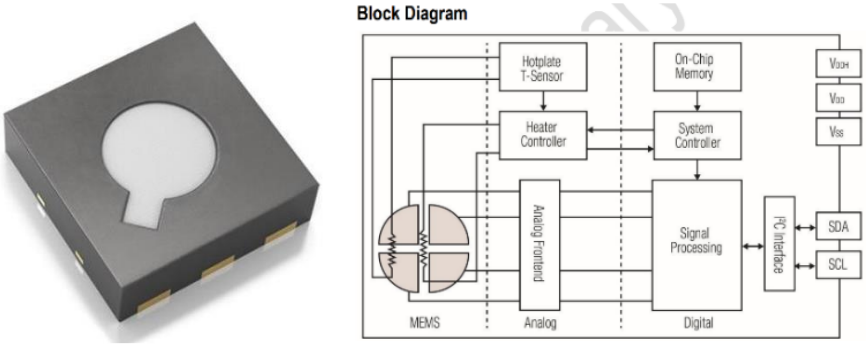
ESP32 adalah nama *microcontroller* yang dirancang oleh Espressif Systems. Espressif merupakan perusahaan Cina yang berbasis di Shanghai. ESP32 diperkenalkan sebagai solusi jaringan WiFi mandiri yang menawarkan sebagai jembatan dari *microcontroller* ke pengontrol ke WiFi selain itu juga mampu menjalankan aplikasi mandiri (Kolban, 2018).



Gambar 2. Mikrokontroler ESP32

Sensor Gas SGP30

SGP30 merupakan sensor gas multi-piksel digital yang dirancang untuk integrasi mudah ke pembersih udara, kontrol ventilasi, dan aplikasi IoT. Sensor ini diperkenalkan oleh Sensirion dengan *interface* I2C. SGP30 menggunakan algoritma *dynamic baseline compensation* dan parameter kalibrasi *on-chip* untuk memberikan dua sinyal kualitas udara yang saling melengkapi. Dua sinyal antara lain sinyal total *VOC* (*Volatile Organic Compound*) dan sinyal ekuivalen CO_2 (Datasheet SGP30, 2017).



Gambar 3. Sensor Gas SGP30

Sensor Gas MQ-2

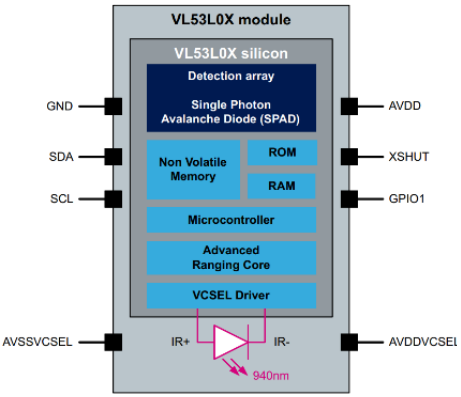
Sensor gas MQ-2 memiliki sensitivitas tinggi terhadap propana dan asap rokok, juga dapat mendeteksi gas alam yang mudah terbakar lainnya diantaranya LPG, iso-butana, metana, alkohol dan hidrogen. Sensor dapat mendeteksi gas dari 200ppm hingga 10000ppm sedangkan ouput sensor berupa tegangan listrik dari 1.0 sampai 5.0

volt. Semakin tinggi nilai output tegangan maka semakin tinggi juga kadar asap yang terdeteksi (Zholehaw, 2019).

Gambar 4. Sensor Gas MQ-2

Sensor *Time of Flight* VL53L0X

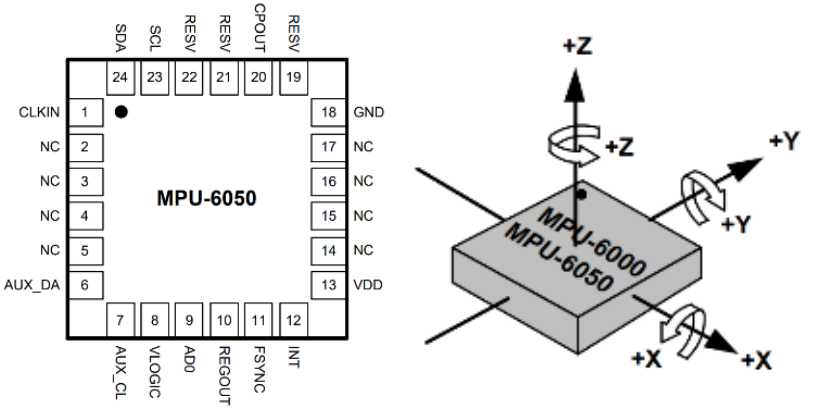
VL53L0X adalah modul sensor *Time-of-Flight* (*ToF*) yang dirancang oleh STMicroelectronics, dengan menggunakan prinsip kerja *laser-ranging* yaitu memancarkan sinar laser dan menerima kembali sinar pantulan. Dengan perhitungan tertentu maka pengukuran jarak yang akurat dapat dilakuan selain itu juga sensor ini dapat mengukur jarak absolut hingga 2 meter (Datasheet VL53L0X, 2018)



Gambar 5. Sensor *Time of Flight* VL53L0X

Sensor MPU6050

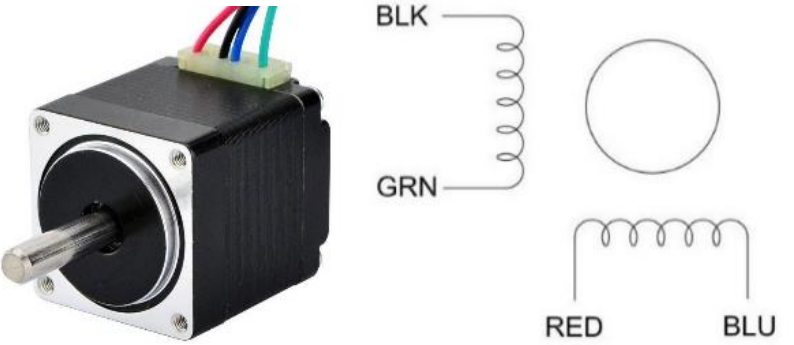
MPU-60X0 merupakan sensor 6 aksis yang menggabungkan 3 sumbu *gyroscope*, 3 sumbu *accelerometer* dan Digital Motion Processor (DMP). Data yang dikirimkan sensor berupa kemiringan pada sumbu X, Y dan Z. Dengan pengaplikasian sensor tersebut maka data sudut kemiringan atau orientasi robot dapat diketahui (MPU-6000 and MPU-6050 Datasheet, 2013).



Gambar 6. Sensor MPU6050

Motor Stepper

Motor stepper adalah salah satu aktuator untuk pengaplikasian pada robotika. Putaran atau rotasi motor memiliki sudut step yang bervariasi tergantung motor yang akan digunakan. Motor stepper dapat berputar atau berotasi dengan sudut step yang bisa bervariasi dari 0.90 hingga 900 derajat (Syahrul, 2021).



Gambar 7. Motor Stepper

Sensor Suhu dan Kelembaban DHT11

DHT11 merupakan salah satu sensor yang sensitif terhadap suhu dan kelembaban. Jangkauan pengukuran temperatur dari sensor ini adalah 0-50°C dan jangkauan pengukuran kelembaban sebesar 20-90%. Sensor DHT11 membutuhkan catu daya sebesar 3 sampai 5,5 Volt DC. Keakuratan untuk kelembaban relatifnya sebesar ± 4% dan keakuratan untuk temperatur sebesar ± 2°C.

Gambar 8. Sensor DHT11

Metode Fuzzy Logic

Logika fuzzy atau fuzzy logic sendiri merupakan salah satu jenis logika multi nilai yang dapat merepresentasikan situasi dunia nyata. Logika fuzzy juga merupakan logika yang memiliki nilai fuzzy antara benar dan salah. Jika dalam logika reguler, suatu elemen memiliki dua pilihan nilai kebenaran pada himpunannya, yaitu pada himpunan tersebut (nilai 1 → TRUE) atau tidak pada himpunan tersebut (nilai 0 → FALSE). Dalam logika fuzzy diketahui terdapat derajat kebenaran pada interval [0,1]. Logika fuzzy juga dapat digambarkan sebagai kotak yang terhubung antara ruang input dan ruang output. Kotak yang dimaksud berisi metode-metode yang digunakan dalam logika fuzzy untuk mengolah data masukan menjadi keluaran.

Metode Trigonometri

Sholih (2017) menjelaskan trigonometri merupakan salah satu cabang ilmu matematika yang merupakan ilmu mengenai hubungan relasi antara sudut dan sisi - sisi pada suatu segitiga. Dalam mempelajari segitiga pada Trigonometri, maka segitiga itu harus mempunyai tepat satu sudutnya (90°) artinya segitiga itu tidak lain adalah segitiga siku-siku. Trigonometri sering digunakan dalam kehidupan, baik secara langsung dan tidak langsung, di antaranya seperti ilmu perbintangan (Astronomi), teknik sipil, dan konstruksi. Pada Trigonometri terdapat salah satunya yaitu Tangen, (lambang tg, an; bahasa Belanda: tangens; bahasa Inggris: tangent), dalam matematika adalah perbandingan sisi segitiga yang ada di depan

sudut dengan sisi segitiga yang terletak di sudut. Nilai tangen positif di kuadran I dan III dan negatif di kuadran II dan IV.

Metode Regresi Linear

Hijriani *et al* (2016) menjelaskan analisis regresi adalah suatu metode statistik yang mengamati hubungan antara variabel terikat Y dan serangkaian variabel bebas X1,…,Xp. Tujuan dari metode ini adalah untuk memprediksi nilai Y untuk nilai X yang diberikan. Model regresi linier sederhana adalah model regresi yang paling sederhana yang hanya memiliki satu variabel bebas X. Analisis regresi memiliki beberapa kegunaan, salah satunya untuk melakukan prediksi terhadap variabel terikat Y. Persamaan untuk model regresi linier sederhana adalah sebagai berikut:

$$\mathrm{Y} = a + bx \tag{1}$$

Y adalah variabel terikat yang diramalkan, X adalah variabel bebas, a adalah intercep, yaitu nilai Y pada saat X=0, dan b adalah slope, yaitu perubahan rata-rata Y terhadap perubahan satu unit X. Koefisien a dan b adalah koefisien regresi dimana nilai a dan b dapat dicari menggunakan persamaan berikut:

$$\mathrm{b} = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2} \tag{2}$$

$$\mathrm{a} = \frac{\sum y - \mathrm{b}(\sum x)}{n} \tag{3}$$

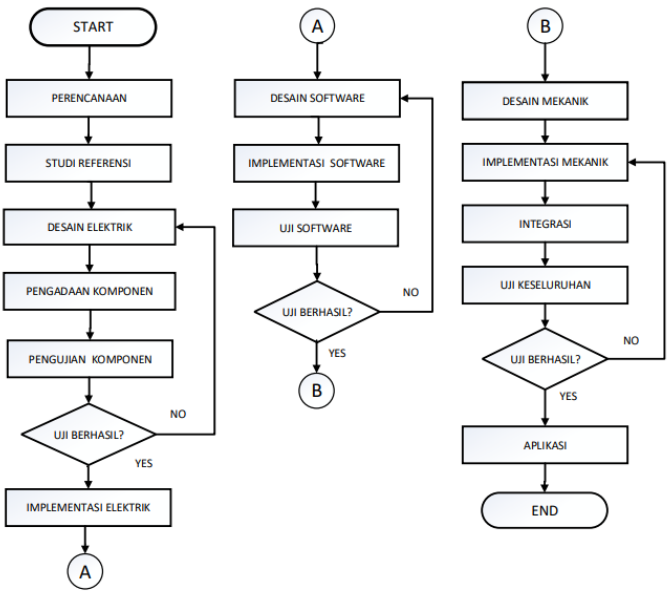
Nilai a adalah *slope*, b adalah *intercep* dan n adalah banyaknya data yang digunakan dalam perhitungan.

Visual Code

Permana dan Romadlon (2019) menjelaskan visual studio code (VS Code) ini adalah sebuah teks editor ringan dan handal yang dibuat oleh Microsoft untuk sistem operasi multiplatform, artinya tersedia juga untuk versi Linux, Mac, dan Windows. Teks editor ini secara langsung mendukung bahasa pemrograman JavaScript, Typescript, dan Node.js, serta bahasa pemrograman lainnya dengan bantuan plugin yang dapat dipasang via marketplace Visual Studio Code (seperti C++, C#, Python, Go, Java, dst). Banyak sekali fitur-fitur yang disediakan oleh Visual Studio Code, diantaranya Intellisense, Git Integration, Debugging, dan fitur ekstensi yang menambah kemampuan teks editor. Fitur-fitur tersebut akan terus bertambah seiring dengan bertambahnya versi Visual Studio Code. Pembaruan versi Visual Studio Code ini juga dilakukan berkala setiap bulan, dan inilah yang membedakan VS Code dengan teks editor-teks editor yang lain. Teks editor VS Code juga bersifat open source, yang mana kode sumbernya dapat kalian lihat dan kalian dapat berkontribusi untuk pengembangannya. Kode sumber dari VS Code ini pun dapat dilihat di link Github. Hal ini juga yang membuat VS Code menjadi favorit para pengembang aplikasi, karena para pengembang aplikasi bisa ikut serta dalam proses pengembangan VS Code ke depannya.

BAB 2
METODE HARDWARE PROGRAMMING, LOGIKA FUZZY MAMDANI, REGRESI LINEAR UNTUK KERANGKA KERJA PERHITUNGAN ROBOT

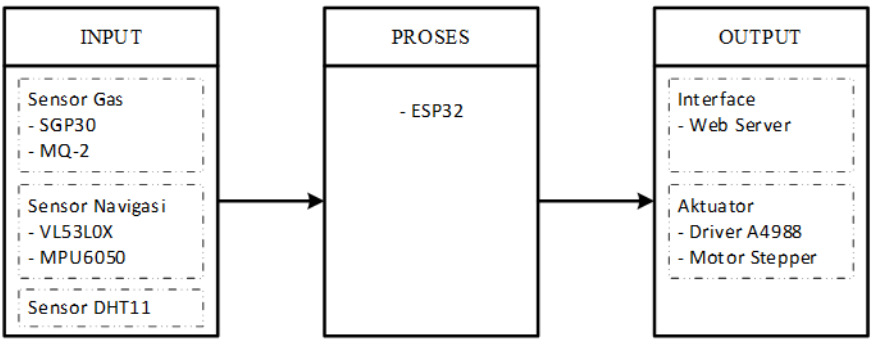
Metode yang digunakan adalah metode bidang hardware programming. Metodologi menjelaskan tentang rencana projek, desain mekanik, desain elektrik, desain perangkat lunak, uji coba fungsi, integrasi, pengujian keseluruhan. Integrasi merupakan hubungan mikrokontroler dan struktur mekanik. Berikut skema penelitiannya dapat dilihat pada Gambar 9. Metode Hardware Programming:



Gambar 9. Metode Hardware Programming

Perencanaan Rancangan Penelitian *(Project Planning)*

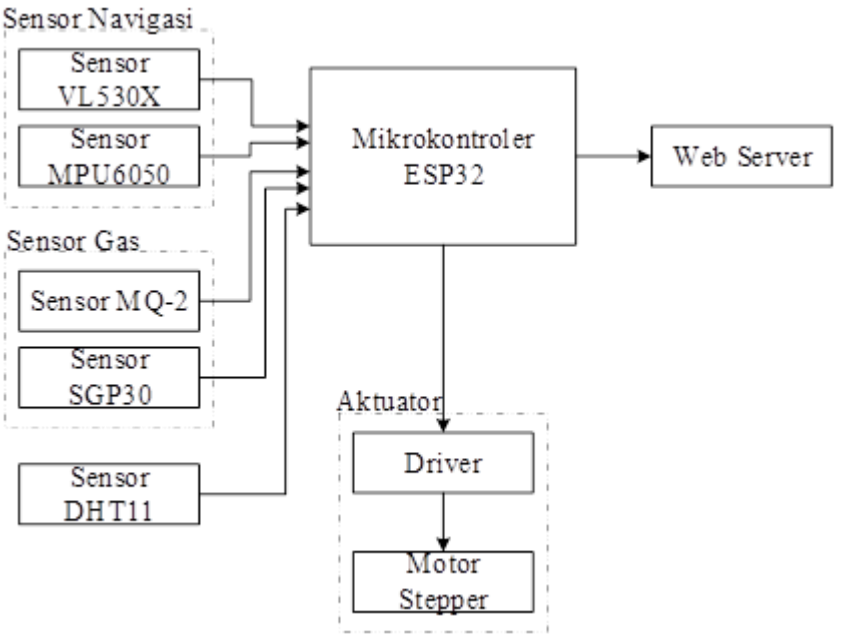
Perencanaan proyek penelitian terdapat beberapa hal yang harus dipertimbangkan agar proyek dapat berlangsung sebagaimana mestinya. Berikut adalah beberapa perencanaan dan rancangan meliputi perencanaan arsitektur, perancangan konfigurasi model dan diagram blok sistem dapat dilihat pada Gambar 10. Diagram Blok Sistem.



Gambar 10. Diagram Blok Sistem

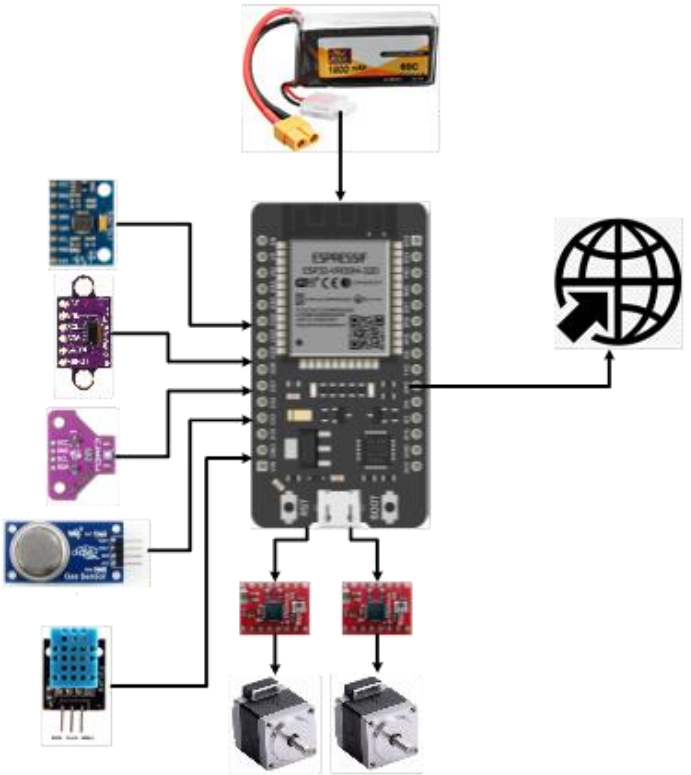
Tahapan pemilihan bahan yang akan digunakan dalam penelitian baik dari hardware maupun software. Adapun komponen hardware yang dibutuhkan yaitu microcontroller ESP32, sensor MPU6050, sensor jarak vl53l0x, sensor gas SGP30, sensor asap MQ-2, sensor temperature DHT11, aktuator motor stepper, catu daya baterai lipo 3s 1600mah. Selain itu komponen software yang digunakan antara lain Visual Studio Code, Processing IDE, Git, Solidworks, Kicad, Ms. Office, Visio, Chrome Browser serta library yang dibutuhkan untuk mempermudah dalam pembuatan robot. Perencanaan arsitektur merupakan penggabungan berbagai unsur untuk menampung suatu proses kegiatan sehingga menghasilkan suatu

keseluruhan. Pada Gambar 11. Perencanaan Arsitektur menjelaskan bahwa sensor navigasi dan sensor gas merupakan hasil nilai input yang akan diproses oleh mikrokontroller, sehingga sistem akan berjalan sesuai dengan tujuan yang telah dibuat. Mikrokontroler ESP32 mengumpulkan data dari beberapa sensor, lalu diproses oleh mikrokontroler untuk selanjutnya dikirimkan ke web server, mikrokontroler juga dapat menerima perintah dari mikrokontroler untuk menggerakan roda *actuator*



Gambar 11. Perencanaan Arsitektur

Adapun perancangan konfigurasi model dapat dilihat pada Gambar 12. Perancangan Konfigurasi Model.



Gambar 12. Perancangan Konfigurasi Model

Penelitian *(Research)*

Setelah perencanaan telah matang dilanjutkan dengan penilitian awal dan aplikasi yang akan dibuat, mulai dari pengetesan komponen (alat dan bahan) yang akan digunakan, kemungkinan rancangan awal dan akhir yaitu "Model Robot *Mapping* Pendeteksi Kualitas Udara Dalam Ruangan Berbasis *Internet of Things (IoT)".*

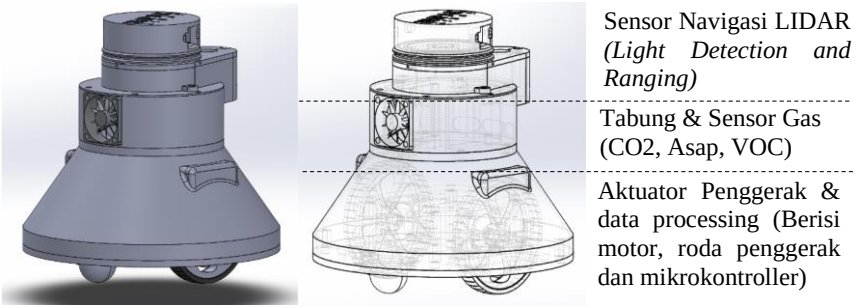
Pengetesan Komponen *(Parts Testing)*

Dalam pengetesan komponen dilakukan pengetesan alat terhadap fungsi kerja komponen berdasarkan kebutuhan sistem yang akan

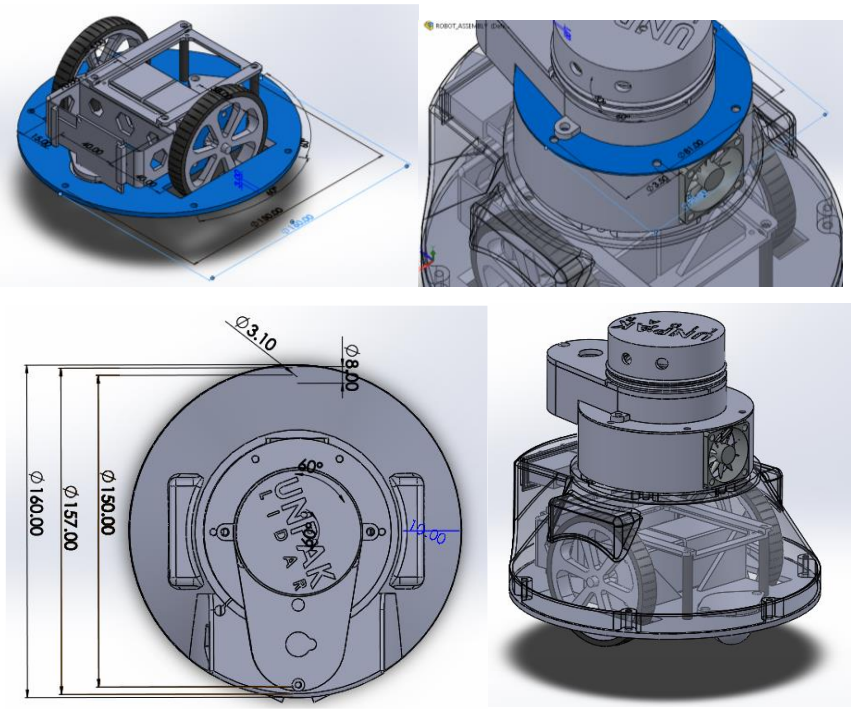
dibuat. pengetesan semua komponen yang akan digunakan pada model sistem ini. Dalam pengetesan ini dilakukan testing terhadap fungsi komponen menggunakan multimeter. Lalu pengetesan menggunakan PlatformIO serial monitor dengan cara melihat output dari masing-masing komponen.

Desain Sistem Mekanik *(Mechanical Design)*

Dalam perancangan perangkat keras *(hardware)*, desain mekanik merupakan hal penting yang harus diperimbangkan. Pada umumnya kebutuhan aplikasi terhadap desain mekanik antara lain bentuk dan ukuran, ketahanan dan fleksibilitas terhadap lingkungan, penempatan modul-modul elektronik, pengetesan sistem mekanik yang telah dirancang. Untuk memudahkan dalam pembuatan robot, desain sistem mekanik menggunakan software bantuan SolidWorks 2019. Dimensi robot harus diperhitungkan dengan kondisi lingkungan yang akan dilalui oleh robot dapat dilihat pada Gambar 13. Rancangan Desain dan 14. Desain Mekanik.



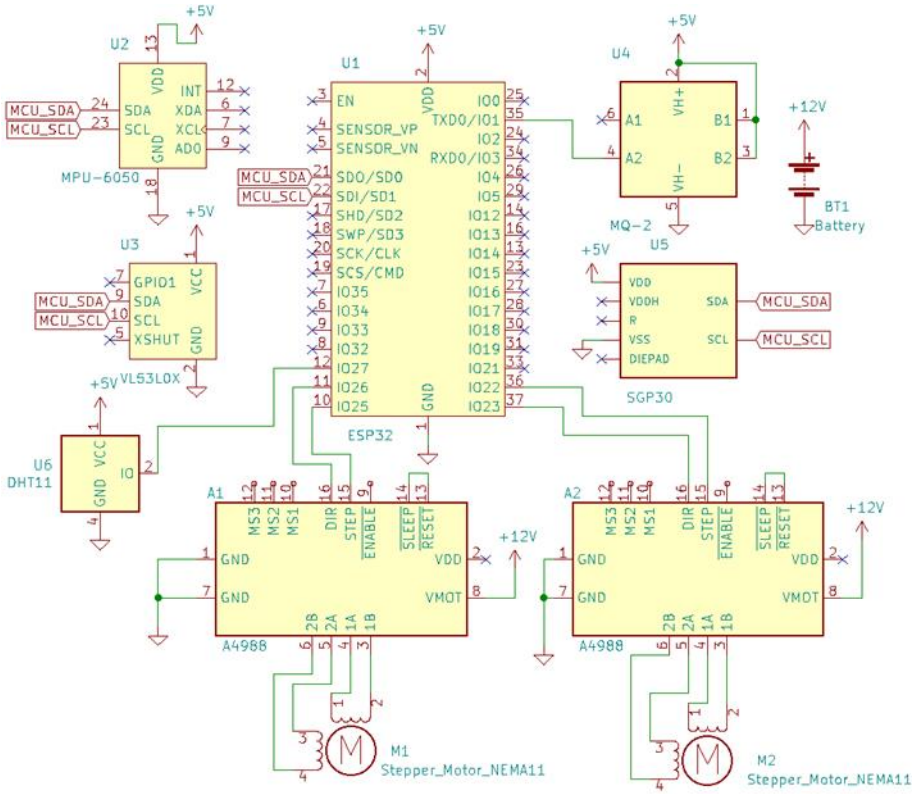
Gambar 13. Rancangan Desain



Gambar 14. Desain Mekanik

Desain Sistem Listrik (*Electrical Design)*

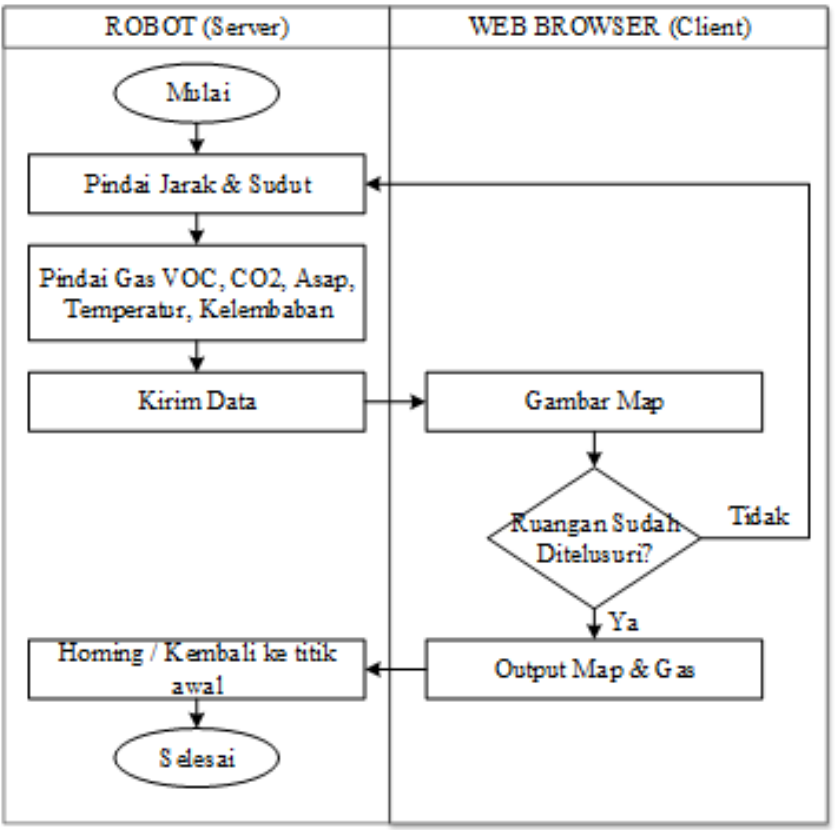
Dalam desain sistem listrik dan mekanik terdapat beberapa hal yang harus diperhatikan, antara lain sumber catu daya dan pembagian daya untuk masing-masing komponen, kebutuhan tegangan dan arus untuk mikrokontroller, sensor dan actuator dan desain skema rangkaian. Desain elektronik menggunakan bantuan software Kicad untuk memudahkan dalam pengkoneksian kabel kabel sehingga dapat mengurangi kesalahan dalam proses penyambungan kabel dapat dilihat pada Gambar 15. Desain Elektronik.



Gambar 15. Desain Elektronik

Desain Perangkat Lunak *(Software Design)*

Desain perangkat lunak yang digunakan menggunakan perangkat lunak Visual Studio Code, Git, Processing, Chrome Browser dan Visio. Untuk Bahasa pemrograman yang digunakan yaitu C++, JavaScript, HTML dan CSS. Desain perangkat lunak dibagi menjadi dua yaitu perangkat lunak pada robot dan perangkat lunak pada web client. Berikut merupakan flowchart keseluruhan sistem dapat dilihat pada Gambar. 16. Flowchart Keseluruhan Sistem.



Gambar 16. Flowchart Keseluruhan Sistem

Desain Perangkat Lunak pada Robot

Perangkat lunak pada robot didesain berdasarkan karakteristik hardware pada robot baik aktuator maupun sensor. Untuk menambah kinerja aktuator lebih optimal maka dilakukan perhitungan berikut dapat dilihat pada Tabel 2. Perhitungan Aktuator.

Tabel 2. Perhitungan Aktuator

No	Keterangan	Rumus	Nilai
1	Diameter Roda (OD)	-	70mm
2	Jarak Roda Kanan & Kiri	-	95mm
3	Spesifikasi Stepper Motor	1.8deg/Step	200 Step/Revolution
4	Microstepping	-	1/8
5	Microstepping Full Rotation	200 * 8	1600 Step / Revolution
6	Panjang Keliling Roda	$K = \pi * d$	219.9 mm
7	Step/mm berdasarkan keliling roda	Full Rotation / Keliling Roda	1600 / 219.91 = 7.28 steps/mm

Selain itu digunakan library FreeRTOS *(Real Time Operating System)*, dengan library tersebut memungkinkan pembuatan *task* atau tugas yang diperlukan oleh robot, selain itu juga dengan *library* tersebut dapat dengan mudah mengatur tugas tugas berdasarkan alokasi memori, prioritas dan pemilihan *core* cpu yang tersedia. Berikut tabel tugas tugas yang terdapat pada robot dapat dilihat pada tabel d bawah ini:

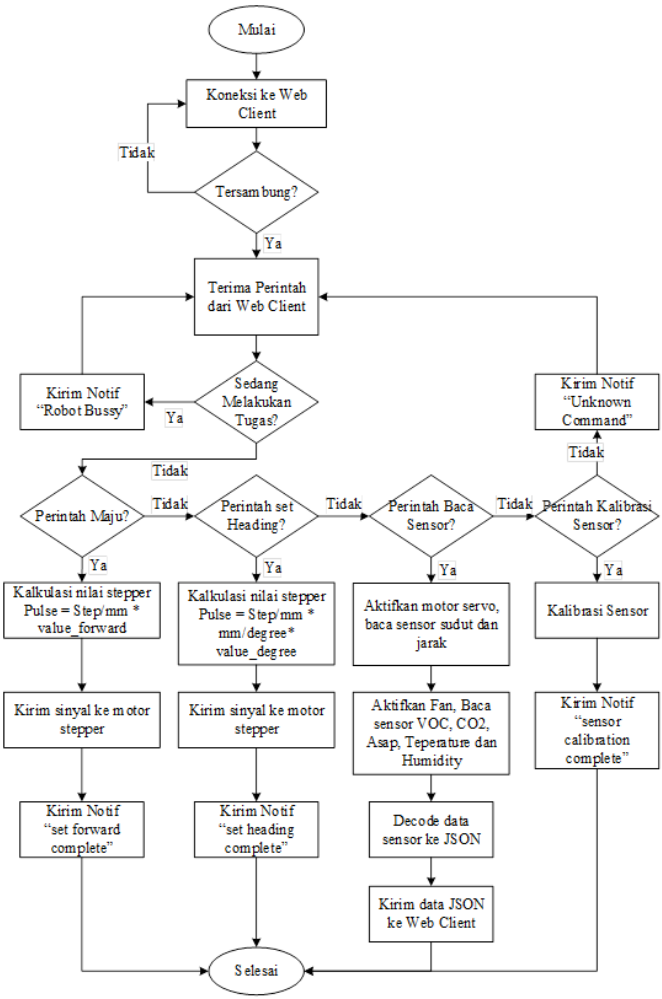
Tabel 3. Tugas-Tugas pada Robot

No	Task Name	Stack Size	Priority	Running Core
1	"async_tcp"	1024*16	1	-1
2	"FlexyStepper"	1024	1	-1
3	"FlexyStepper"	1024	1	-1
4	"Watch_Command_Task"	1024*6	1	-1

Dalam proses komunikasi robot dengan Web UI digunakan API *(Application Programming Interface)* dengan format JSON *(JavaScript Object Notation)*, Berikut struktur data JSON pada robot.

{
"wall" : [
index 1 = j,
index 2 = j,
index n = j,
]
"gas": {
"voc": k,
"co2": l,
"smoke": m,
"temp": n,
"hum": o,
"batt": p
}
}
	Key "wall" merupakan data dinding
	Index Array merukapan representasi dari sudut/angle
	Nilai j merupakan jarak
	
	Key "gas" merupakan data gas
	Key "voc" merupakan kadar voc
	Key "co2" merupakan kadar co2
	Key "smoke" merupakan kadar asap
	Key "temp" merupakan temperature
	Key "hum" merupakan kelembaban
	Key "batt" merupakan kapasitas baterai

Desain software pada robot dapat dilihat pada Gambar 17. Flowchart Robot.



Gambar 17. Flowchart Robot

Desain Perangkat Lunak pada Web UI

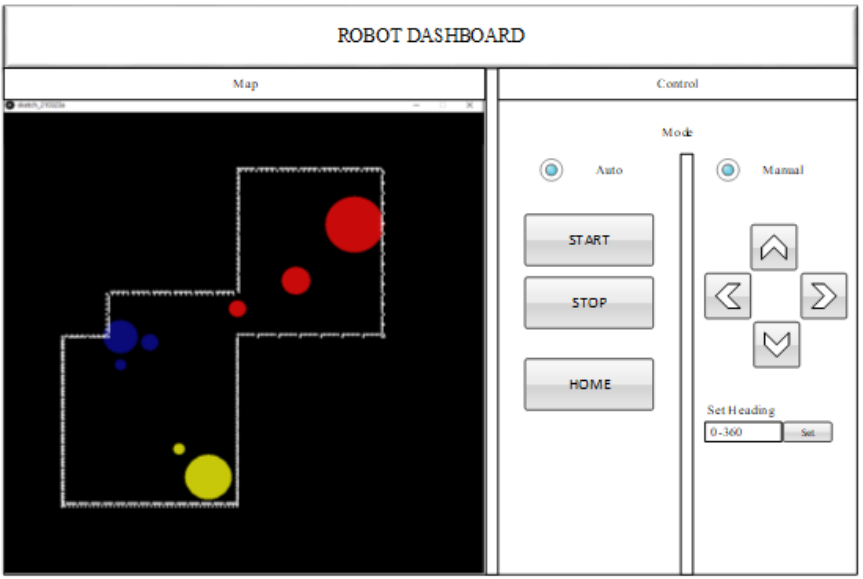
Perangkat lunak pada web ui mayoritas menggunakan bahasa pemrograman Javascript. Selain itu juga digunakan *library* P5.js untuk memudahkan dalam pembuatan grafik map berdasarkan data

yang diterima dari robot. Untuk mengkonversi data jarak ke grafik map, digunakan skala yaitu 1 berbanding 3.33, artinya setiap satu pixel pada web ui mewakili 3.33 milimeter pada lingkungan sebenarnya. Untuk menyajikan data map, web ui menggunakan format data JSON. Berikut merupakan struktur data JSON untuk data map.

dataMap = {
robotCor : [[x,y,h] ,…,[x,y,h]],
wall : [[x,y], …, [x,y]],
gas : {
voc : [[x,y,v], …, [x,y,v]],
co2 : [[x,y,v], …, [x,y,v]],
smoke : [[x,y,v], …, [x,y,v]],
temp : [[x,y,v], …, [x,y,v]],
hum : [[x,y,v], …, [x,y,v]],
quality : [[x,y,v], …, [x,y,v]],
battVolt : [[x,y,v]],
battPers : [[x,y,v]]
}
	x = koordinat x (mm)
	y = koordinat y (mm)
	h = heading (degree)
	
	x = koordinat x (mm)
	y = koordinat y (mm)
	v = nilai kadar gas

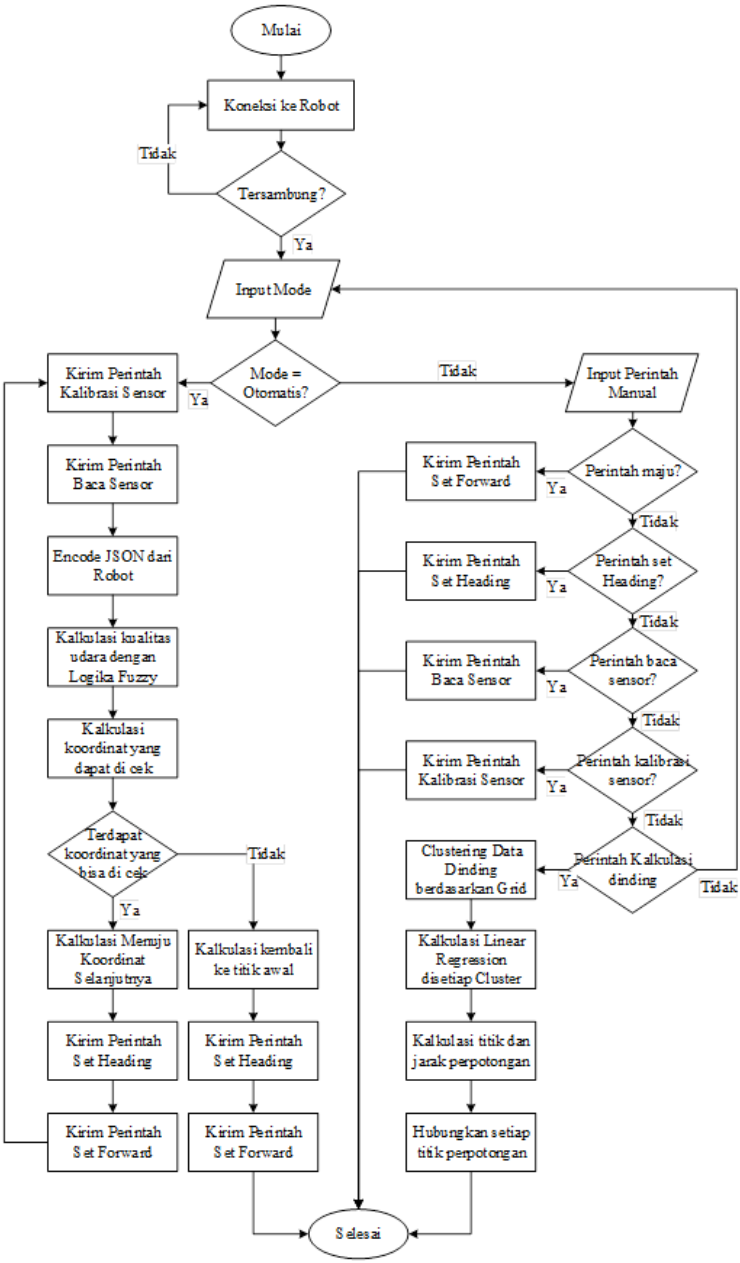
};	

Desain perangkat lunak web interface dapat dilihat pada Gambar 18. Desain WEB UI.



Gambar 18. Desain WEB UI

Adapun flowchart dapat dilihat pada Gambar 19. Flowchart Web UI



Gambar 19. Flowchart Web UI

Tes Fungsional *(Functional Test)*

Tes fungsional meliputi pengetesan fungsional sistem yang telah terintegrasi antara desain listrik dan desain perangkat lunak. Proses tes ini dilakukan untuk meningkatkan kinerja dari perangkat lunak dalam pengontrolan terhadap desain listrik dan mengeliminasi serta antisipasi error dari software yang dibuat, bila sistem software telah selesai diuji maka selanjutnya masuk ke proses perakitan.

Integrasi atau Perakitan *(Integration)*

Modul listrik yang diintegrasi dengan *software* di dalam kontrolernya, diintegrasikan dalam struktur mekanik yang telah dirancang, lalu dilakukan tes fungsional keseluruhan sistem.

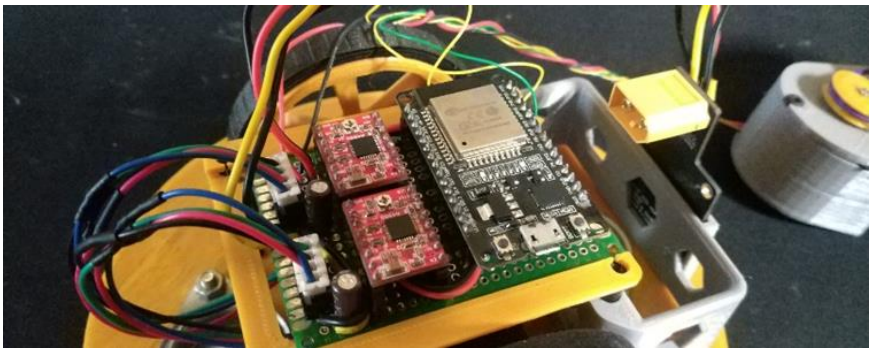
Perakitan Hardware

Perakitan hardware dilakukan dengan bantuan mesin 3D printer, mencetak komponen komponen yang telah di desain diproses sebelumnya, berikut proses pencetakan komponen dapat dilihat pada Gambar 20. Perakitan Hardware.

Gambar 20. Perakitan Hardware

Perakitan Kelistrikan

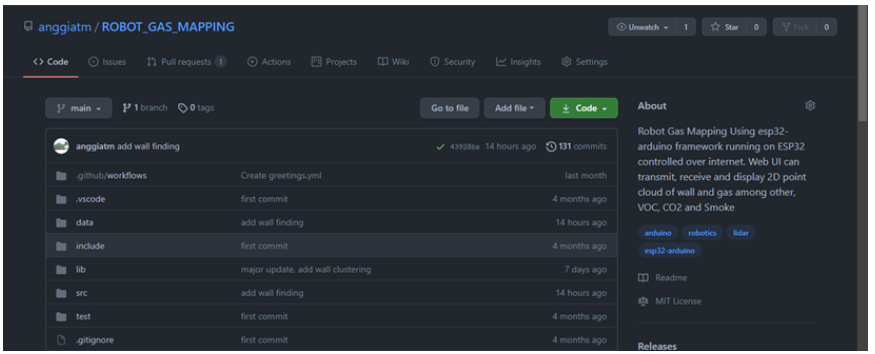
Pada proses perakitan kelistrikan pertama kali dibuat sebuah papan induk dengan bahan PCB kosong, papan induk ini berfungsi sebagai penyangga komponen utama yaitu microcontroller. Selain itu juga papan induk berfungsi sebagai jalur jalur listrik pada sensor dan actuator dapat dilihat pada Gambar 21. Perakitan Kelistrikan.



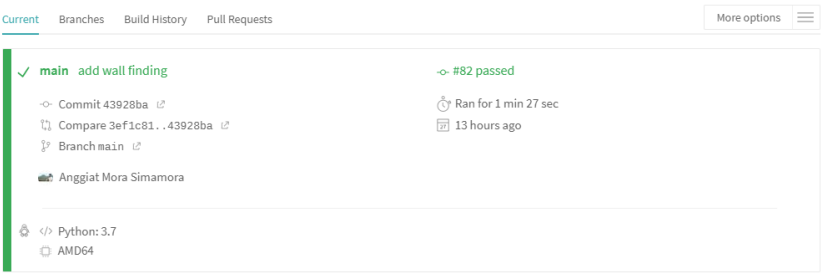
Gambar 21. Perakitan Kelistrikan

Perakitan Software

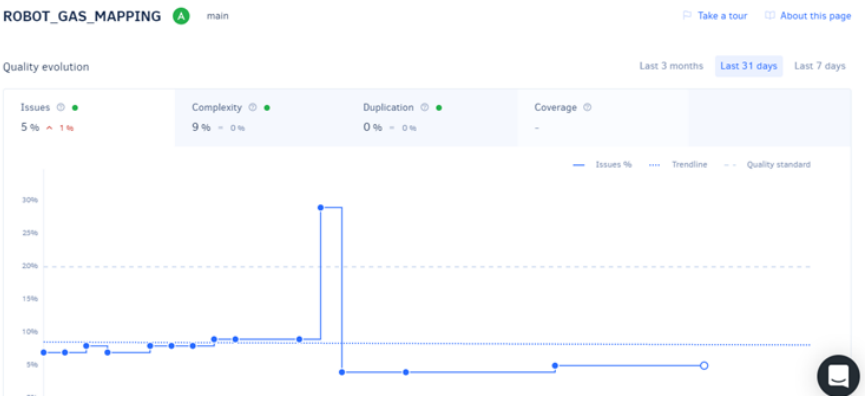
Proses perakitan software menggunakan code editor visual studio code, selain itu juga hasil kode program yang telah dibuat, diupload ke ropositori online Github yang terintegrasi dengan beberapa layanan Continous Intergration diantaranya TravisCI, Github Workflows dan Codacy. Layanan ini berfungsi sebagai compiler dan review kualitas kode sekaligus mencatat setiap perubahan pada kode untuk memudahkan pencarian baris kode jika ada yang tidak sesuai atau error dapat dilihat pada gambar 22, 23 dan 24.



Gambar 22. Perakitan Software Github



Gambar 23. Perakitan Software TravisCI



Gambar 24. Perakitan Software Codacy

Adapun penjelasan *code library* secara detail dapat dilihat pada Tabel 4. *Header Library* dan Tabel 5. Fungsi Utama *Serverside* pada *servideside*, kemudian untuk *clientside* dapat dilihat pada Tabel 6. Fungsi Utama *Clientside* selengkapnya dapat dilihat pada Lampiran 1. Serverside *(Robot Code)*.

a. *Serverside (Robot Code)*

Tabel 4. *Header Library*

No	Nama Code	Fungsi
1	Wifi.h	Untuk konfigurasi wifi

2	AsyncTCP.h	Untuk konfigirasi websocket
3	ESPAsyncWebServer.h	Untuk konfigurasi webserver
4	SPIFFS.h	Untuk mengorganisasi filesystem seperti file .html .js .css untuk keperluan server
5	ESP_FlexyStepper.h	Untuk konfigurasi task motor stepper
6	MPU6050_6Axis_MotionApps20.h	Untuk memproses data sensor mpu6050
7	VL53L0X.h	Untuk konfigurasi dan proses data jarak dari sensor VL53L0X
8	ESP32Servo	Untuk driver microservo
9	ArduinoJson.h	Untuk mengorganisasi data JSON untuk keperluan kmomunikasi dengan client

Tabel 5. Fungsi Utama *Serverside*

No	Nama Code	Fungsi

1	void setup()	Saat pertamakali alat dinyalakan akan menjalankan fungsi ini, beberapa fungsi didalamnya yaitu : membuat task motor kanan dan kiri, membuat task web server dan web socket, task membaca sensor, routing web server, menyiapkan koneksi wifi, menyiapkan konfigurasi pin, menyiapkan konfigurasi sensor dan komunikasi sensor
2	void handleWebSocketMessage()	Task handle untuk websocket menerima pesan dari client
3	void onEvent(……)	Debug status koneksi websocket
4	void forward()	Fungsi untuk perintah maju / mundur
5	void setHeading()	Fungsi untuk perintah belok kiri / kanan
6	void task_display(void *pvParameters)	Fungsi task untuk membaca sensor jarak dan angle, pada fungsi ini data tersebut di kirimkan ke client dengan format JSON

7	void sensorAlignment()	Funsi untuk memposisikan sudut sensor navigasi
8	String splitString(String data, char separator, int index)	Fungsi untuk memecah string perintah dari client dengan separator
9	Int getAngle()	return selected string index

b. *Clientside (Web UI Code)*

Tabel 6. Fungsi Utama *Clientside*

No	Nama Code	Fungsi
1	function setup()	Saat pertama kali server diakses maka akan menjalankan code ini beberapa fungsi didalammya yaitu : menentukan panjang dan lebar canvas, membuat canvas, menentukan warna background

2	function draw()	Setiap update dari robot akan langsung digambar pada canvas diantaranya status koneksi, koordinat robot, heading robot, data dinding, data gas.
3	function initWebSocket()	Fungsi untuk meyiapkan konfigurasi websocket diantaranya : variable websocket, fungsi jika koneksi terbuka dan tertutup, fungsi juka ada pesan
4	function onMessage(event)	Fungsi untuk menerima pesan dari robot berupa format JSON, pada fungsi ini data JSON akan di parsing untuk selanjutnya data akan disajikan pada function draw()

5	function initButton()	Fungsi untuk menyiapkan event pada tombol tombol
6	void function setHeading()	Fungsi untuk mengirim perintah belok kiri / kanan pada robot
7	function setForward()	Fungsi untuk mengirim perintah maju/mundur pada robot

Tes Fungsional Keseluruhan Sistem *(Overall Testing)*

Pada tahapan ini dilakukan pengetesan fungsi dari keseluruhan sistem. Apakah dapat berfungsi sesuai dengan konsep atau tidak. Bila ada sistem yang tidak dapat bekerja dengan baik maka harus dilakukan proses perakitan ulang pada setiap desain sistemnya.

Application

Application untuk meningkatkan performa dari aplikasi yang telah dirancang. Optimasi ditekankan pada desain mekanik agar penggunaan lebih maksimal serta optimal.

Proses Implementasi *(Implementation)*

Proses implementasi dilakukan dengan mengoperasikan robot pada model ruangan yang telah dibuat. Model ruangan yang dibuat

berbentuk persegi dan beralaskan karpet dapat dilihat pada gambar 25 dan 26.

Gambar 25. Implementasi Robot

Gambar 26. Implementasi Web UI

Penerapan Metode pada Robot

Digunakan beberapa metode untuk menentukan output agar lebih maksimal. Untuk menentukan kualitas gas pada ruangan digunakan metode logika fuzzy sederhana. Sedangkan untuk menentukan dinding digunakan metode regresi linear sederhana.

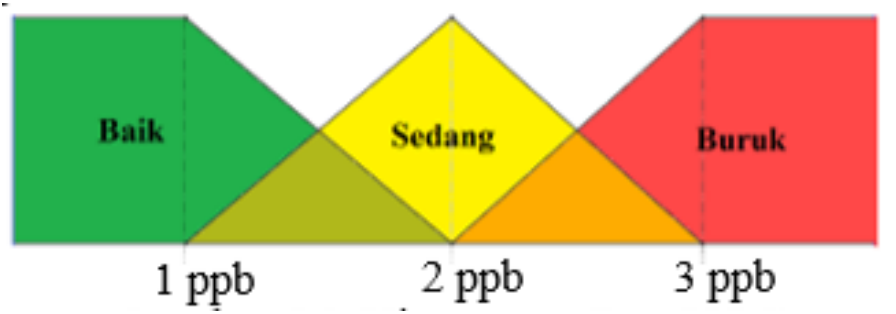
Logika Fuzzy Mamdani

Menentukan kualitas apakah kualitas gas baik, sedang atau buruk digunakan logika fuzzy. Langkah langkah sistem penentuan kualitas udara yaitu Fuzifikasi, inferensi dan defuzifikasi.

a. Proses Fuzzifikasi

Pada proses ini dibuat beberapa fungsi keangotaan, yang digunakan sebagai input adalah membership fungsi kadar VOC, CO2, Asap, Temperature dan Kelembaban. Sedangkan output adalah persentase 0% - 100% kualitas udara dapat dilihat pada gambar 27, 28, 29, 30, 31 dan 32.

a. Variabel input gas VOC



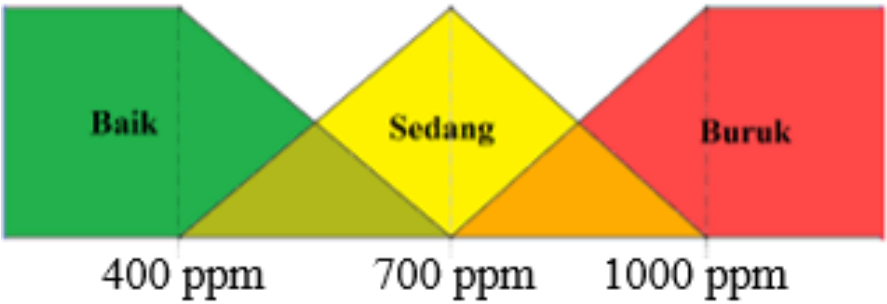
Gambar 27. Himpunan Gas VOC

Fungsi keanggotaan gas VOC:

$$VOC\ Rendah[voc] = \begin{cases} 1; voc \leq 0 \\ \dfrac{3 - voc}{3 - 0}; 0 \leq voc \leq 3 \\ 0; voc > 3 \end{cases}$$

$$VOC\ Tinggi[voc] = \begin{cases} 1; \geq 3 \\ \dfrac{voc - 0}{3 - 0}; 0 \leq voc \leq 3 \\ 0; voc \leq 0 \end{cases}$$

b. Variabel input gas CO2



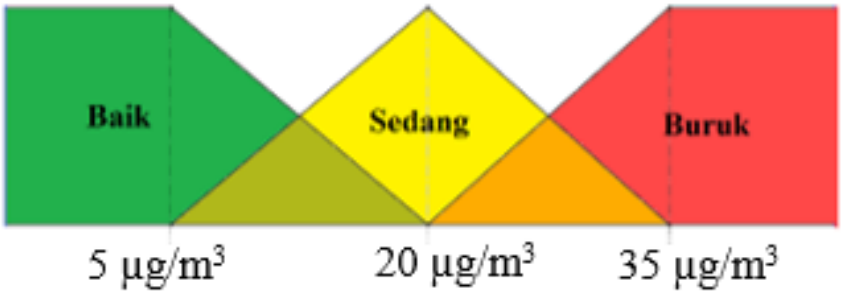
Gambar 28. Himpunan Gas CO2

Fungsi keanggotaan gas CO2:

$$CO2\ Rendah[co2] = \begin{cases} 1; co2 \leq 0 \\ \dfrac{1000 - co2}{1000 - 0}; 0 \leq co2 \leq 1000 \\ 0; co2 > 1000 \end{cases}$$

$$CO2\ Tinggi[co2] = \begin{cases} 1; \geq 1000 \\ \dfrac{co2 - 0}{1000 - 0}; 0 \leq co2 \leq 1000 \\ 0; co2 \leq 0 \end{cases}$$

c. Variabel input Asap



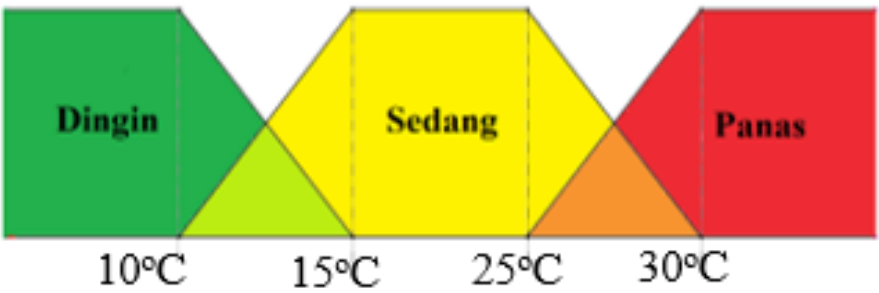
Gambar 29. Himpunan Asap

Fungsi keanggotaan gas Asap:

$$Asap\ Rendah[asap] = \begin{Bmatrix} 1; asap \leq 0 \\ \frac{35 - asap}{35 - 0}; 0 \leq asap \leq 35 \\ 0; asap > 35 \end{Bmatrix}$$

$$Asap\ Tinggi[asap] = \begin{Bmatrix} 1; \geq 35 \\ \frac{asap - 0}{35 - 0}; 0 \leq asap \leq 35 \\ 0; asap \leq 0 \end{Bmatrix}$$

d. Variabel input Temperature



Gambar 30. Himpunan Temperatur

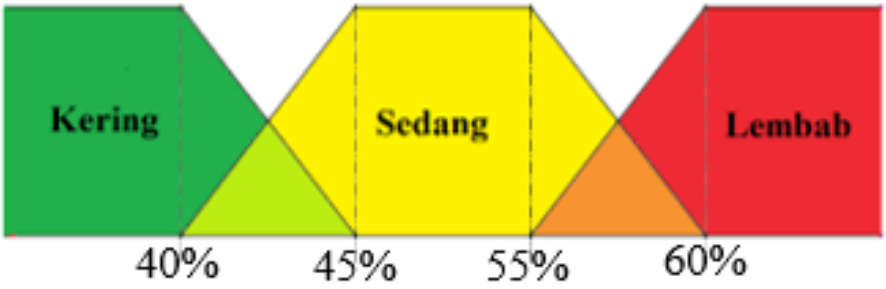
Fungsi Keanggotaan Temperatur

$$Temperatur\ Baik[temp] = \begin{Bmatrix} 1; 10 \leq temp \leq 30 \\ \frac{temp - 5}{10 - 5}; 5 \leq temp \leq 10 \\ \frac{35 - temp}{35 - 30}; 30 \leq temp \leq 35 \\ 0; temp > 35\ atau, temp < 5 \end{Bmatrix}$$

$$Temperatur\ Buruk[temp]$$

$$= \begin{cases} 1; temp < 5\ atau, temp > 35 \\ \dfrac{temp - 30}{35 - 30};\ 30 \leq temp\ \leq 35 \\ \dfrac{10 - temp}{10 - 5}; 5 \leq temp \leq 10 \\ 0; 10\ \leq temp \leq 30 \end{cases}$$

e. Variabel input Kelembaban



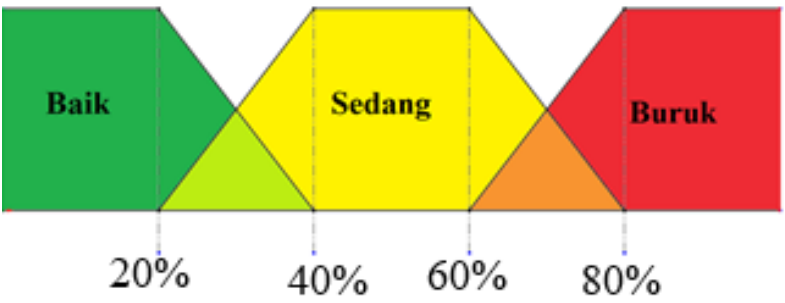
Gambar 31. Himpunan Kelembaban

Fungsi keanggotaan Kelembaban

$$Kelembaban\ Baik[hum] = \begin{cases} 1; 40 \leq hum \leq 60 \\ \dfrac{hum - 35}{40 - 35}; 35 \leq hum \leq 40 \\ \dfrac{65 - hum}{65 - 60};\ 60 \leq hum\ \leq 65 \\ 0; hum > 65\ atau, hum < 35 \end{cases}$$

$$Kelembaban\ Buruk[hum] = \begin{cases} 1; hum < 35\ atau, hum > 65 \\ \dfrac{hum - 60}{65 - 60};\ 60 \leq hum\ \leq 65 \\ \dfrac{40 - hum}{40 - 35}; 35 \leq hum \leq 40 \\ 0;\ 35\ \leq hum \leq 60 \end{cases}$$

f. Variable output Kualitas Gas



Gambar 32. Himpunan Kualitas Gas

Fungsi keanggotaan kualitas udara

$$Kualitas\ Baik[z] = \begin{cases} 0; z \leq 60 \\ \dfrac{z-80}{80-60};\ 60 \leq z\ \leq 80 \\ 1; z > 80 \end{cases}$$

$$Kualitas\ Sedang[z] = \begin{cases} 1;\ 40\ \leq z\ \leq 60 \\ \dfrac{z-20}{40-20};\ 20 \leq z\ \leq 40 \\ \dfrac{80-z}{80-60}; 60 \leq z \leq 80 \\ 0;\ z < 20\ atau\ z > 80 \end{cases}$$

$$Kualitas\ Buruk[z] = \begin{cases} 0; z > 40 \\ \dfrac{20-z}{40-20};\ 20 \leq z\ \leq 40 \\ 1; z < 20 \end{cases}$$

b. Proses Inferensi

Proses inferensi atau aturan dasar merupakan proses untuk mendapatkan keluaran dari rule set yang digunakan rule metode combs dapat dilihat pada Tabel 7. Rule Logika Fuzzy.

Tabel 7. Rule Logika Fuzzy Mamdani

Rule	Input					Output
	VOC	CO2	Asap	Temperatur	Kelembaban	
1	Rendah	Rendah	Rendah	Baik	Baik	Baik
2	Rendah	Rendah	Rendah	Baik	Buruk	Sedang
3	Rendah	Rendah	Rendah	Buruk	Baik	Sedang
4	Rendah	Rendah	Rendah	Buruk	Buruk	Sedang
5	Rendah	Rendah	Tinggi	Baik	Baik	Sedang
6	Rendah	Rendah	Tinggi	Baik	Buruk	Sedang
7	Rendah	Rendah	Tinggi	Buruk	Baik	Sedang
8	Rendah	Rendah	Tinggi	Buruk	Buruk	Buruk
9	Rendah	Tinggi	Rendah	Baik	Baik	Sedang
10	Rendah	Tinggi	Rendah	Baik	Buruk	Sedang
11	Rendah	Tinggi	Rendah	Buruk	Baik	Sedang
12	Rendah	Tinggi	Rendah	Buruk	Buruk	Buruk
13	Rendah	Tinggi	Tinggi	Baik	Baik	Sedang
14	Rendah	Tinggi	Tinggi	Baik	Buruk	Buruk
15	Rendah	Tinggi	Tinggi	Buruk	Baik	Buruk
16	Rendah	Tinggi	Tinggi	Buruk	Buruk	Buruk
17	Tinggi	Rendah	Rendah	Baik	Baik	Sedang
18	Tinggi	Rendah	Rendah	Baik	Buruk	Sedang
19	Tinggi	Rendah	Rendah	Buruk	Baik	Sedang
20	Tinggi	Rendah	Rendah	Buruk	Buruk	Buruk
21	Tinggi	Rendah	Tinggi	Baik	Baik	Sedang
22	Tinggi	Rendah	Tinggi	Baik	Buruk	Buruk
23	Tinggi	Rendah	Tinggi	Buruk	Baik	Buruk
24	Tinggi	Rendah	Tinggi	Buruk	Buruk	Buruk
25	Tinggi	Tinggi	Rendah	Baik	Baik	Sedang
26	Tinggi	Tinggi	Rendah	Baik	Buruk	Buruk
27	Tinggi	Tinggi	Rendah	Buruk	Baik	Buruk
28	Tinggi	Tinggi	Rendah	Buruk	Buruk	Buruk
29	Tinggi	Tinggi	Tinggi	Baik	Baik	Buruk
30	Tinggi	Tinggi	Tinggi	Baik	Buruk	Buruk

31	Tinggi	Tinggi	Tinggi	Buruk	Baik	Buruk
32	Tinggi	Tinggi	Tinggi	Buruk	Buruk	Buruk

c. Proses Defuzifikasi

Metode defuzzifikasi yang digunakan adalah centroid atau *center of area* (COA). Dimana nilai tegas outputnya diperoleh berdasarkan titik berat dari kurva hasil proses pengambilan keputusan. Dengan menggunakan persamaan sebagai berikut:

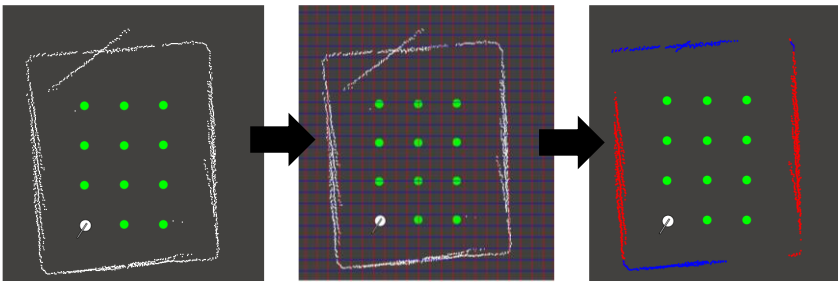
$$Z^* = \frac{\int \mu_{[x]}(z) z \, dz \qquad Momen}{\int \mu_{[x]}(z) z \, dz \qquad Luas\ Daerah} \tag{4}$$

Regresi Linear Sederhana

Data dinding yang disajikan pada Web UI masih berupa *2D point cloud* atau titik titik. Oleh karena itu, untuk menentukan dinding digunakan salah satu metode perhitungan adalah yaitu regresi linear sederhana. Namun sebelum mengkalkulasi data dibutuhkan pemilahan data atau *clustering* yang diyakini sebagai dinding vertikal atau horizontal. Berikut merupakan langkah langkah penentuan dinding.

a. Proses *Clustering*

Pada proses ini data map dibagi berdasarkan sumbu horizontal dan sumbu vertikal, setelah itu dilakukan pengelompokan data titik vertikal dan data titik horizontal dapat dilihat pada Gambar 33. Proses *Clustering*.



Gambar 33. Proses Clustering

b. Proses Kalkulasi

Pada proses ini setiap kelompok data dihitung menggunakan regresi linear sederhana untuk mendapatkan fungsi persamaan sebagai berikut:

$$Y = a + bX$$

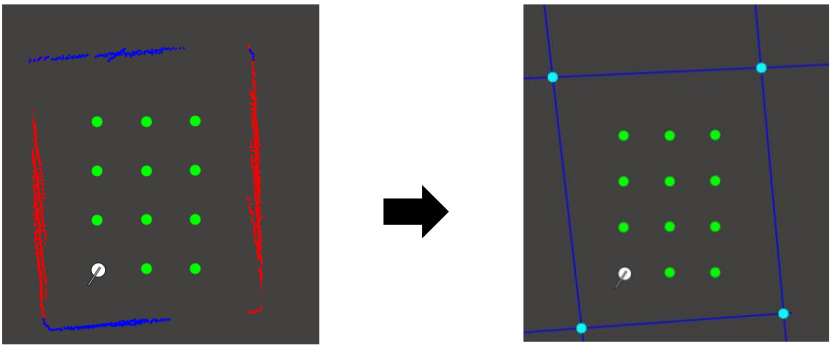
Y : Variabel dependen

a : konstanta

b : koefisien variabel X

X : Variabel Independen

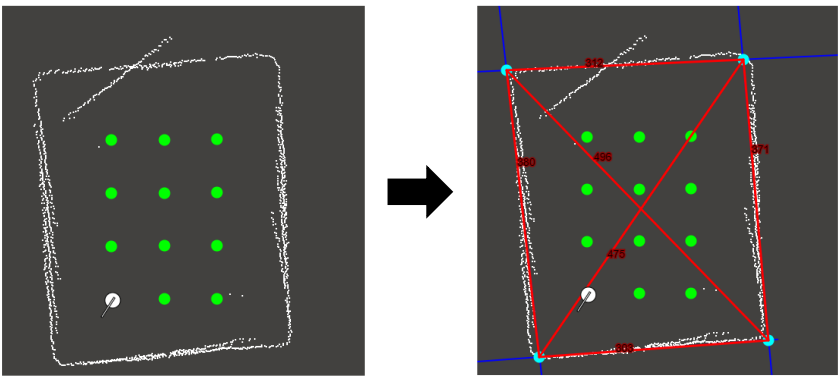
Setelah mendapatkan persamaan dari masing masing kelompok data selanjutnya didapatkan sejumlah garis lurus yang diyakini sebagai dinding, yang ditunjukan garis warna biru pada gambar. Lalu setiap perpotongan pada garis dari hasil perhitungan tersebut merupakan sudut dari sebuah ruangan yang ditunjukan titik warna biru muda pada Gambar 34. Proses Regresi Linear Sederhana



Gambar 34. Proses Regresi Linear Sederhana

c. Proses Evaluasi

Pada proses ini terlihat output yang relatif mirip dengan data mentah, sehingga perhitungan panjang dinding dan jarak sudut dapat dikalkulasi dengan mudah dapat dilihat pada Gambar 35. Proses Evaluasi.



Gambar 35. Proses Evaluasi

Trigonometri

Robot akan kembali ke titik awal jika beberapa kondisi terpenuhi yaitu ruangan telah ditelusuri atau kapasitas baterai yang tersisa kurang dari 40%. Untuk kembali ke titik awal, robot menghitung jarak terpendek yaitu dengan menarik garis lurus dari koordinat

terakhir robot ke titik awal selain itu juga robot akan mengkalkulasi sudut orientasi yang sesuai.

a. Proses mencari jarak terpendek

Panjang jarak yang akan ditempuh robot dapat diselesaikan dengan persamaan phytagoras.

$$[jarak] = \sqrt{X^2 + Y^2} \quad (5)$$

jarak : jarak yang akan dilalui robot

X : Koordinat X Robot

Y : Koordinat Y Robot

b. Proses mencari target orientasi

Untuk menentukan target orientasi robot dapat diselesaikan dengan persamaan trigonometri:

1. Orientasi robot terakhir kuadran 1 (0 – 90 derajat)

$$[orientasi] = Rh + 90 + \frac{cosec\left(\frac{X}{jarak}\right) x\ 180}{\pi} \quad (6)$$

2. Orientasi robot terakhir kuadran 2 (90 – 180 derajat)

$$[orientasi] = Rh + 180 + \frac{cosec\left(\frac{X}{jarak}\right) x\ 180}{\pi} \quad (7)$$

3. Orientasi robot terakhir kuadran 3 (180 – 270 derajat)

$$[orientasi] = Rh + 270 + \frac{cosec\left(\frac{X}{jarak}\right) x\ 180}{\pi} \quad (8)$$

4. Orientasi robot terakhir kuadran 4 (270 – 360 derajat)

$$[orientasi] = Rh + \frac{cosec\left(\frac{X}{jarak}\right) x\ 180}{\pi} \quad (9)$$

orientasi : target orientasi robot

Rh : orientasi robot terakhir

jarak : jarak tempuh yang akan dilalui robot

Nilai dari perhitungan jarak tempuh dan orientasi robot akan dikonversi ke bilangan bulat mengingat robot hanya menerima dan mengeksekusi perintah dengan nilai bilangan bulat. Sehingga dari perhitungan tersebut akan mengurangi akurasi perpindahan robot dengan jarak sebesar ±0.5mm dan sudut sebesar ±0.5derajat.

BAB 3
PEMBAHASAN

Tes Fungsional Keseluruhan Sistem *(Overall Testing)*

Robot dapat bekerja berdasarkan perintah dari web client, selain itu robot juga mengirimkan data data sensor ke web client yang selanjutnya dapat diproses untuk menentukan perintah selanjutnya. Pada tahap ini dilakukan pengetesan fungsi dari keseluruhan sistem. Apakah dapat berfungsi sesuai konsep yang telah dibuat. Bila ada sistem yang tidak dapat bekerja dengan baik, maka harus dilakukan proses perakitan ulang. Pengujian ini meliputi pengujian struktural, fungsional dan validasi.

Pengujian Struktural

Pengujian yang bertujuan untuk mengetahui apakah jalur rangkaian hardware sudah terhubung dengan benar sehingga sistem dapat berfungsi dengan baik. Pengujian ini dilakukan dengan mencoba jalur-jalur rangkaian dengan menggunakan multimeter. Berikut tabel hasil pengujian struktural sistem dapat dilihat pada Tabel 8. Pengujian Struktural.

Tabel 8. Pengujian Struktural

No.	ESP32	Komponen	Hambatan	Kondisi
1	Pin I2C SCL	MPU6050, VL53L0X, SGP30 SCL	0.55 Ω	Baik

2	Pin I2C SDA	MPU6050, VL53L0X, SGP30 SDA	0.57 Ω	Baik
3	Pin 32	Motor 1 Step	0.60 Ω	Baik
4	Pin 33	Motor 1 Dir	0.73 Ω	Baik
5	Pin 25	Motor 2 Step	0.32 Ω	Baik
6	Pin 26	Motor 2 Dir	0.44 Ω	Baik
7	Pin 16	DHT11 D0	0.57 Ω	Baik
8	Pin 39	Voltage Divider Baterai	0.82 Ω	Baik

Pengujian Fungsional

Pengujian yang bertujuan untuk mengetahui apakah tegangan yang mengalir di dalam rangkaian sudah sesuai dengan yang dibutuhkan. Pengujian ini dilakukan dengan cara melihat output robot baik pada aktuator maupun dari sensor.

a. Pengujian *Microcontroller* ESP32

   ESP32 membutuhkan tegangan setidaknya 5v untuk dapat bekerja dengan baik, selain itu ESP32 membutuhkan 3.3v untuk menghidupkan mikroprosesor. Pada pengujian mikrokontroller ESP32 dilakukan dengan cara memberikan tegangan 5V. Setelah itu output tegangan dicek pada pin 5V dan pin 3.3v. Berikut pengujian Mikrokontroler ESP32 dapat dilihat pada Tabel 9. Pengujian *Microcontroller* ESP32.

   Tabel 9. Pengujian *Microcontroller* ESP32

No	Suplai Voltase	Pengukuran	Kondisi
1	5v	4.89 v	Baik

2	3.3v	3.23	Baik

b. Pengujian Aktuator

Pengujian aktuator dilakukan untuk memastikan bahwa aktuator dapat berfungsi dengan baik. Pengujian motor stepper dilakukan dengan cara mengetes hambatan pada setiap coil, jika terdapat hambatan >4Ω dan tidak putus maka kondisi motor baik. Tabel pengujian aktuator terdapat pada Tabel 10. Pengujian Aktuator.

Tabel 10. Pengujian Aktuator

No.	Coil	Hambatan	Keterangan
1	Motor 1 Coil A	4,76 Ω	Kondisi Baik
2	Motor 1 Coil B	4,86 Ω	Kondisi Baik
3	Motor 2 Coil A	5,06 Ω	Kondisi Baik
4	Motor 2 Coil B	4,88 Ω	Kondisi Baik

c. Pengujian Sensor

Pengujian pada beberapa sensor meliputi sensor gas dan sensor navigasi. Pengujian ini dilakukan dengan memberikan tegangan yang dibutuhan oleh sensor dan melihat data perubahan data pada serial monitor, jika sensor memberikan nilai serta perubahan maka sensor dapat bekerja dengan baik. Tabel pengujian sensor dapat dilihat pada Tabel 11. Pengujian Sensor.

Tabel 11. Pengujian Sensor

No.	Sensor	Tegangan	Serial Monitor	Keterangan

1	SGP30	4.83	✓	Kondisi Baik
2	MQ-2	4.86	✓	Kondisi Baik
3	VL53L0X	4.88	✓	Kondisi Baik
4	MPU6050	4.92	✓	Kondisi Baik
5	DHT11	4.75	✓	Kondisi Baik

d. Pengujian Keseluran Sistem

Pengujian keseluruhan sistem dilakukan untuk mengetahui integrasi antara mikrokontroller dengan sensor dan aktuator. Dari sisi hardware dipastikan bahwa aktuator dapat bergerak dengan bebas. Selain itu dari sisi sofware dipastikan bahwa kode program tidak ada yang error. Untuk pengujian software digunakan layanan continous integration CI yaitu TravisCI dan github Build. Beberapa pengujian untuk keseluruhan sistem dapat dilihat pada Tabel 12. Pengujian Keseluruhan Sistem.

Tabel 12. Pengujian Keseluruhan Sistem

No	Perintah Web UI	Aksi Robot	Keterangan
1	Koneksi ke Alamat IP Robot http://192.168.43.222/	Robot terkoneksi, siap menerima perintah	Sesuai
2	Kirim Perintah ke Robot	Menerima perintah, parsing	Sesuai

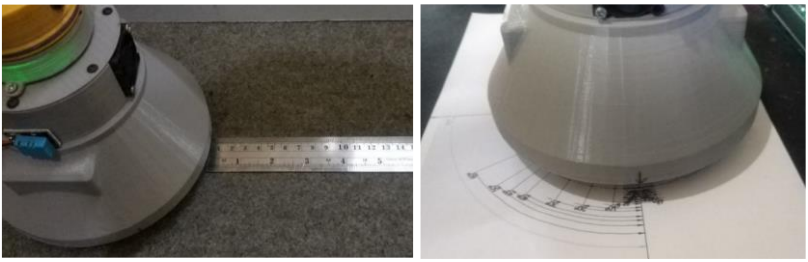
		perintah dan nilai	
3	Perintah Maju	Maju	Sesuai
4	Perintah Belok	Belok	Sesuai
5	Perintah Kalibrasi Sensor	Sensor di kalibrasi & kalkulasi offset sensor	Sesuai
6	Perintah Baca Sensor	Scan Dinding, Baca Sensor Gas, Kirim data format JSON	Sesuai

Uji Coba Validasi

Tahap ini dilakukan dengan tujuan untuk mengetahui cara kerja dan fungsi apakah dapat berjalan dengan baik sesuai dengan kalibrasi maupun perhitungan yang sudah diterapkan. Pada uji coba validasi dibedakan menjadi 3 yaitu uji coba validasi aktuator, uji coba validasi sensor dan uji coba validasi penerapan robot *mapping*.

a. Uji Coba Validasi Aktuator

   Uji coba validasi aktuator dilakukan dengan cara mengirimkan perintah pada robot maju atau berbelok, yaitu dengan nilai maju 50 sampai 500 milimeter, dan nilai belok 10 – 180 derajat. Setelah robot melakukan perintah, lalu dihitung jarak perpindahan robot dapat dilihat pada Gambar 36. Uji Coba Validasi Aktuator.



Gambar 36. Uji Coba Validasi Aktuator

Berikut tabel uji coba dapat dilihat pada Tabel 13, dan 14.

Tabel 13. Uji Coba Validasi Aktuator Perintah Maju.

No	Perintah Maju (mm)	Perpindahan Robot (mm)	Selisih (mm)	Error (%)
1	50	53	3	6.00
2	100	101	1	1.00
3	150	154	4	2.67
4	200	197	3	1.50
5	250	252	2	0.80
6	300	301	1	0.33
7	350	348	2	0.57
8	400	396	4	1.00
9	450	455	5	1.11
10	500	508	8	1.60
Rata Rata			**3.3**	**1.66**

Tabel 14. Uji Coba Validasi Aktuator Perintah Belok

No	Perintah Belok (derajat)	Perpindahan Robot (derajat)	Selisih (derajat)	Error (%)
1	10	10	0	0.00
2	20	20	0	0.00
3	30	30	0	0.00
4	40	41	1	2.50
5	50	49	1	2.00
6	60	60	0	0.00
7	70	72	2	2.86
8	80	81	1	1.25
9	90	88	2	2.22
10	100	99	1	1.00
11	110	108	2	1.82
12	120	120	0	0.00
13	130	131	1	0.77
14	140	139	1	0.71
15	150	150	0	0.00
16	160	160	0	0.00
17	170	173	3	1.76
18	180	178	2	1.11
Rata Rata			**0.94**	**1.00**

b. Uji Coba Validasi Sensor

Sensor diuji dengan membandingkan antara nilai pembacaan dari sensor dengan alat ukur yang sebenarnya. Pengujian ini dilakukan dengan cara menguji dari nilai kemungkinan kesalahan yang dapat terjadi pada komponen-komponen yang diimplementasikan model ini. Dari hasil perbandingan tersebut akan didapatkan selisih dan merupakan error. Uji coba validasi sensor gas dilakukan dengan cara menempatkan robot pada datu ruangan tertutup, setelah itu pembacaan sensor gas dibandingkan dengan pembancaan sensor gas dengan tipe yang berbeda dapat dilihat pada Gambar 37. Uji Coba Validasi Sensor Gas, acuan pembacaan sensor gas yaitu untuk gas VOC menggunakan sensor CCS811, kadar asap menggunakan Optical sensor GP2Y1010AU0F, kadar CO2 menggunakan MQ-135, suhu dan kelembaban menggunakan sensor AHT10.

Gambar 37. Uji Coba Validasi Sensor Gas

Berikut tabel uji coba validasi gas dapat dilihat pada Tabel 15 dan 16, sedangkan uji coba validasi asap dapat dilihat pada Tabel 17. Uji Coba Validasi Asap. Uji coba validasi temperature dapat dilihat pada Tabel 18. Uji Coba Validasi Temperatur dan kelembaban dapat dilihat pada Tabel 19. Uji Coba Validasi Kelembaban.

Tabel 15. Uji Coba Validasi Gas VOC

No	Pembacaan Sensor CCS811 (ppb)	Pembacaan Robot (ppb)	Selisih (ppb)	Error (%)
1	0	0	0	0.00
2	1	1	0	0.00
3	0	0	0	0.00
4	12	10	2	16.67
5	14	12	2	14.29
6	9	10	1	11.11
7	7	8	1	14.29
8	4	3	1	25.00
9	3	3	0	0.00
10	2	1	1	50.00
11	0	0	0	0.00
12	0	0	0	0.00
Rata Rata			**0.67**	**10.95**

Tabel 16. Uji Coba Validasi Gas CO2

No	Pembacaan MQ-2 (ppm)	Pembacaan Robot (ppm)	Selisih (ppm)	Error (%)
1	380.23	400	19.77	5.20
2	390.26	439	48.74	12.49
3	405.28	420	14.72	3.63

4	411.12	483	71.88	17.48
5	401.72	523	121.28	30.19
6	478.32	536	57.68	12.06
7	523.21	508	15.21	2.91
8	719.48	671	48.48	6.74
9	578.71	493	85.71	14.81
10	688.21	600	88.21	12.82
11	579.55	539	40.55	7.00
12	721.3	705	16.3	2.26
Rata Rata			**52.38**	**10.63**

Tabel 17. Uji Coba Validasi Asap

No	Pembacaan Sensor Optical GP2Y10 (µg/m3)	Pembacaan Robot (µg/m3)	Selisih (µg/m3)	Error (%)
1	11.55	11.89	0.34	2.94
2	12.32	13.25	0.93	7.55
3	15.67	18.63	2.96	18.89
4	16.22	18.33	2.11	13.01
5	15.98	19.27	3.29	20.59
6	17.59	21.54	3.95	22.46
7	26.46	26.45	0.01	0.04
8	24.75	25.77	1.02	4.12
9	28.56	29.45	0.89	3.12
10	19.23	20.26	1.03	5.36
11	24.87	28.51	3.64	14.64
12	22.56	28.76	6.2	27.48
Rata Rata			**2.20**	**11.68**

Tabel 18. Uji Coba Validasi Temperatur

No	Pembacaan Sensor AHT10 (^oC)	Pembacaan Robot (^oC)	Selisih (^oC)	Error (%)
1	30.65	32.2	1.55	5.06

2	31.87	33.3	1.43	4.49
3	31.06	33.3	2.24	7.21
4	30.44	32.4	1.96	6.44
5	28.32	30.6	2.28	8.05
6	29.45	33.2	3.75	12.73
7	28.56	31.8	3.24	11.34
8	28.22	31.7	3.48	12.33
9	27.49	31.7	4.21	15.31
10	26.07	28.6	2.53	9.70
11	26.36	28.9	2.54	9.64
12	26.45	29.9	3.45	13.04
Rata Rata			**2.72**	**9.61**

Tabel 19. Uji Coba Validasi Kelembaban

No	Pembacaan Pada AHT10 (% RH)	Pembacaan Robot (% RH)	Selisih (% RH)	Error (%)
1	61.47	58	3.47	5.65
2	63.55	62	1.55	2.44
3	63.23	61	2.23	3.53
4	64.22	62	2.22	3.46
5	62.54	58	4.54	7.26
6	63.26	58	5.26	8.31
7	61.59	59	2.59	4.21
8	61.26	58	3.26	5.32
9	62.91	58	4.91	7.80
10	62.39	61	1.39	2.23
11	62.89	61	1.89	3.01
12	61.22	61	0.22	0.36
Rata Rata			**2.79**	**4.46**

Uji coba voltase baterai dilakukan dengan cara mengukur voltase baterai dengan multitester dan membandingkan dengan pembaca robot dapat dilihat pada Gambar 38. Uji Coba Validasi Voltase Baterai.

Gambar 38. Uji Coba Validasi Voltase Baterai

Berikut tabel uji coba validasi voltase baterai dapat dilihat pada Tabel 20. Uji Coba Validasi Voltase Baterai, sedangkan uji coba validasi sensor jarak dapat dilihat pada Tabel 21. Uji Coba Validasi Sensor Jarak.

Tabel 20. Uji Coba Validasi Voltase Baterai

No	Multitester (volt)	Pembacaan Robot (volt)	Selisih (volt)	Error (%)
1	11.52	11.17	0.35	3.04
2	11.72	11.43	0.29	2.47

3	11.91	11.77	0.14	1.18
4	12.55	12.93	0.38	3.03
5	12.6	13.11	0.51	4.05
6	12.31	12.49	0.18	1.46
7	12.2	12.33	0.13	1.07
8	12.03	12.06	0.03	0.25
9	11.33	11.01	0.32	2.82
10	11.01	10.46	0.55	5.00
Rata Rata			**0.288**	**2.44**

Uji coba sensor jarak dilakukan dengan cara mengukur jarak sebenarnya pada objek dan membandingkan dengan pembacaan robot dapat dilihat pada Tabel 21. Uji Coba Validasi Sensor Jarak dan Gambar 39. Uji Coba Validasi Sensor.

Gambar 39. Uji Coba Validasi Sensor Jarak

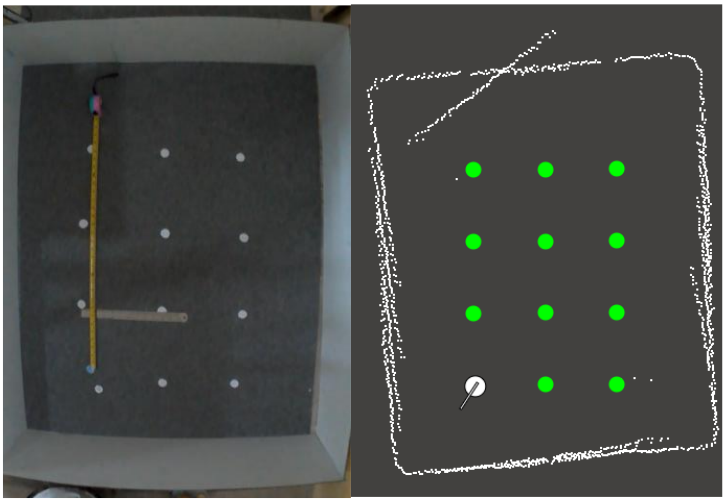
Tabel 21. Uji Coba Validasi Sensor Jarak

No	Pengukuran Mistar (mm)	Pembacaan Robot (mm)	Selisih (mm)	Error (%)
1	100	141	41	41.00
2	150	193	43	28.67
3	200	252	52	26.00
4	250	304	54	21.60
5	300	353	53	17.67
6	350	398	48	13.71
7	400	453	53	13.25
8	450	513	63	14.00
9	500	561	61	12.20
10	550	619	69	12.55
Rata Rata			53.7	20.06

c. Uji Coba Validasi Penerapan Robot Mapping

Pengujian robot maping yaitu mengkalkulasi error titik koordinat gas, dimensi ruangan dan ketepatan robot untuk kembali ke titik awal atau homing. Uji coba dilakukan dengan cara membandingkan navigasi robot dengan koordinat sebenarnya pada ruangan.

Titik koordinat pengecekan gas dilakukan untuk mengetahui akurasi koordinat dari robot dapat dilihat pada Gambar 40. Uji Coba Validasi Koordinat Gas dan Tabel 22. Uji Coba Validasi Koordinat Gas.

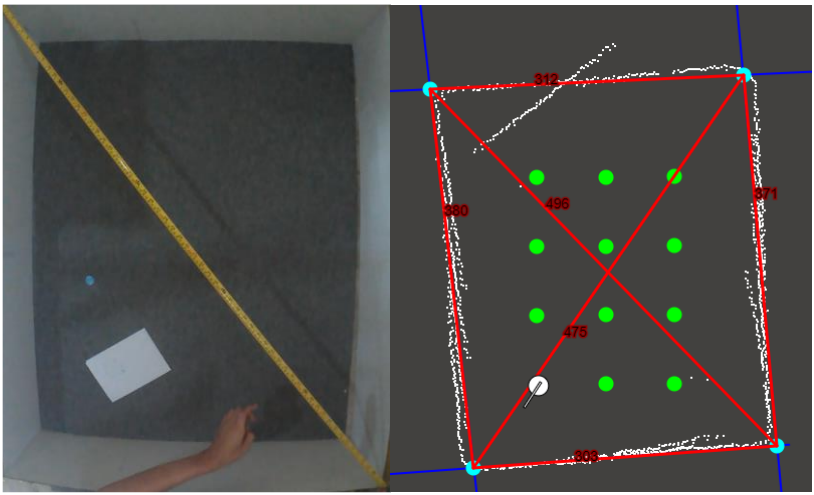


Gambar 40. Uji Coba Validasi Koordinat Gas

Tabel 22. Uji Coba Validasi Koordinat Gas

No	Set Point X (mm)	Set Point Y (mm)	Aktual X (mm)	Aktual Y (mm)	Selisih X (mm)	Selisih Y(mm)	Error X (%)	Error Y (%)
1	0	0	0	0	0	0	0.00	0.00
2	0	230	0	220	0	10	0.00	4.35
3	0	460	0	460	0	0	0.00	0.00
4	0	690	0	690	0	0	0.00	0.00
5	230	0	240	0	10	0	4.35	0.00
6	230	230	250	240	20	10	8.70	4.35
7	230	460	240	470	10	10	4.35	2.17
8	230	690	250	710	20	20	8.70	2.90
9	460	0	500	0	40	0	8.70	0.00
10	460	230	500	270	40	40	8.70	17.39
11	460	460	470	510	10	50	2.17	10.87
12	460	690	440	750	20	60	4.35	8.70
Rata Rata					**14.17**	**16.67**	**4.17**	**4.23**

Dimensi dinding pengujian dilakukan untuk mengetahui akurasi pemetaan yang dilakukan oleh robot dapat dilihat pada Gambar 41. Uji Coba Validasi Dimensi Dinding.



Gambar 41. Uji Coba Validasi Dimensi Dinding

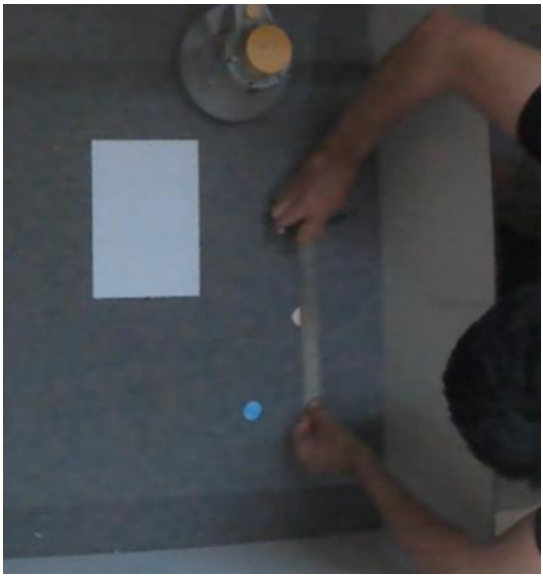
Data output map pada web interface masih berupa 2d point cloud atau titik-titik, sehingga diperlukan metode regresi linear untuk mengetahui garis lurus yang di yakini adalah dinding dapat dilihat pada Tabel 23. Uji Coba Validasi Dimensi Dinding.

Tabel 23. Uji Coba Validasi Dimensi Dinding

No	Ket Dinding	Pengukuran (mm)	Pembacaan Web UI (px)	Konversi ke milimeter (mm)	Selisih (mm)	Error (%)
1	Vertikal	1220	382	1272.06	52.06	4.27
2	Horizontal	960	349	1162.17	202.17	21.06

3	Vertikal	1220	375	1248.75	28.75	2.36
4	Horizontal	970	265	882.45	87.55	9.03
5	Diagonal	1580	488	1625.04	45.04	2.85
6	Diagonal	1550	484	1611.72	61.72	3.98
7	Vertikal	1220	375	1248.75	28.75	2.36
8	Horizontal	740	238	792.54	52.54	7.10
9	Vertikal	1220	376	1252.08	32.08	2.63
10	Horizontal	760	235	782.55	22.55	2.97
11	Diagonal	1430	448	1491.84	61.84	4.32
12	Diagonal	1440	440	1465.2	25.2	1.75
Rata Rata					**58.35**	**5.39**

Pengujian *homing* dilakukan untuk mengetahui ketepatan robot untuk kembali ke titik awal *(homing)* dapat dilihat pada Gambar 42. Uji Coba Validasi *Homing* dan Tabel 24. Uji Coba Validasi *Homing.*



Gambar 42. Uji Coba Validasi *Homing*

Tabel 24. Uji Coba Validasi Homing

No	Set point X (cm)	Set Point Y (cm)	Koodinat Robot X (cm)	Koordinat Robot Y (cm)	Selisih X (cm)	Selisih Y (cm)	Jarak Ke Home (cm)
1	0	0	13	-8	13	8	15.26
2	0	0	4.5	-5	4.5	5	6.73
3	0	0	6	-8	6	8	10.00
4	0	0	7	-3	7	3	7.62
5	0	0	4	-4.5	4	4.5	6.02
6	0	0	5	-3	5	3	5.83
7	0	0	10	-4	10	4	10.77
8	0	0	5	-6	5	6	7.81
9	0	0	6	-9	6	9	10.82
10	0	0	6	-8	6	8	10.00
Rata Rata					**6.65**	**5.85**	**9.09**

d. Uji Coba Validasi Logika Fuzzy Mamdani

Pengujian dilakukan dengan cara membandingkan output kualitas udara pada robot dengan pembacaan sensor aktual sebagai acuan dan dilakukan perhitunggan logika fuzzy mamdani dengan cara manual.

Tabel 25. Sensor VOC

Titik	Pembacaan Sensor		Output Kualitas Udara Fuzzy (%)		Selisih (%)	Error (%)
	Aktual	Robot	Aktual	Robot		
1	**VOC: 0,** CO2: 380.23, Asap: 11.55, Temp: 30.65, Hum: 61.47	**VOC: 0,** CO2: 380.23, Asap: 11.55, Temp: 30.65, Hum: 61.47	57.95	57.95	0.00	0.00
2	**VOC: 1,** CO2: 390.26, Asap: 12.32, Temp: 31.87, Hum: 63.55	**VOC: 1,** CO2: 390.26, Asap: 12.32, Temp: 31.87, Hum: 63.55	48.25	48.25	0.00	0.00
3	**VOC: 0,** CO2: 405.28, Asap: 15.67, Temp: 31.06, Hum: 63.23	**VOC: 0,** CO2: 405.28, Asap: 15.67, Temp: 31.06, Hum: 63.23	48.68	48.68	0.00	0.00
4	**VOC: 12,**	**VOC: 10,** CO2: 411.12,	38.78	38.78	0.00	0.00

	CO2: 411.12, Asap: 16.22, Temp: 30.44, Hum: 64.22	Asap: 16.22, Temp: 30.44, Hum: 64.22				
5	**VOC: 14,** CO2: 401.72, Asap: 15.98, Temp: 28.32, Hum: 62.54	**VOC: 12,** CO2: 401.72, Asap: 15.98, Temp: 28.32, Hum: 62.54	38.43	38.43	0.00	0.00
6	**VOC: 9,** CO2: 478.32, Asap: 17.59, Temp: 29.45, Hum: 63.26	**VOC: 10,** CO2: 478.32, Asap: 17.59, Temp: 29.45, Hum: 63.26	37.56	37.56	0.00	0.00
7	**VOC: 7,** CO2: 523.21, Asap: 26.46, Temp: 28.56, Hum: 61.59	**VOC: 8,** CO2: 523.21, Asap: 26.46, Temp: 28.56, Hum: 61.59	36.93	36.93	0.00	0.00
8	**VOC: 4,** CO2: 719.48, Asap: 24.75, Temp: 28.22,	**VOC: 3,** CO2: 719.48, Asap: 24.75, Temp: 28.22, Hum: 61.26	30.11	30.11	0.00	0.00

	Hum: 61.26					
9	**VOC: 3,** CO2: 578.71, Asap: 28.56, Temp: 27.49, Hum: 62.91	**VOC: 3,** CO2: 578.71, Asap: 28.56, Temp: 27.49, Hum: 62.91	34.94	34.94	0.00	0.00
10	**VOC: 2,** CO2: 688.21, Asap: 19.23, Temp: 26.07, Hum: 62.39	**VOC: 1,** CO2: 688.21, Asap: 19.23, Temp: 26.07, Hum: 62.39	45.07	46.2	1.13	2.51
11	**VOC: 0,** CO2: 579.55, Asap: 24.87, Temp: 26.36, Hum: 62.89	**VOC: 0,** CO2: 579.55, Asap: 24.87, Temp: 26.36, Hum: 62.89	43.16	43.16	0.00	0.00
12	**VOC: 0,** CO2: 721.3, Asap: 22.56, Temp: 26.45, Hum: 61.22	**VOC: 0,** CO2: 721.3, Asap: 22.56, Temp: 26.45, Hum: 61.22	50.9	50.9	0.00	0.00
				Rata Rata	**0.09**	**0.21**

Tabel 26. Sensor C02

Titik	Pembacaan Sensor		Output Kualitas Udara Fuzzy (%)		Selisih (%)	Error (%)
	Aktual	Robot	Aktual	Robot		
1	VOC: 0, **CO2: 380.23,** Asap: 11.55, Temp: 30.65, Hum: 61.47	VOC: 0, **CO2: 400**, Asap: 11.55, Temp: 30.65, Hum: 61.47	57.95	57.39	0.56	0.97
2	VOC: 1, **CO2: 390.26,** Asap: 12.32, Temp: 31.87, Hum: 63.55	VOC: 1, **CO2: 439,** Asap: 12.32, Temp: 31.87, Hum: 63.55	48.25	48.16	0.09	0.19
3	VOC: 0, **CO2: 405.28,** Asap: 15.67, Temp: 31.06, Hum: 63.23	VOC: 0, **CO2: 420,** Asap: 15.67, Temp: 31.06, Hum: 63.23	48.68	48.45	0.23	0.47
4	VOC: 12, **CO2: 411.12,** Asap: 16.22, Temp: 30.44, Hum: 64.22	VOC: 12, **CO2: 483,** Asap: 16.22, Temp: 30.44, Hum: 64.22	38.78	38.18	0.60	1.55
5	VOC: 14, **CO2: 401.72,** Asap: 15.98, Temp: 28.32,	VOC: 14, **CO2: 523,** Asap: 15.98, Temp: 28.32,	38.43	37.24	1.19	3.10

	Hum: 62.54	Hum: 62.54				
6	VOC: 9, **CO2: 478.32,** Asap: 17.59, Temp: 29.45, Hum: 63.26	VOC: 9, **CO2: 536,** Asap: 17.59, Temp: 29.45, Hum: 63.26	37.56	36.98	0.58	1.54
7	VOC: 7, **CO2: 523.21,** Asap: 26.46, Temp: 28.56, Hum: 61.59	VOC: 7, **CO2: 508,** Asap: 26.46, Temp: 28.56, Hum: 61.59	36.93	37.24	0.31	0.84
8	VOC: 4, **CO2: 719.48,** Asap: 24.75, Temp: 28.22, Hum: 61.26	VOC: 4, **CO2: 671,** Asap: 24.75, Temp: 28.22, Hum: 61.26	30.11	31.68	1.57	5.21
9	VOC: 3, **CO2: 578.71,** Asap: 28.56, Temp: 27.49, Hum: 62.91	VOC: 3, **CO2: 493,** Asap: 28.56, Temp: 27.49, Hum: 62.91	34.94	36.13	1.19	3.41
10	VOC: 2, **CO2: 688.21,** Asap: 19.23, Temp: 26.07, Hum: 62.39	VOC: 2, **CO2: 600,** Asap: 19.23, Temp: 26.07, Hum: 62.39	45.07	45.65	0.58	1.29
11	VOC: 0, **CO2: 579.55,**	VOC: 0, **CO2: 539,**	43.16	44.2	1.04	2.41

	Asap: 24.87, Temp: 26.36, Hum: 62.89	Asap: 24.87, Temp: 26.36, Hum: 62.89				
12	VOC: 0, **CO2: 721.3,** Asap: 22.56, Temp: 26.45, Hum: 61.22	VOC: 0, **CO2: 705,** Asap: 22.56, Temp: 26.45, Hum: 61.22	50.9	51.12	0.22	0.43
				Rata Rata	**0.68**	**1.78**

Tabel 27. Sensor Asap

Titik	Pembacaan Sensor		Output Kualitas Udara Fuzzy (%)		Selisih (%)	Error (%)
	Aktual	Robot	Aktual	Robot		
1	VOC: 0, CO2: 380.23, **Asap: 11.55,** Temp: 30.65, Hum: 61.47	VOC: 0, CO2: 380.23, **Asap: 11.89,** Temp: 30.65, Hum: 61.47	57.95	57.95	0.00	0.00
2	VOC: 1, CO2: 390.26, **Asap: 12.32,** Temp: 31.87, Hum: 63.55	VOC: 1, CO2: 390.26, **Asap: 13.25,** Temp: 31.87, Hum: 63.55	48.25	48.03	0.22	0.46
3	VOC: 0, CO2: 405.28, **Asap: 15.67,** Temp: 31.06,	VOC: 0, CO2: 405.28, **Asap: 18.63,** Temp: 31.06,	48.68	48.66	0.02	0.04

	Hum: 63.23	Hum: 63.23				
4	VOC: 12, CO2: 411.12, **Asap: 16.22,** Temp: 30.44, Hum: 64.22	VOC: 12, CO2: 411.12, **Asap: 18.33,** Temp: 30.44, Hum: 64.22	38.78	36.93	1.85	4.77
5	VOC: 14, CO2: 401.72, **Asap: 15.98,** Temp: 28.32, Hum: 62.54	VOC: 14, CO2: 401.72, **Asap: 19.27,** Temp: 28.32, Hum: 62.54	38.43	37.24	1.19	3.10
6	VOC: 9, CO2: 478.32, **Asap: 17.59,** Temp: 29.45, Hum: 63.26	VOC: 9, CO2: 478.32, **Asap: 21.54,** Temp: 29.45, Hum: 63.26	37.56	35.19	2.37	6.31
7	VOC: 7, CO2: 523.21, **Asap: 26.46,** Temp: 28.56, Hum: 61.59	VOC: 7, CO2: 523.21, **Asap: 26.45,** Temp: 28.56, Hum: 61.59	36.93	36.93	0.00	0.00
8	VOC: 4, CO2: 719.48, **Asap: 24.75,** Temp: 28.22, Hum: 61.26	VOC: 4, CO2: 719.48, **Asap: 25.77,** Temp: 28.22, Hum: 61.26	30.11	29.71	0.40	1.33
9	VOC: 3, CO2: 578.71,	VOC: 3, CO2: 578.71,	34.94	34.94	0.00	0.00

	Asap: 28.56, Temp: 27.49, Hum: 62.91	**Asap: 29.45,** Temp: 27.49, Hum: 62.91				
10	VOC: 2, CO2: 688.21, **Asap: 19.23,** Temp: 26.07, Hum: 62.39	VOC: 2, CO2: 688.21, **Asap: 20.26,** Temp: 26.07, Hum: 62.39	45.07	44.89	0.18	0.40
11	VOC: 0, CO2: 579.55, **Asap: 24.87,** Temp: 26.36, Hum: 62.89	VOC: 0, CO2: 579.55, **Asap: 28.51,** Temp: 26.36, Hum: 62.89	43.16	40.46	2.70	6.26
12	VOC: 0, CO2: 721.3, **Asap: 22.56,** Temp: 26.45, Hum: 61.22	VOC: 0, CO2: 721.3, **Asap: 28.76,** Temp: 26.45, Hum: 61.22	50.9	48.7	2.20	4.32
					0.93	2.25

Tabel 28. Temperature

Titik	Pembacaan Sensor		Output Kualitas Udara Fuzzy (%)		Selisih (%)	Error (%)
	Aktual	Robot	Aktual	Robot		
1	VOC: 0, CO2: 380.23, Asap: 11.55, Temp: 30.65,	VOC: 0, CO2: 380.23, Asap: 11.55, Temp: 32.2,	57.95	55.29	2.66	4.59

	Hum: 61.47	Hum: 61.47				
2	VOC: 1, CO2: 390.26, Asap: 12.32, Temp: 31.87, Hum: 63.55	VOC: 1, CO2: 390.26, Asap: 12.32, Temp: 33.3, Hum: 63.55	48.25	47.83	0.42	0.87
3	VOC: 0, CO2: 405.28, Asap: 15.67, Temp: 31.06, Hum: 63.23	VOC: 0, CO2: 405.28, Asap: 15.67, Temp: 33.3, Hum: 63.23	48.68	47.59	1.09	2.24
4	VOC: 12, CO2: 411.12, Asap: 16.22, Temp: 30.44, Hum: 64.22	VOC: 12, CO2: 411.12, Asap: 16.22, Temp: 32.4, Hum: 64.22	38.78	38.18	0.60	1.55
5	VOC: 14, CO2: 401.72, Asap: 15.98, Temp: 28.32, Hum: 62.54	VOC: 14, CO2: 401.72, Asap: 15.98, Temp: 30.6, Hum: 62.54	38.43	38.43	0.00	0.00
6	VOC: 9, CO2: 478.32, Asap: 17.59, Temp: 29.45, Hum: 63.26	VOC: 9, CO2: 478.32, Asap: 17.59, Temp: 33.2, Hum: 63.26	37.56	35.13	2.43	6.47
7	VOC: 7, CO2: 523.21,	VOC: 7, CO2: 523.21,	36.93	36.93	0.00	0.00

	Asap: 26.46, Temp: 28.56, Hum: 61.59	Asap: 26.46, Temp: 31.8, Hum: 61.59				
8	VOC: 4, CO2: 719.48, Asap: 24.75, Temp: 28.22, Hum: 61.26	VOC: 4, CO2: 719.48, Asap: 24.75, Temp: 31.7, Hum: 61.26	30.11	30.77	0.66	2.19
9	VOC: 3, CO2: 578.71, Asap: 28.56, Temp: 27.49, Hum: 62.91	VOC: 3, CO2: 578.71, Asap: 28.56, Temp: 31.7, Hum: 62.91	34.94	34.94	0.00	0.00
10	VOC: 2, CO2: 688.21, Asap: 19.23, Temp: 26.07, Hum: 62.39	VOC: 2, CO2: 688.21, Asap: 19.23, Temp: 28.6, Hum: 62.39	45.07	45.07	0.00	0.00
11	VOC: 0, CO2: 579.55, Asap: 24.87, Temp: 26.36, Hum: 62.89	VOC: 0, CO2: 579.55, Asap: 24.87, Temp: 28.9, Hum: 62.89	43.16	43.16	0.00	0.00
12	VOC: 0, CO2: 721.3, Asap: 22.56, Temp: 26.45, Hum: 61.22	VOC: 0, CO2: 721.3, Asap: 22.56, Temp: 29.9, Hum: 61.22	50.9	50.9	0.00	0.00

				Rata Rata	**0.66**	**1.49**

Tabel 29. Sensor Humidity

Titik	Pembacaan Sensor		Output Kualitas Udara Fuzzy (%)		Selisih (%)	Error (%)
	Aktual	Selisih	Aktual	Selisih		
1	VOC: 0, CO2: 380.23, Asap: 11.55, Temp: 30.65, Hum: 61.47	VOC: 0, CO2: 380.23, Asap: 11.55, Temp: 30.65, Hum: 58	57.95	62.61	4.66	8.04
2	VOC: 1, CO2: 390.26, Asap: 12.32, Temp: 31.87, Hum: 63.55	VOC: 1, CO2: 390.26, Asap: 12.32, Temp: 31.87, Hum: 62	48.25	55.31	7.06	14.63
3	VOC: 0, CO2: 405.28, Asap: 15.67, Temp: 31.06, Hum: 63.23	VOC: 0, CO2: 405.28, Asap: 15.67, Temp: 31.06, Hum: 61	48.68	53.34	4.66	9.57
4	VOC: 12,	VOC: 12, CO2: 411.12,	38.78	39.73	0.95	2.45

	CO2: 411.12, Asap: 16.22, Temp: 30.44, Hum: 64.22	Asap: 16.22, Temp: 30.44, Hum: 62				
5	VOC: 14, CO2: 401.72, Asap: 15.98, Temp: 28.32, Hum: 62.54	VOC: 14, CO2: 401.72, Asap: 15.98, Temp: 28.32, Hum: 58	38.43	39.93	1.50	3.90
6	VOC: 9, CO2: 478.32, Asap: 17.59, Temp: 29.45, Hum: 63.26	VOC: 9, CO2: 478.32, Asap: 17.59, Temp: 29.45, Hum: 58	37.56	37.92	0.36	0.96
7	VOC: 7, CO2: 523.21, Asap: 26.46, Temp: 28.56, Hum: 61.59	VOC: 7, CO2: 523.21, Asap: 26.46, Temp: 28.56, Hum: 59	36.93	36.93	0.00	0.00
8	VOC: 4,	VOC: 4, CO2: 719.48,	30.11	30.11	0.00	0.00

	CO2: 719.48, Asap: 24.75, Temp: 28.22, Hum: 61.26	Asap: 24.75, Temp: 28.22, Hum: 58				
9	VOC: 3, CO2: 578.71, Asap: 28.56, Temp: 27.49, Hum: 62.91	VOC: 3, CO2: 578.71, Asap: 28.56, Temp: 27.49, Hum: 58	34.94	34.94	0.00	0.00
10	VOC: 2, CO2: 688.21, Asap: 19.23, Temp: 26.07, Hum: 62.39	VOC: 2, CO2: 688.21, Asap: 19.23, Temp: 26.07, Hum: 61	45.07	44.46	0.61	1.35
11	VOC: 0, CO2: 579.55, Asap: 24.87, Temp: 26.36, Hum: 62.89	VOC: 0, CO2: 579.55, Asap: 24.87, Temp: 26.36, Hum: 61	43.16	44.64	1.48	3.43
12	VOC: 0,	VOC: 0, CO2: 721.3,	50.9	51.81	0.91	1.79

	CO2: 721.3, Asap: 22.56, Temp: 26.45, Hum: 61.22	Asap: 22.56, Temp: 26.45, Hum: 61				
					1.85	3.84

BAB 4
KESIMPULAN

Buku ini menyimpulkan bahwa model robot pendeteksi kualitas gas dalam ruangan dapat menghasilkan *output* dan ditampilkan menggunakan web *user interface*. Namun output kualitas udara pada setiap titik masih belum mencapai 100% sempurna, hal ini masih terdapat error dengan rata rata sebesar 5.6% dari kualitas sebenarnya. Selisih error tersebut dipengaruhi oleh pembacaan sensor diantaranya sensor VOC memiliki error 10.9%, sensor CO2 10.6%, sensor asap 11.7%, sensor temperatur 9.6%, sensor kelembaban 4.5%. Pengaruh error setiap sensor terhadap proses logika mamdani yaitu sensor VOC 0.2%, sensor CO2 1.8%, sensor asap 2.3%, sensor temperatur 1.5%, sensor kelembaban 3.8%. Kemudian, implementasi pemetaan pada robot memiliki error diantaranya koordinat gas X rata rata error 4.2%, koordinat gas Y rata rata error 4.2%, dimensi dinding rata rata error 5.4%. Selisih tersebut dipengaruhi oleh perpindahan robot yang memiliki sensitivitas sebesar 1-millimeter dan perputaran atau belok memiliki sensitivitas 1 derajat pada setiap perintah. Selain itu, selisih atau error dipengaruhi juga oleh beberapa faktor diantaranya dari akurasi pembacaan sensor jarak dan sudut sehingga terdapat selisih antara lingkungan sebenarnya dengan data yang disajikan. Setiap pemetaan yang dilakukan membutuhkan koneksi antara web *user interface* dan robot, jika koneksi terputus maka robot tidak akan menerima perintah dan berhenti. Setiap data dari sensor akan di-*encode* oleh

robot ke dalam format JSON lalu dikirimkan kepada web *user interface*, setiap data JSON yang diterima oleh web *user interface* akan diolah, dicatat dan disajikan dalam bentuk map. Selain itu data dapat di download dengan bentuk file .json.

Model robot pendeteksi kualitas udara dalam ruangan belum sempurna secara keseluruhan, masih memiliki keterbatasan yang harus disempurnakan, baik dari sisi hardware maupun software. Untuk pengembangan lebih lanjut disarankan untuk diimplementasikan pada ruangan yang sebenarnya dan menggunakan sensor yang spesifikasinya dengan standar tinggi sehingga hasil yang didapatkan lebih baik lagi.

DAFTAR PUSTAKA

Denih, A. 2020. *Dasar-Dasar Pengembangan Integrasi GIS & IOT*. Komojoyo Press. DIY.

Hijriani, A., Muludi, K & Andini, E. A. 2016. Implementasi Metode Regresi Liniear Sederhana Pada Penyajian Hasil Prediksi Pemakaian Air Bersih PDAM WAY RILAU Kota Bandar Lampung Dengan Sistem Informasi Geografis. Jurnal Informatika Mulawarman. 11(2): 37 - 42.

Mugirahayu, A. S., Linawati. L & Setiawan, A. 2021. Penentuan Status Kewaspadaan COVID-19 Pada Suatu Wilayah Menggunakan Metode Fuzzy Inference System (FIS) Mamdani. Jurnal Sains dan Edukasi Sains. 4(1): 28 - 39.

Permana, A. Y & Romadlon, P. 2019. Perancangan Sistem Informasi Penjualan Perumahan Menggunakan Metode SDLC Pada PT. Mandiri Land Prosperous Berbasis Mobile.

Prabowo., Kuat & Muslim, B. 2017. *Buku ajar kesehatan lingkungan*. Penyehatan udara. Kementrian Kesehatan Republik Indonesia.

Renaldi & Bima. 2020. Rancang Bangun Robot SAR Sebagai Pendeteksi Gas Beracun Pra Evakuasi. JATI (Jurnal Mahasiswa Teknik Informatika).

Sensirion Gas Platform. 2017. Datasheet SGP30 Version 0.9. Diakses pada 20 Maret 2021 melalui https://sensirion.com.

Sholih, F. A. 2017. Implementasi Trigonometri Pada Pengukuran Tinggi Badan Menggunakan Arduino Mega 2560. Jurnal Simki-Techsain. 1(10).

STMicroelectronics. 2018. *VL53L0X World's smallest Time-of-Flight ranging and gesture detection sensor Datasheet - production data*. Diakses pada 17 Maret 2021 melalui https://www.st.com.

Suriansyah., Iqbal, M & Harsani, P. 2018 Air Pollution Monitoring for Bogor Smart City Based Internet of Things and *SocialMedia* (Twitter) IOP Conf. Series: Materials Science and Engineering 621 (2019) 012006.

Syahrul. 2021. *Majalah Ilmiah Unikom Vol.6, No. 2 Motor Stepper: Teknologi, Metoda Dan Rangkaian Kontrol*. Jurusan Teknik Komputer Universitas Komputer Indonesia Majalah Ilmiah Unikom © Unikom Center.

The National Institute for Occupational Safety and Health (NIOSH). 2017.

Widyantara., Helmy., Rivai, M & Purwanto, D. 2018. Gas Source Localization Using an Olfactory Mobile Robot Equipped with Wind Direction Sensor. International Conference on Computer Enginering, Network and Intelegent Multimedia (CENIM). Surabaya Indonesia.

Xingdong., Liu. X, L & Zhong, X. 2018. Research on simultaneous localization and mapping of indoor mobile robot. IOP Conf. Series: Journal of Physics: Conf. Series 1074 (2018) 012099.

Zholehaw., Sri. Ali Basrah Pulungan & Hamdani 2019. Sistem Monitoring Realtime Gas CO Pada Asap Rokok Berbasis Mikrokontroler. Program Studi Teknik Elektro Industri Jurusan Teknik Elektro, Universitas Negeri Padang.

INDEKS

S

T

U

V

LAMPIRAN

Lampiran 1. Serverside *(Robot Code)*

```
#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <SPIFFS.h>
#include <ESP_FlexyStepper.h>
// #include "Arduino_FreeRTOS.h"
// #include <driver/i2c.h>
// #include <esp_log.h>
// #include <esp_err.h>
// #include <freertos/FreeRTOS.h>
// #include <freertos/task.h>
#include "MPU6050_6Axis_MotionApps20.h"
// #include "sdkconfig.h"
#include <VL53L0X.h>
#include <ESP32Servo.h>
#include <ArduinoJson.h>

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIR
E
    #include "Wire.h"
#endif

#ifndef M_PI
    #define M_PI 3.14159265358979323846
#endif

#define INTERRUPT_PIN 5
#define OUTPUT_READABLE_YAWPITCHROLL

//#define PIN_SDA 21
//#define PIN_SCL 22
```

```
#define STEP_PER_MM 3.637827270671893389
//#define MM_PER_DEGREE 0.82903139469730654904   //95m
m Diameter
#define MM_PER_DEGREE 0.84648468721724984481     //97mm
 Diameter

#define DIR_R 32
#define STEP_R 33
#define DIR_L 25
#define STEP_L 26

#define SPEED_MM_PER_SECOND 35
#define ACCELERATION_MM_PER_SECOND 140
#define DECELERATION_MM_PER_SECOND 140

#define SERVO_NEUTRAL 99
#define SERVO_RUN_CW 94
#define SERVO_RUN_ALIGNMENT 96

#define HALL_SENSOR 19  // INVERTED !!! || ON = 0 || OFF =
1
#define LED 2

//#define MM_PER_DEGREE 0.82903139469730654904

ESP_FlexyStepper MOTOR_R;
ESP_FlexyStepper MOTOR_L;

//TaskHandle_t MPU_TaskInit_Handle;
TaskHandle_t MPU_TaskRun_Handle;
TaskHandle_t Client_Task_Handle;

VL53L0X sensor;
Servo motor;
MPU6050 mpu;

// StaticJsonDocument<9216> doc;
```

```
DynamicJsonDocument doc(9216); // fixed size 9216
JsonObject root = doc.to<JsonObject>();
char buffer[9216]; // create temp buffer

bool dmpReady = false;
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MP
U
uint8_t devStatus;
uint16_t packetSize;
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

int angle;
int angle_old = 0;
int scanTask;
uint16_t count;
bool hall_detect;

Quaternion q;           // [w, x, y, z]         quaternion container
VectorFloat gravity;    // [x, y, z]            gravity vector
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and
 gravity vector

// const char* ssid = "RP";
// const char* password = "rumahpenelitian123";
const char* ssid = "MIX";
const char* password = "123456789";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);
AsyncWebSocket ws("/ws");

String splitString(String data, char separator, int index){
  int found = 0;
  int strIndex[] = {0, -1};
  int maxIndex = data.length()-1;
```

```
  for(int i=0; i<=maxIndex && found<=index; i++){
    if(data.charAt(i)==separator || i==maxIndex){
      found++;
      strIndex[0] = strIndex[1]+1;
      strIndex[1] = (i == maxIndex) ? i+1 : i;
    }
  }
  return found>index ? data.substring(strIndex[0], strIndex[1]) : "";
}

void sendHeading(){
  ws.textAll(String("heading-value"));
}

void forward(int target){
  MOTOR_R.resumeService();
  MOTOR_L.resumeService();
  long target_step = target*STEP_PER_MM;

  MOTOR_R.setCurrentPositionInSteps(0);
  MOTOR_L.setCurrentPositionInSteps(0);
  MOTOR_R.setTargetPositionInSteps(target_step);
  MOTOR_L.setTargetPositionInSteps(target_step);

  while (MOTOR_L.getCurrentPositionInSteps() != target_step){
    Serial.print("DEBUG : MOTOR RUNNING !! POS = ");
    Serial.println(MOTOR_L.getCurrentPositionInSteps());
    // Serial.println(MOTOR_L.getTaskStackHighWaterMark());
  }

  MOTOR_R.suspendService();
  MOTOR_L.suspendService();
}

void setHeading(int target){
  MOTOR_R.resumeService();
  MOTOR_L.resumeService();
```

```
  long target_step = target*STEP_PER_MM;

  MOTOR_R.setCurrentPositionInSteps(0);
  MOTOR_L.setCurrentPositionInSteps(0);
  MOTOR_R.setTargetPositionInMillimeters(-target_step);
  MOTOR_L.setTargetPositionInMillimeters(target_step);

  while (MOTOR_L.getCurrentPositionInSteps() != target_step){
    Serial.print("DEBUG : MOTOR RUNNING !! POS = ");
    Serial.println(MOTOR_L.getCurrentPositionInSteps());
    // Serial.println(MOTOR_L.getTaskStackHighWaterMark());
  }
  MOTOR_R.suspendService();
  MOTOR_L.suspendService();
}

int getAngle(){
    mpu.dmpGetCurrentFIFOPacket(fifoBuffer);
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
    return (ypr[0] * 57.2958)+180;
}

// MPU REALTIME DEBUG
void task_display(void *pvParameters){
  (void) pvParameters;
  vTaskDelay(200);

  while (1){
    vTaskDelay(1);
    count = 0;
    int hall = 0;
    int old_hall = 0;

    if (scanTask > 0){
      motor.write(SERVO_RUN_CW);
```

```
    while (count <= 1){
      hall = digitalRead(HALL_SENSOR);
      if (hall != old_hall){
        old_hall = hall;
        count = count + 1;
      }

      angle = getAngle();
      if (angle != angle_old){
        root["a"+String(angle)] = sensor.readRangeSingleMillimeter
s();
        angle_old = angle;
        Serial.println(angle);
      }
    }

    motor.write(SERVO_NEUTRAL);
    size_t len = serializeJson(root, buffer);  // serialize to buffer
    ws.textAll(buffer, len);
    scanTask = 0;
   }
  }
  vTaskDelete(NULL);
}

void scanWall(){
  count = 0;
  int hall = 0;
  int old_hall = 0;

  motor.write(SERVO_RUN_CW);
  while (count <= 1){
    hall = digitalRead(HALL_SENSOR);
    if (hall != old_hall){
      old_hall = hall;
      count = count + 1;
    }
```

```
    angle = getAngle();
    if (angle != angle_old){
      root["a"+String(angle)] = sensor.readRangeSingleMillimeters()
;
      angle_old = angle;
      Serial.println(angle);
    }
  }
  motor.write(SERVO_NEUTRAL);
  root["voc"] = 0.0;
  root["co2"] = 0.0;
  root["asap"] = 0.0;
  root["temp"] = 0.0;
  root["hum"] = 0.0;
  size_t len = serializeJson(root, buffer);  // serialize to buffer
  ws.textAll(buffer, len); // send buffer to web socket
  // Serial.println(buffer);
}

void handleWebSocketMessage(void *arg, uint8_t *data, size_t len
) {
  AwsFrameInfo *info = (AwsFrameInfo*)arg;
  if (info->final && info->index == 0 && info-
>len == len && info->opcode == WS_TEXT) {
    data[len] = 0;
    String command = splitString((char*)data, '=', 0);
    String value = splitString((char*)data, '=', 1);

    if (command == "setheading"){
      Serial.println("SET HEADING");
      int val_head = value.toInt();
      setHeading(val_head);
    }

    else if (command == "setforward"){
      Serial.println("SET FORWARD");
      int val_forward = value.toInt();
```

```
      forward(val_forward);
      //clearPosition();
      //sendHeading();
    }

    else if (command == "readsensor"){
      // MOTOR_R.suspendService();
      // MOTOR_L.suspendService();
      Serial.println("DEBUG : Reading Sensor.....");
      scanTask = 1;
      // scanWall();
      Serial.println("DEBUG : Read Sensor Complete.....");
      // MOTOR_R.resumeService();
      // MOTOR_L.resumeService();
      //clearPosition();
      //sendHeading();
    }

    else {
      Serial.println("UNKNOWN COMMAND");
    }
  }
}

void onEvent(AsyncWebSocket *server, AsyncWebSocketClient *
client, AwsEventType type,
             void *arg, uint8_t *data, size_t len) {
  switch (type) {
    case WS_EVT_CONNECT:
      Serial.printf("WebSocket client #%u connected from %s\n", cli
ent->id(), client->remoteIP().toString().c_str());
      break;
    case WS_EVT_DISCONNECT:
      Serial.printf("WebSocket client #%u disconnected\n", client-
>id());
      break;
    case WS_EVT_DATA:
```

```
      handleWebSocketMessage(arg, data, len);
      break;
    case WS_EVT_PONG:
      Serial.println("WS_EVT_PONG");
      break;
    case WS_EVT_ERROR:
      Serial.println("ERROR");
      break;
  }
}

void initWebSocket() {
  ws.onEvent(onEvent);
  server.addHandler(&ws);
}

String processor(const String& var){
  Serial.println(var);
  if(var == "STATE"){
    if (1){
      return "ON";
    }
    else{
      return "OFF";
    }
  }
}

void task_web_client(void*){
  while(1){
    ws.cleanupClients();
    vTaskDelay(1);
  }
}
volatile bool mpuInterrupt = false;
void dmpDataReady() {
    mpuInterrupt = true;
```

```
}

void sensorAlignment(){
  hall_detect = digitalRead(HALL_SENSOR);
  if (hall_detect){
    motor.write(SERVO_RUN_ALIGNMENT);
  }
  while(hall_detect){
    hall_detect = digitalRead(HALL_SENSOR);
  }
  motor.write(SERVO_NEUTRAL);
}

void setup(){
  scanTask = 0;
  #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WI
RE
    Wire.begin();
    Wire.setClock(400000);
  #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FA
STWIRE
    Fastwire::setup(400, true);
  #endif

  Serial.begin(115200);
  motor.attach(23);
  motor.write(SERVO_NEUTRAL);
  if(!SPIFFS.begin(true)){
    Serial.println("An Error has occurred while mounting SPIFFS");
    return;
  }

  pinMode(HALL_SENSOR, INPUT);
  pinMode(LED, OUTPUT);
  sensorAlignment();

  sensor.setTimeout(500);
```

```
  if (!sensor.init()){
     Serial.println("Failed to detect and initialize sensor!");
     while (1) {}
  }

  // reduce timing budget to 20 ms (default is about 33 ms)
  sensor.setMeasurementTimingBudget(20000);

  // initialize device
  Serial.println(F("Initializing I2C devices..."));
  mpu.initialize();
  pinMode(INTERRUPT_PIN, INPUT);

  Serial.println(F("Testing device connections..."));
  Serial.println(mpu.testConnection() ? F("MPU6050 connection su
ccessful") : F("MPU6050 connection failed"));

  Serial.println(F("Initializing DMP..."));
  devStatus = mpu.dmpInitialize();

  mpu.setXGyroOffset(220);
  mpu.setYGyroOffset(76);
  mpu.setZGyroOffset(0);   // -85 default
  mpu.setZAccelOffset(1888); // 1688 factory default for my test ch
ip

  if (devStatus == 0) {
    mpu.CalibrateAccel(6);
    mpu.CalibrateGyro(6);
    mpu.PrintActiveOffsets();
    Serial.println(F("Enabling DMP..."));
    mpu.setDMPEnabled(true);
    Serial.print(F("Enabling interrupt detection (Arduino external int
errupt "));
    Serial.print(digitalPinToInterrupt(INTERRUPT_PIN));
    Serial.println(F(")..."));
```

```
    attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpD
ataReady, RISING);
    mpuIntStatus = mpu.getIntStatus();
    Serial.println(F("DMP ready! Waiting for first interrupt..."));
    dmpReady = true;
    packetSize = mpu.dmpGetFIFOPacketSize();
  } else {
      Serial.print(F("DMP Initialization failed (code "));
      Serial.print(devStatus);
      Serial.println(F(")"));
  }

  MOTOR_R.connectToPins(STEP_R, DIR_R);
  MOTOR_L.connectToPins(STEP_L, DIR_L);
  MOTOR_R.setStepsPerMillimeter(STEP_PER_MM);
  MOTOR_L.setStepsPerMillimeter(STEP_PER_MM);
  MOTOR_R.setSpeedInMillimetersPerSecond(SPEED_MM_PER
_SECOND);
  MOTOR_L.setSpeedInMillimetersPerSecond(SPEED_MM_PER
_SECOND);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }

  initWebSocket();

  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request)
{
    request->send(SPIFFS, "/index.html", String(), false, processor);
  });
```

```
  server.on("/bootstrap-
grid.min.css", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/bootstrap-grid.min.css");
  });

  server.on("/style.css", HTTP_GET, [](AsyncWebServerRequest *
request){
    request->send(SPIFFS, "/style.css");
  });

  server.on("/p5.min.js", HTTP_GET, [](AsyncWebServerRequest
*request){
    request->send(SPIFFS, "/p5.min.js");
  });

  server.on("/app.js", HTTP_GET, [](AsyncWebServerRequest *re
quest){
    request->send(SPIFFS, "/app.js");
  });

  MOTOR_R.startAsService(-1);
  MOTOR_L.startAsService(-1);

  xTaskCreateUniversal(task_display, "MPU_RUN_TASK", 1024*
2, NULL, 1, &MPU_TaskRun_Handle, -1);
  server.begin();
  xTaskCreate(task_web_client, "WEB_CLIENT_TASK", 1024, N
ULL, 1, &Client_Task_Handle);
  //digitalWrite(ledPin, HIGH);
  delay(3000);
  dmpReady = true;
  MOTOR_R.suspendService();
  MOTOR_L.suspendService();
  digitalWrite(LED, HIGH);
  // Print ESP Local IP Address
  Serial.println(WiFi.localIP());
}
```

```
void loop() {
}
```

CLIENTSIDE (web ui code) JavaScript

```
var gateway = "ws://192.168.43.27/ws";
// var gateway = "ws://192.168.1.11/ws";

var websocket;

var corX = 0;
var corY = 0;
var angle = 0;
var wall = [];
var scale = 0.5;
var connectionStatus = "...";

var wallObject = {};
var dotCount = 0;

window.addEventListener("load", onLoad);

function initWebSocket() {
  console.log("Trying to open a WebSocket connection...");
  websocket = new WebSocket(gateway);
  websocket.onopen    = onOpen;
  websocket.onclose   = onClose;
  websocket.onmessage = onMessage; // <-- add this line
}

function onOpen(event) {
  // eslint-disable-next-line no-console
  // console.log("Connection opened");
  connectionStatus = "Connection opened";
```

```
}
function onClose(event) {
  console.log("Connection closed");
  connectionStatus = "Connection closed";
  setTimeout(initWebSocket, 5000);
}

function onMessage(event) {
  console.log(event.data);
  var data = JSON.parse(event.data);
  var keys = Object.keys(data);
  for (var j=0; j<360; j++){
    if(keys[j] != "voc" && keys[j] != "co2" && keys[j] != "asap" &
& keys[j] != "temp" && keys[j] != "hum"){
      var name = keys[j];
      var strIndex = String(name);
      var splitIndex = strIndex.split("a");
      var intIndex = parseInt(splitIndex[1], 10);

      var value = data[name];
      wall[intIndex] = value;
    }
  }

  for (var i=0; i<wall.length; i++){
    var a = (i * 0.0174533);
    var num = parseInt(wall[i], 10);
    if (num){
      var x = ((sin(a) * num * 0.3) + robotCorX);
      var y = ((cos(a) * num * 0.3) + robotCorY);

      wallObject[dotCount.toString()] = x + "," + y;
      console.log(wallObject);
      dotCount += 1;
    }
    else {
      console.log("DEBUG : Array kosong/error di index #"+i);
```

```
    }
  }

  console.log(wallObject);

  document.getElementById('state').innerHTML = wall;
  //document.getElementById('state').innerHTML = event.data;
}

function onLoad(event) {

  // var keys = Object.keys(wallObject);
  // for (var j=0; j<360; j++){
  //   if(keys[j] != "voc" && keys[j] != "co2" && keys[j] != "asap"
&& keys[j] != "temp" && keys[j] != "hum"){
  //     var name = keys[j];
  //     var strIndex = String(name);
  //     var splitIndex = strIndex.split("a");
  //     var intIndex = parseInt(splitIndex[1], 10);

  //     var value = data[name];
  //     wall[intIndex] = value;
  //   }
  // }
  initWebSocket();
  initButton();
}

function initButton() {
  document.getElementById('button_set_heading').addEventListene
r('click', setHeading);
  document.getElementById('button_set_forward').addEventListene
r('click', setForward);
  document.getElementById('button_read_sensor').addEventListene
r('click', readSensor);
  //document.getElementById('button').addEventListener('click', tog
gle);
```

```
}

function setHeading(){
  var headingValue = document.getElementById("input_set_headin
g").value;
  websocket.send('setheading='+headingValue);
  angle = angle + parseInt(headingValue);
  console.log(typeof(angle));
}

function setForward(){
  var forwardValue = document.getElementById("input_set_forwar
d").value;
  websocket.send('setforward='+forwardValue);
  angleMode(RADIANS);
  corX = corX + parseInt((sin(angle * 0.0174533) * parseInt(forwar
dValue)));
  corY = corY + parseInt((cos(angle * 0.0174533) * parseInt(forwar
dValue)));
}

function readSensor(){
  websocket.send('readsensor');
}

function drawScanValue(){

}

function setup() {
  let canvas = createCanvas(850, 560);
  canvas.position(20, 80);
  background("#42413F");
}

function draw() {
```

```
  var robotCorX = (0.3*corX) + 70;
  var robotCorY = -(0.3*corY) + 500;
  fill(255);
  noStroke();
  textSize(15);
  textStyle(NORMAL);
  text("Connection :" + connectionStatus, 20, 20);
  text("Coordinate (x, y):"+ corX + ", " + corY, 350, 20);  // offset fr
om edge 30px
  text("Heading :" + angle, 700, 20);

  if (wall.length > 1){
    stroke(255);
    strokeWeight(2);
    angleMode(RADIANS);
    for (var i=0; i<wall.length; i++){
      var a = (i * 0.0174533);
      var num = parseInt(wall[i], 10);
      if (num){
        var x = ((sin(a) * num * 0.3) + robotCorX);
        var y = ((cos(a) * num * 0.3) + robotCorY);
      }
      else {
        console.log("DEBUG : Array kosong/error di index #"+i);
      }
      point (x , y);
    }
    wall = [];
  }
  stroke(0);
  angleMode(DEGREES);
  strokeWeight(1);
  circle(robotCorX, robotCorY, 20);
  translate(robotCorX, robotCorY);
  rotate(angle);
  rect(-1, 4, 2, -30);
}
```

PROFIL PENULIS

Asep Denih, S. Kom., M. Sc., Ph.D. adalah Dosen di Program Studi Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Pakuan, Bogor sejak tahun 1997. Penulis mendapatkan gelar Sarjana Komputer pada tahun 1996 dari Universitas Gunadarma, Depok - Indonesia.

Kemudian, penulis melanjutkan studi S2 di bidang Information Technology for Natural Resources Management dan lulus dengan mendapatkan gelar Master of Science pada tahun 2005 dari IPB University, Bogor – Indonesia.

Setelah lulus S2, penulis melanjutkan studi S3 di bidang Environmental Informatics dan lulus pada tahun 2019 dari University of Miyazaki, Jepang. Setelah lulus dari Jepang, penulis mendapat amanah sebagai Dekan FMIPA UNPAK Periode 2020-2025. Saat ini, penulis sebagai Editor in Chief di Journal Komputasi, Program Studi Ilmu Komputer, FMIPA UNPAK, Bogor.

Beberapa pengalaman berorganisasi yang pernah diikuti oleh penulis, diantaranya adalah sebagai Ketua Perhimpunan Pelajar Indonesia Jepang Komisariat Miyazaki, Ketua Perhimpunan Pelajar Indonesia Jepang Koordinator Wilayah Jepang Selatan yang meliputi Kyushu Prefecture dan Okinawa, Anggota Badan Pengawas Perhimpunan Pelajar Indonesia Jepang, dan President Muslim Community of Miyazaki, Jepang.

Profil Penulis

Asep Denih, S.Kom., M.Sc., Ph.D. adalah Dosen di Program Studi Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Pakuan, Bogor sejak tahun 1997. Penulis mendapatkan gelar Sarjana Komputer pada tahun 1996 dari Universitas Gunadarma, Depok - Indonesia.

Kemudian, penulis melanjutkan studi S2 di bidang Information Technology for Natural Resources Management dan lulus dengan mendapatkan gelar Master of Science pada tahun 2005 dari IPB University, Bogor – Indonesia.

Setelah lulus S2, penulis melanjutkan studi S3 di bidang Environmental Informatics dan lulus pada tahun 2019 dari University of Miyazaki, Jepang. Setelah lulus dari Jepang, penulis mendapat amanah sebagai Dekan FMIPA UNPAK Periode 2020-2025. Saat ini, penulis sebagai Editor in Chief di Journal Komputasi, Program Studi Ilmu Komputer, FMIPA UNPAK, Bogor.

Beberapa pengalaman berorganisasi yang pernah diikuti oleh penulis, diantaranya adalah sebagai Ketua Perhimpunan Pelajar Indonesia Jepang Komisariat Miyazaki, Ketua Perhimpunan Pelajar Indonesia Jepang Koordinator Wilayah Jepang Selatan yang meliputi Kyushu Prefecture dan Okinawa, Anggota Badan Pengawas Perhimpunan Pelajar Indonesia Jepang, dan President Muslim Community of Miyazaki, Jepang.