

DOKUMEN CD



Sistem Navigasi Cerdas Robot Pengantar Makanan Dengan Pendekatan *Machine Learning*

Oleh

Muhammad Tsabit 1101213103
Fernando Amanda Nikola 1101213156
Nadya Alifia Chairunnisa 1101213001

**PRODI S1 TEKNIK TELEKOMUNIKASI
FAKULTAS TEKNIK ELEKTRO
UNIVERSITAS TELKOM
BANDUNG
2025**

LEMBAR PENGESAHAN
BUKU CAPSTONE DESIGN

**Sistem Navigasi Cerdas Robot Pengantar Makanan Dengan
Pendekatan *Machine Learning***

*(INTELLIGENT NAVIGATION SYSTEM OF FOOD DELIVERY ROBOTS WITH
MACHINE LEARNING APPROACH)*

Telah disetujui dan disahkan sebagai bagian dari Capstone Design
Program S1 Teknik Telekomunikasi
Fakultas Teknik Elektro
Universitas Telkom
Bandung

Disusun oleh:

Muhammad Tsabit / 1101213103

Fernando Amanda Nikola / 1101213156

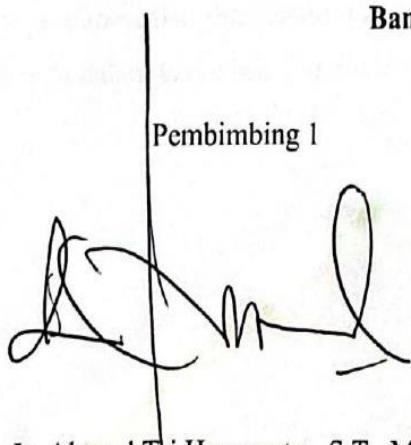
Nadya Alifia Chairunnisa / 1101213001

Bandung, 18 Juni 2025

Menyetujui,

Pembimbing 1

Pembimbing 2



Ir. Ahmad Tri Hanuranto., S.T., M.T.
NIP. : 93660031



Dr. Ir. Sony Sumaryo., S.T., M.T.
NIP. 93670031

LEMBAR PERNYATAAN ORISINALITAS

Saya, yang bertanda tangan di bawah ini

Nama : Muhammad Tsabit
NIM : 1101213103
Alamat : Jalan Hateng 2 No.33 RT 01/07
No. Telepon : 085888426135
Email : tsabit.bit18@gmail.com

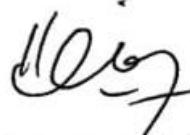
Menyatakan bahwa Buku Capstone Design ini merupakan karya orisinal saya sendiri bersama dengan kelompok Capstone Design saya, dengan judul:

Sistem Navigasi Cerdas Robot Pengantar Makanan Dengan Pendekatan *Machine Learning*

*(INTELLIGENT NAVIGATION SYSTEM OF FOOD DELIVERY ROBOTS WITH
MACHINE LEARNING APPROACH)*

Atas pernyataan ini, saya siap menanggung resiko/sanksi yang dijatuhkan kepada saya apabila dikemudian ditemukan adanya pelanggaran terhadap kejujuran akademik atau etika keilmuan dalam karya ini, atau ditemukan bukti yang menunjukkan ketidak aslian karya ini.

Bandung, 18 Juni 2025



Muhammad Tsabit

1101213103

LEMBAR PERNYATAAN ORISINALITAS

Saya, yang bertanda tangan di bawah ini

Nama : Fernando Amanda Nikola
NIM : 1101213156
Alamat : Bumi Serang Baru Blok DD13 No8
No. Telepon : 088808783704
Email : fernandonikola0@gmail.com

Menyatakan bahwa Buku Capstone Design ini merupakan karya orisinal saya sendiri bersama dengan kelompok Capstone Design saya, dengan judul:

Sistem Navigasi Cerdas Robot Pengantar Makanan Dengan Pendekatan *Machine Learning*

*(INTELLIGENT NAVIGATION SYSTEM OF FOOD DELIVERY ROBOTS WITH
MACHINE LEARNING APPROACH)*

Atas pernyataan ini, saya siap menanggung resiko/sanksi yang dijatuhkan kepada saya apabila dikemudian ditemukan adanya pelanggaran terhadap kejujuran akademik atau etika keilmuan dalam karya ini, atau ditemukan bukti yang menunjukkan ketidak aslian karya ini.

Bandung, 18 Juni 2025



Fernando Amanda Nikola
1101213156

LEMBAR PERNYATAAN ORISINALITAS

Saya, yang bertanda tangan di bawah ini

Nama : Nadya Alifia Chairunnisa
NIM : 1101213001
Alamat : Jln Puri Permata, Cipondoh Makmur, Cipondoh, Tangerang, Banten
No. Telepon : 081284395059
Email : nanadyanugraha@gmail.com

Menyatakan bahwa Buku Capstone Design ini merupakan karya orisinal saya sendiri bersama dengan kelompok Capstone Design saya, dengan judul:

Sistem Navigasi Cerdas Robot Pengantar Makanan Dengan Pendekatan *Machine Learning*

*(INTELLIGENT NAVIGATION SYSTEM OF FOOD DELIVERY ROBOTS WITH
MACHINE LEARNING APPROACH)*

Atas pernyataan ini, saya siap menanggung resiko/sanksi yang dijatuhkan kepada saya apabila dikemudian ditemukan adanya pelanggaran terhadap kejujuran akademik atau etika keilmuan dalam karya ini, atau ditemukan bukti yang menunjukkan ketidak aslian karya ini.

Bandung, 18 Juni 2025



Nadya Alifia Chairunnisa
1101213001

ABSTRAK

Perkembangan teknologi otomasi kini semakin nyata dirasakan di berbagai bidang, termasuk dalam sektor layanan seperti restoran. Salah satu inovasi menarik yang mulai banyak diterapkan adalah penggunaan robot pengantar makanan. Inovasi ini hadir untuk meningkatkan efisiensi dan ketepatan dalam proses pelayanan. Namun, penerapan robot di lingkungan restoran yang dinamis bukan tanpa tantangan. Perubahan posisi meja, lalulalang pelanggan, dan kondisi ruang yang tidak selalu sama membuat navigasi menjadi hal yang kompleks. Sistem navigasi konvensional yang masih bergantung pada titik tetap dan tidak mampu merespons perubahan secara real-time menjadi salah satu kendala utama.

Untuk menjawab tantangan tersebut, dikembangkanlah sistem navigasi cerdas berbasis *Machine Learning* yang mampu memetakan lingkungan secara langsung (*real-time*). Sistem ini menggunakan sensor RPLiDAR dan metode SLAM "Simultaneous Localization and Mapping" untuk membangun peta lingkungan secara otomatis dan akurat. Proses pengolahan data dilakukan oleh Raspberry Pi sebagai pusat kendali. Dalam perencanaan jalur, sistem ini menggabungkan algoritma K-Nearest Neighbors (KNN) untuk pengenalan lingkungan dan Breadth-First Search (BFS) untuk pencarian rute terbaik. Setelah rute ditentukan, jalur tersebut disempurnakan dengan teknik *smoothing* agar robot dapat bergerak lebih mulus dan bebas dari hambatan.

Berdasarkan hasil pengujian, sistem ini mampu memetakan 2 lingkungan sekitar yang berbeda dengan akurasi mencapai 99.55% dan akurasi lokalisasi dengan pengujian 30 titik kordinat yang berbeda mencapai 99.29 %. Rata-rata deviasi pengukuran kurang dari 5 cm, menunjukkan performa navigasi yang akurat. Robot dapat menentukan jalur dari titik awal ke tujuan secara efisien dan mampu menghindari rintangan tanpa mengalami tabrakan. Selain itu, kecepatan gerak robot yang stabil di kisaran $\pm 0,3$ m/s memungkinkan pergerakan yang aman dan responsif terhadap perubahan posisi objek di sekitarnya. Dengan keunggulan tersebut, sistem navigasi ini terbukti dapat meningkatkan keandalan dan efektivitas robot pengantar makanan, terutama saat beroperasi di lingkungan yang penuh dinamika seperti restoran.

Kata kunci: navigasi robot, *machine learning*, SLAM, RPLiDAR, robot pengantar makanan.

ABSTRACT

The advancement of automation technology is increasingly evident across various sectors, including service industries such as restaurants. One notable innovation that has begun to be widely implemented is the use of food delivery robots. This innovation aims to enhance efficiency and accuracy in service processes. However, deploying robots in dynamic environments like restaurants presents its own set of challenges. Changes in table positions, the movement of customers, and ever-shifting spatial conditions make navigation particularly complex. Conventional navigation systems, which rely heavily on fixed points and lack the ability to respond to real-time changes, often become major limitations.

To address these challenges, an intelligent navigation system based on Machine Learning has been developed, capable of mapping the environment in real-time. This system utilizes RPLiDAR sensors and the SLAM (Simultaneous Localization and Mapping) method to construct accurate and automated environmental maps. Data processing is handled by a Raspberry Pi, functioning as the central control unit. For path planning, the system combines the K-Nearest Neighbors (KNN) algorithm for environmental recognition and Breadth-First Search (BFS) for optimal route finding. Once the route is determined, it is refined using a smoothing technique to ensure the robot moves smoothly and avoids obstacles.

Based on testing results, the system was able to map two different environments with an accuracy of 99.55%, and achieved a localization accuracy of 99.29% through testing at 30 distinct coordinate points. The average measurement deviation was less than 5 cm, indicating precise navigation performance. The robot can determine the optimal path from the starting point to the destination efficiently and is capable of avoiding obstacles without collisions. In addition, the robot maintains a stable movement speed of approximately ± 0.3 m/s, allowing for safe and responsive motion in reaction to changes in the surrounding environment. With these advantages, the navigation system has proven to enhance the reliability and effectiveness of the food delivery robot, particularly when operating in dynamic environments such as restaurants.

Keywords : *robot navigation, machine learning, SLAM, RPLiDAR, food delivery robot.*

KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa atas segala limpahan rahmat dan karunia-Nya, sehingga tim penulis dapat menyelesaikan Capstone Design yang berjudul “Sistem Navigasi Cerdas Robot Pengantar Makanan dengan Pendekatan *Machine Learning*”. Laporan ini disusun sebagai salah satu syarat untuk menyelesaikan studi pada jenjang Sarjana Program Studi S1 Teknik Telekomunikasi, Universitas Telkom.

Dalam proses penyusunan Capstone Design ini, tim penulis mendapatkan banyak arahan dan bimbingan. Oleh karena itu, tim penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada Bapak Ir. Ahmad Tri Hanuranto, M.T. selaku pembimbing pertama, dan Bapak Dr. Ir. Sony Sumaryo, M.T. selaku pembimbing kedua, yang telah meluangkan waktu, tenaga, dan pikiran dalam membimbing kami selama proses penyusunan hingga penyelesaian Capstone Design ini.

Tim penulis menyadari bahwa Capstone Design ini masih jauh dari kesempurnaan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan untuk menjadi bahan evaluasi dan perbaikan di masa mendatang. Besar harapan kami bahwa karya ini dapat memberikan kontribusi positif serta menjadi referensi yang bermanfaat dalam pengembangan ilmu pengetahuan dan teknologi, khususnya di bidang sistem navigasi cerdas.

Bandung, 18 Juni 2025

Tim Penulis

UCAPAN TERIMAKASIH

Dengan segala hormat, penulis menyampaikan rasa terima kasih yang mendalam atas segala bentuk dukungan, bantuan, serta motivasi yang telah diberikan dalam proses penyelesaian tugas akhir berjudul “*Sistem Navigasi Cerdas Robot Pengantar Makanan dengan Pendekatan Machine Learning*”. Penyelesaian tugas akhir ini pun tidak lepas dari kontribusi berbagai pihak. Oleh karena itu, dengan penuh rasa syukur, penulis menyampaikan penghargaan dan ucapan terima kasih sebesar-besarnya kepada:

1. Allah SWT atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan membawa manfaat positif.
2. Kedua orang tua atas segala dukungan melalui doa, dukungan berupa materi, dan motivasinya serta menjadi semangat penulis selama mengerjakan Tugas Akhir.
3. Bapak Ir. Ahmad Tri Hanuranto, S.T., M.T. selaku Dosen Pembimbing 1 dan Bapak Dr. Ir. Sony Sumaryo, S.T., M.T. selaku Dosen Pembimbing 2 atas segala bimbingan dan arahannya selama proses penggeraan Tugas Akhir.
4. Ibu Dr. Ir. Rina Pudji Astuti, S.T., M.T. selaku Dosen Wali kelas TT-45-01, Bapak Dr. Nachwan Mufti A., S.T., M.T. selaku Dosen Wali kelas TT-45-02, dan Bapak Nasrullah Armi, ST, M.Eng., Ph.D. selaku Dosen Wali kelas TT-45-06 yang telah memberikan banyak informasi dan arahannya selama menjalani masa perkuliahan.
5. Bapak Dr. Iman Hedi Santoso, S.T., M.T. selaku Dosen Pengampu Mata Kuliah Tugas Akhir dan Bapak Dhoni Putra Setiawan, S.T., M.T., Ph.D. selaku Dosen Pengampu Mata Kuliah Proposal Tugas Akhir yang telah mendampingi proses penulisan dokumen proposal capstone design.
6. Tim Proyek Telkom University Interactive Food Assistant (T.I.F.A) dan Laboratorium Aeromodelling Payload Telemetry Research Group (APTRG) yang telah mengizinkan penulis menggunakan fasilitas laboratorium untuk membantu penggeraan Tugas Akhir.
7. Sahabat dan teman-teman seperjuangan yang selalu hadir dengan semangat dan dukungan selama penulisan Tugas Akhir.

Penulis turut menyampaikan apresiasi yang sebesar-besarnya kepada seluruh pihak yang telah memberikan dukungan. Semoga Tugas Akhir ini dapat menjadi wujud kontribusi bermanfaat dan mengambil peran kecil dalam mendorong kemajuan ilmu pengetahuan dan teknologi.

DAFTAR ISI

LEMBAR PENGESAHAN	I
LEMBAR PERNYATAAN ORISINALITAS	II
ABSTRAK	V
ABSTRACT	VI
KATA PENGANTAR.....	VII
UCAPAN TERIMA KASIH.....	VIII
DAFTAR ISI.....	IX
DAFTAR GAMBAR.....	XII
DAFTAR TABEL	XV
DAFTAR ISTILAH	XVI
BAB I PENDAHULUAN.....	1
1.1 Deskripsi Umum Masalah.....	1
1.2 Analisa Masalah	2
1.3 Analisa Solusi yang Ada	3
BAB II DESAIN KONSEP SOLUSI	5
2.1 Dasar Teori.....	5
2.1.1 <i>Robot Pengantar Makanan</i>	5
2.1.2 <i>SLAM (Simultaneous Localization and Mapping)</i>	5
2.1.3 <i>Navigasi</i>	5
2.1.4 <i>Machine Learning</i>	5
2.1.5 <i>ROS (Robotic Operating System)</i>	6
2.2 Dasar Penentuan Spesifikasi	6
2.2.1 <i>Sumber Literatur dan Standar Industri</i>	7
2.2.2 <i>Spesifikasi Produk yang Ada</i>	7
2.2.3 <i>Wawancara dengan Pengguna atau Ahli</i>	8
2.2.4 <i>Penggunaan Teknologi Machine Learning</i>	8
2.2.5 <i>Pengembangan di Masa Depan</i>	8
2.3 Batasan dan Spesifikasi.....	9

2.3.1 <i>Tabel Spesifikasi Produk</i>	9
2.3.2 <i>Pergerakan Robot ke Titik Tujuan yang Telah ditentukan</i>	9
2.3.3 <i>Robot dapat Bekerja pada Lingkungan Dinamis</i>	10
2.3.4 <i>Robot mampu menentukan rute tercepat ketujuannya</i>	10
2.4 Pengukuran/verifikasi spesifikasi.....	11
2.4.1 <i>Pergerakan Robot</i>	11
2.4.2 <i>Navigasi Robot</i>	12
BAB III DESAIN RANCANGAN SOLUSI.....	13
3.1 Alternatif Usulan Solusi.....	13
3.1.1 <i>Alternatif Solusi 1: Light Detection and Ranging (LiDAR)</i>	13
3.1.2 <i>Alternatif Solusi 2: Sensor Ultrasonik</i>	15
3.1.3 <i>Alternatif Solusi 3: RPLiDAR</i>	17
3.2 Analisis dan Pemilihan Solusi.....	19
3.2.1 <i>Kriteria Pemilihan Solusi</i>	19
3.2.2 <i>Matriks Keputusan Solusi</i>	21
3.3 Desain Solusi Terpilih.....	21
3.3.1 <i>Blok Diagram Sistem</i>	22
3.3.2 <i>Flowchart Sistem Mapping</i>	23
3.3.3 <i>Desain Perangkat Keras Sistem Navigasi</i>	24
3.4 Jadwal dan Anggaran	24
3.4.1 <i>Jadwal Perancangan</i>	25
3.4.2 <i>Rancangan Anggaran Biaya</i>	25
BAB IV IMPLEMENTASI	26
4.1 Deskripsi Umum Implementasi.....	26
4.2 Detil Implementasi	27
4.2.1 <i>Perangkat Keras</i>	27
4.2.1.1 <i>RPLIDAR A2</i>	27
4.2.1.2 <i>Raspberry Pi 4 Model B</i>	28
4.2.2 <i>Perangkat Lunak</i>	30
4.2.2.1 <i>DBScan</i>	30
4.2.2.2 <i>KNN</i>	31
4.2.2.3 <i>Pengambilan Koordinat Path</i>	32

4.2.2.4	Rute Navigasi	33
4.3	Prosedur Pengoperasian	35
BAB V PENGUJIAN	39
5.1	Skenario Umum Pengujian	39
5.2	Detil Pengujian.....	40
5.2.1	<i>Pengukuran jarak dengan RPLidar</i>	40
5.2.2	<i>Hasil Pengujian Mapping</i>	68
5.2.3	<i>Hasil Pengujian Lokalisasi</i>	72
5.2.4	<i>Hasil Pengujian Navigasi</i>	74
5.3	Analisa Hasil Pengujian	83
5.3.1	<i>Analisa Hasil Pengukuran RPLidar</i>	83
5.3.2	<i>Analisa Hasil Pengujian Mapping</i>	83
5.3.3	<i>Analisa Hasil Pengujian Lokalisasi</i>	84
5.3.4	<i>Analisa Hasil Pengujian Navigasi</i>	84
BAB VI KESIMPULAN	86
6.1	Kesimpulan	86
6.2	Saran.....	87
DAFTAR PUSTAKA	89
LAMPIRAN	92

DAFTAR GAMBAR

Gambar 1. 1 Path Planning.....	3
Gambar 1. 2 SLAM.....	4
Gambar 3. 1 LiDAR.....	13
Gambar 3. 2 Sensor Ultrasonic	15
Gambar 3. 3 RPLiDAR	17
Gambar 3. 4 Blok Diagram Sistem	22
Gambar 3. 5 Flowchart Sistem Mapping	23
Gambar 3. 8 Desain Perangkat Keras	24
Gambar 4. 1 Pkode RPLiDAR A2	28
Gambar 4. 2 Raspberry Pi 4 Model B	28
Gambar 4. 3 Implementasi Visual.....	29
Gambar 4. 4 Flowchart Proses Area Navigasi	32
Gambar 4. 5 Flowchart Pembuatan Jalur	34
Gambar 4. 6 File yaml & pgm	35
Gambar 4. 7 "Map_Read.P.y"	35
Gambar 4. 8 "KNN_Path.py"	36
Gambar 4. 9 KNN Path Correction	36
Gambar 4. 10 "knn_corrected_path_smoothed.csv".....	37
Gambar 4. 11 DBSCAN Path	37
Gambar 4. 12 Path with Cluster Priority	38
Gambar 5. 1 Pengukuran Jarak	40
Gambar 5. 2 Hasil pengukuran dengan RPLidar (0° ,30 cm)	41
Gambar 5. 3 Hasil pengukuran dengan alat pengukur manual (0° ,30 cm)	41
Gambar 5. 4 Hasil pengukuran dengan RPLidar (0° ,60 cm)	42
Gambar 5. 5 Hasil pengukuran dengan alat pengukur manual (0° ,60 cm)	42
Gambar 5. 6 Hasil pengukuran dengan RPLidar (45° ,30 cm)	43
Gambar 5. 7 Hasil pengukuran manual (45° ,30 cm).....	43
Gambar 5. 8 Hasil pengukuran (90° ,30 cm)	44
Gambar 5. 9 Hasil pengukuran (90° ,60 cm)	44
Gambar 5. 10 Hasil pengukuran (135° ,30 cm)	45
Gambar 5. 11 Hasil pengukuran (135° ,60 cm)	45
Gambar 5. 12 Hasil pengukuran (180° ,30 cm)	46

Gambar 5. 13 Hasil pengukuran (225° ,30 cm)	46
Gambar 5. 14 Hasil pengukuran (270° ,30 cm)	47
Gambar 5. 15 Hasil pengukuran (225° ,60 cm)	47
Gambar 5. 16 Hasil pengukuran (270° ,60 cm)	48
Gambar 5. 17 Hasil pengukuran (315° ,30 cm)	48
Gambar 5. 18 Hasil pengukuran (315° ,60 cm)	49
Gambar 5. 19 Hasil pengukuran jarak 0°	55
Gambar 5. 20 Hasil Pengukuran Jarak 45°	56
Gambar 5. 21 Hasil Pengukuran Jarak 90°	57
Gambar 5. 22 Hasil Pengukuran Jarak 135°	58
Gambar 5. 23 Hasil Pengukuran Jarak 180°	59
Gambar 5. 24 Hasil Pengukuran Jarak 225°	60
Gambar 5. 25 Hasil Pengukuran Jarak 270°	61
Gambar 5. 26 Hasil Pengukuran Jarak 315°	62
Gambar 5. 27 Visualisasi perbandingan pengukuran.....	67
Gambar 5. 28 Grafik error Pengukuran.....	67
Gambar 5. 29 Hasil pengujian <i>mapping</i>	68
Gambar 5. 30 pengujian <i>mapping</i> 1	69
Gambar 5. 31 pengujian <i>mapping</i> 2	69
Gambar 5. 32 pengujian <i>mapping</i> 3	70
Gambar 5. 33 pengujian <i>mapping</i> 4	70
Gambar 5. 34 Pengujian <i>mapping</i> 5.....	70
Gambar 5. 35 Simulasi Pengujian Lokalisasi	72
Gambar 5. 36 Simulasi Navigasi.....	78
Gambar 5. 37 Skenario 2 pengujian simulasi 2	79
Gambar 5. 38 Jarak grafik skenario 2	79
Gambar 5. 39 Skenario 3 pengujian simulasi 2	79
Gambar 5. 40 Jarak grafik skenario 3	80
Gambar 5. 41 Skenario 3, Rute penuh	80
Gambar 5. 42 Jarak grafik rute penuh	80
Gambar 5. 43 Skenario 3, 2 obstacle	81
Gambar 5. 44 Jarak grafik rute obstacle.....	81
Gambar 5. 45 Skenario 1, 4 Rute penuh	81
Gambar 5. 46 Skenario 1 Jarak rute penuh	82

Gambar 5. 47 Skenario 1, 4 obstacle	82
Gambar 5. 48 Skenario 1 Jarak rute obstacle.....	82

DAFTAR TABEL

Tabel 3. 1 Matriks Penilaian	21
Tabel 3. 2 Jadwal Perancangan Biaya.....	25
Tabel 3. 3 Rancangan Anggaran Biaya.....	25
Tabel 4. 1 Spesifikasi RPLIDAR A2M8	27
Tabel 4. 2 Spesifikasi Raspberry Pi 4 Model B	29
Tabel 5. 1 Hasil Pengukuran Sudut dan Jarak	50
Tabel 5. 2 Hasil pengukuran Sudut dan Derajat (180-315°).....	51
Tabel 5. 3 Hasil pengukuran Selisih	52
Tabel 5. 4 Hasil pengukuran Selisih (180-315°).....	52
Tabel 5. 5 Hasil Pengukuran Persentase Error (%).....	53
Tabel 5. 6 Hasil Pengukuran Persentase Error (%), (180-315°)	54
Tabel 5. 7 Hasil Pengukuran Akurasi (%)	54
Tabel 5. 8 Hasil Pengukuran Akurasi (%), (180-315°).....	55
Tabel 5. 9 Hasil Pngukuran Sudut dan Jarak	65
Tabel 5. 10 Hasil Pengukuran Selisih	66
Tabel 5. 11 Hasil Pengukuran Persentase Error.....	67
Tabel 5. 12 Hasil Pengukuran Akurasi	68
Tabel 5. 13 Hasil Pengujian Mapping.....	73
Tabel 5. 14 Hasil Pengujian Lokalisasi.....	76
Tabel 5. 15 Hasil Pengujian Skenario 1	77
Tabel 5. 16 Hasil Pengujian Skenario 2	78
Tabel 5. 17 Hasil Pengujian Skenario 3	79

DAFTAR ISTILAH

Algoritma	Langkah-langkah logis dan sistematis yang dirancang untuk menyelesaikan suatu masalah atau tujuan dalam proses komputasi.
BFS	Breadth-First Search, menemukan jalur terpendek dalam graf
DBSCAN	Density-Based Spatial Clustering of Applications with Noise, Algoritma unsupervised learning yang digunakan untuk mengelompokkan data (<i>clustering</i>) berdasarkan kepadatan titik data.
KNN	K-Nearest Neighbors, Menentukan kelas berdasarkan mayoritas tetangga
Koordinat	Posisi suatu titik dalam ruang, biasanya dalam format (x, y) atau (x, y, z), digunakan untuk menunjukkan lokasi pada peta atau lingkungan.
Triangulation/ Triangulasi	metode untuk menentukan posisi suatu objek atau titik dengan menggunakan pengukuran sudut dan jarak dari dua atau lebih titik referensi yang sudah diketahui posisinya.
Lokalisasi	Proses menentukan posisi titik saat ini dalam suatu peta atau lingkungan berdasarkan data yang ada.
Machine Learning	Cabang kecerdasan buatan yang membuat komputer belajar dari data untuk membuat prediksi atau keputusan.
Mapping	Proses membuat peta dari lingkungan sekitar berdasarkan data yang dikumpulkan.
Navigasi	Navigasi dijalankan dengan algoritma KNN dan BFS
Odometri	Teknik menghitung posisi berdasarkan pergerakan dari waktu ke waktu.
Path Planning	Proses menentukan rute atau jalur terbaik dari posisi awal ke tujuan, sambil menghindari rintangan.
Sensor	Alat yang mendeteksi dan merespon perubahan di lingkungan fisik
Python	Bahasa Pemrograman
SLAM	Simultaneous Localization and Mapping , Teknologi untuk

	pemetaan dan lokalisasi simultan.
YAML	YAML Ain't Markup Language, Format file konfigurasi hasil pemetaan.
PGM	Format gambar hasil visualisasi pemetaan.
ROS	Robot Operating System, perangkat lunak untuk pengembangan aplikasi robotika
IMU	Inertial Measurement Unit, sensor gabungan yang mengukur percepatan, kecepatan sudut, dan kadang orientasi.
ISO	International Organization for Standardization, adalah organisasi internasional yang mengembangkan dan menerbitkan standar di berbagai bidang, termasuk teknologi, manufaktur, keselamatan, dan sistem manajemen.

BAB I

PENDAHULUAN

1.1 Deskripsi Umum Masalah

Perkembangan teknologi yang semakin maju setiap harinya, sistem otomatis menjadi penggunaan untuk meningkatkan efisiensi berbagai industri seperti pelayanan restoran. Implementasi sistem otomatis pada pelayanan restoran seperti robot pengantar makanan merupakan ide inovatif yang diharapkan mampu membantu manusia dalam tugas-tugas rutin seperti mengantarkan pesanan. Menurut data yang ada, penggunaan robot pengantar makanan mampu mengurangi biaya tenaga kerja sebesar 30-50% dan kesalahan pengiriman menurun sebesar 90%[1]. Selain itu, robot pengantar makanan juga mampu meningkatkan kepuasan pelanggan sampai 70% [2]. Hal ini menunjukkan bahwa sistem otomatis memiliki potensi besar dalam meningkatkan efisiensi dan kualitas layanan.

Pada tahun lalu, Universitas Telkom telah membuat sebuah robot pengantar makanan yang mampu bergerak dengan baik. Tapi robot ini masih dikendalikan secara manual oleh penggunanya untuk mencapai tujuan [3]. Hal ini tidak memenuhi harapan yang diinginkan untuk mendapatkan efisiensi yang sesungguhnya dari robot pengantar makanan. Robot ini membutuhkan pengembangan lanjutan untuk membuat sistem navigasi yang baik dan mampu dijalankan secara otonom. Pada sistem ini, komunikasi data menjadi inti dari seluruh proses operasional. Pertukaran informasi antara robot, sensor, dan infrastruktur jaringan menjadi komponen utama untuk memastikan sistem dapat bekerja dengan sangat efisien dan *real-time*. Robot pengantar makanan yang dilengkapi dengan berbagai sensor seperti kamera, LiDAR, dan ultrasonik untuk memetakan lingkungan sekitarnya. Data yang dihasilkan dari sensor-sensor ini kemudian dikirimkan melalui jaringan untuk diproses oleh algoritma *Machine Learning* yang ada di server pusat.

Pada akhirnya, tantangan utama yang dihadapi oleh robot pengantar makanan adalah bagaimana mengintegrasikan komunikasi data yang cepat dan andal dengan sistem kecerdasan buatan yang mampu memproses informasi lingkungan secara *real-time*. Masalah inilah yang menjadi dasar dari proyek ini, yang bertujuan untuk mengembangkan sistem navigasi cerdas berbasis *Machine Learning* yang didukung oleh infrastruktur

telekomunikasi berlatensi rendah dan mempunyai *bandwidth* tinggi, sehingga dapat meningkatkan kinerja robot dalam pengantaran makanan di lingkungan dinamis.

1.2 Analisa Masalah

Dalam sistem telekomunikasi yang mendukung komunikasi data antara robot dan infrastruktur jaringan, terdapat beberapa permasalahan yang perlu dianalisis berdasarkan berbagai aspek. Fokus utama dari analisis ini adalah aspek teknis, karena terkait dengan keilmuan Teknik Telekomunikasi, namun aspek lain seperti sosial juga memiliki relevansi yang penting. Komunikasi data yang cepat dan efisien sangat bergantung pada kualitas jaringan telekomunikasi. Dalam konteks ini, tiga permasalahan utama muncul:

1. Aspek Teknik

- **Navigasi di Lingkungan Statis**

Dalam pengembangan sistem sebelumnya, robot pengantar makanan hanya dirancang untuk bekerja di lingkungan yang bersifat statis, di mana tata letak dan posisi objek tidak mengalami perubahan signifikan. Hal ini menyebabkan robot tidak memiliki kemampuan adaptif terhadap perubahan lingkungan, seperti adanya perubahan posisi meja dan kursi, atau keberadaan orang yang tidak terdeteksi. Akibatnya, navigasi menjadi tidak efektif dan dapat menimbulkan kesalahan dalam pengantaran.

2. Aspek Sosial

- **Kepuasan Pelanggan**

Keterbatasan robot dalam menyesuaikan diri dengan lingkungan yang dinamis berdampak langsung terhadap tingkat kepuasan pelanggan. Robot yang tidak mampu mengenali atau menghindari orang yang bergerak di sekitarnya, atau yang mengalami kesulitan menavigasi saat tata letak ruangan berubah, akan menimbulkan gangguan dalam pelayanan. Hal ini bisa menimbulkan keterlambatan, kesalahan pengiriman, atau bahkan menabrak pelanggan, yang pada akhirnya menurunkan kualitas layanan otomatis yang disediakan.

3. Aspek Interaksi dengan Lingkungan

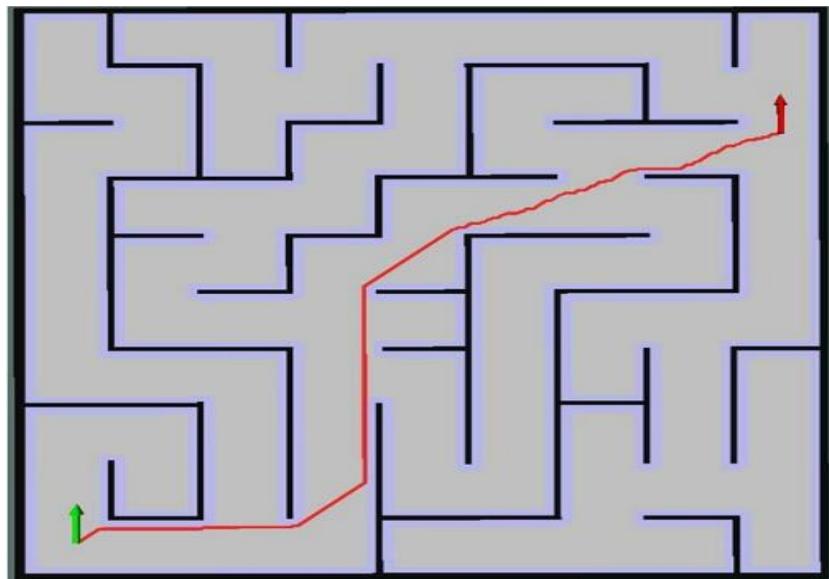
- **Visualisasi**

Tidak adanya sistem visualisasi yang baik pada robot, seperti kamera atau sistem pengolahan citra, menyebabkan robot kesulitan dalam mengenali dan

memahami kondisi aktual di sekitarnya. Tanpa kemampuan visualisasi ini, robot tidak dapat mendeteksi secara akurat perubahan tata letak, keberadaan pelanggan yang berdiri di jalurnya, atau objek tak terduga yang muncul di lingkungan. Hal ini menyebabkan keterbatasan dalam pengambilan keputusan secara real-time dan dapat mengganggu kelancaran navigasi maupun proses pengantaran makanan

1.3 Analisa Solusi yang Ada

Cara kerja navigasi robot pengantar makanan saat ini umumnya menggunakan gabungan dari beberapa teknologi untuk menentukan jalur, melakukan navigasi adaptif, dan mencapai tujuan. Sistem ini biasanya menggabungkan odometri dan sensor LiDAR untuk menentukan lokasi robot secara *real-time* dan merencanakan rute optimal ke titik tujuan. Robot juga dilengkapi dengan berbagai sensor untuk menghindari rintangan di sepanjang jalan secara otomatis, yang merupakan bagian dari sistem navigasi adaptif.



Gambar 1. 1 Path Planning

Sumber : <https://www.researchgate.net> (2022)

Namun, meskipun ini termasuk Solusi yang efektif, sistem masih mempunyai beberapa kekurangan. Salah satu kekurangan yang utama adalah ketergantungan pada peta statis yang dibuat di awal, sehingga ketika ada perubahan di lingkungan (misalnya kursi atau meja dipindahkan), robot pengantar makanan sering kali mengalami kesulitan untuk beradaptasi. Selain itu, tantangan dalam navigasi pada permukaan lantai yang tidak rata atau slip roda juga masih menjadi kendala dalam akurasi navigasi jangka panjang.

Untuk mengatasi kekurangan ini, pengembangan yang lebih lanjut sangat penting karena dapat difokuskan pada penggunaan teknologi *machine learning*, yang memungkinkan robot belajar dari lingkungannya dengan lebih dinamis[14]. Salah satu hal yang akan dikembangkan ke depan, yaitu kombinasi antara SLAM "Simultaneous Localization and Mapping", *machine learning*, dan sensor canggih dapat meningkatkan kemampuan adaptasi robot, sehingga robot bisa lebih mandiri dalam menemukan jalur dan menavigasi di lingkungan yang berubah-ubah, sekaligus mengurangi ketergantungan pada input manual atau pemeliharaan sistem.



Gambar 1. 2 SLAM

Sumber : https://msadowski.github.io/hands-on-with-slam_toolbox (2019)

Gambar di atas adalah hasil pemetaan yang dilakukan menggunakan sensor RPLiDAR, dan bisa divisualisasikan melalui software seperti RViz atau diekstrak menjadi gambar. Hasil pemetaan berbasis SLAM ini biasanya disimpan dalam format file .pgm, yaitu semacam gambar digital yang berisi kumpulan titik-titik koordinat yang membentuk peta lingkungan sekitar. Titik-titik inilah yang mewakili area yang telah dipindai—seperti tembok, rintangan, atau ruang kosong—and nantinya digunakan sebagai dasar bagi robot untuk melakukan navigasi. Jadi, data ini bukan hanya sekadar tampilan visual, tapi menjadi fondasi yang penting bagi robot untuk bisa menentukan arah, menghindari halangan, dan bergerak dengan aman dari satu titik ke titik lainnya.

BAB II

DESAIN KONSEP SOLUSI

2.1 Dasar Teori

2.1.1 Robot Pengantar Makanan

Robot pengantar makanan adalah salah satu jenis robot yang dirancang untuk mengantar makanan atau pesanan menuju pelanggan [4]. Robot ini membutuhkan berbagai bantuan teknologi seperti sensor dan mikrokontroler yang saling berkerja satu sama lain untuk menciptakan navigasi robot yang baik sehingga robot mampu beropersi secara otonom. Tujuan dari robot ini adalah untuk meningkatkan efisiensi tenaga dalam pengriman makanan, mengurangi biaya tenaga kerja, dan meningkatkan kepuasan pelanggan [5].

2.1.2 SLAM (Simultaneous Localization and Mapping)

SLAM adalah sebuah metode yang digunakan untuk membuat peta lingkungan sekitar dan menentukan posisinya berdasarkan peta yang telah dibuat secara bersamaan dengan menggunakan data sensor yang mendukung [7]. SLAM merupakan metode yang sering digunakan untuk mendukung navigasi robot sehingga robot dapat beroperasi secara otonom. SLAM memiliki berbagai algoritma untuk beroperasi seperti GMapping dan Hector SLAM. Algoritma ini memungkinkan robot untuk berjalan secara otonom di lingkungan yang kompleks dan dinamis [7].

2.1.3 Navigasi

Navigasi adalah proses yang memungkinkan robot untuk secara otonom menentukan jalur dari titik awal menuju tujuan yang diinginkan sambil menghindari rintangan di sepanjang perjalannya [7]. Sistem navigasi robot modern terdiri dari tiga komponen utama, yaitu lokalisasi, perencanaan lintasan (*path planning*), dan kontrol gerakan. Lokalisasi merujuk pada kemampuan robot untuk mengetahui posisi dan orientasinya di dalam peta lingkungan, yang sering kali dihasilkan melalui metode SLAM (Simultaneous Localization and Mapping).

2.1.4 Machine Learning

Machine Learning (ML) merupakan cabang dari kecerdasan buatan (AI) yang memungkinkan sistem untuk belajar dari data secara otomatis [15]. Dengan sistem ML

bekerja dengan membangun model matematis berdasarkan pola dalam data, yang kemudian dapat digunakan untuk melakukan prediksi, klasifikasi, atau pengambilan keputusan secara otomatis [15].

2.1.5 ROS (Robotic Operating System)

Robot Operating System (ROS) bukanlah sistem operasi dalam pengertian tradisional, melainkan sebuah kerangka kerja perangkat lunak berbasis *middleware* yang dirancang untuk mendukung pengembangan aplikasi robotika secara modular dan terdistribusi [11]. ROS menyediakan serangkaian alat bantu (*tools*), pustaka (*libraries*), dan konvensi yang memungkinkan para pengembang untuk merancang dan mengelola sistem perangkat lunak robot secara efisien. Beberapa fitur utama ROS antara lain:

- Komunikasi antarproses (inter-process communication) melalui *topics*, *services*, dan *actions*, yang memungkinkan pertukaran data secara real-time antara berbagai komponen.
- Manajemen paket (package management) yang terstruktur, mendukung pemisahan fungsi, dan memungkinkan kolaborasi pengembangan.
- Dukungan untuk simulasi dan visualisasi, seperti melalui Gazebo untuk simulasi fisik dan Rviz untuk visualisasi sensor dan peta.
- Kompatibilitas dengan berbagai bahasa pemrograman, terutama Python dan C++, menjadikan ROS fleksibel untuk berbagai kebutuhan pengembangan.

Selain itu, ROS juga mendukung berbagai perangkat keras seperti sensor, aktuator, dan pengendali motor melalui driver yang telah dikembangkan oleh komunitas. Sistem ini sangat cocok digunakan dalam pengembangan robot otonom, sistem pemetaan dan navigasi (SLAM), serta integrasi kecerdasan buatan untuk pengambilan keputusan [11].

Dengan menggunakan ROS, pengembang tidak perlu membangun seluruh sistem dari awal, melainkan cukup mengintegrasikan berbagai node yang sudah ada atau dikembangkan sendiri ke dalam arsitektur ROS yang modular.

2.2 Dasar Penentuan Spesifikasi

Dalam penentuan spesifikasi dasar mencakup beberapa aspek penting. Pertama, robot harus mampu mendeteksi dan memetakan lingkungan sekitarnya secara *real-time*

menggunakan sensor seperti "RPLiDAR" (RoboPeak Light Detection and Ranging) [8]. Selanjutnya, algoritma *machine learning* akan diterapkan untuk menganalisis data yang diperoleh, memungkinkan robot untuk belajar dari interaksi sebelumnya dan meningkatkan efisiensi navigasinya.

2.2.1 Sumber Literatur dan Standar Industri

Ketika mengembangkan sistem navigasi untuk robot, aturan dan standar yang sudah ada di industri robotika. Misalnya, *International Organization of Standardization* (ISO) tentang aturan keselamatan penggunaan robot di tempat umum seperti adanya sistem kendali jarak jauh pada robot yang memungkinkan teknisi dan admin dapat mengetahui kapan harus melakukan *check* dan *maintenance* pada robot.

2.2.2 Spesifikasi Produk yang Ada

Saat ini, berbagai robot navigasi otonom telah beredar di pasaran dan banyak dimanfaatkan di sektor layanan seperti restoran. Salah satu fungsinya adalah untuk mengantar makanan secara otomatis. Umumnya, robot-robot ini dilengkapi dengan sensor canggih seperti LiDAR untuk mendeteksi lingkungan sekitar, serta sistem komputasi dan algoritma navigasi untuk menentukan jalur terbaik secara mandiri.

Salah satu produk yang sudah digunakan secara luas adalah PuduBot dari Pudu Robotics. Robot ini dirancang khusus untuk pengantaran makanan dan mampu beroperasi secara otonom di lingkungan yang kompleks [12]. PuduBot dilengkapi dengan sensor LiDAR 360°, sensor ultrasonik, kamera visual, dan IMU yang membantu menjaga arah dan stabilitas saat bergerak. Navigasi dilakukan dengan bantuan algoritma SLAM, yang memungkinkan robot untuk memetakan ruangan dan mengetahui posisinya sendiri secara real-time [12]. Selain itu, sistemnya juga mampu menghindari rintangan dan beradaptasi terhadap kondisi lingkungan.

Meskipun PuduBot memiliki performa yang tinggi dan fitur yang lengkap, produk ini masih memiliki beberapa keterbatasan. Di antaranya adalah biaya perangkat keras yang cukup tinggi, sistem yang tertutup sehingga sulit dimodifikasi, dan kurangnya fleksibilitas untuk pengembangan dalam konteks penelitian atau proyek berskala kecil.

Melihat hal tersebut, sistem yang dikembangkan dalam tugas akhir ini bertujuan untuk menghadirkan robot pengantar makanan yang lebih dinamis dan terjangkau dari segi harga. Sistem ini dibuat menggunakan RPLiDAR A2, Raspberry Pi 4, serta algoritma

navigasi KNN dan BFS. Dengan pendekatan ini, sistem dapat disesuaikan lebih mudah sesuai kebutuhan pengguna.

2.2.3 Wawancara dengan Pengguna atau Ahli

Kami belajar dari pengalaman penggunaan robot di sektor pengantaran makanan. Pengalaman ini memberikan acuan tentang masalah sehari-hari yang sering dihadapi robot, seperti kesulitan beradaptasi, terutama jika ada perubahan di lingkungan, seperti ketika meja atau kursi dipindahkan.

2.2.4 Penggunaan Teknologi *Machine Learning*

Algoritma *machine learning* diterapkan sebagai bagian dari sistem navigasi robot, khususnya melalui penggunaan K-Nearest Neighbors (KNN). KNN merupakan salah satu algoritma *supervised learning* yang digunakan untuk mengklasifikasikan data berdasarkan kedekatannya dengan data yang sudah ada sebelumnya [16].

Pada sistem robot pengantar makanan ini, KNN dimanfaatkan untuk mengklasifikasikan jenis area atau kondisi lingkungan berdasarkan data hasil pemetaan menggunakan sensor RPLiDAR. Contohnya, data hasil pemindaian area dapat diklasifikasikan menjadi area bebas, area dengan rintangan, atau jalur yang bisa dilalui[16]. Hasil klasifikasi ini kemudian digunakan sebagai input untuk proses perencanaan jalur dengan algoritma Breadth First Search (BFS), yang mencari jalur optimal menuju tujuan.Dengan *Machine Learning* robot bisa menyesuaikan rute dan responnya setiap kali menemukan tantangan baru, tanpa harus bergantung sepenuhnya pada peta yang sudah ada [17].

2.2.5 Pengembangan di Masa Depan

Menggabungkan teknologi seperti “SLAM” (Simultaneous Localization and Mapping), *machine learning*, dan sensor canggih akan membuat robot lebih mampu menyesuaikan diri dengan lingkungan yang berubah.

Seperti mengoptimalkan penggunaan sensor pada robot untuk navigasi yang lebih handal dengan menambahkan beberapa sensor penunjang seperti sensor IR (*infrared*) dan sensor ultrasonik

2.3 Batasan dan Spesifikasi

Batasan yang ada pada perancangan ini adalah :

1. Robot tidak dapat mendeteksi benda transparan.
2. Robot hanya dapat bergerak 0.3 m/s

2.3.1 Tabel Spesifikasi Produk

Tabel 2.1 Spesifikasi Produk

No	Hal	Rincian
1	Sistem Navigasi dapat bergerak ke titik tujuan yang telah ditentukan dalam ruangan bidang datar.	Sistem Navigasi bergerak ke titik tujuan yang telah ditentukan dengan kecepatan $\pm 0,3\text{m/s}$.
2	Sistem Navigasi mampu bekerja pada lingkungan dinamis yang mana objek disekitarnya dapat berubah posisi.	Dengan adanya sensor jarak dan <i>positioning</i> , robot dapat menjaga jarak dan menentukan posisinya secara <i>real-time</i> , dengan pembaruan data setiap 0,073934 detik.
3	Sistem Navigasi mampu menemukan rute tercepat ketujuannya.	Dengan menggunakan metode <i>path planning</i> Sistem Navigasi mampu menemukan rute tercepat ketujuannya.

Waktu pembacaan RPLIDAR dalam satu rotasi yaitu sebesar 0,073934 detik memiliki pengaruh penting terhadap kemampuan sistem navigasi dalam merespons lingkungan sekitar. Dengan kecepatan pergerakan robot sekitar $\pm 0,3 \text{ m/s}$, maka dalam satu rotasi RPLIDAR, robot dapat menempuh jarak sekitar 0,02218 meter (2,2 cm). Artinya, sistem navigasi mendapatkan pembaruan data lingkungan setiap 2,2 cm pergerakan robot, yang cukup rapat untuk mendeteksi perubahan di lingkungan secara akurat dan *real-time*[8].

2.3.2 Pergerakan Robot ke Titik Tujuan yang Telah ditentukan

Robot dapat menuju ke meja atau titik tujuan yang telah ditentukan berdasarkan pesanan pelanggan. Saat pelanggan memesan makanan atau minuman, mereka juga menentukan lokasi meja tempat mereka akan duduk. Setelah pesanan siap, robot akan secara otomatis bergerak menuju meja pelanggan yang telah ditentukan.

Odometri menggunakan *positioning* menjadi metode navigasi yang dipilih. Dengan sensor jarak, mendukung *obstacle avoidance* untuk menghindari halangan di jalur robot serta *positioning* untuk menentukan koordinat X dan Y dari posisi robot dan meja pelanggan. Saat robot bergerak menuju titik tujuan, koordinat X dan Y yang diperoleh.

Positioning System menjadi faktor utama dalam navigasi. Koordinat ini memungkinkan robot untuk mengetahui posisi aktualnya dan lokasi tujuan dengan akurat. Selain itu, kecepatan linear robot menjadi parameter penting yang harus diperhatikan. Jika kecepatan terlalu tinggi, robot akan bergerak terlalu cepat, sehingga makanan atau minuman yang dibawa berisiko tumpah atau berantakan. Rata-rata kecepatan robot pengantar makanan biasanya berada di kisaran 0,3–0,5 m/s, sementara kecepatan robot ini dirancang sekitar $\pm 0,3$ m/s untuk memastikan stabilitas selama perjalanan.

2.3.3 Robot dapat Bekerja pada Lingkungan Dinamis

Robot ini dirancang untuk bekerja di lingkungan dinamis, di mana posisi objek seperti meja atau kursi dapat berubah sewaktu-waktu akibat interaksi manusia, atau keberadaan orang yang bergerak di sekitarnya. Untuk mengatasi kondisi tersebut, robot ini dilengkapi dengan sensor *ranging*, sebuah sensor yang mampu melakukan pemindai jarak secara 360 derajat. Sensor ini digunakan untuk *obstacle avoidance* dan juga mendukung sistem penentuan posisi (*positioning system*) yang memungkinkan robot mengetahui posisi dirinya sendiri serta lokasi tujuan yang telah ditentukan sebelumnya.

Cara kerja sensor *ranging* adalah dengan memancarkan laser dan mengukur pantulanannya menggunakan sensor untuk menghitung jarak objek yang terkena laser. Dengan informasi ini, robot dapat mendeteksi keberadaan rintangan di sekitarnya. Sensor *ranging* memungkinkan robot menghindari tabrakan, baik dengan objek maupun orang yang mungkin melintas di jalur pergerakannya. Jika ada rintangan di jalur yang dilalui, robot dapat secara otomatis menentukan rute alternatif untuk mencapai tujuannya dengan aman dan efisien. Penggunaan sensor *ranging* ini memberikan kemampuan adaptif yang sangat penting, terutama untuk operasi di lingkungan yang kompleks dan dinamis.

2.3.4 Robot mampu menentukan rute tercepat ketujuannya

Menentukan rute tercepat untuk mencapai tujuan merupakan salah satu faktor utama agar robot dapat sampai ke tujuannya dengan efisien. Dengan memilih rute terpendek, robot tidak hanya menghemat waktu, tetapi juga mengurangi konsumsi daya

baterai, sehingga memungkinkan robot bekerja lebih lama. Untuk mencapai hal ini, robot menggunakan metode *path planning* dengan bantuan *positioning system*.

Cara kerjanya adalah robot terlebih dahulu melakukan proses *mapping* menggunakan positioning system untuk menghasilkan peta ruangan beserta koordinat X dan Y dari area tersebut. Setelah peta ruangan dan koordinat diperoleh, metode *path planning* dapat diterapkan. Robot kemudian menentukan koordinat X dan Y tujuan, lalu bergerak menuju tujuan dengan mengikuti rute terpendek berdasarkan peta yang telah dibuat. Metode ini memungkinkan robot untuk bekerja lebih efisien, menghindari rintangan, dan mencapai tujuannya dengan cepat serta hemat energi.

2.4 Pengukuran/verifikasi spesifikasi

Robot yang sudah dirancang dengan produk untuk mendukung sistem navigasi cerdas robot harus menjalani beberapa tahap pengukuran terhadap kinerja produk tersebut. Hal ini dilakukan sesuai dengan pengukuran yang telah ditetapkan oleh produk tersebut, sehingga hasil yang diperoleh merupakan hasil yang akurat.

2.4.1 Pergerakan Robot

Proses verifikasi pergerakan robot dilakukan untuk memastikan robot dapat bergerak dengan baik tanpa ada kendala menggunakan alat ukur jarak. Selain itu, proses ini juga untuk melihat pergerakan robot sudah sesuai dengan program yang sudah diberikan.

Tabel 2.2 Verifikasi Pergerakan Robot

Hal	Rincian
Metode Pengujian	Membandingkan arah dan jarak pergerakan robot pada dunia nyata dengan program yang sudah dibuat
Program	Program dibuat menggunakan bahasa <i>python</i> dengan perintah untuk maju dan mundur sejauh 2 meter.
Prosedur Pengujian	Menjalankan robot menggunakan program <i>python</i> yang membuat robot bergerak maju dan mundur sejauh 2 meter. Setelah itu, menggunakan penggaris untuk memastikan arah dan jarak pergerakan robot sudah sesuai.

2.4.2 Navigasi Robot

Proses verifikasi navigasi robot dilakukan untuk memastikan robot mampu bernaligasi secara otomatis menuju titik tujuan yang telah ditentukan tanpa hambatan. Selain itu, proses ini bertujuan untuk mengevaluasi apakah navigasi robot sudah sesuai dengan algoritma dan jalur yang dirancang pada program.

Tabel 2.3 Verifikasi Navigasi Pergerakan Robot

Hal	Rincian
Metode Pengujian	Membandingkan jalur yang diambil robot pada dunia nyata dengan jalur yang dihasilkan oleh algoritma navigasi yang dibuat menggunakan ROS (<i>Robot Operating System</i>).
Program	Program dibuat menggunakan ROS untuk perencanaan jalur serta algoritma A* untuk menentukan rute optimal. Selain itu, pemetaan dilakukan menggunakan SLAM yang menghasilkan peta lingkungan untuk membantu navigasi.
Prosedur Pengujian	Robot dijalankan menggunakan algoritma navigasi dan pergerakannya diamati untuk memastikan robot mampu mencapai tujuan tanpa menabrak objek dari jalur optimal. Setelah itu, mengukur dan membandingkan kesesuaian posisi akhir robot dengan titik tujuan pada program dan dunia nyata.

menjadi dasar penting dalam memastikan bahwa robot pengantar makanan siap digunakan di lingkungan yang nyata.

BAB III

DESAIN RANCANGAN SOLUSI

3.1 Alternatif Usulan Solusi

Beberapa metode yang disusun untuk mengatasi masalah yang ada pada Batasan-batasan yang ada, terbentuklah beberapa alternatif solusi yang tersedia.

3.1.1 Alternatif Solusi 1: Light Detection and Ranging (LiDAR)



Gambar 3. 1 LiDAR

Sumber : <https://www.desertcart.uy> (2020)

Alternatif solusi 1 merupakan pilihan sensor yang akan dipakai untuk pemetaan dan *machine learning* pada robot makanan. LiDAR memiliki berbagai kelebihan dan kekurangan dalam aplikasinya, antara lain:

a. Akurasi dan Detail Pemindaian

Mampu memberikan pemetaan lingkungan dengan akurasi yang baik, deteksi objek dapat dilakukan dengan ketelitian hingga beberapa sentimeter dengan ketelitian ± 1 cm hingga ± 2 cm dalam kondisi optimal, meskipun mungkin kurang presisi pada objek yang sangat kecil.

b. Independensi terhadap Kondisi Pencahayaan

LiDAR bekerja efektif di berbagai kondisi pencahayaan, baik siang maupun malam hari. Hal ini menjadikannya ideal untuk pemetaan yang membutuhkan operasional sepanjang waktu, meskipun kemampuan deteksi pada kondisi cahaya sangat rendah bisa terpengaruh.

c. Kemampuan Mengetahui Posisi dan Lingkungan yang Kompleks

LiDAR efektif bekerja di berbagai jenis lingkungan, Meskipun jangkauannya terbatas, LiDAR ini tetap efektif di area yang tidak terlalu padat atau rumit.

d. Pengukuran Jarak

LiDAR dapat mengukur jarak dengan sangat akurat menggunakan pulsa cahaya yang dipantulkan, menghasilkan data posisi objek dalam 3D.

e. Pemetaan 3D

Meskipun resolusinya lebih rendah dibandingkan model yang lebih mahal, LiDAR ini masih mampu menghasilkan model 3D yang berguna untuk navigasi, perencanaan rute, dan deteksi hambatan dalam konteks yang lebih sederhana.

f. Kemampuan Integrasi dengan Sensor Lain

LiDAR dapat dengan mudah diintegrasikan dengan sensor lain seperti kamera, radar. kemampuan integrasi ini mungkin tidak seefektif LiDAR yang lebih mahal. Ini meningkatkan ketepatan aplikasi di berbagai sistem.

g. Mendeteksi Objek Transparan

LiDAR memiliki kemampuan mendeteksi objek transparan seperti kaca atau air yang sulit dideteksi oleh sensor lain.

h. Fungsi Kalibrasi Otomatis:

Beberapa sistem LiDAR dilengkapi dengan kalibrasi otomatis untuk memudahkan perawatan dan menjaga akurasi. Namun tidak selalu efektif pada versi yang dipakai.

Namun, LiDAR juga memiliki kekurangan:

- Keterbatasan dalam Deteksi Objek Kecil**

LiDAR kurang efektif dalam mendeteksi objek kecil atau sangat halus, serta permukaan reflektif atau gelap.

- Terpengaruh oleh Permukaan Tertentu**

LiDAR kesulitan mendeteksi objek atau permukaan yang sangat reflektif atau sangat gelap.

Namun, LiDAR dengan harga terjangkau memiliki jangkauan yang terbatas dan mungkin kesulitan mendeteksi objek kecil atau permukaan reflektif seperti kaca atau air. Meskipun demikian, sistem ini tetap efektif untuk aplikasi yang membutuhkan pemetaan

dan navigasi dasar, serta dapat diintegrasikan dengan sensor lain untuk meningkatkan akurasi dan fungsionalitasnya [6].

3.1.2 Alternatif Solusi 2: Sensor Ultrasonik



Gambar 3. 2 Sensor Ultrasonic

Sumber : <https://khurslabs.com> (2021)

Alternatif solusi 2 merupakan pilihan sensor yang akan dipakai untuk pemetaan dan *machine learning* pada robot makanan. Sensor Ultarsonik memiliki berbagai kelebihan dan kekurangan dalam aplikasinya, antara lain:

a. Akurasi dan Detail Pemindaian

Sensor ultrasonik dapat mengukur jarak dengan cukup akurat, meskipun ketelitiannya tidak setinggi LiDAR, dengan akurasi biasanya dalam rentang beberapa sentimeter ± 1 cm hingga ± 3 cm adalah akurasi standar untuk sensor ultrasonik yang umum.

b. Independensi terhadap Kondisi Pencahayaan

Sensor ultrasonik tidak terpengaruh oleh kondisi pencahayaan, sehingga dapat berfungsi baik di siang maupun malam hari, cocok untuk aplikasi dalam berbagai kondisi cahaya.

c. Kemampuan Mengetahui Posisi dan Lingkungan yang Kompleks

Sensor ultrasonik efektif digunakan dalam lingkungan dengan hambatan terbatas atau area yang relatif lebih terbuka, meskipun kurang efektif di ruang dengan banyak objek kecil atau permukaan reflektif.

d. Pengukuran Jarak

Sensor ultrasonik mengukur jarak dengan memanfaatkan gelombang suara yang dipantulkan dari objek. Pengukuran ini dapat dilakukan dengan cukup cepat, meski dengan batasan akurasi tergantung pada jenis sensor.

e. Pemetaan 3D

Meskipun sensor ultrasonik dapat digunakan untuk pengukuran jarak, mereka tidak menghasilkan model 3D secara langsung. Sensor ultrasonik membutuhkan penggabungan dengan teknologi lain untuk menghasilkan model 3D.

f. Kemampuan Integrasi dengan Sensor Lain

Sensor ultrasonik sering digunakan bersama dengan sensor lain, seperti kamera atau IMU, untuk meningkatkan ketepatan dalam aplikasi seperti robotika atau kendaraan otomatis.

g. Mendeteksi Objek Transparan

Sensor ultrasonik lebih efektif dalam mendeteksi objek padat dan tidak dapat mendeteksi objek transparan seperti kaca atau air, yang sering menjadi tantangan bagi sensor lain seperti LiDAR.

h. Fungsi Kalibrasi Otomatis

Beberapa sensor ultrasonik dilengkapi dengan kemampuan kalibrasi otomatis untuk menjaga akurasi dalam jangka Panjang

Namun, sensor ultrasonik juga memiliki kekurangan :

- **Biaya Terjangkau, Tetapi Keterbatasan Fitur**

Dibandingkan dengan teknologi lain seperti LiDAR, sensor ultrasonik lebih terjangkau, namun fitur dan jangkauannya terbatas, biasanya dengan harga mulai dari sekitar Rp 100.000 hingga Rp 2.000.000 untuk model yang lebih canggih.

- **Keterbatasan dalam Deteksi Objek Kecil**

Sensor ultrasonik kurang efektif dalam mendeteksi objek kecil atau halus, serta tidak cocok untuk mendeteksi objek yang sangat jauh (lebih dari beberapa meter).

- **Terpengaruh oleh Permukaan dan Kondisi Lingkungan**

Sensor ultrasonik bisa kesulitan dalam mendeteksi objek pada permukaan yang sangat lembut, berbentuk kompleks, atau di lingkungan dengan banyak gangguan suara. Sensor ultrasonik merupakan teknologi pengukur jarak yang terjangkau, dengan harga mulai dari Rp100.000 hingga Rp2.000.000. Kinerjanya dipengaruhi oleh beberapa

parameter seperti jangkauan deteksi, frekuensi gelombang, akurasi, dan kualitas pengolahan sinyal. Sensor ini mampu beroperasi dalam kondisi cahaya rendah maupun terang, serta cukup akurat dalam rentang beberapa sentimeter hingga desimeter. Meski sederhana dan cocok untuk aplikasi dasar, sensor ultrasonik memiliki keterbatasan, seperti kesulitan mendeteksi objek transparan, kecil, atau sangat jauh. Mereka tidak menghasilkan model 3D secara langsung, dan sering dikombinasikan dengan sensor lain seperti kamera untuk aplikasi robotik atau otonom. Beberapa model mendukung kalibrasi otomatis guna menjaga akurasi jangka panjang [9].

3.1.3 Alternatif Solusi 3: RPLiDAR



Gambar 3. 3 RPLiDAR

Sumber : <https://store.husarion.com/products/rplidar-a2m12> (2025)

Alternatif solusi 3 merupakan pilihan sensor yang akan dipakai untuk pemetaan dan *machine learning* pada robot makanan. RPLiDAR memiliki berbagai kelebihan dan kekurangan dalam aplikasinya, antara lain:

a. Akurasi dan Detail Pemindaian:

RPLiDAR memberikan pemetaan dengan resolusi tinggi dan akurasi yang sangat baik, dengan ketelitian hingga beberapa sentimeter misalnya pada jarak 0.15 meter hingga 6 meter: $\pm 1\%$ dari jarak sebenarnya. Odometri 360 membantu melacak posisi sensor dengan presisi, meningkatkan keakuratan data yang dihasilkan.

b. Pemetaan 360 Derajat:

RPLiDAR mampu memetakan lingkungan secara menyeluruh dengan pemindaian 360 derajat, sementara odometri 360 melacak pergerakan dan posisi sensor dalam ruang yang kompleks, meningkatkan kemampuan navigasi.

c. Independensi Terhadap Pencahayaan:

RPLiDAR berfungsi baik di segala kondisi pencahayaan, baik siang maupun malam, berkat teknologi laser yang tidak bergantung pada cahaya tampak.

d. Pengukuran Jarak dan Pemetaan 3D:

Dengan kemampuan pengukuran jarak yang sangat akurat, RPLiDAR menghasilkan model 3D dari lingkungan, yang berguna untuk perencanaan rute dan navigasi.

e. Integrasi dengan Sensor Lain:

RPLiDAR dapat dengan mudah diintegrasikan dengan sensor lain seperti kamera dan IMU untuk meningkatkan akurasi dalam aplikasi robotika dan kendaraan otonom.

Namun, ada juga beberapa **kekurangan** yang perlu diperhatikan:

- **Biaya Tinggi:**

Meskipun lebih terjangkau dibandingkan LiDAR kelas atas, harga RPLiDAR dengan odometri 360 tetap signifikan, mulai dari sekitar Rp 5.000.000 hingga Rp 30.000.000 tergantung pada model dan aplikasi.

- **Kompleksitas Integrasi:**

Penggabungan RPLiDAR dengan odometri 360 memerlukan perangkat lunak khusus untuk pemrosesan data dan kalibrasi, yang menambah kompleksitas integrasi dan pengoperasian sistem.

- **Keterbatasan Deteksi Objek Reflektif:**

RPLiDAR dengan odometri 360 merupakan kombinasi sistem yang efektif untuk pemetaan dan navigasi presisi tinggi. Sensor ini mampu melakukan pemindaian 360 derajat dengan frekuensi hingga 8000 titik per detik dan jangkauan antara 5 hingga 12 meter, menghasilkan pemetaan detail dengan akurasi 2–10 cm. Teknologi laser yang digunakan membuatnya dapat berfungsi optimal dalam berbagai kondisi pencahayaan, baik siang maupun malam. Dari sisi integrasi, sistem ini memerlukan perangkat lunak khusus, kalibrasi, dan keterampilan pemrograman, karena kompleksitas pengolahan data dan sinkronisasi posisi. Odometri 360 berperan dalam meningkatkan pelacakan posisi sensor, memperkuat hasil pemetaan dan navigasi dalam ruang kompleks. Namun, RPLiDAR memiliki keterbatasan, seperti kesulitan mendeteksi objek gelap atau sangat reflektif, serta biaya tinggi, dengan kisaran Rp 5.000.000 – Rp 30.000.000, termasuk odometri dan biaya pemeliharaan tahunan sekitar 5–10%. Meski lebih terjangkau dibandingkan LiDAR kelas industri, sistem ini tetap memerlukan investasi besar dan pengaturan yang kompleks [8].

3.2 Analisis dan Pemilihan Solusi

Bagian ini akan membahas mengenai Analisis dan Pemilihan Solusi dari rencana penelitian yang akan dilakukan kedepannya. Analisis dilakukan dengan cara menentukan kriteria dan nilai bobot menggunakan matriks keputusan solusi.

3.2.1 Kriteria Pemilihan Solusi

Penulis menentukan pemilihan solusi terbaik untuk diimplementasikan pada robot pengantar makanan dengan mempertimbangkan beberapa kriteria, yang akan dijelaskan sebagai berikut :

a. Biaya

Biaya yang terjangkau menjadi salah satu kriteria dalam pemilihan solusi. Hal ini penting diperhatikan karena semakin canggih atau kompleks komponen yang digunakan, biaya yang diperlukan juga akan semakin tinggi. Untuk penilaian yang rendah menandakan bahwa solusi tersebut memerlukan biaya lebih.

$$Biaya = Biaya Alat + Biaya lainnya \quad (3.1)$$

b. Efektifitas dan Efisiensi

Solusi yang dipilih harus efektif dan efisien untuk mencapai tujuan penelitian. Efektivitas solusi ditentukan oleh sejauh mana solusi yang diterapkan dapat mencapai tujuan yang telah ditetapkan. Sementara itu, efisiensi menjadi faktor utama yang mencakup penggunaan waktu dan biaya secara optimal. Dengan kata lain, solusi yang efisien adalah solusi yang mampu mencapai tingkat efektivitas yang tinggi dengan memanfaatkan sumber daya seminimal mungkin. Untuk penilaian solusi yang tinggi menandakan bahwa solusi tersebut lebih efektif dan efisien.

c. Akurasi

Solusi yang dipilih tentu harus memberikan hasil data yang akurat untuk memberikan hasil yang sesuai. Untuk penilaian solusi yang tinggi menandakan bahwa solusi tersebut memiliki tingkat akurasi yang lebih baik, Berikut Rumus dari Akurasi itu sendiri :

$$Akurasi = \frac{Jumlah Benar}{Jumlah Total} \times 100\% \quad (3.2)$$

d. Keandalan

Solusi yang diimplementasikan harus dapat beroperasi dengan andal dalam berbagai kondisi. Sistem yang sering mengalami kegagalan akan menimbulkan ketidaknyamanan bagi pengguna dan memengaruhi tingkat kepercayaan terhadap teknologi yang digunakan. Untuk penilaian solusi yang tinggi menandakan bahwa solusi tersebut memiliki keandalan yang baik.

e. Fleksibilitas

Solusi yang fleksibel akan lebih mudah untuk beradaptasi terhadap perubahan kebutuhan atau lingkungan operasional. Misalnya, robot dapat berfungsi di berbagai jenis lantai atau kondisi ruang yang berbeda. Untuk rating solusi yang tinggi menandakan bahwa solusi tersebut memiliki fleksibilitas yang lebih baik.

Berikut adalah keterangan skala singkat untuk setiap parameter pada tabel penilaian:

A. Biaya (Bobot: 15%)

Mengukur seberapa efisien biaya yang dibutuhkan untuk solusi. Skala: 1 (sangat tinggi) hingga 5 (sangat rendah dan efisien).

B. Efektifitas dan Efisiensi (Bobot: 20%)

Mengukur seberapa baik solusi mencapai tujuan dengan cara yang efisien. Skala: 1 (sangat tidak efektif) hingga 5 (sangat efektif dan efisien).

C. Akurasi (Bobot: 25%)

Mengukur ketepatan hasil yang diberikan oleh solusi. Skala: 1 (banyak kesalahan) hingga 5 (sangat akurat).

D. Keandalan (Bobot: 20%)

Mengukur sejauh mana solusi dapat diandalkan dalam jangka panjang. Skala: 1 (sering gagal) hingga 5 (sangat dapat diandalkan).

E. Fleksibilitas (Bobot: 20%)

Mengukur kemampuan solusi untuk beradaptasi dengan perubahan. Skala: 1 (tidak fleksibel) hingga 5 (sangat fleksibel).

3.2.2 Matriks Keputusan Solusi

Untuk menentukan solusi dari permasalahan yang dihadapi, diperlukan berbagai pertimbangan. Agar proses pengambilan solusi terpilih menjadi lebih mudah, digunakanlah Matriks Keputusan Solusi seperti berikut:

Tabel 3.1 Matriks Penilaian

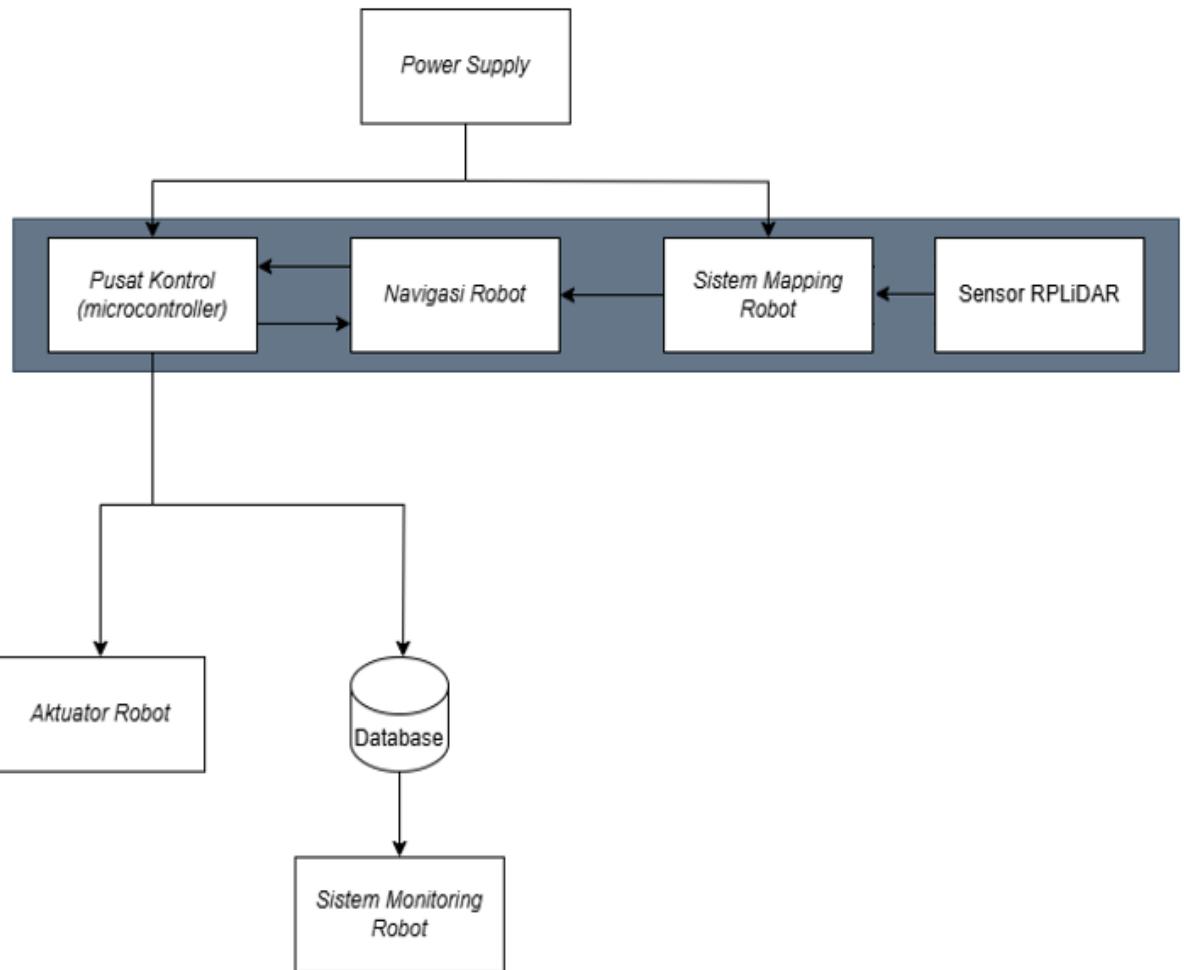
Kriteria Solusi	Bobot	Solusi 1		Solusi 2		Solusi 3	
		Rating	Nilai Bobot	Rating	Nilai Bobot	Rating	Nilai Bobot
Biaya	15%	3	0.45	4	0.6	2	0.3
Efektifitas dan Efisiensi	20%	4	0.8	3	0.6	5	1
Akurasi	25%	4	1	3	0.75	5	1.25
Keandalan	20%	4	0.8	2	0.4	4	0.8
Fleksibilitas	20%	3	0.6	2	0.4	4	0.8
Total Nilai		3.65		2.75		4.15	
Peringkat		2		3		1	
Alternatif Solusi Terpilih		Alternatif Solusi 3					

3.3 Desain Solusi Terpilih

Berdasarkan analisis dari berbagai solusi yang ada, penulis memilih alternatif solusi 3 yang menggunakan Sensor RPLiDAR sebagai pemetaan lingkungan sekitar dan navigasi pergerakan robot. Hal ini dikarenakan solusi tersebut mampu membaca lingkungan sekitar secara menyeluruh dengan detail dan luas, sehingga memungkinkan robot dapat membuat peta lingkungan yang akurat dalam waktu yang lebih singkat dibandingkan dengan alternatif solusi yang lain.

Meskipun solusi ini lebih mahal dibandingkan alternatif solusi yang lain. Hal ini mampu diimbangi dalam hal efektivitas, efisiensi, dan akurasi, sehingga keunggulan tersebut dapat menutupi kekurangan yang ada. Oleh karena itu, berdasarkan analisis menyeluruh, solusi ini menjadi solusi yang terbaik untuk sistem navigasi pada robot pengantar makanan.

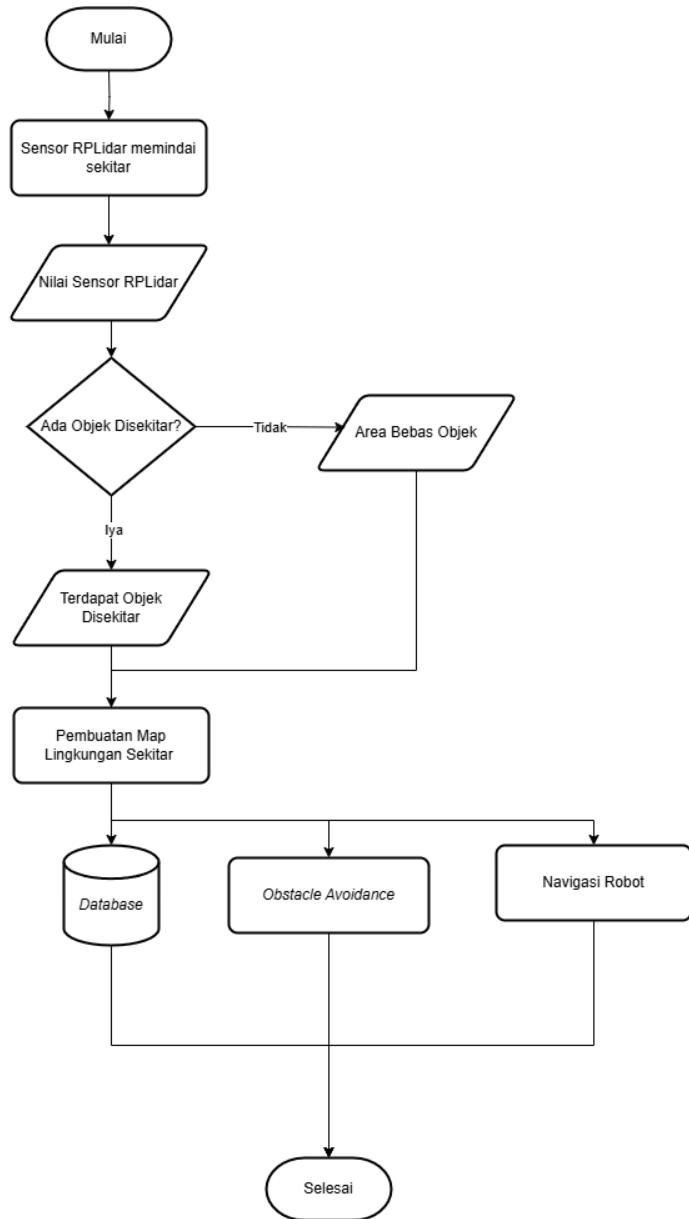
3.3.1 Blok Diagram Sistem



Gambar 3.4 Blok Diagram Sistem

Gambar 3.4 merupakan diagram blok keseluruhan sistem pada robot pengantar makanan, dengan fokus penggerjaan pada kotak berwarna biru. Sistem *mapping* robot akan mendapatkan daya melalui *power supply* untuk mendapatkan data terkait objek disekitar robot melalui sensor RPLiDAR. Data tersebut akan dikumpulkan menjadi sebuah peta secara menyeluruh. peta yang terbentuk akan menjadi acuan robot untuk bergerak dan menghindari objek disekitar, sehingga hal ini akan membentuk sistem navigasi robot. Saat sistem ini terbentuk, data yang didapat akan dikirimkan menuju pusat kontrol untuk menggerakkan aktuator robot.

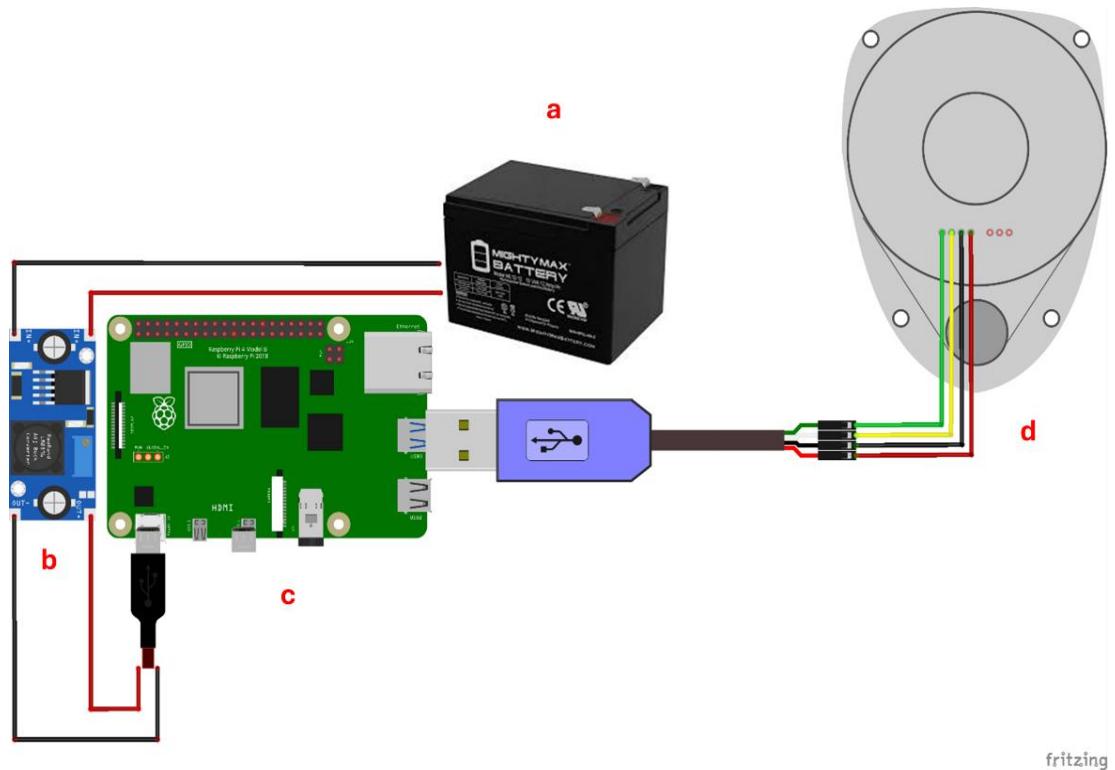
3.3.2 Flowchart Sistem Mapping



Gambar 3.5 Flowchart Sistem Mapping

Pada Gambar 3.5 merupakan proses sistem *mapping* pada robot bekerja. Proses dimulai dengan inisialisasi sensor RPLidar untuk memindai area sekitar. Data dari sensor kemudian diterima oleh sistem untuk dilakukan deteksi objek. Jika terdeteksi ada objek, area sekitar diolah menjadi peta dengan memperhatikan kondisi lingkungan dan hambatan yang ada. Jika tidak ada objek, sistem tetap memperbarui peta berdasarkan kondisi kosong. Selanjutnya, hasil pemetaan dikirimkan ke *database* dan digunakan untuk perencanaan navigasi robot. Robot kemudian dapat bergerak berdasarkan peta yang telah diperoleh.

3.3.3 Desain Perangkat Keras Sistem Navigasi



Gambar 3. 6 Desain Perangkat Keras

Pada **Gambar 3.6** (a) merupakan DC *Power Supply* 12V yang digunakan sebagai sumber daya utama untuk sistem ini yang terhubung dengan komponen pada **Gambar 3.6** (b) yaitu *step down converter* yang menurunkan tegangan dari 12V menjadi 5V. *Output* dari *step down converter* ini diteruskan ke komponen pada **Gambar 3.6** (c) yaitu Raspberry Pi, yang berfungsi sebagai pusat pengolahan data dalam sistem ini. Raspberry Pi terhubung dengan sensor RPLiDAR seperti yang ditunjukkan pada **Gambar 3.6** (d) melalui pin TX, RX, GND, dan 5V. Pin TX dan RX digunakan untuk komunikasi serial antara Raspberry Pi dan RPLiDAR, sementara pin GND dan 5V digunakan untuk penyediaan daya listrik ke RPLiDAR.

3.4 Jadwal dan Anggaran

Jadwal dan Anggaran dirancang dimulai dari awal menjalankan hingga 5 bulan kedepanya, Penulis memulai dari bulan januari 2025 hingga bulan juni 2025.

BAB IV

IMPLEMENTASI

4.1 Deskripsi Umum Implementasi

Implementasi umum dari sistem navigasi cerdas ini mempunyai fokus pada pengembangan robot pengantar makanan dengan sistem mapping robot sebagai elemen utama dalam navigasi. Sistem ini dirancang agar mampu melakukan pemetaan lingkungan yang diperbarui secara *real-time* dan menentukan jalur pergerakan yang optimal, sehingga meningkatkan efisiensi dan akurasi untuk pengantaran makanan. Untuk membantu kerja fungsinya, robot menggunakan DC *Power Supply* 12V yang dialirkan melalui *step down converter* untuk menurunkan tegangan menjadi 5V, yang kemudian digunakan oleh Raspberry Pi 4B 8GB Kit sebagai pusat kendali utama. Raspberry Pi terhubung dengan sensor RPLIDAR A2, yang berfungsi untuk memindai lingkungan sekitar dengan resolusi yang tinggi.

Proses pemetaan dimulai dengan inisialisasi sensor RPLIDAR, yang mengumpulkan data mengenai objek di sekitar robot dan memperbarui peta secara adaptif, informasi hasil pemetaan akan disimpan dalam *database* yang digunakan oleh pusat kontrol untuk menentukan sistem navigasi robot. Sistem pergerakan navigasi robot dikendalikan oleh aktuator motor DC dengan spesifikasi kecepatan rotasi untuk memastikan pergerakan yang stabil dan sesuai dengan jalur yang direncanakan. Selain pemetaan yang berbasis sensor, sistem ini juga mengimplementasikan *machine learning* untuk meningkatkan akurasi navigasi. Data yang diperoleh dari sensor diproses dan digunakan dalam pelatihan model *machine learning* menggunakan algoritma KNN (*K-Nearest Neighbors*) dan BFS (*Breadth-First Search*).

Model yang telah dilatih akan digunakan untuk menentukan jalur navigasi yang lebih dinamis, dengan metode jalur agar pergerakan robot lebih efisien dan halus. Dengan spesifikasi perangkat keras yang telah dijelaskan serta penerapan algoritma navigasi berbasis *machine learning*, sistem ini memastikan bahwa robot dapat bergerak secara halus, menghindari rintangan, dan mengantarkan makanan dengan efisien sesuai dengan spesifikasi yang telah kami jelaskan pada desain solusi terpilih.

4.2 Detil Implementasi

4.2.1 Perangkat Keras

4.2.1.1 RPLIDAR A2

Pada **Gambar 3.3** adalah sensor yang dapat melakukan pemetaan lingkungan sekitar secara 360 derajat menggunakan laser. Sensor ini menggunakan teknologi *Triangulation* atau metode untuk menentukan posisi suatu objek atau titik dengan menggunakan pengukuran sudut dan jarak dari dua atau lebih titik referensi yang sudah diketahui posisinya[14]. *Triangulation* dimanfaatkan melalui data sudut dan jarak yang diperoleh dari RPLiDAR, untuk menentukan posisi objek dan posisi robot itu sendiri secara relatif terhadap lingkungan, dan digunakan sebagai dasar dalam proses SLAM dan navigasi otomatis.untuk mengukur jarak ke objek di sekitarnya dengan akurasi tinggi seperti yang tertera pada spesifikasi pada **Tabel 4.1**.

Tabel Spesifikasi RPLIDAR A2M8

Tabel 4. 1 Spesifikasi RPLIDAR A2M8

Spesifikasi	Keterangan
Jarak Pemindaian	Hingga 12 meter
Sudut Pemindaian	360°
Kecepatan Rotasi	5 – 15 Hz
Resolusi Sudut	±0.45° (pada 10 Hz)
Jarak Pemindaian	Hingga 12 meter
Frekuensi Data Output	Hingga 8000 sampel/detik
Antarmuka Komunikasi	UART (3.3V TTL)
Tegangan Operasi	5V DC

Sumber:https://cdn.sparkfun.com/assets/e/a/f/9/8/LD208_SLAMTEC_rplidar_datasheet_A2M8_v1.0_en.pdf

Setelah dihubungkan dengan mikrokomputer Raspberry Pi, RPLIDAR A2 dapat mengumpulkan data lingkungan dan memprosesnya menggunakan algoritma "SLAM" Simultaneous Localization and Mapping. Hasil pemetaan dapat digunakan untuk pembuatan peta 2D, navigasi robot dan penghindaran rintangan.

```

1   <launch>
2     <node name="rplidarNode"          pkg="rplidar_ros" type="rplidarNode" output="screen">
3       <param name="serial_port"      type="string" value="/dev/ttyUSB0"/>
4       <param name="serial_baudrate" type="int"    value="115200"/>
5       <param name="frame_id"        type="string" value="laser"/>
6       <param name="inverted"        type="bool"   value="false"/>
7       <param name="angle_compensate" type="bool"   value="true"/>
8     </node>
9   </launch>

```

Gambar 4. 1 Pkode RPLiDAR A2

Pada **gambar 4.1** merupakan pkode program implementasi RPLIDAR A2 untuk pemetaan lingkungan sekitar. Pada baris pertama, merupakan tanda untuk memulai program dengan file bertipe *launch*. Pada baris kedua berfungsi untuk membuat node dengan nama “rplidarNode” sehingga data yang didapat dari sensor RPLIDAR A2 dapat disimpan ke node tersebut. Selanjutnya, kita membuat parameter yang dibutuhkan untuk menjalankan sensor RPLIDAR A2 seperti “serial_port” berfungsi untuk menentukan port serial yang terhubung ke sensor dan parameter “serial_baudrate” untuk menentukan kecepatan komunikasi serial antara komputer dan sensor. Parameter yang kami gunakan saat ini menggunakan “USB0” sebagai “serial_port” dan “115200 bps” sebagai “serial_baudrate”. Parameter yang lain bisa digunakan sesuai kebutuhan penggunaan perangkat. Pada baris terakhir berfungsi untuk menutup program yang ada[8].

4.2.1.2 Raspberry Pi 4 Model B



Gambar 4. 2 Raspberry Pi 4 Model B

Sumber : <https://www.hestanto.web.id/perbedaan-dari-versi-raspberry-pi> (2024)

Tabel 4. 2 Spesifikasi Raspberry Pi 4 Model B

Spesifikasi	Keterangan
Prosesor	Broadcom BCM2711, Quad-core Cortex-A72 (64-bit) @ 1.5GHz
RAM	2GB / 4GB / 8GB LPDDR4-3200
Jaringan	Gigabit Ethernet, 2.4GHz & 5GHz Wi-Fi, Bluetooth 5.0
Tegangan Operasi	5V via USB-C (min. 3A disarankan)
Sistem Operasi	Raspberry Pi OS (Linux), atau OS lain berbasis ARM

Sumber : <https://www.raspberrypi.org/documentation/hardware/> (2024)

Raspberry Pi 4 Model B adalah komputer mini dengan performa tinggi yang dilengkapi dengan prosesor quad-core ARM Cortex-A72 berkecepatan 1.5 GHz. Perangkat ini memungkinkan pengguna menjalankan berbagai aplikasi, termasuk pemrosesan data sensor, IoT, dan robotika. Raspberry Pi 4 memiliki konektivitas dengan dukungan Wi-Fi yang handal untuk komunikasi yang lebih cepat [10].



Gambar 4. 3 Implementasi Visual

Dengan sistem operasi berbasis Linux seperti Raspberry Pi OS, perangkat ini dapat digunakan untuk berbagai keperluan, termasuk pengendalian robot, server IoT, serta pemrosesan data dari sensor seperti RPLIDAR A2 untuk pemetaan lingkungan. Hal

tersebut menjadikannya pilihan ideal untuk pengembangan proyek berbasis *embedded system*.

Pada **gambar 4.3** merupakan implementasi visual pemrosesan data pada Raspberry Pi Model B. data yang divisualisasikan didapat dari node “rplidarNode” yang berisi data pemetaan lingkungan sekitar. Data tersebut diproses oleh Raspberry Pi sehingga data yang ada dapat divisualisasikan.

4.2.2 Perangkat Lunak

4.2.2.1 DBScan

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) adalah salah satu algoritma clustering yang bekerja berdasarkan kepadatan data[17]. Berbeda dengan metode seperti K-Means yang membutuhkan jumlah cluster sebagai input, DBSCAN tidak memerlukan jumlah cluster secara eksplisit dan dapat membentuk cluster dengan bentuk yang tidak teratur (non-linear).

DBSCAN mengelompokkan data berdasarkan kedekatan antar titik dan jumlah titik dalam suatu wilayah. Algoritma ini mampu mengidentifikasi:

- *Core Point*: titik yang memiliki cukup banyak tetangga di sekitarnya.
- *Border Point*: titik yang berada di sekitar core point, tetapi tidak memiliki cukup tetangga untuk menjadi core.
- *Noise (Outlier)*: titik yang tidak termasuk dalam cluster mana pun karena tidak memiliki cukup kedekatan dengan titik-titik lain.

Keunggulan DBSCAN:

- Tidak perlu menentukan jumlah cluster di awal.
- Dapat menemukan cluster dengan bentuk arbitrer (tidak terikat bentuk).
- Mampu menangani outlier (data yang tidak termasuk dalam cluster).

Parameter Utama DBSCAN:

- *eps* (*epsilon*): jarak maksimum antar titik untuk dianggap bertetangga.
- *min_samples*: jumlah minimum titik tetangga yang dibutuhkan untuk membentuk sebuah core point.

Cara Kerja DBSCAN:

1. Menghitung Jarak Antar Titik

Untuk mengetahui seberapa dekat dua titik satu sama lain, digunakan rumus jarak Euclidean dua dimensi sebagai berikut:

$$d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

2. Mengukur Jarak Antar Pasangan Titik

Jarak antara tiap dua titik dalam dataset dihitung menggunakan rumus Euclidean untuk menghasilkan sebuah matriks yang merepresentasikan kedekatan antar titik.

3. Menentukan Titik Tetangga Berdasarkan Parameter eps

Titik-titik yang memiliki jarak sama dengan atau kurang dari nilai ambang eps akan diklasifikasikan sebagai tetangga. Langkah ini penting untuk menilai hubungan lokal antara titik-titik dalam ruang vector.

3. Menentukan Titik Pusat Cluster Berdasarkan min_samples

Sebuah titik akan dianggap sebagai bagian dari cluster apabila jumlah tetangganya (berdasarkan eps) memenuhi atau melebihi nilai min_samples . Titik tersebut kemudian disebut sebagai *core point* atau titik inti dalam proses pembentukan cluster.

4.2.2.2 KNN

K-Nearest Neighbors (KNN) adalah salah satu algoritma *machine learning* yang digunakan untuk klasifikasi dan regresi [16]. KNN termasuk metode *lazy learning* karena tidak mempelajari model secara eksplisit, tetapi menyimpan seluruh data latih dan melakukan prediksi hanya saat ada data uji (*inference time*).

Cara kerja KNN:

1. Tentukan Nilai K

K adalah jumlah tetangga terdekat yang akan dipertimbangkan.

2. Hitung Jarak

Untuk setiap data uji, algoritma menghitung jarak keseluruhan data. Jarak yang paling umum digunakan adalah Euclidean Distance mirip seperti yang digunakan oleh DBSCAN sebelumnya.

3. Pilih K Tetangga Terdekat

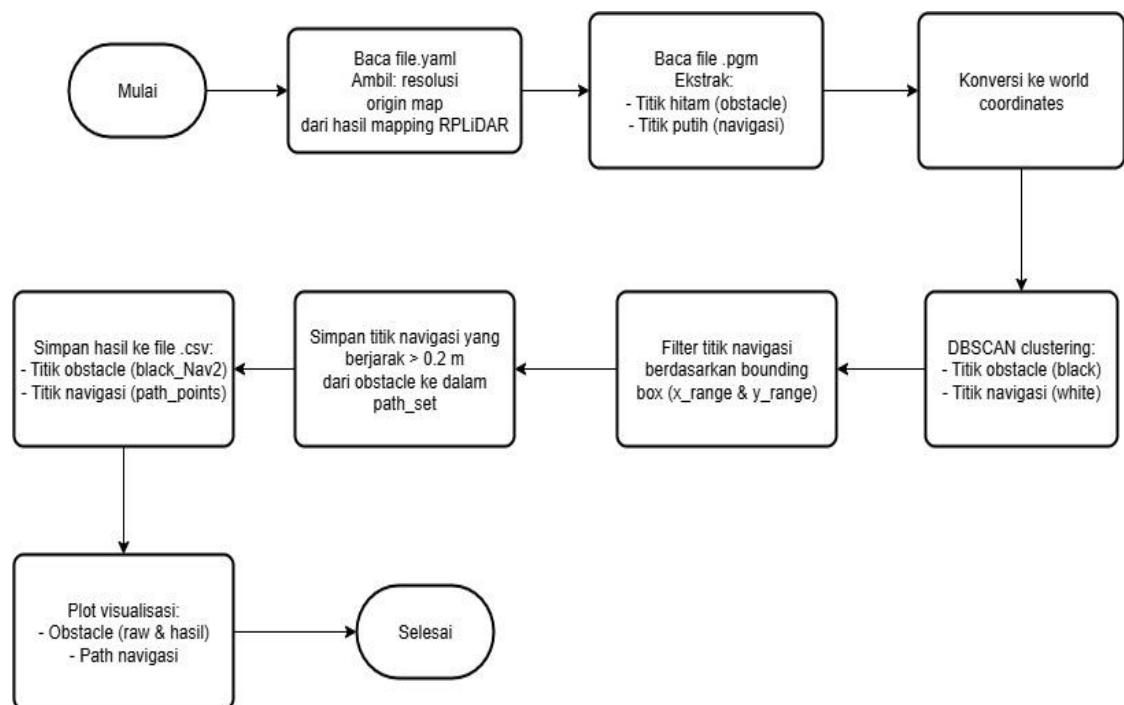
Urutkan seluruh data berdasarkan jarak ke data uji, lalu ambil K data dengan jarak paling kecil.

4. Buat Edge pada Graph

Hubungkan titik utama ke K tetangganya dengan membuat edge atau sisi. Setiap edge dapat menyimpan informasi seperti jarak sebagai bobot pada graf.

4.2.2.3 Pengambilan Koordinat Path

Setelah mendapatkan hasil *mapping* dari rplidar A2 yang merupakan file yaml dan pgm selanjutnya kami perlu memisahkan antara koordinat *obstacle* dan koordinat yang bukan *obstacle* (koordinat yang dapat dilalui robot). Menggunakan Bahasa pemrograman *python* untuk mengcluster yang mana *obstacle* dan yang bukan dengan algoritma *machine learning* DBScan. Berikut adalah flowchart programnya:



Gambar 4. 4 Flowchart Proses Area Navigasi

lingkungan dalam format .pgm dan metadata dari file .yaml. Peta ini menggambarkan area navigasi dalam bentuk citra hitam-putih, di mana titik hitam merepresentasikan obstacle dan titik putih merepresentasikan area yang dapat dilalui.

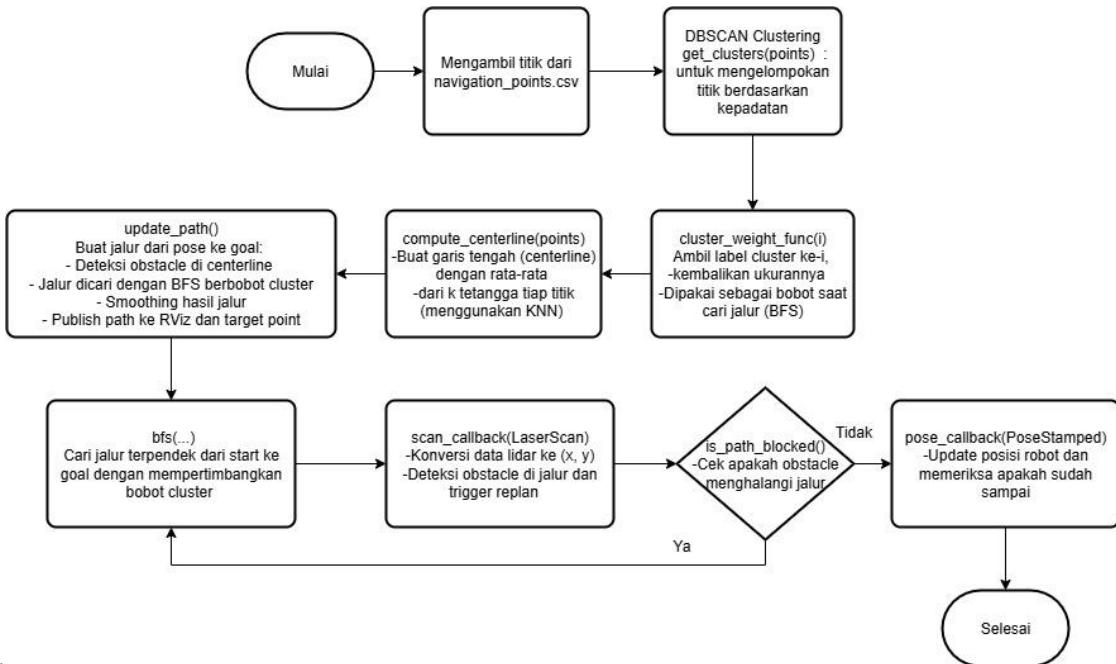
Setelah itu, data peta dikonversi ke koordinat dunia menggunakan informasi resolusi dan origin dari file .yaml. Titik-titik obstacle dan area navigasi kemudian diproses menggunakan metode clustering DBSCAN untuk mengelompokkan area-area yang relevan dan mengurangi noise dari data mentah.

Selanjutnya, dilakukan proses filtering terhadap area navigasi agar hanya mencakup wilayah dalam batas yang telah ditentukan (range X dan Y). Titik-titik navigasi yang memiliki jarak lebih dari 0.2 meter dari obstacle dianggap aman dan dipilih sebagai titik jalur yang dapat dilalui robot.

Hasil akhir dari proses ini berupa kumpulan titik jalur navigasi yang disimpan ke dalam file .csv untuk digunakan dalam perencanaan dan pelaksanaan navigasi robot. Selain itu, visualisasi juga ditampilkan untuk menunjukkan distribusi obstacle dan jalur yang telah direncanakan. Setelah jalur ditemukan, sistem akan melakukan penyederhanaan (*smoothing*) terhadap jalur dengan cara menghitung rata-rata dari setiap segmen titik agar jalur menjadi lebih halus dan efisien untuk dilalui. Robot akan mulai bergerak mengikuti jalur yang telah direncanakan dengan terus memantau keberadaan *obstacle* secara *real-time* melalui sensor Lidar. Jika di sepanjang jalur terdapat halangan baru yang terdeteksi, sistem akan secara otomatis melakukan *replanning* (perencanaan ulang jalur) untuk menghindari tabrakan dan mencari rute alternatif ke tujuan.

4.2.2.4 Rute Navigasi

Berdasarkan kode di point sebelumnya yang menyimpan koordinat yang bukan *obstacle* (koordinat yang dapat dilalui robot) kedalam file csv selanjutnya adalah untuk memprediksi rute tercepat dari titik mulai ke titik tujuan berdasarkan file csv tersebut menggunakan algoritma *machine learning* KNN (*K-Nearest Neighbors*) dan BFS (*Breadth-First Search*).



Gambar 4. 5 Flowchart Pembuatan Jalur

Proses navigasi robot dalam sistem ini diawali dengan memuat titik-titik jalur navigasi (*path*) dari file CSV hasil pemrosesan sebelumnya, seperti yang diperoleh dari DBSCAN. Titik-titik ini kemudian digunakan untuk membentuk centerline atau garis tengah jalur yang dihasilkan menggunakan algoritma K-Nearest Neighbors (KNN).

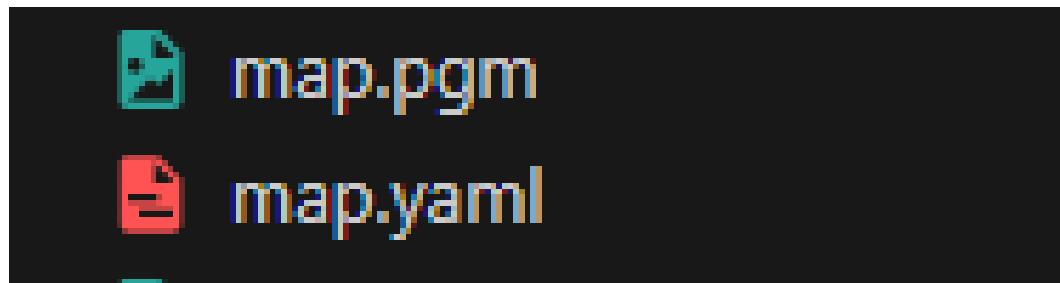
Selanjutnya, sistem menunggu pose awal robot dari topik `/slam_out_pose` dan data sensor Lidar dari topik `/scan`. Saat posisi robot dan lingkungan telah diketahui, sistem akan mulai melakukan perencanaan jalur dinamis dari posisi saat ini menuju titik tujuan menggunakan algoritma kombinasi:

- KNN: untuk mencari tetangga terdekat dari setiap titik navigasi,
- Breadth-First Search (BFS): untuk menentukan rute terpendek antar titik-titik yang dapat dilalui.

Sebelum BFS dilakukan, sistem akan memeriksa apakah ada halangan (*obstacle*) di sepanjang jalur. Titik-titik pada centerline yang berada dalam radius tertentu dari obstacle (0.3 meter) akan ditandai sebagai titik yang diblokir dan tidak digunakan dalam pencarian jalur.

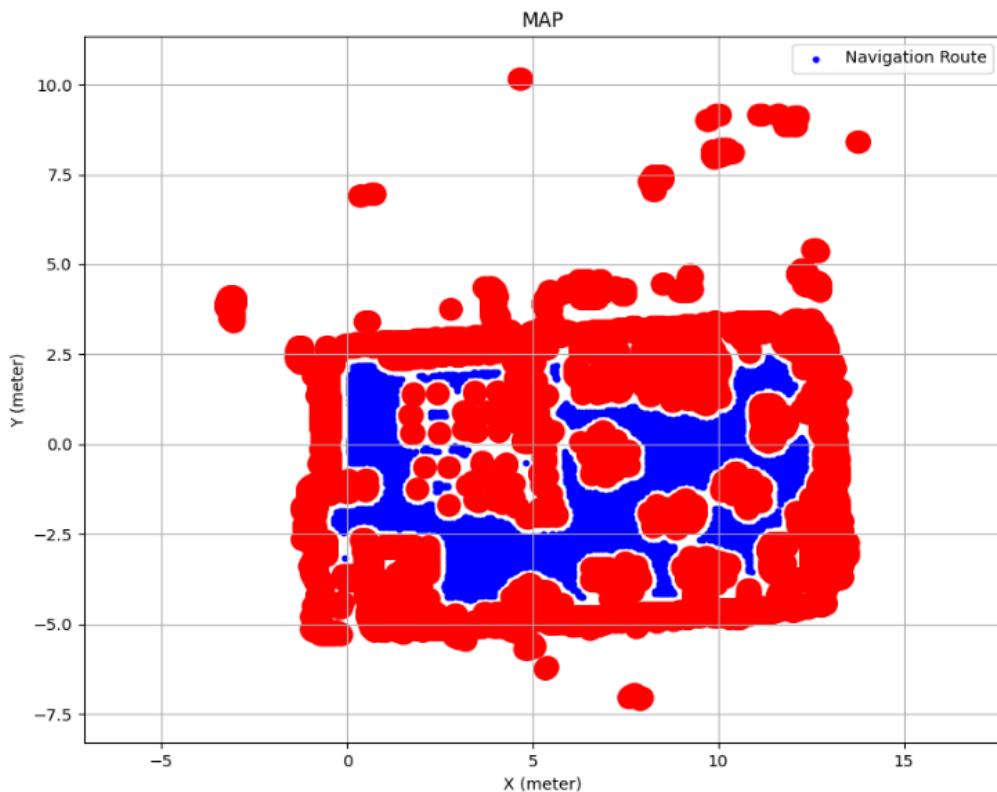
4.3 Prosedur Pengoperasian

Untuk menggunakan program dengan *source code* diatas yang terdapat 2 buah file yang pertama “Map_Read.py” dan yang kedua “KNN_Path.py” perlu melakukan mapping menggunakan “HectorSlam” dengan RPlidar untuk mendapatkan Mapnya dan akan di peroleh 2 file yaitu yaml dan pgm.



Gambar 4. 6 File yaml & pgm

File inilah yang akan di proses oleh file “Map_Read.py” untuk mengambil navigasi area yang akan di simpan kedalam file csv. Dan berikut hasil dari “Map_Read.py” untuk menampilkan Mapnya.



Setelah memiliki file csv dari hasil “Map_Read.py” seperti gambar dibawah ini. Selanjutnya dapat di proses di file “KNN_Path.py”

```

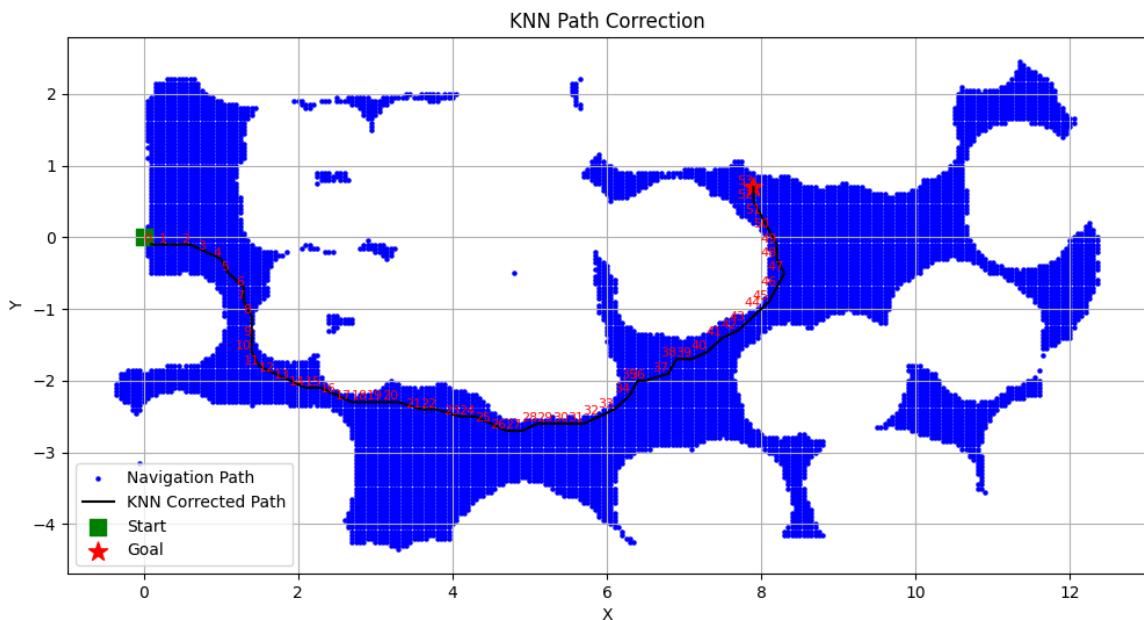
path_points.csv
SVM > path_points.csv
1 x,y
2 5.899976000000002,1.1500239999999948
3 2.1999760000000066,-2.1999760000000066
4 0.8499760000000052,-1.7999760000000001
5 2.8499760000000005,-3.1999760000000066
6 2.8499760000000005,-3.4499760000000066
7 2.8499760000000005,-3.6999760000000066
8 2.8499760000000005,-3.9499760000000066
9 10.949976000000007,2.0000239999999962
10 8.699976000000007,0.6000239999999977
11 8.699976000000007,0.35002399999999767
12 8.699976000000007,0.10002399999999767
13 8.699976000000007,-0.14997600000000233

```

Gambar 4. 8 “KNN_Path.py”

Di file “KNN_Path.py” kami dapat menentukan titik mulai dan tujuannya dengan mengubah kode dengan variable *start_point* dan *goal_point*. Pada contoh di bawah ini akan menggunakan *start_point* “0, 0” dan *goal_point* “7.9, 0.7.” Nilai tersebut adalah koordinat X dan Y. Berikut adalah hasilnya.

Berikut ini ada contoh data dari visualisasi Pengujian



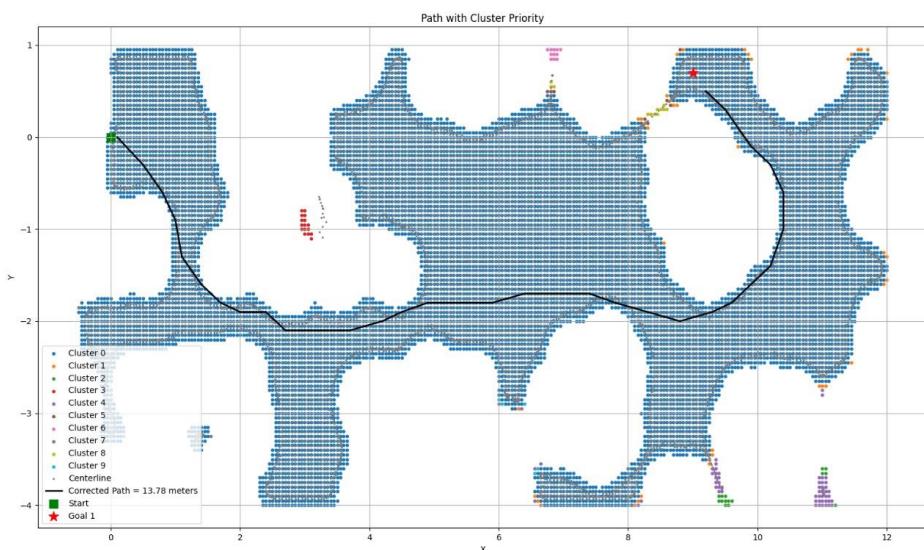
Gambar 4. 9 KNN Path Correction

Setelah itu kami akan mendapatkan file csv yang berisi Path Coordinate yang dapat dilalui oleh robot dengan nama “knn_corrected_path_smoothed.csv” dan hasilnya seperti ini.

	x,y
1	0.1,-0.1
2	0.3,-0.1
3	0.6,-0.1
4	0.8,-0.2
5	1.0,-0.3
6	1.1,-0.5
7	1.3,-0.7
8	1.3,-0.9
9	1.4,-1.1
10	1.4,-1.4
11	1.4,-1.6
12	1.5,-1.8
13	1.7,-1.9
14	

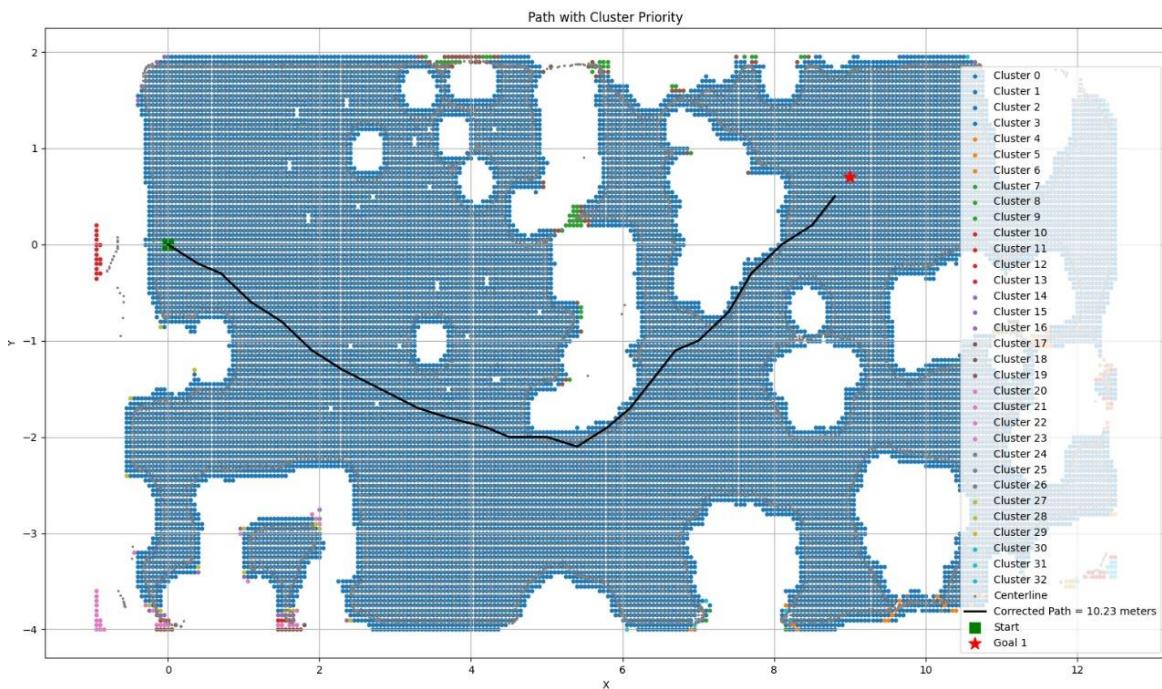
Gambar 4. 10 “knn_corrected_path_smoothed.csv”

Dengan melakukan *smoothing* pada *path* dapat menghasilkan *Path Coordinate* yang lebih sedikit sehingga tidak membuat kerja sistem menjadi lebih berat dan dengan melakukan *smoothing* dapat membuat *path* menjadi lebih bagus dan menghindari koordinat berulang yang membuat koordinat menjadi banyak.



Gambar 4. 11 DBSCAN Path

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) berfungsi untuk menentukan cluster untuk dijadikan prioritas



Gambar 4.12 Path with Cluster Priority

Gambar 4.10 memperlihatkan visualisasi dari proses pemetaan, pengelompokan area, dan perencanaan jalur yang dilakukan oleh robot pengantar makanan di dalam ruangan. Peta divisualisasikan dalam bentuk *occupancy grid*, di mana warna biru menunjukkan area yang aman untuk dilalui robot, sementara bagian putih atau kosong menggambarkan rintangan atau area yang tidak dapat diakses. Titik berwarna hijau (X) menandakan titik awal keberangkatan robot, sedangkan bintang merah menunjukkan titik tujuan. Garis hitam yang menghubungkan keduanya merupakan jalur yang telah dihitung dan dikoreksi secara otomatis, dengan panjang sekitar 10,23 meter

Selain pemetaan, sistem juga melakukan proses *clustering* terhadap data lingkungan yang diperoleh dari sensor RPLIDAR. Proses ini menggunakan algoritma untuk membagi area ke dalam 33 kelompok atau *cluster* berbeda. Setiap cluster memiliki warna dan label tersendiri (Cluster 0 hingga Cluster 32), yang berfungsi untuk membedakan objek atau area berdasarkan kepadatan titik-titik yang terdeteksi. Pengelompokan ini membantu sistem mengenali bagian-bagian lingkungan, seperti meja, kursi, atau area layanan.

BAB V

PENGUJIAN

5.1 Skenario Umum Pengujian

Pengujian sistem robot dilakukan untuk memastikan bahwa setiap komponen sistem dapat bekerja dengan baik, akurat, sesuai dengan tujuan jelas. Sistem pengujian ini mencakup aspek penting seperti pengukuran jarak, pemetaan lingkungan, penentuan posisi (lokalisasi), serta navigasi otomatis yang terintegrasi melalui sensor RPLiDAR dan metode SLAM. Setiap pengujian bertujuan untuk mengevaluasi kinerja sistem secara menyeluruh dan mengetahui sejauh mana sistem navigasi mampu beroperasi dengan presisi dan efisien.

Pengujian dimulai dengan menguji akurasi sensor RPLiDAR dalam mengukur jarak terhadap objek pantul pada jarak-jarak referensi di setiap 45° sudut dalam kondisi lingkungan yang terkendali. Sensor diuji dari berbagai sudut terhadap sumbu depannya untuk mengamati konsistensi hasil pengukuran. Data yang diperoleh dibandingkan dengan nilai referensi untuk menilai deviasi dan akurasi pengukuran. Selanjutnya, sistem diuji dalam hal pemetaan menggunakan metode SLAM dalam arena yang memiliki dimensi serta rintangan yang bervariasi. Tujuannya adalah untuk melihat kemampuan sistem dalam membangun peta lingkungan secara *real-time* berdasarkan data yang diperoleh dari sensor. Peta yang dihasilkan menjadi dasar bagi proses navigasi selanjutnya.

Pengujian berikutnya adalah Sistem lokalisasi, yaitu kemampuan robot untuk mengenali posisinya dalam rute yang telah dibuat. Estimasi posisi ini diperoleh dari data SLAM dan dibandingkan secara manual pada titik-titik tertentu untuk memastikan akurasinya. Setelah itu, dilakukan pengujian navigasi baik melalui simulasi maupun uji langsung. Navigasi dijalankan dengan menggunakan kombinasi algoritma K-Nearest Neighbor (KNN) dan Breadth-First Search (BFS), yang dirancang agar robot mampu bergerak dari titik awal ke titik tujuan sambil menghindari rintangan di sekitarnya. Melalui rangkaian pengujian skenario ini, diperolehnya gambaran menyeluruh mengenai alur kerja sistem navigasi. Hasil dari uji coba pengujian ini bisa menjadi acuan penting untuk pengembangan sistem navigasi ke depannya, sekaligus memastikan bahwa robot dapat berfungsi dengan baik saat digunakan pada lingkungan.

5.2 Detil Pengujian

5.2.1 Pengukuran jarak dengan RPLidar



Gambar 5. 1 Pengukuran Jarak

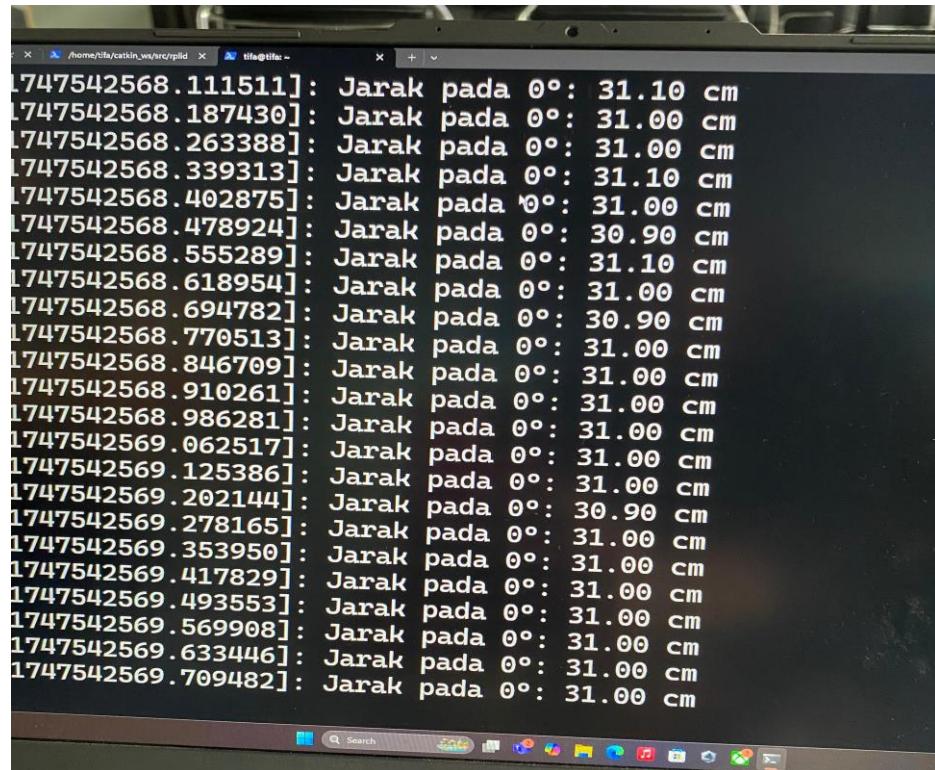
Pengujian ini dilakukan untuk mengevaluasi akurasi pengukuran jarak oleh sensor RPLiDAR di 3 jarak referensi, yaitu pada jarak 30 cm, 60 cm dan 2 m, dengan jumlah pengambilan 30 data dari setiap 45° sudut pemindaian. Tujuan dari pengujian ini adalah untuk memastikan bahwa RPLiDAR mampu mengukur jarak secara konsisten dan akurat dalam kondisi lingkungan yang dikendalikan. Pengujian dilakukan dengan menempatkan objek pantul/target pada jarak tertentu dari sensor dan mencatat hasil pengukuran pada berbagai sudut terhadap sumbu depan sensor. Berikut adalah hasil pengambilan data jarak menggunakan RPLiDAR.

Untuk mendapatkan berapa waktu 1 rotasi rplidar digunakan rumus berikut yang didapat dari timestamp ros.

$$Delay = \sum_{n=0}^{\infty} timestamp^{n+1} - timestamp^n \quad (5.1)$$

Untuk mengetahui berapa nilai timestampnya dapat dilihat di **Gambar 5.2** yang menggunakan ros untuk mengetahui berapa timestampnya.

- Sudut 0 derajat dengan jarak 30 cm



```

1747542568.111511]: Jarak pada 0°: 31.10 cm
1747542568.187430]: Jarak pada 0°: 31.00 cm
1747542568.263388]: Jarak pada 0°: 31.00 cm
1747542568.339313]: Jarak pada 0°: 31.10 cm
1747542568.402875]: Jarak pada 0°: 31.00 cm
1747542568.478924]: Jarak pada 0°: 30.90 cm
1747542568.555289]: Jarak pada 0°: 31.10 cm
1747542568.618954]: Jarak pada 0°: 31.00 cm
1747542568.694782]: Jarak pada 0°: 30.90 cm
1747542568.770513]: Jarak pada 0°: 31.00 cm
1747542568.846709]: Jarak pada 0°: 31.00 cm
1747542568.910261]: Jarak pada 0°: 31.00 cm
1747542568.986281]: Jarak pada 0°: 31.00 cm
1747542569.062517]: Jarak pada 0°: 31.00 cm
1747542569.125386]: Jarak pada 0°: 31.00 cm
1747542569.202144]: Jarak pada 0°: 30.90 cm
1747542569.278165]: Jarak pada 0°: 31.00 cm
1747542569.353950]: Jarak pada 0°: 31.00 cm
1747542569.417829]: Jarak pada 0°: 31.00 cm
1747542569.493553]: Jarak pada 0°: 31.00 cm
1747542569.569908]: Jarak pada 0°: 31.00 cm
1747542569.633446]: Jarak pada 0°: 31.00 cm
1747542569.709482]: Jarak pada 0°: 31.00 cm

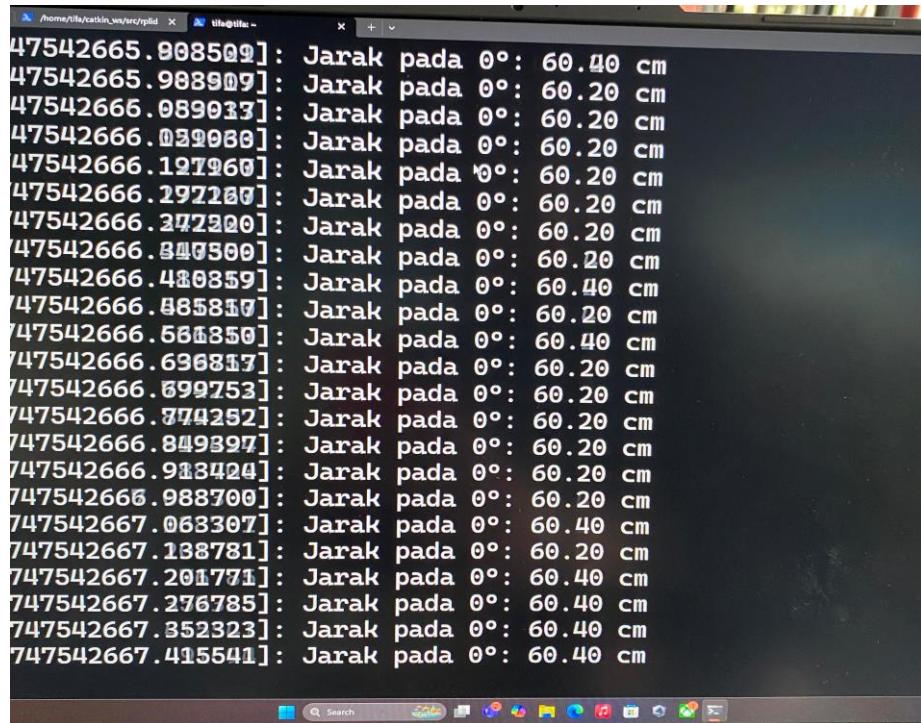
```

Gambar 5. 2 Hasil pengukuran dengan RPLidar (0 °,30 cm)



Gambar 5. 3 Hasil pengukuran dengan alat pengukur manual (0 °,30 cm)

- Sudut 0 derajat dengan jarak 60 cm



```

47542665.008509]: Jarak pada 0°: 60.40 cm
47542665.988909]: Jarak pada 0°: 60.20 cm
47542666.089033]: Jarak pada 0°: 60.20 cm
47542666.122080]: Jarak pada 0°: 60.20 cm
47542666.127960]: Jarak pada 0°: 60.20 cm
47542666.192200]: Jarak pada 0°: 60.20 cm
47542666.272200]: Jarak pada 0°: 60.20 cm
47542666.840500]: Jarak pada 0°: 60.20 cm
47542666.480859]: Jarak pada 0°: 60.40 cm
47542666.585817]: Jarak pada 0°: 60.20 cm
47542666.586850]: Jarak pada 0°: 60.40 cm
47542666.696817]: Jarak pada 0°: 60.20 cm
47542666.699753]: Jarak pada 0°: 60.20 cm
47542666.774252]: Jarak pada 0°: 60.20 cm
47542666.849397]: Jarak pada 0°: 60.20 cm
47542666.918424]: Jarak pada 0°: 60.20 cm
47542666.988700]: Jarak pada 0°: 60.20 cm
47542667.068307]: Jarak pada 0°: 60.40 cm
47542667.188781]: Jarak pada 0°: 60.20 cm
47542667.201771]: Jarak pada 0°: 60.40 cm
47542667.276785]: Jarak pada 0°: 60.40 cm
47542667.352323]: Jarak pada 0°: 60.40 cm
47542667.415541]: Jarak pada 0°: 60.40 cm

```

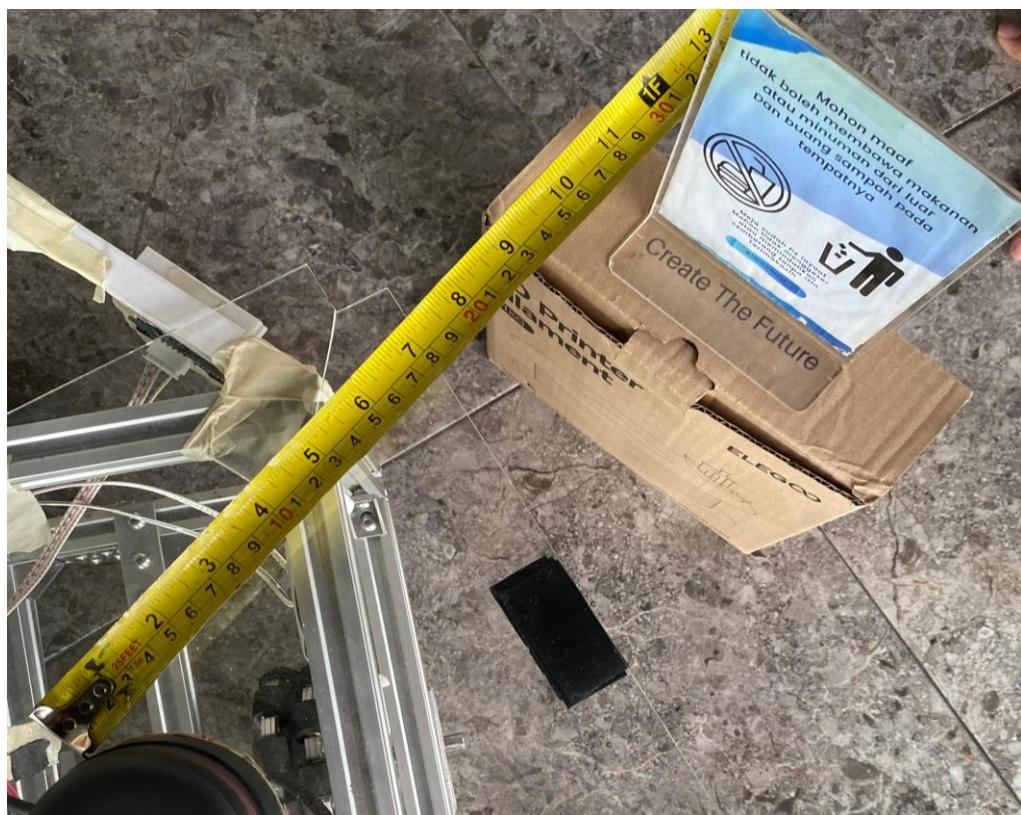
Gambar 5. 4 Hasil pengukuran dengan RPLidar ($0^\circ, 60$ cm)



Gambar 5. 5 Hasil pengukuran dengan alat pengukur manual ($0^\circ, 60$ cm)

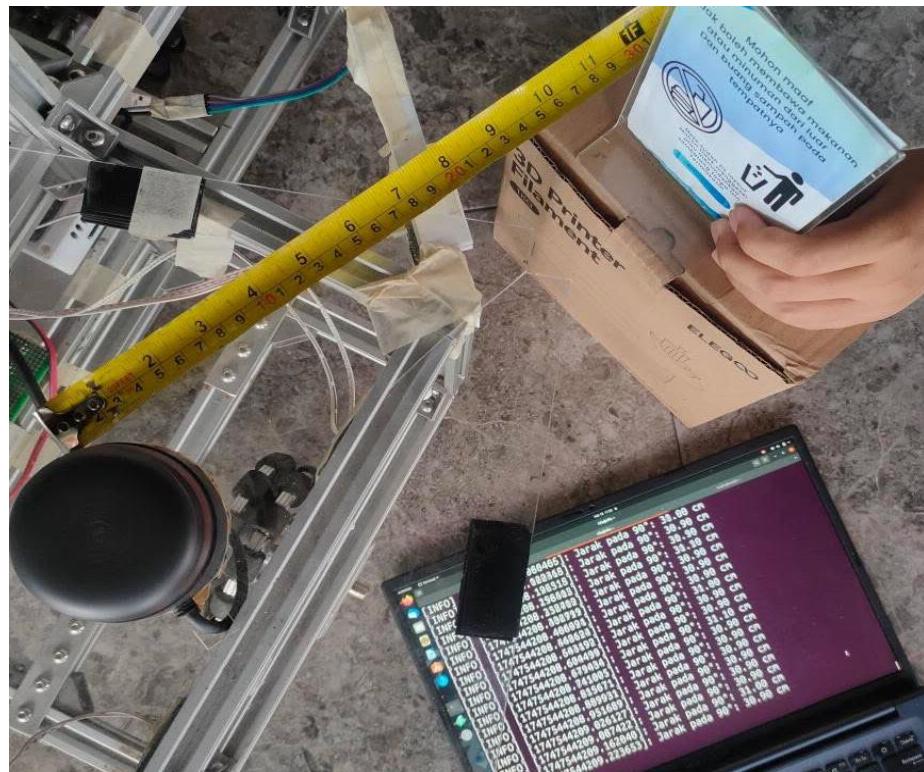
3184.648169]	: Jarak pada 45° : 31.90 cm
3184.711440]	: Jarak pada 45° : 32.30 cm
3184.787196]	: Jarak pada 45° : 32.10 cm
3184.863136]	: Jarak pada 45° : 32.30 cm
3184.926162]	: Jarak pada 45° : 32.30 cm
3185.001908]	: Jarak pada 45° : 32.30 cm
3185.065120]	: Jarak pada 45° : 31.80 cm
3185.140586]	: Jarak pada 45° : 32.30 cm
3185.216391]	: Jarak pada 45° : 32.10 cm
3185.279528]	: Jarak pada 45° : 32.20 cm
3185.355356]	: Jarak pada 45° : 32.30 cm
185.431281]	: Jarak pada 45° : 32.00 cm
185.494169]	: Jarak pada 45° : 32.30 cm
185.569911]	: Jarak pada 45° : 32.30 cm
185.633074]	: Jarak pada 45° : 31.80 cm
185.708533]	: Jarak pada 45° : 32.30 cm
185.784562]	: Jarak pada 45° : 32.10 cm
185.847365]	: Jarak pada 45° : 32.40 cm
185.923267]	: Jarak pada 45° : 32.50 cm
185.986403]	: Jarak pada 45° : 31.80 cm
86.062022]	: Jarak pada 45° : 32.30 cm
86.137975]	: Jarak pada 45° : 32.00 cm
86.200592]	: Jarak pada 45° : 32.40 cm

Gambar 5. 6 Hasil pengukuran dengan RPLidar (45° ,30 cm)



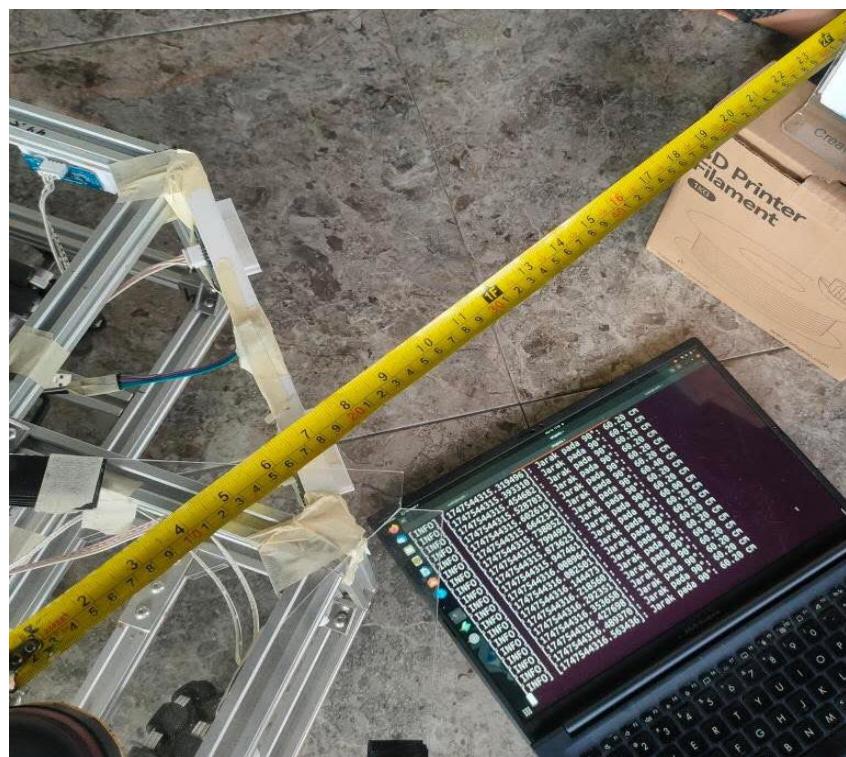
Gambar 5. 7 Hasil pengukuran manual (45° ,30 cm)

- Sudut 90 derajat dengan jarak 30 cm



Gambar 5. 8 Hasil pengukuran ($90^\circ, 30\text{ cm}$)

- Sudut 90 derajat dengan jarak 60 cm



Gambar 5. 9 Hasil pengukuran ($90^\circ, 60\text{ cm}$)

- Sudut 135 derajat dengan jarak 30 cm



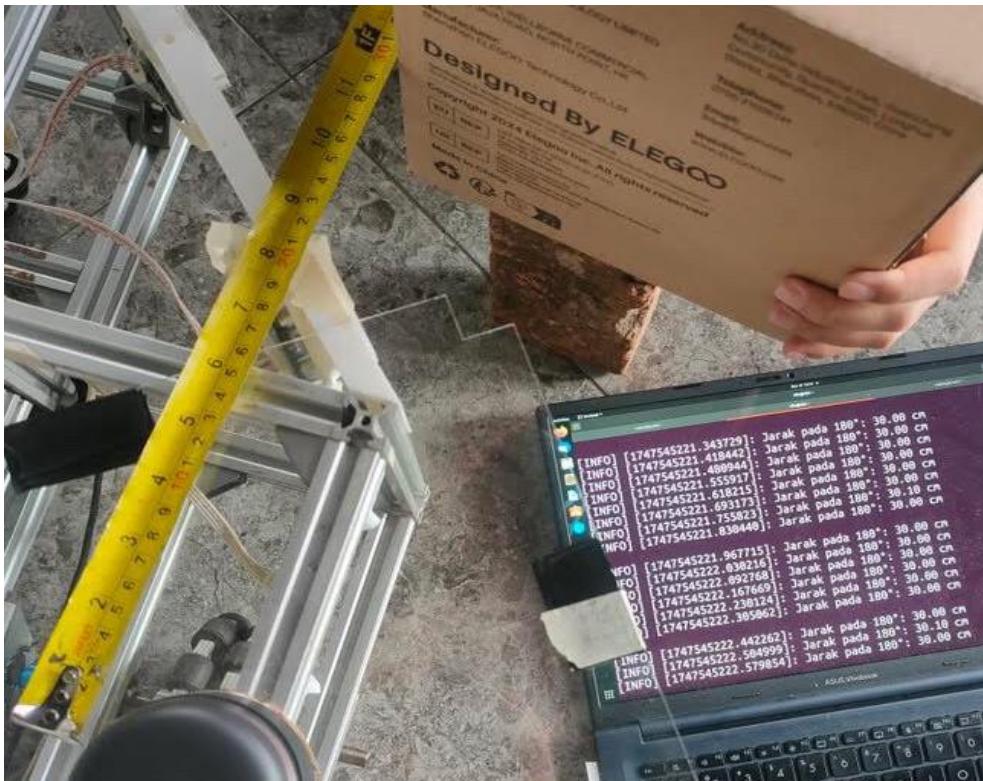
Gambar 5. 10 Hasil pengukuran ($135^\circ, 30\text{ cm}$)

- Sudut 135 derajat dengan jarak 60 cm



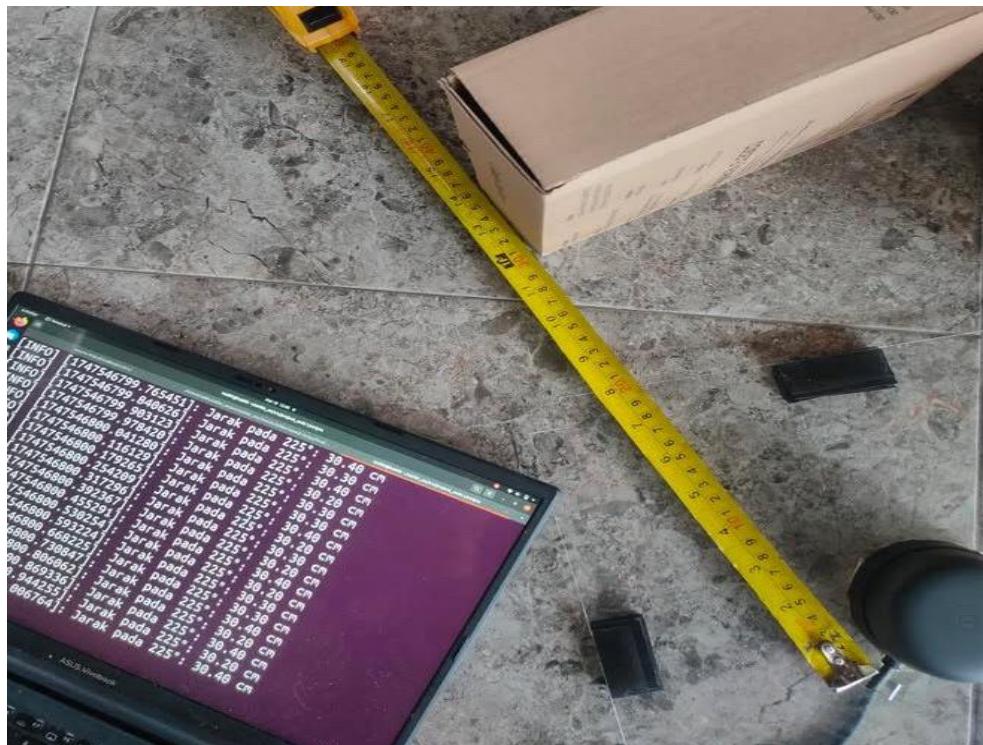
Gambar 5. 11 Hasil pengukuran ($135^\circ, 60\text{ cm}$)

- Sudut 180 derajat dengan jarak 30 cm



Gambar 5. 12 Hasil pengukuran (180° ,30 cm)

- Sudut 225 derajat dengan jarak 30 cm



Gambar 5. 13 Hasil pengukuran (225° ,30 cm)

- Sudut 225 derajat dengan jarak 60 cm

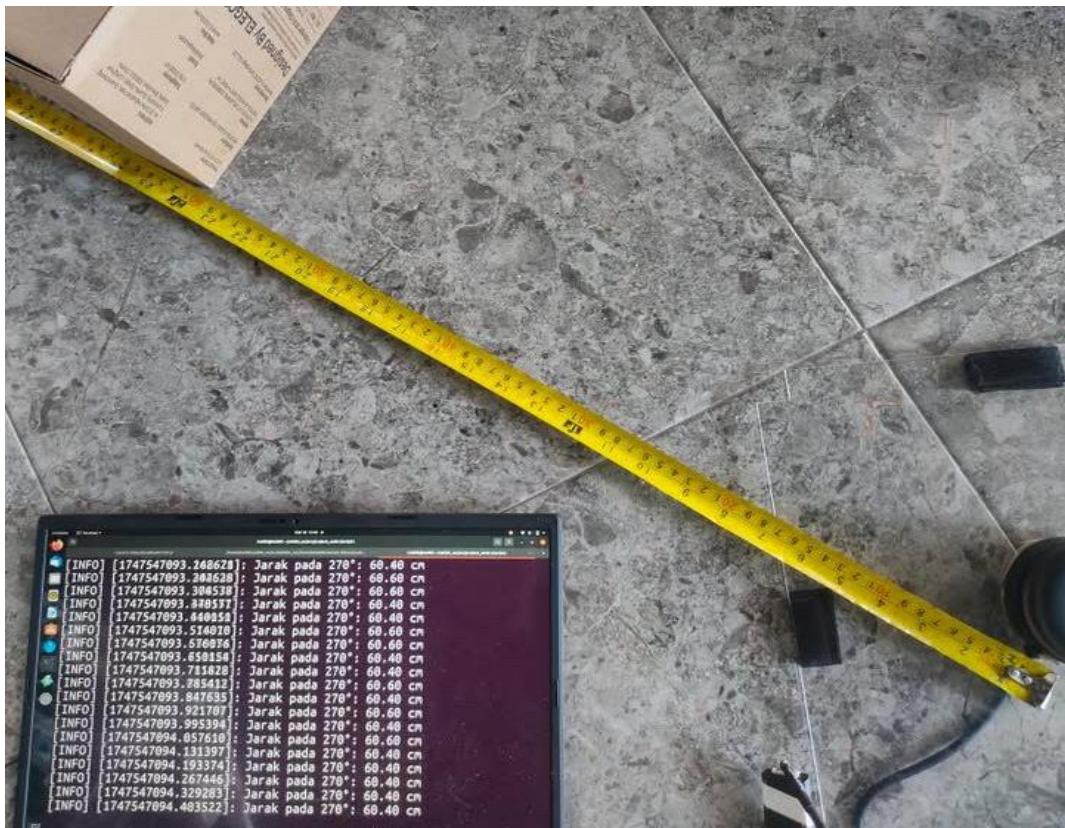


Gambar 5. 15 Hasil pengukuran ($225^\circ, 60$ cm)

- Sudut 270 derajat dengan jarak 30 cm

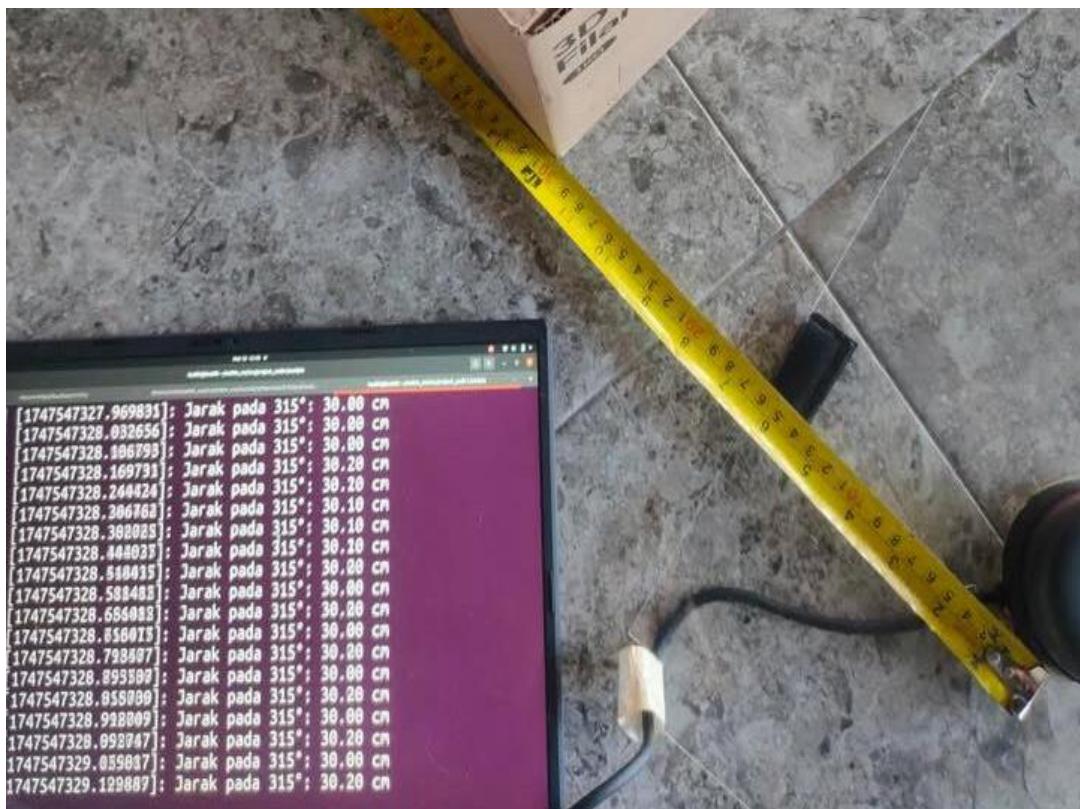


Gambar 5. 14 Hasil pengukuran ($270^\circ, 30$ cm)



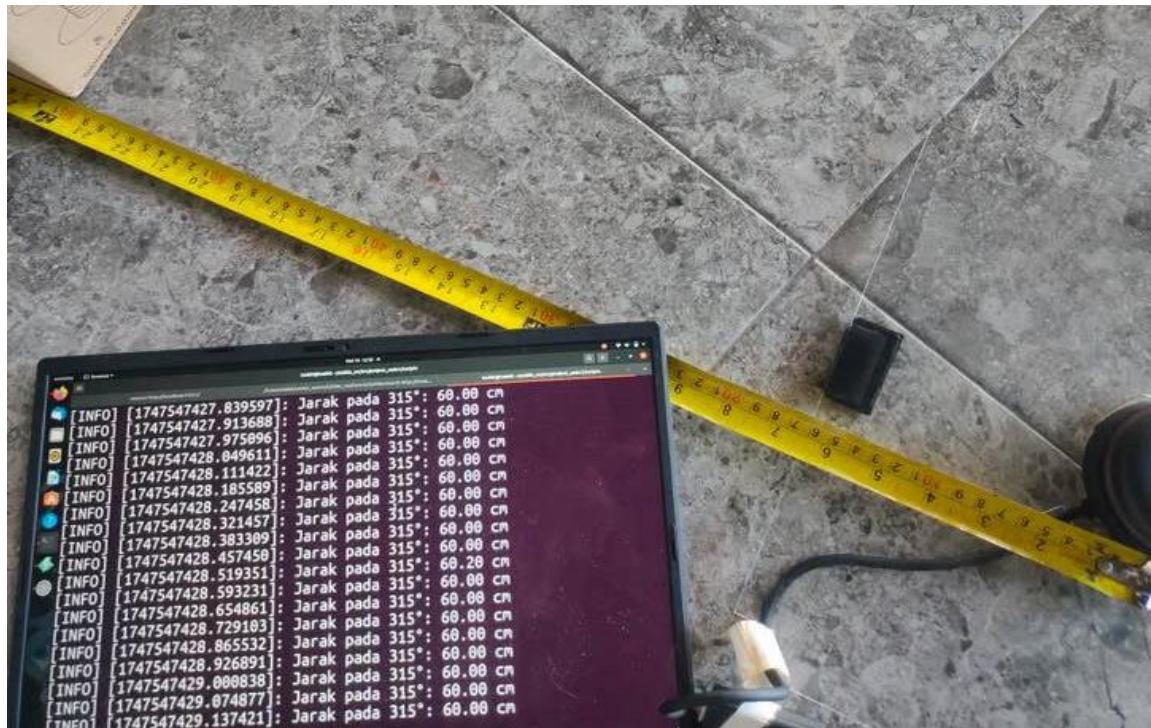
Gambar 5. 16 Hasil pengukuran (270° ,60 cm)

- Sudut 315 derajat dengan jarak 30 cm



Gambar 5. 17 Hasil pengukuran (315° ,30 cm)

- Sudut 315 derajat dengan jarak 60 cm



Gambar 5. 18 Hasil pengukuran ($315^\circ, 60\text{ cm}$)

- Data Tabel Pengukuran Sudut dan Jarak

Tabel 5. 1 Hasil Pengukuran Sudut dan Jarak

No	Sudut (derajat) dan Jarak (cm)							
	0		45		90		135	
	30	60	30	60	30	60	30	60
1	31.10	60.40	31.90	60.40	30.00	60.20	30.90	60.00
2	31.00	60.20	32.30	60.20	30.90	60.20	30.90	60.00
3	31.00	60.20	32.10	60.20	30.90	60.20	30.90	60.40
4	31.10	60.20	32.30	60.20	31.10	60.20	30.90	60.00
5	31.00	60.20	32.10	60.20	30.90	60.20	30.90	60.00
6	30.90	60.20	32.10	60.20	30.90	60.20	30.90	60.00
7	31.10	60.20	32.10	60.20	30.90	60.40	30.90	60.00
8	31.00	60.20	32.10	60.20	31.00	60.20	30.90	60.00
9	30.90	60.40	32.10	60.40	30.90	60.20	30.90	60.00

- Sudut 45 derajat dengan jarak 2 m

```
[INFO] [1751374853.094826]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374853.156120]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374853.229624]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374853.291665]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374853.364767]: Sudut 45.0°: Jarak = 2.00 meter
[INFO] [1751374853.426455]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374853.499650]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374853.561922]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374853.634698]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374853.696445]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374853.770158]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374853.832353]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374853.905679]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374853.966644]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374854.028911]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374854.101598]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374854.162867]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374854.236921]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374854.299253]: Sudut 45.0°: Jarak = 2.00 meter
[INFO] [1751374854.372005]: Sudut 45.0°: Jarak = 2.00 meter
[INFO] [1751374854.433325]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374854.506899]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374854.568288]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374854.642112]: Sudut 45.0°: Jarak = 2.00 meter
[INFO] [1751374854.703618]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374854.778694]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374854.838477]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374854.912886]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374854.973591]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374855.047662]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374855.109454]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374855.182368]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374855.243688]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374855.305359]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374855.378739]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374855.440408]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374855.515862]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374855.575819]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374855.648972]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374855.710684]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374855.784045]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374855.845713]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374855.919546]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374855.981376]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374856.054553]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374856.116252]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374856.189892]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374856.251384]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374856.324893]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374856.386404]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374856.447827]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374856.521285]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374856.582769]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374856.655224]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374856.718237]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374856.791997]: Sudut 45.0°: Jarak = 2.01 meter
[INFO] [1751374856.853187]: Sudut 45.0°: Jarak = 2.01 meter
```

Gambar 5. 20 Hasil Pengukuran Jarak 45°

- Sudut 90 derajat dengan jarak 2 m

```
[INFO] [1751374907.631844]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374907.693181]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374907.766906]: Sudut 90.0°: Jarak = 2.01 meter
[INFO] [1751374907.828747]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374907.902424]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374907.964015]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374908.038164]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374908.099345]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374908.173075]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374908.234653]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374908.296112]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374908.370002]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374908.431345]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374908.505269]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374908.567251]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374908.640308]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374908.702360]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374908.775640]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374908.837415]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374908.911252]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374908.972789]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374909.046126]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374909.107816]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374909.181227]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374909.243152]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374909.300466]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374909.378148]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374909.439735]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374909.513415]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374909.574856]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374909.648766]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374909.710393]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374909.783897]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374909.845381]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374909.919182]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374909.980619]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374910.054331]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374910.116092]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374910.189457]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374910.251771]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374910.312342]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374910.386280]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374910.447614]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374910.521512]: Sudut 90.0°: Jarak = 2.01 meter
[INFO] [1751374910.583326]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374910.657080]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374910.718235]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374910.792467]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374910.853450]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374910.927280]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374910.989810]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374911.062425]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374911.123826]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374911.198289]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374911.259210]: Sudut 90.0°: Jarak = 2.00 meter
[INFO] [1751374911.320495]: Sudut 90.0°: Jarak = 2.00 meter
```

Gambar 5. 21 Hasil Pengukuran Jarak 90°

- Sudut 135 derajat dengan jarak 2 m

```
[INFO] [1751374976.554615]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374976.628224]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374976.689535]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374976.763279]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374976.824895]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374976.898500]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374976.960161]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374977.034050]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374977.095746]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374977.168972]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374977.239464]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374977.304225]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374977.365729]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374977.427304]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374977.501130]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374977.562740]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374977.636325]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374977.697869]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374977.773431]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374977.833005]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374977.907636]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374977.968325]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374978.042184]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374978.103952]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374978.177338]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374978.238890]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374978.300601]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374978.374579]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374978.435630]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374978.509609]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374978.572697]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374978.645088]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374978.706419]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374978.780108]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374978.841520]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374978.915179]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374978.977826]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374979.050692]: Sudut 135.0°: Jarak = 2.00 meter
[INFO] [1751374979.112115]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374979.173753]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374979.247189]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374979.308889]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374979.382460]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374979.443955]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374979.517897]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374979.579118]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374979.653429]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374979.714622]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374979.788041]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374979.849612]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374979.923791]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374979.985389]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374980.058847]: Sudut 135.0°: Jarak = 2.02 meter
[INFO] [1751374980.119896]: Sudut 135.0°: Jarak = 2.01 meter
[INFO] [1751374980.181531]: Sudut 135.0°: Jarak = 2.02 meter
[INFO] [1751374980.255243]: Sudut 135.0°: Jarak = 2.02 meter
```

Gambar 5. 22 Hasil Pengukuran Jarak 135°

- Sudut 180 derajat dengan jarak 2 m

```
[INFO] [1751375002.493717]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375002.567494]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375002.628958]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375002.702365]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375002.764009]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375002.837885]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375002.899142]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375002.973199]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375003.034236]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375003.107725]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375003.169454]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375003.243331]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375003.305014]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375003.365935]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375003.442943]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375003.501275]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375003.575136]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375003.636228]: Sudut 180.0°: Jarak = 2.01 meter
[INFO] [1751375003.709904]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375003.771452]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375003.845136]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375003.906870]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375003.980474]: Sudut 180.0°: Jarak = 2.01 meter
[INFO] [1751375004.042456]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375004.115937]: Sudut 180.0°: Jarak = 2.01 meter
[INFO] [1751375004.178725]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375004.239251]: Sudut 180.0°: Jarak = 2.01 meter
[INFO] [1751375004.312715]: Sudut 180.0°: Jarak = 2.01 meter
[INFO] [1751375004.373902]: Sudut 180.0°: Jarak = 2.01 meter
[INFO] [1751375004.448127]: Sudut 180.0°: Jarak = 2.01 meter
[INFO] [1751375004.509310]: Sudut 180.0°: Jarak = 2.01 meter
[INFO] [1751375004.583523]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375004.644855]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375004.718778]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375004.779423]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375004.854242]: Sudut 180.0°: Jarak = 1.99 meter
[INFO] [1751375004.915326]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375004.988978]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375005.059184]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375005.111426]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375005.185428]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375005.246953]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375005.320681]: Sudut 180.0°: Jarak = 1.99 meter
[INFO] [1751375005.382223]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375005.457677]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375005.517278]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375005.591291]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375005.652458]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375005.713840]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375005.787695]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375005.849593]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375005.922994]: Sudut 180.0°: Jarak = 2.01 meter
[INFO] [1751375005.984793]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375006.058145]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375006.119489]: Sudut 180.0°: Jarak = 2.00 meter
[INFO] [1751375006.193348]: Sudut 180.0°: Jarak = 2.00 meter
```

Gambar 5. 23 Hasil Pengukuran Jarak 180°

- Sudut 225 derajat dengan jarak 2 m

```
[INFO] [1751375024.854544]: Sudut 225.0°: Jarak = 2.01 meter
[INFO] [1751375024.915791]: Sudut 225.0°: Jarak = 2.01 meter
[INFO] [1751375024.990391]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375025.051501]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375025.126923]: Sudut 225.0°: Jarak = 1.99 meter
[INFO] [1751375025.186745]: Sudut 225.0°: Jarak = 1.99 meter
[INFO] [1751375025.259996]: Sudut 225.0°: Jarak = 1.99 meter
[INFO] [1751375025.322279]: Sudut 225.0°: Jarak = 1.99 meter
[INFO] [1751375025.383451]: Sudut 225.0°: Jarak = 1.99 meter
[INFO] [1751375025.457252]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375025.519327]: Sudut 225.0°: Jarak = 1.99 meter
[INFO] [1751375025.592649]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375025.653940]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375025.728890]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375025.791544]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375025.863100]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375025.926295]: Sudut 225.0°: Jarak = 1.99 meter
[INFO] [1751375026.081277]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375026.059417]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375026.133141]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375026.194785]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375026.256761]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375026.330021]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375026.393021]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375026.465162]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375026.527616]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375026.684872]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375026.663116]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375026.735586]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375026.797813]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375026.875549]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375026.932486]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375026.994519]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375027.069248]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375027.129467]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375027.203075]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375027.263553]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375027.338228]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375027.399062]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375027.472867]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375027.534510]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375027.688856]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375027.669644]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375027.743267]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375027.884782]: Sudut 225.0°: Jarak = 2.01 meter
[INFO] [1751375027.866210]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375027.939834]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375028.081498]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375028.075117]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375028.137416]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375028.218381]: Sudut 225.0°: Jarak = 2.01 meter
[INFO] [1751375028.271754]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375028.345797]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375028.407268]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375028.481004]: Sudut 225.0°: Jarak = 2.00 meter
[INFO] [1751375028.542472]: Sudut 225.0°: Jarak = 2.00 meter
```

Gambar 5. 24 Hasil Pengukuran Jarak 225°

- Sudut 270 derajat dengan jarak 2 m

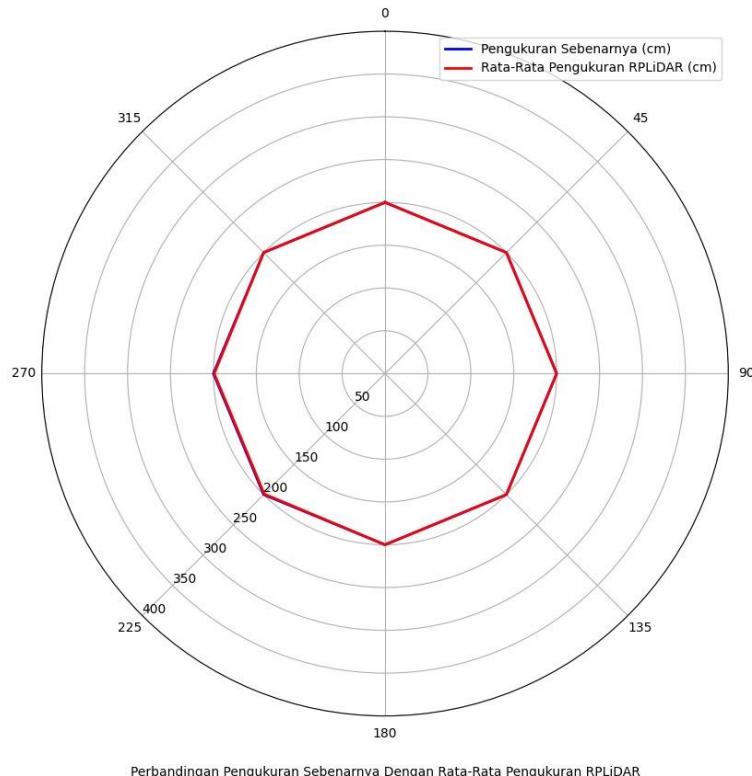
```
[INFO] [1751375045.934775]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375045.996479]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375046.069911]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375046.131336]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375046.205714]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375046.266659]: Sudut 270.0°: Jarak = 1.99 meter
[INFO] [1751375046.342813]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375046.401476]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375046.462918]: Sudut 270.0°: Jarak = 1.99 meter
[INFO] [1751375046.536622]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375046.598949]: Sudut 270.0°: Jarak = 1.99 meter
[INFO] [1751375046.671998]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375046.733823]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375046.806749]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375046.868256]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375046.942005]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375047.003469]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375047.077327]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375047.138720]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375047.212129]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375047.273485]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375047.347362]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375047.409167]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375047.470226]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375047.544388]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375047.605737]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375047.679891]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375047.749302]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375047.814618]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375047.875656]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375047.949793]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375048.010348]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375048.084957]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375048.145522]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375048.206967]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375048.289874]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375048.343793]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375048.415625]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375048.477727]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375048.552111]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375048.612454]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375048.686277]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375048.749172]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375048.821301]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375048.882509]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375048.956200]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375049.017579]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375049.079096]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375049.152916]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375049.214199]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375049.287659]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375049.349299]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375049.422721]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375049.484481]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375049.558130]: Sudut 270.0°: Jarak = 2.00 meter
[INFO] [1751375049.620485]: Sudut 270.0°: Jarak = 2.00 meter
```

Gambar 5. 25 Hasil Pengukuran Jarak 270°

- Sudut 315 derajat dengan jarak 2 m

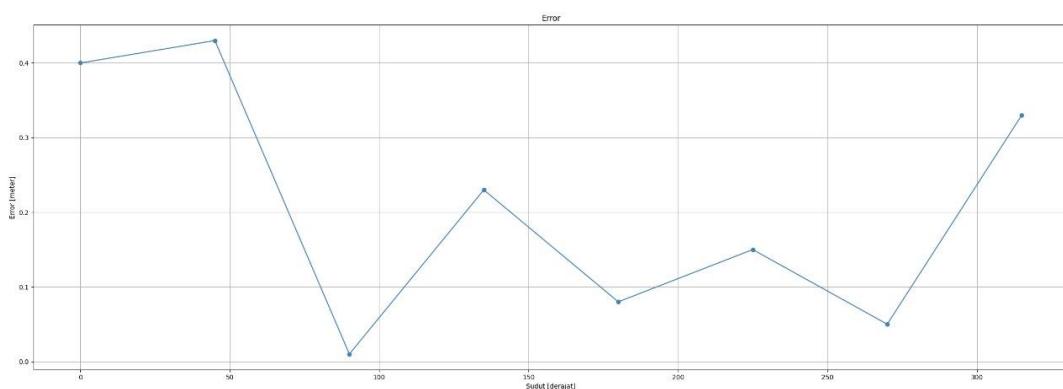
```
[INFO] [1751375075.943569]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375075.184990]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375075.178698]: Sudut 315.0°: Jarak = 2.00 meter
[INFO] [1751375075.240097]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375075.301603]: Sudut 315.0°: Jarak = 2.00 meter
[INFO] [1751375075.375379]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375075.437006]: Sudut 315.0°: Jarak = 2.00 meter
[INFO] [1751375075.510637]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375075.572178]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375075.645298]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375075.707068]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375075.780911]: Sudut 315.0°: Jarak = 2.00 meter
[INFO] [1751375075.842165]: Sudut 315.0°: Jarak = 2.00 meter
[INFO] [1751375075.915655]: Sudut 315.0°: Jarak = 2.00 meter
[INFO] [1751375075.977191]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375076.050709]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375076.112230]: Sudut 315.0°: Jarak = 2.00 meter
[INFO] [1751375076.174818]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375076.247234]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375076.309732]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375076.382561]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375076.444151]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375076.517970]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375076.579305]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375076.652658]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375076.714159]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375076.787538]: Sudut 315.0°: Jarak = 2.00 meter
[INFO] [1751375076.849447]: Sudut 315.0°: Jarak = 2.00 meter
[INFO] [1751375076.923427]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375076.984530]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375077.058025]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375077.119266]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375077.193566]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375077.254339]: Sudut 315.0°: Jarak = 2.00 meter
[INFO] [1751375077.315835]: Sudut 315.0°: Jarak = 2.00 meter
[INFO] [1751375077.389739]: Sudut 315.0°: Jarak = 2.00 meter
[INFO] [1751375077.451400]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375077.524723]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375077.586148]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375077.659671]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375077.721328]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375077.795412]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375077.856477]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375077.930184]: Sudut 315.0°: Jarak = 2.00 meter
[INFO] [1751375077.991363]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375078.065193]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375078.126376]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375078.187782]: Sudut 315.0°: Jarak = 2.00 meter
[INFO] [1751375078.261709]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375078.323557]: Sudut 315.0°: Jarak = 2.00 meter
[INFO] [1751375078.396531]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375078.459670]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375078.532068]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375078.593697]: Sudut 315.0°: Jarak = 2.01 meter
[INFO] [1751375078.666976]: Sudut 315.0°: Jarak = 2.03 meter
[INFO] [1751375078.728740]: Sudut 315.0°: Jarak = 2.04 meter
```

Gambar 5. 26 Hasil Pengukuran Jarak 315°



Gambar 5. 27 Visualisasi perbandingan pengukuran

Gambar ini menunjukkan perbandingan antara jarak sebenarnya dan rata-rata hasil pengukuran RPLiDAR pada berbagai sudut. Garis biru mewakili jarak sebenarnya, sedangkan garis merah menunjukkan hasil dari RPLiDAR. Kedua garis tampak berdekatan, menandakan bahwa RPLiDAR memiliki akurasi tinggi dan konsistensi yang baik dalam mendekripsi jarak di segala arah



Gambar 5. 28 Grafik error Pengukuran

Gambar menampilkan grafik kesalahan (galat) pengukuran RPLiDAR terhadap sudut pengukuran. Sumbu X menunjukkan sudut dalam derajat, sementara sumbu Y menunjukkan besar galat dalam meter. Terlihat bahwa nilai galat bervariasi pada setiap

sudut, dengan galat tertinggi berada di sekitar 0° dan 45° , serta peningkatan kembali mendekati 315° . Meskipun demikian, sebagian besar nilai galat tetap berada di bawah 0,2 meter. Grafik ini menunjukkan bahwa meskipun terdapat fluktuasi, RPLiDAR secara umum masih memberikan hasil yang cukup akurat dengan tingkat kesalahan yang kecil.

Selisih dapat dihitung dengan menggunakan rumus di bawah ini untuk mengetahui seberapa jauh perbedaan antara jarak hasil yang didapat dengan jarak sebenarnya.

$$Selisih = Jarak Hasil - Jarak Sebenarnya \quad (5.2)$$

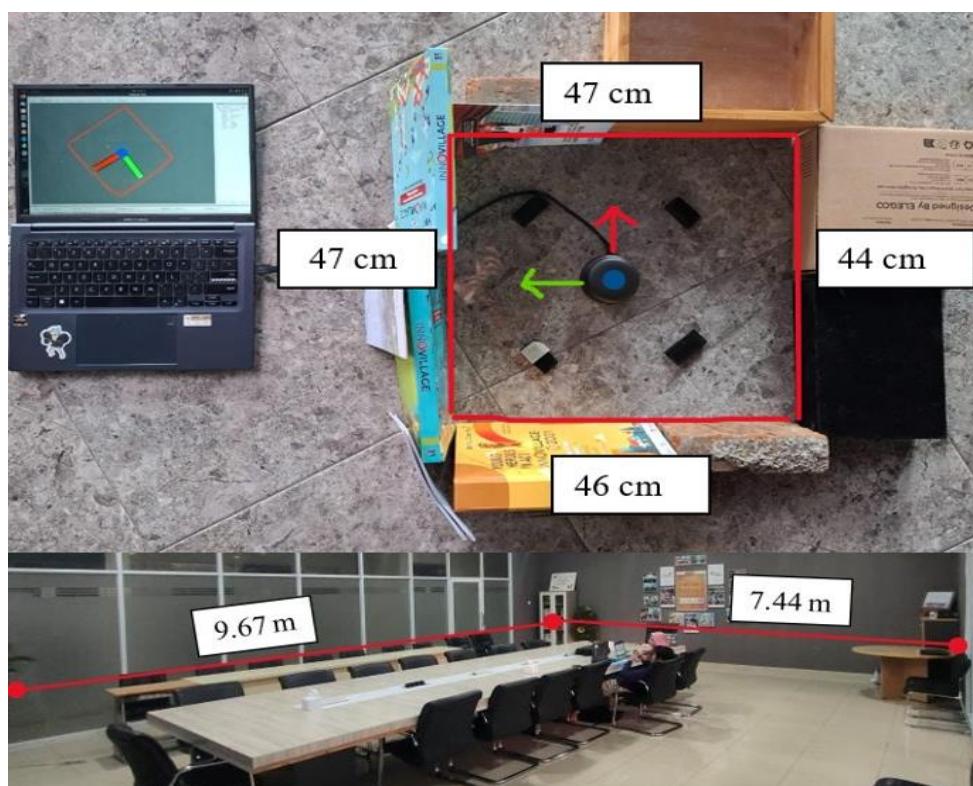
seberapa besar kesalahan dalam bentuk persentase dari nilai sebenarnya.

$$Percentase Error(%) = \frac{Selisih}{Jarak Sebenarnya} \times 100 \quad (5.3)$$

Persentase Akurasi dapat dihitung dengan rumus di bawah ini untuk memberikan angka seberapa tepat hasilnya dibandingkan nilai sebenarnya, jika galat 0% berarti akurasi 100% (hasil sempurna) dan jika galat besar maka akurasi akan rendah.

$$Akurasi(%) = 100\% - Percentase Error(%) \quad (5.4)$$

5.2.2 Hasil Pengujian Mapping



Gambar 5. 29 Hasil pengujian mapping

Pengujian *Mapping* dilakukan untuk memastikan robot mampu membuat peta lingkungan sekitar dengan akurat secara *real-time* menggunakan metode SLAM. Hal ini diperlukan untuk membantu navigasi robot menghindari rintangan dan menentukan jalur tujuan robot dengan otomatis. Hasil *mapping* didapat dari pembacaan sensor RPLidar A2 yang diproses menggunakan SLAM. Pengujian dilakukan didalam arena yang telah dibuat dengan panjang dan lebar yang berbeda sebanyak 5 kali percobaan disetiap panjangnya. Hal ini bertujuan untuk menghasilkan data dalam berbagai kondisi.

- Pengujian *mapping* 1



Gambar 5. 30 pengujian *mapping* 1

- Pengujian *mapping* 2



Gambar 5. 31 pengujian *mapping* 2

- Pengujian mapping 3



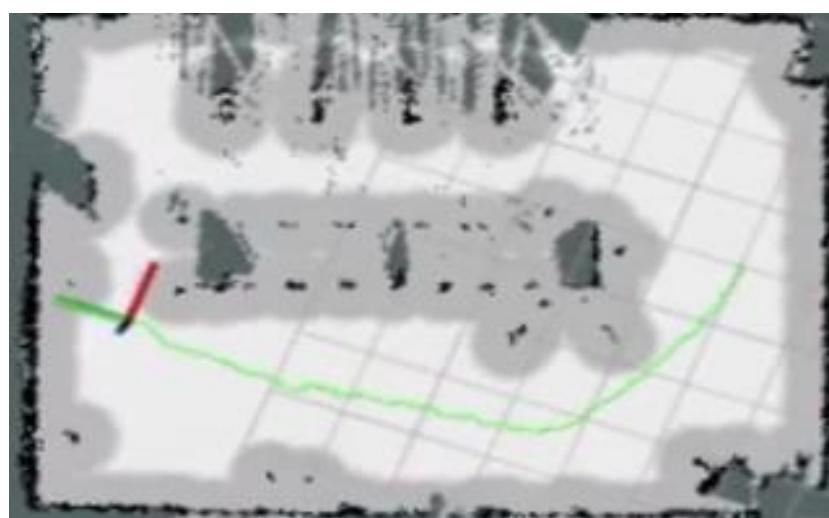
Gambar 5. 32 pengujian mapping 3

- Pengujian mapping 4



Gambar 5. 33 pengujian mapping 4

- Pengujian mapping 5



Gambar 5. 34 Pengujian mapping 5

• **Tabel Hasil Pengambilan Data**

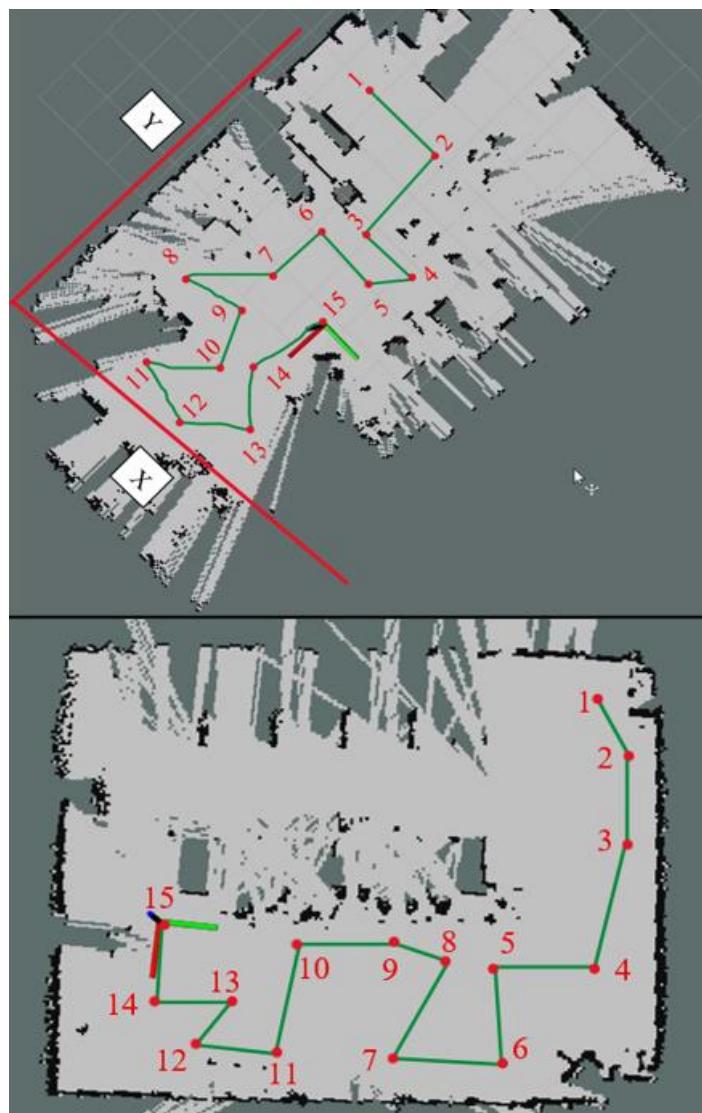
Tabel 5. 13 Hasil Pengujian Mapping

Data ke-	Panjang Simulasi (cm)	Panjang Arena (cm)	Selisih Pengukuran (cm)	Error (%)	Akurasi (%)
1	47.57	47	0.57	1.21	98.79
2	47.51	47	0.51	1.09	98.91
3	47.53	47	0.53	1.13	98.87
4	47.53	47	0.53	1.13	98.87
5	47.54	47	0.54	1.15	98.85
6	44.02	44	0.02	0.05	99.95
7	44.02	44	0.02	0.05	99.95
8	44.01	44	0.01	0.02	99.98
9	44.02	44	0.02	0.05	99.95
10	44.02	44	0.02	0.05	99.95
11	46.52	46	0.52	1.13	98.87
12	46.53	46	0.53	1.15	98.85
13	46.53	46	0.53	1.15	98.85
14	46.53	46	0.53	1.15	98.85
15	46.51	46	0.51	1.11	98.89
16	46.94	47	0.06	0.13	99.87
17	46.95	47	0.05	0.11	99.89
18	46.95	47	0.05	0.11	99.89
19	46.94	47	0.06	0.13	99.87
20	46.94	47	0.06	0.13	99.87
21	968.11	967	1.32	0.14	99.86
22	968.16	967	1.36	0.14	99.86
23	968.17	967	1.37	0.14	99.86
24	968.16	967	1.36	0.14	99.86
25	968.15	967	1.35	0.14	99.86
26	744.89	744	0.89	0.12	99.88
27	744.88	744	0.88	0.12	99.88

28	744.88	744	0.88	0.12	99.88
29	744.88	744	0.88	0.12	99.88
30	744.89	744	0.89	0.12	99.88
Rata-rata Error dan Akurasi			0.45	99.55	

Berdasarkan hasil pengujian *mapping* didapatkan bahwa akurasi robot untuk memetakan lingkungan sekitar mencapai 99.39%. Nilai ini menunjukkan metode yang digunakan untuk memetakan lingkungan sekitar memiliki akurasi yang mendekati lingkungan aslinya. Akurasi *mapping* yang tinggi akan berdampak terhadap peningkatan navigasi yang lebih baik

5.2.3 Hasil Pengujian Lokalisasi



Gambar 5. 35 Simulasi Pengujian Lokalisasi

22	674	518	678	522	4	4	0.59	0.77	99.41	99.23
23	536	614	540	618	4	4	0.74	0.65	99.26	99.35
24	585	512	588	516	3	4	0.51	0.78	99.49	99.22
25	585	327	589	330	4	3	0.68	0.91	99.32	99.09
26	668	315	672	318	4	3	0.60	0.94	99.40	99.06
27	655	215	660	217	5	2	0.76	0.92	99.24	99.08
28	637	233	642	236	5	3	0.78	1.27	99.22	98.73
29	644	174	648	175	4	1	0.62	0.57	99.38	99.43
30	579	175	582	176	3	1	0.52	0.57	99.48	99.43
Rata-rata Error dan Akurasi							0.71	99.29		

Hasil pengujian lokalisasi menunjukkan bahwa estimasi posisi robot memiliki akurasi sebesar 99,29% dibandingkan dengan posisi aktual di dunia nyata. Nilai akurasi yang tinggi ini menunjukkan bahwa metode SLAM mampu menentukan posisi robot secara presisi, sehingga mendukung navigasi yang aman.

5.2.4 Hasil Pengujian Navigasi

Pengujian navigasi dilakukan untuk mengevaluasi efektivitas system navigasi dalam membantu robot menavigasi lintasan menuju tujuan (goal) dengan menghindari rintangan yang ada di lingkungan. Pengujian ini bertujuan untuk memastikan bahwa jalur yang dihasilkan oleh algoritma mampu memberikan jalur yang optimal dan mengurangi potensi menabrak rintangan.

Pengujian sistem navigasi dilakukan pada dua tahap, yaitu melalui simulasi dan uji coba terbatas menggunakan robot nyata di lingkungan sebenarnya. Fokus utama pengujian adalah pada validasi rute hasil kombinasi algoritma K-Nearest Neighbor (KNN) dan Breadth-First Search (BFS), dengan parameter keberhasilan berupa tidak adanya lintasan yang menabrak halangan (obstacle) pada peta lingkungan. Berikut parameter yang digunakan dalam setiap pengujian:

- $k_1 = 30$
- $k_2 = 5$

Berikut hasil pengujian di setiap Skenario:

Skenario 1

Tabel 5. 15 Hasil Pengujian Skenario 1

No	Start – Goal	Panjang Jalur (m)	Jarak terdekat ke obstacle (m)		
			0.0 – 0.40	0.41 – 0.80	0.81 – 1.0
1	(3.0, -3.0) – (7.0, -0.1)	4.94	0%	17%	18%
2	(3.0, -3.0) – (9.4, -2.4)	6.83	0%	12%	40%
3	(3.0, -3.0) – (0.0, 0.0)	4.20	0%	10%	29%
4	(3.0, -3.0) – (11.0, -0.3)	9.10	0%	19%	32%
5	(3, 0) - (4.4, -0.6)	3.44	0%	9%	49%
6	(3.0, -3.0) – (0.2,-2.0)	3.11	0%	32%	14%
7	(3.0, -3.0) – (5.8,-1.1)	3.35	0%	26%	13%
8	(3.0, -3.0) – (7.8,-0.2)	5.58	0%	16%	8%
9	(3.0, -3.0) – (10.0,-2.0)	7.36	0%	47%	12%
10	(3.0, -3.0) – (8.0, -1.0)	5.37	0%	34%	0%
Rata-Rata		5.32	0%	21%	21%

Tabel ini menunjukkan jalur navigasi dari titik (3.0, -3.0) ke berbagai tujuan dengan rata-rata panjang jalur 4.64 meter. Semua lintasan aman dari zona bahaya (0.0–0.40 m) dengan 0% insidensi rentang tersebut. Mayoritas jalur berada dalam jarak sedang ke rintangan (0.41–0.80 m) sebesar 31%, sementara hanya 9% yang berada dalam jarak yang sangat aman (0.81–1.0 m). Hal ini menunjukkan bahwa meskipun robot bergerak lebih dekat ke rintangan dibanding kasus sebelumnya, semua rute tetap aman.

Skenario 2

Tabel 5. 16 Hasil Pengujian Skenario 2

No	Start – Goal	Panjang Jalur (m)	Jarak terdekat ke obstacle (m)		
			0.0 – 0.40	0.41 – 0.80	0.81 – 1.0
1	(0, 0) – (6.6, -2.2)	7.9	0%	18%	21%
2	(0, 0) – (8.0, -0.6)	9.7	0%	22%	21%
3	(0, 0) – (5.6, -0.1)	8.4	0%	6%	25%
4	(0, 0) – (11, -2.0)	11.8	0%	22%	44%
5	(0, 0) – (3.0, -3.0)	4.3	0%	10%	38%
6	(0.0, 0.0) – (10.0, -2.0)	10.98	0%	31%	20%
7	(0.0, 0.0) – (0.2, -2.0)	2.58	0%	0%	83%
8	(0.0, 0.0) – (7.0, -0.1)	8.38	0%	22%	18%
9	(0.0, 0.0) – (4.6, 0.2)	7.53	0%	31%	29%
10	(0.0, 0.0) – (6.25, -2.0)	7.08	0%	33%	20%
Rata-Rata		7.86	0%	18%	28%

Tabel ini menunjukkan hasil evaluasi jalur sistem dari titik awal (0,0) ke berbagai tujuan, dilihat dari panjang lintasan dan seberapa dekat sistem berada pada *obstacle* di sekitarnya. Hasilnya tidak ada satu pun jalur yang memasuki zona (0.0–0.40 m), sistem menjaga jarak aman dari *obstacle*. Sebagian besar jarak terdekat berada dalam rentang 0.41–1.0 meter, dengan dominasi di rentang 0.81–1.0 m (rata-rata 34%), menandakan lintasan tersebut aman.

Skenario 3

Tabel 5. 17 Hasil Pengujian Skenario 3

No	Start – Goal	Panjang Jalur (m)	Jarak terdekat ke obstacle (m)		
			0.0 – 0.40	0.41 – 0.80	0.81 – 1.0
1	(7.0, -0.2) - (9.5, -2.4)	3.16	0%	27%	16%
2	(7.0, -0.2) - (3.0, -3.2)	4.89	0%	19%	15%
3	(7.0, -0.2) – (6.25, -2.0)	1.66	0%	0%	0%
4	(7.0, -0.2) – (0.0, 0.0)	8.25	0%	16%	33%
5	(7.0, -0.2) – (4.6, 0.2)	2.25	0%	0%	56%
6	(7.0, -0.2) – (10.0, -2.0)	3.43	0%	40%	41%
7	(7.0, -0.2) - (8.0, -1.0)	1.16	0%	0%	27%
8	(7.0, -0.2) – (5.8, -1.1)	1.36	0%	0%	0%
9	(7.0, -0.2) - (11, -2.0)	4.43	0%	31%	50%
10	(7.0, -0.2) - (6.0, -2.0)	2.03	0%	0%	0%
Rata-Rata		3.26	0%	10%	24%

Tabel di atas menggambarkan bagaimana sistem menavigasi jalurnya berdasarkan panjang lintasan dan seberapa dekat sistem berada dengan rintangan. Semua jalur yang diuji menunjukkan bahwa sistem menjaga jarak aman karena tidak ada satu pun bagian lintasan yang terlalu dekat (kurang dari 0.4 meter) dengan rintangan. keseluruhan jalur justru berada pada jarak yang aman, terutama di rentang 0.81–1.0 meter, yang artinya sistem mampu menjaga jarak secara stabil dari hambatan di sekitarnya.

Untuk Panjang jalur dan jarak terdekat ke *obstacle* dihitung menggunakan fungsi dari pustaka NumPy dengan perintah

```
distances = np.linalg.norm(points[1:] - points[:-1], axis=1) atau
```

$$distances = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (5.5)$$

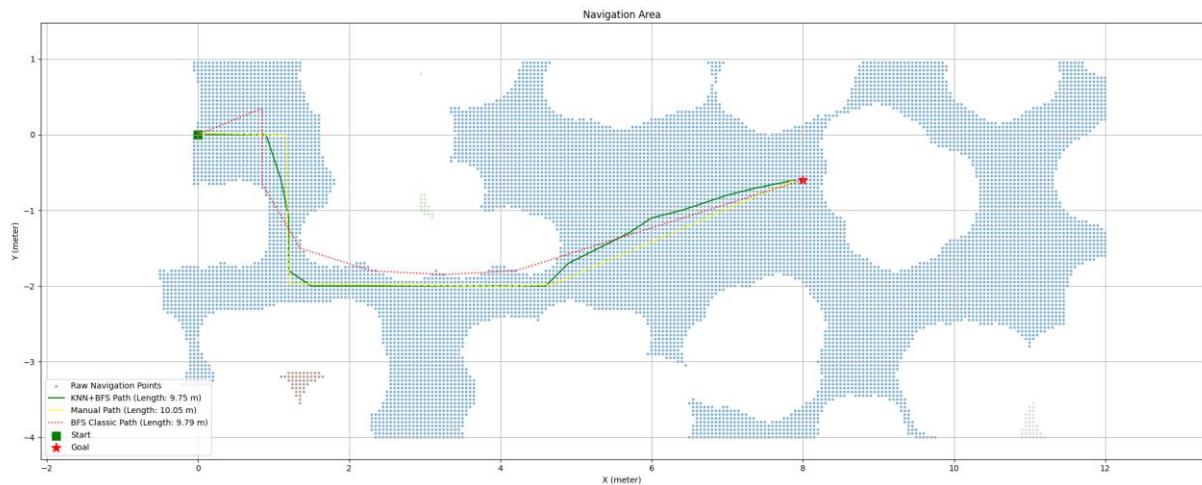
Berikut salah satu hasil pengujian yang diperoleh ditunjukkan melalui perbandingan antara hasil simulasi dan uji coba di lapangan.



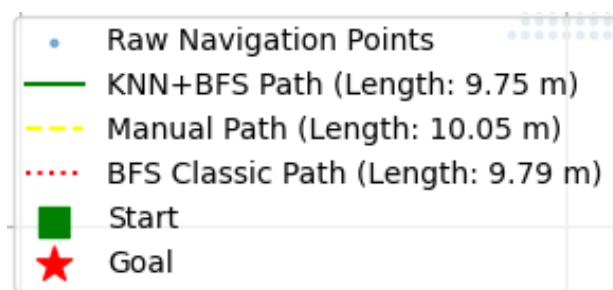
Gambar 5. 36 Simulasi Navigasi

Garis merah menunjukkan rute hasil dari navigasi dan untuk garis hijau menunjukkan lintasan yang dilalui oleh robot. Kami juga melakukan perbandingan antara hasil rute navigasi dari sistem yang kami buat dengan rute navigasi yang dilakukan secara manual dan BFS *Classic*, berikut hasilnya :

Skenario 2 Pengujian 2

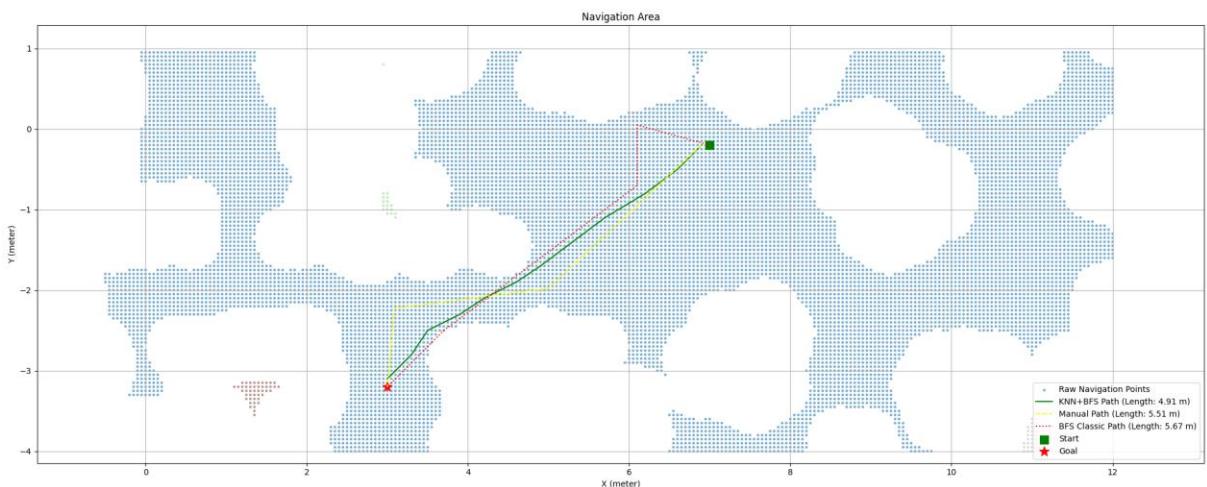


Gambar 5.37 Skenario 2 pengujian simulasi 2

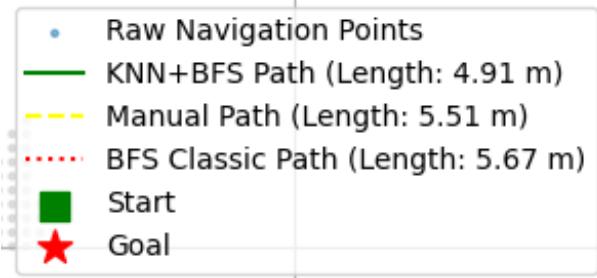


Gambar 5.38 Jarak grafik skenario 2

Skenario 3 Pengujian 2



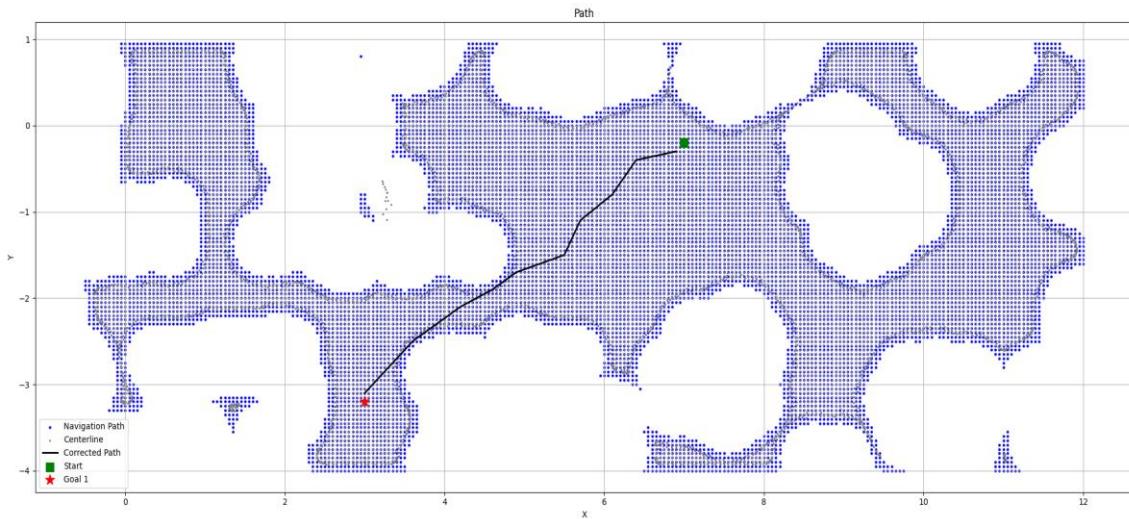
Gambar 5.39 Skenario 3 pengujian simulasi 2



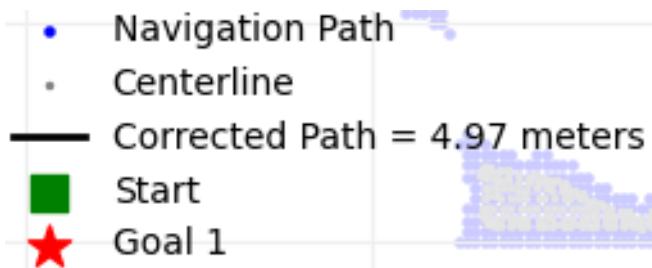
Gambar 5. 40 Jarak grafik skenario 3

Perbandingan hasil menunjukkan bahwa metode K-NN+BFS menghasilkan rute dengan jarak tempuh yang berbeda dari rute manual dan BFS Klasik. Untuk Skenario dan Pengujian lainnya juga menghasilkan hasil yang sama Dimana KNN+BFS lebih cepat untuk komputasi, lebih aman, dan lebih pendek rute yang dipilihnya.

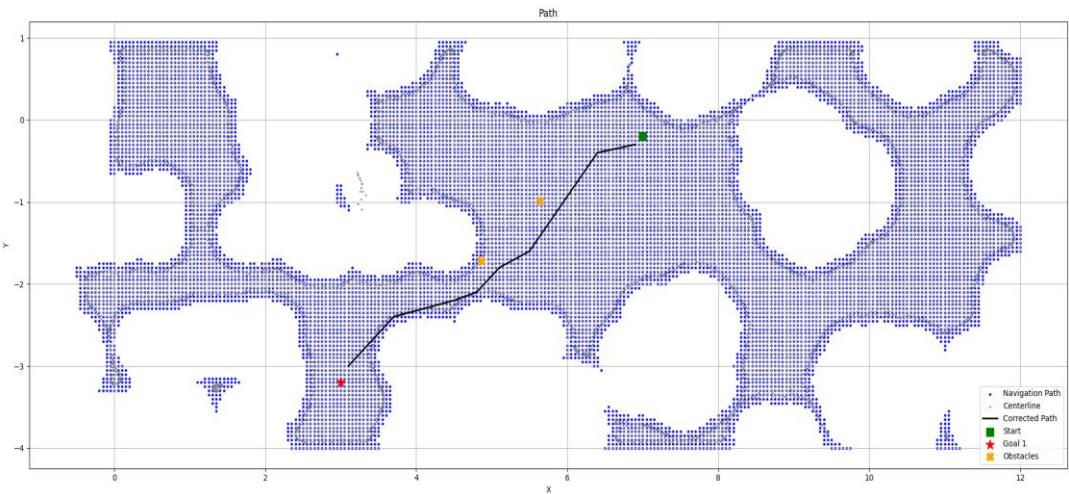
Skenario 3 Pengujian 2 Obstacle



Gambar 5. 41 Skenario 3, Rute penuh



Gambar 5. 42 Jarak grafik rute penuh



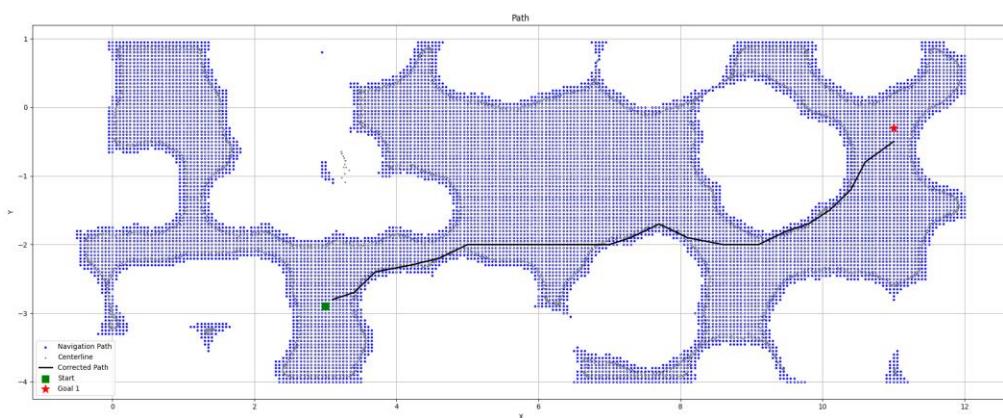
Gambar 5. 43 Skenario 3, 2 obstacle

- Navigation Path
- Centerline
- Corrected Path = 5.05 meters
- Start
- ★ Goal 1
- ✖ Obstacles

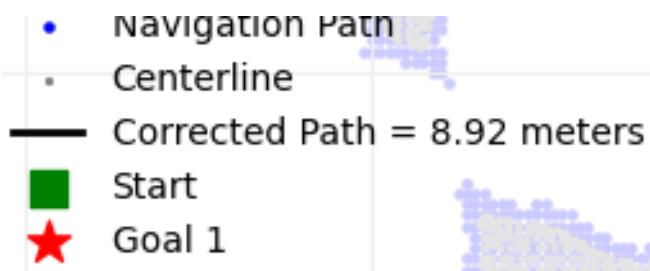
Gambar 5. 44 Jarak grafik rute obstacle

Dari kedua gambar Skenario 3, terlihat bahwa sistem navigasi mampu menyesuaikan jalur dan menghasilkan rute baru untuk menghindari *obstacle*. Meskipun tidak ada perubahan posisi *obstacle*. Pada kondisi dengan **Gambar 5.40**, jalur lurus untuk mencapai tujuan dengan jarak 4.97m tanpa *obstacle* ditambahkan, sedangkan pada **Gambar 5.42** jalur ditambahkan *obstacle* sehingga jaraknya menjadi 5.05m.

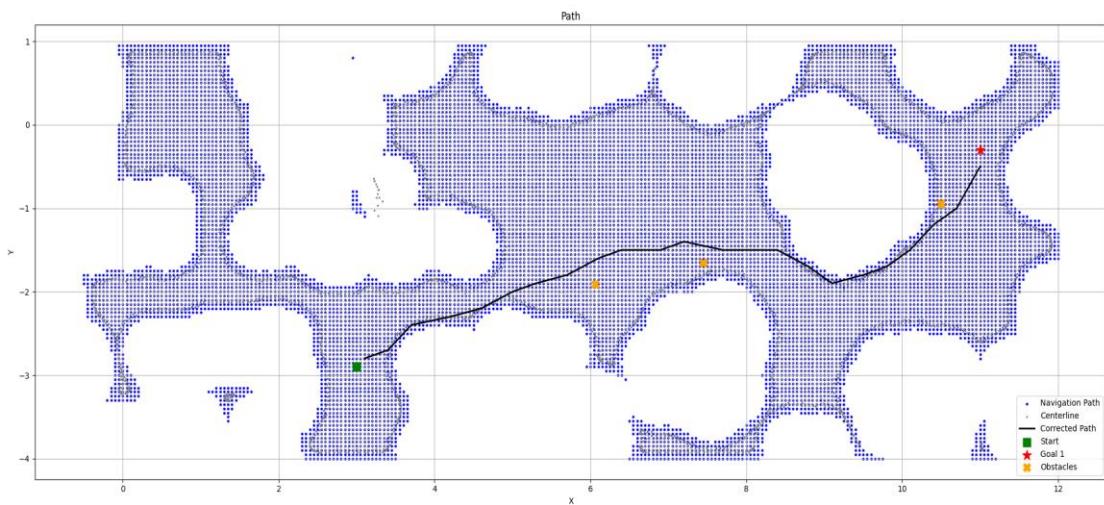
Skenario 1 Pengujian 4 Obstacle



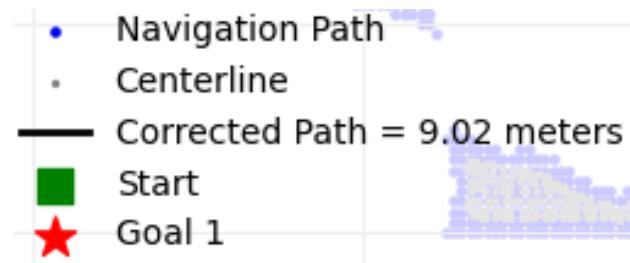
Gambar 5. 45 Skenario 1, 4 Rute penuh



Gambar 5.46 Skenario 1 Jarak rute penuh



Gambar 5.47 Skenario 1, 4 obstacle



Gambar 5.48 Skenario 1 Jarak rute obstacle

Kedua gambar pada Skenario 1 sistem navigasi dalam menghadapi lingkungan dengan tiga *obstacle*. Pada **Gambar 5.44**, merupakan rute penuh dari titik mulai ke tujuan dengan total jarak 8.92 m. Sementara itu, pada **Gambar 5.46**, dengan beberapa *obstacle* sehingga jarak rute bertambah menjadi 9.02 m karena sistem harus mencari rute baru untuk menghindari *obstacle*. Meskipun ditambahkan 3 *obstacle* pada rute, sistem navigasi akan menghasilkan rute baru untuk menghindari *obstacle*.

5.3 Analisa Hasil Pengujian

5.3.1 Analisa Hasil Pengukuran RPLidar

Pengujian dilakukan sebanyak 15x pengujian dengan jarak 30cm dan 60cm dan pada sudut yang berbeda beda. Diperoleh bahwa semakin dekat objek terdeteksi akan mempengaruhi Tingkat akurasi, Tingkat akurasi rata-rata terendah ada di 93.42% pada jarak 30cm.

Pada pengujian ini diperoleh beberapa Faktor Pendukung dan Faktor Penghambat sebagai berikut:

Faktor Pendukung:

- Jangkauan deteksi RPLidar yang jauh membuat pemetaan menjadi luas jangkauannya.
- RPLidar juga memiliki posisioning sensor sehingga dapat menggantikan peran odometri konvensional ditambah keakurasiannya yang mencapai 98%.

Faktor Penghambat:

- Karena RPLidar ini menggunakan laser sehingga ia tidak dapat mendeteksi benda seperti kaca atau benda yang memantulkan Cahaya seperti cermin.
- RPLidar juga tidak bisa melakukan gerakan spontan karena akan membuat pembacaannya kacau.

5.3.2 Analisa Hasil Pengujian *Mapping*

Berdasarkan hasil pengujian yang telah dilakukan, sistem navigasi robot menunjukkan performa yang sangat baik dalam aspek *mapping*. Pada pengujian *mapping*, robot berhasil membangun peta lingkungan dengan akurasi sebesar 99,39%. Angka ini menunjukkan bahwa metode SLAM yang diterapkan mampu merepresentasikan kondisi lingkungan secara akurat dan real-time. Tingkat keberhasilan ini menegaskan bahwa solusi yang digunakan telah mampu menjawab kebutuhan akan pemetaan yang presisi untuk mendukung sistem navigasi robot secara keseluruhan.

Keberhasilan tersebut didukung oleh kualitas sensor RPLIDAR A2 yang memiliki jangkauan dan resolusi memadai, serta stabilitas algoritma SLAM yang digunakan. Selain itu, desain arena uji yang terstruktur turut memberikan kondisi yang ideal bagi proses pemetaan. Meski demikian, terdapat beberapa faktor penghambat seperti kemungkinan gangguan akibat refleksi dari permukaan kaca, guncangan pada robot, serta keterbatasan resolusi peta yang tergantung pada parameter SLAM. Keterbatasan utama dari sistem ini

adalah belum dilakukannya pengujian pada lingkungan dinamis atau kompleks, serta ketergantungan pada sensor tunggal tanpa dukungan visual atau inersial. Oleh karena itu, rencana pengembangan selanjutnya mencakup integrasi sensor tambahan seperti kamera atau IMU, pengujian pada kondisi lingkungan yang lebih bervariasi, serta optimalisasi algoritma SLAM agar lebih efisien dan adaptif terhadap perubahan lingkungan.

5.3.3 Analisa Hasil Pengujian Lokalisasi

Pada pengujian lokalisasi, sistem berhasil memperkirakan posisi robot dengan akurasi sebesar 99,28% terhadap posisi aktual di dunia nyata. Hasil ini menunjukkan bahwa estimasi posisi dari SLAM sangat presisi dan dapat diandalkan untuk mendukung proses navigasi dan pengambilan keputusan secara otonom. Beberapa faktor yang berkontribusi terhadap keberhasilan lokalisasi meliputi kinerja algoritma SLAM yang konsisten, kondisi lingkungan yang stabil, serta sinkronisasi sensor yang baik. Namun, hambatan yang mungkin muncul meliputi kemungkinan terjadinya *drift* posisi dalam jangka waktu tertentu, kurangnya fitur unik dalam lingkungan yang dapat digunakan sebagai referensi, serta potensi gangguan akibat pergerakan robot yang tidak stabil.

Keterbatasan lain dari sistem lokalisasi ini adalah belum adanya pembandingan dengan sistem referensi eksternal seperti GPS atau sistem pelacakan visual, serta pengujian yang masih terbatas pada kondisi lingkungan statis. Untuk meningkatkan kemampuan sistem di masa mendatang, pengembangan akan difokuskan pada pengujian dalam kondisi lingkungan dinamis dan kompleks. Selain itu, penggunaan algoritma lokalisasi lanjutan seperti Adaptive Monte Carlo Localization (AMCL) juga menjadi pertimbangan untuk meningkatkan ketelitian estimasi posisi.

5.3.4 Analisa Hasil Pengujian Navigasi

Pengujian dilakukan dalam simulasi dan lingkungan nyata *indoor* dengan hambatan statis. Berdasarkan uji coba pada 3 Skenario dengan masing-masing skenario berisi 5 kali percobaan dengan total 15 kali percobaan dengan variasi titik start dan tujuan, diperoleh bahwa rute yang dihasilkan semuanya berhasil mencapai tujuan, dengan jarak terdekat ke obstacle berada di 0.4 meter.

Pada pengujian ini diperoleh beberapa Faktor Pendukung dan Faktor Penghambat sebagai berikut:

Faktor Pendukung:

- Dataset memiliki kepadatan titik yang mencapai 11439 titik, sehingga memungkinkan KNN membentuk *graph* yang baik.
- BFS dengan prioritas berdasarkan jarak menjamin jalur optimal dalam konteks jarak tempuh antar titik dengan jarak rata-rata terpendek ada di skenario 3 mencapai 4.11 meter.
- Dengan rute yang dekat dengan obstacle pada rentang 0.0-0.40 ada di 0%
- Dengan penggunaan KNN untuk membentuk *graph* konektivitas antara titik-titik, sebagai dasar bagi BFS.

Faktor Penghambat:

- Pada pengujian dengan robot nyata, permukaan lantai yang tidak rata dapat membuat robot *slip* bahkan hingga keluar jalur navigasi yang ditentukan.
- Pada pengujian dengan robot nyata, robot harus dijalankan dari titik 0 saat *mapping* awal.

Namun demikian, dengan dukungan sensor jarak, sistem tetap dapat menghindari hambatan secara dinamis meskipun perencanaan rute tidak berubah secara langsung. Selain itu, sistem saat ini hanya mampu menentukan jalur menuju tujuan berdasarkan peta hasil proses pemetaan awal (*mapping*), sehingga ketergantungan terhadap peta awal masih tinggi.

Sebagai langkah pengembangan berkelanjutan, direncanakan agar sistem navigasi dapat menghasilkan rute yang dapat diperbarui secara dinamis saat robot sedang bergerak. Hal ini bertujuan untuk mengantisipasi munculnya hambatan baru secara tiba-tiba di depan robot, sehingga sistem navigasi menjadi lebih adaptif dan responsif terhadap kondisi lingkungan yang dinamis.

BAB VI

KESIMPULAN

6.1 Kesimpulan

Implementasi sistem navigasi cerdas pada robot pengantar makanan membuktikan bahwa teknologi dapat menyederhanakan tugas-tugas harian secara signifikan. Robot ini dirancang untuk mampu mengenali lingkungan, menghindari rintangan, dan mencapai tujuan secara mandiri dengan memanfaatkan sensor RPLiDAR A2 dan unit pemrosesan Raspberry Pi 4B sebagai pusat kendali. Data yang dikumpulkan dari sensor dianalisis menggunakan algoritma DBSCAN untuk mengenali area aman, serta K-Nearest Neighbor (KNN) dan Breadth-First Search (BFS) untuk menentukan jalur optimal. Proses navigasi dilengkapi dengan teknik smoothing yang membuat pergerakan robot terasa lebih halus dan efisien.

Untuk memastikan semua komponen dalam sistem robot berfungsi dengan baik dan sesuai tujuan, dilakukan serangkaian pengujian menyeluruh. Pengujian ini mencakup empat aspek utama, yaitu pengukuran jarak, pemetaan lingkungan, pelacakan posisi (lokalisasi), serta navigasi otomatis, yang semuanya terintegrasi dengan sensor RPLiDAR dan metode SLAM. Setiap tahap pengujian dirancang untuk mengevaluasi kemampuan sistem dalam bekerja secara akurat, stabil, dan efisien di lingkungan nyata.

Kesimpulan dari hasil pengujian antara lain:

Pengukuran Jarak:

- Sensor RPLiDAR diuji terhadap objek pada jarak 30 cm dan 60 cm di berbagai sudut.
- Hasil pengukuran menunjukkan konsistensi tinggi, dengan selisih nilai yang sangat kecil terhadap jarak aktual, membuktikan keandalan sensor dalam mendeteksi jarak secara presisi.

Pemetaan Lingkungan (SLAM):

- Dilakukan pada arena uji dengan variasi ukuran dan rintangan fisik.
- Sistem mencatat akurasi pemetaan hingga 99,39%, menunjukkan kemampuannya membentuk representasi lingkungan yang mendekati kondisi nyata.

Lokalisasi Posisi:

- Posisi estimasi yang dihasilkan oleh sistem SLAM dibandingkan dengan titik acuan manual.
- Dari 15 titik pengujian, didapatkan tingkat akurasi sebesar 99,28%, yang berarti robot mampu melacak posisinya sendiri secara akurat saat bergerak.

Navigasi Otomatis:

- Dengan memanfaatkan kombinasi KNN dan BFS, robot dapat menganalisis peta, mengenali rintangan, dan mencari jalur terbaik untuk mencapai tujuan.
- Sistem menunjukkan pergerakan stabil dengan kecepatan operasional $\pm 0,3$ m/s, dan mampu beradaptasi terhadap perubahan lingkungan, tanpa melewati batas kecepatan maksimum 2 m/s.

Secara keseluruhan, sistem navigasi yang dikembangkan menunjukkan kinerja yang efisien, baik dalam hal pengolahan data sensor, pemetaan, pelacakan posisi, maupun pelaksanaan navigasi secara mandiri. Meskipun masih terdapat beberapa keterbatasan seperti kesulitan dalam mendeteksi objek transparan, sistem ini telah terbukti memenuhi tujuan perancangan dan layak untuk diaplikasikan dan dikembangkan lebih lanjut. Dengan tingkat akurasi tinggi, kemampuan beradaptasi di lingkungan nyata, serta sistem dalam menjalankan tugas navigasi secara otonom, robot pengantar makanan ini dinilai bisa digunakan sebagai solusi teknologi dalam industri jasa makanan. Sistem ini tidak hanya berpotensi meningkatkan efisiensi operasional, tetapi juga menghadirkan inovasi berbasis teknologi yang praktis untuk kebutuhan di dunia nyata.

6.2 Saran

Berdasarkan hasil pengujian sistem navigasi robot pengantar makanan, ada beberapa arah pengembangan yang dapat dilakukan ke depan. Pertama, untuk mengatasi kelemahan dalam mendeteksi objek transparan atau reflektif, sistem dapat dilengkapi dengan sensor tambahan atau ditingkatkan melalui pengolahan citra yang lebih canggih. Selain itu, penggantian perangkat seperti prosesor ke versi yang lebih kuat dapat mempercepat kinerja sistem secara *real-time*.

Dari sisi algoritma, metode navigasi klasik seperti KNN dan BFS bisa ditingkatkan menggunakan pendekatan pembelajaran seperti deep learning atau reinforcement learning agar robot bisa beradaptasi dan belajar dari pengalaman. Pengujian juga sebaiknya diperluas ke area nyata seperti restoran atau ruang publik, agar sistem dapat beradaptasi dengan kondisi lingkungan yang lebih kompleks.

Untuk mendukung keberlanjutan, disarankan menggunakan baterai isi ulang dengan energi terbarukan, manajemen daya efisien, dan material ramah lingkungan. Pengembangan sistem *multi-robot* dan aplikasi pemantau berbasis *web* atau *mobile* juga penting untuk efisiensi operasional. Terakhir, aspek keselamatan perlu dijaga melalui desain fisik yang kuat dan skenario darurat untuk memastikan robot tetap aman dalam berbagai situasi.

DAFTAR PUSTAKA

- [1] Marche, “Making robot delivery a reality,” *Marche Mates*, Oct. 2, 2024. [Daring]. Available: <https://marchemates.ca/making-robot-delivery-a-reality>
- [2] ScaleZone Tech, “The power of restaurant automation,” *ScaleZone Technologies*. [Daring]. Available: <https://www.scalezonetech.com/the-power-of-restaurant-automation>
- [3] F. Rifqy, N. Gebby, and R. Muhammad, *Buku Tugas Akhir Capstone Design: Mobilisasi Robot Pengantar Makanan Berbasis Odometry dan QR Detection*, Bandung, Indonesia, Dec. 2023.
- [4] Parenrengi, K. M. (2021). Rancang Bangun Prototype Alat Pengantar Makanan Pada Rumah Makan. *Jurnal MOSFET*, 1(2).
- [5] Nugraha, B. K., Santia, A. S., Aurellia, Z., Pangaribuan, P., & Rodiana, I. M. (2023). Robot Pengantar Makanan berbasis Line Follower dengan Sensor Warna TCS3200 dan Internet of Things (IoT). *JIIP – Jurnal Ilmiah Ilmu Pendidikan*, 6(11), 8928–8933. <https://doi.org/10.54371/jiip.v6i11.2774>
- [6] M. A. Maulana, R. Novian, M. I. Nugraha, and I. M. A. Setiawan, “Autonomous mobile robot dengan menggunakan metode Simultaneous Localization and Mapping berbasis LIDAR,” in *Prosiding Seminar Nasional Inovasi Teknologi Terapan*, Politeknik Manufaktur Negeri Bangka Belitung, 2022.
- [7] Y. Louise, Y. Susanthi, and Muliady, “Mapping dan navigasi untuk robot pengantar makanan di restoran berbasis ROS,” *Techné: Jurnal Ilmiah Elektroteknika*, vol. 22, no. 1, pp. 111–128, 2023.
- [8] SLAMTEC, *RPLIDAR A2M8 datasheet*. [Daring]. Available: https://cdn.sparkfun.com/assets/e/a/f/9/8/LD208_SLAMTEC_rplidar_datasheet_A2M8_v1.0_en.pdf
- [9] MaxBotix Inc., “Ultrasonic Sensor Overview,” [Online]. Available: https://www.maxbotix.com/Articles/Ultrasonic_Sensors_Overview.htm. [Accessed: Jul. 7, 2025].
- [10] Raspberry Pi Trading Ltd., *Raspberry Pi 4 Model B Datasheet*, Release 1.1, Mar. 2024. [Daring]. Available: <https://www.raspberrypi.org/documentation/hardware/>
- [11] Computer Engineering BINUS. (2024, Desember 9). *Apa itu ROS? Sistem operasi untuk robot modern*. BINUS University. <https://comp-eng.binus.ac.id/2024/12/09/apa-itu-ros-sistem-operasi-untuk-robot-modern/>

- [12] Pudu Robotics, “PuduBot: Intelligent Delivery Robot,” *Pudu Robotics Official Website*, 2024. [Online]. Available: <https://www.pudurobotics.com/products/pudubot>
- [13] ResearchGate, “ResearchGate,” [Online]. Available: <https://www.researchgate.net>. [Accessed: 14-Jul-2025].
- [14] I. Ima, “Triangulasi Data: Memahami dan Menerapkannya dalam Penelitian,” Khurslabs.com, Mar. 19, 2025. [Online]. Available: <https://tesis.id/blog/triangulasi-data-memahami-dan-menerapkannya-dalam-penelitian/>
- [15] Dicoding Indonesia, “Machine Learning Adalah: Pengertian, Cara Kerja, dan Contohnya,” *Dicoding Blog*, Sep. 22, 2022. [Online]. Available: <https://www.dicoding.com/blog/machine-learning-adalah/>
- [16] IBM, “K-Nearest Neighbors (KNN),” *IBM Think*, 2024. [Online]. Available: <https://www.ibm.com/id-id/think/topics/knn>
- [17] D. Tamara, “BFS (Breadth First Search): Pengertian, Kekurangan, Kelebihan dan Contohnya,” *Medium*, Mar. 17, 2023. [Online]. Available: <https://medium.com/@defytamara2610/bfs-breadth-first-search-pengertian-kekurangan-kelebihan-dan-contohnya-775a7d808fbc>
- [18] M. Sadowski, “Hands-on with slam_toolbox,” [Online]. Available: https://msadowski.github.io/hands-on-with-slam_toolbox. [Accessed: 14-Jul-2025].
- [19] Desertcart, “Desertcart Uruguay - Online shopping,” [Online]. Available: <https://www.desertcart.uy>. [Accessed: 14-Jul-2025].
- [20] Khurslabs, “Khurs Labs – Robotics & AI Solutions,” [Online]. Available: <https://khurslabs.com>. [Accessed: 14-Jul-2025].
- [21] Husarion Store, “RPLiDAR A2M12 (Slamtec),” [Online]. Available: <https://store.husarion.com/products/rplidar-a2m12>. [Accessed: 14-Jul-2025].
- [22] A. Hestanto, “Perbedaan versi Raspberry Pi,” [Online]. Available: <https://www.hestanto.web.id/perbedaan-dari-versi-raspberry-pi>. [Accessed: 14-Jul-2025].
- [23] ROS Wiki, “sensor_msgs/LaserScan,” [Online]. Available: http://wiki.ros.org/sensor_msgs/LaserScan. [Accessed: Jul. 7, 2025].
- [24] ROS Wiki, “geometry_msgs/PoseStamped,” [Online]. Available: http://wiki.ros.org/geometry_msgs/PoseStamped. [Accessed: Jul. 7, 2025].

- [25] Quigley et al. (2009). ROS: An Open-Source Robot Operating System. ICRA Workshop.
- [26] GeeksforGeeks, “Python PIL | Image.open() method.” [Daring]. Available: <https://www.geeksforgeeks.org/python/python-pil-image-open-method/>
- [27] Scikit-learn, “sklearn.neighbors.NearestNeighbors.” [Daring]. Available: <https://scikitlearn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html>
- [28] FavTutor, “Breadth First Search (BFS) Algorithm in Python,” Jan. 3, 2023. [Daring]. Available: <https://favitutor.com/blogs/breadth-first-search-python>
- [29] GeeksforGeeks, “Euclidean distance in Python.” [Daring]. Available: <https://www.geeksforgeeks.org/math/euclidean-distance/>
- [30] ROS Wiki, “geometry_msgs.” [Daring]. Available: http://wiki.ros.org/geometry_msgs
- [31] ROS Wiki, “sensor_msgs.” [Daring]. Available: http://wiki.ros.org/sensor_msgs
- [32] ROS Wiki, “ROS tutorials.” [Daring]. Available: <http://wiki.ros.org/ROS/Tutorials>
- [33] Scikit-learn, “sklearn.cluster.DBSCAN — scikit-learn documentation.” [Daring]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html>
- [34] Scikit-learn, “sklearn.neighbors.NearestNeighbors — scikit-learn documentation.” [Daring]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html>
- [35] GeeksforGeeks, “heapq in Python.” [Daring]. Available: <https://www.geeksforgeeks.org/python/heap-queue-or-heappq-in-python/>
- [36] GeeksforGeeks, “Breadth-First Search (BFS) for a graph.” [Daring]. Available: <https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>

LAMPIRAN

- **LAMPIRAN 1 : RAB**

No	Nama Barang	Satuan	Jumlah	Harga
1	Raspberry Pi 4B 8GB Kit	Set	1	Rp1.580.000,00
2	RP Lidar A2	Unit	1	Rp3.591.000,00
Jumlah				Rp 5.171.000,00

- **LAMPIRAN 2 : GAMBAR Kode RPLidar A2**

File Launch

```
<launch>
    <node name="rplidarNode"          pkg="rplidar_ros" type="rplidarNode" output="screen">
        <param name="serial_port"        type="string" value="/dev/ttyUSB0"/>
        <param name="serial_baudrate"    type="int"   value="115200"/>
        <param name="frame_id"          type="string" value="laser"/>
        <param name="inverted"          type="bool"  value="false"/>
        <param name="angle_compensate"   type="bool"  value="true"/>
    </node>
</launch>
```

File XML

```
1   <?xml version="1.0"?>
2   <package>
3     <name>rplidar_ros</name>
4     <version>1.5.7</version>
5     <description>The rplidar ros package, support rplidar A2/A1 </description>
6
7     <maintainer email="ros@slamtec.com">Slamtec ROS Maintainer</maintainer>
8     <license>BSD</license>
9
10    <buildtool_depend>catkin</buildtool_depend>
11    <build_depend>roscpp</build_depend>
12    <build_depend>rosconsole</build_depend>
13    <build_depend>sensor_msgs</build_depend>
14    <build_depend>std_srvs</build_depend>
15    <run_depend>roscpp</run_depend>
16    <run_depend>rosconsole</run_depend>
17    <run_depend>sensor_msgs</run_depend>
18    <run_depend>std_srvs</run_depend>
19
20  </package>
```

File cmakelist.txt

```
1  cmake_minimum_required(VERSION 2.8.3)
2  project(rplidar_ros)
3
4  set(RPLIDAR_SDK_PATH "./sdk/")
5
6  FILE(GLOB RPLIDAR_SDK_SRC
7      "${RPLIDAR_SDK_PATH}/src/arch/linux/*.cpp"
8      "${RPLIDAR_SDK_PATH}/src/hal/*.cpp"
9      "${RPLIDAR_SDK_PATH}/src/*.cpp"
10 )
11
12 find_package(catkin REQUIRED COMPONENTS
13     roscpp
14     rosconsole
15     sensor_msgs
16 )
17
18 include_directories(
19     ${RPLIDAR_SDK_PATH}/include
20     ${RPLIDAR_SDK_PATH}/src
21     ${catkin_INCLUDE_DIRS}
22 )
23 |
24 catkin_package()
25
26 add_executable(rplidarNode src/node.cpp ${RPLIDAR_SDK_SRC})
27 target_link_libraries(rplidarNode ${catkin_LIBRARIES})
28
29 add_executable(rplidarNodeClient src/client.cpp)
30 target_link_libraries(rplidarNodeClient ${catkin_LIBRARIES})
31
32 install(TARGETS rplidarNode rplidarNodeClient
33         ARCHIVE DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
34         LIBRARY DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
35         RUNTIME DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
36 )
37
38 install(DIRECTORY launch rviz sdk
39         DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}
40         USE_SOURCE_PERMISSIONS
41     )
```

Referensi: [GitHub - NickL77/RPLidar_Hector_SLAM: Hector SLAM without odometry data on ROS with the RPLidar A1](#)

- **LAMPIRAN 3 : Gambar Kode *Mapping* Menggunakan SLAM**

File Launch

```
1   <?xml version="1.0"?>
2
3   <launch>
4
5     <arg name="geotiff_map_file_path" default="$(find hector_geotiff)/maps"/>
6
7     <param name="/use_sim_time" value="false"/>
8
9     <node pkg="rviz" type="rviz" name="rviz"
10       args="-d $(find hector_slam_launch)/rviz_cfg/mapping_demo.rviz"/>
11
12     <include file="$(find hector_mapping)/launch/mapping_default.launch"/>
13
14     <include file="$(find hector_geotiff)/launch/geotiff_mapper.launch">
15       <arg name="trajectory_source_frame_name" value="scanmatcher_frame"/>
16       <arg name="map_file_path" value="$(arg geotiff_map_file_path)"/>
17     </include>
18
19   </launch>
```

File XML

```
<?xml version="1.0"?>
<package>
  <name>hector_slam_launch</name>
  <version>0.3.5</version>
  <description>
    hector_slam_launch contains launch files for launching hector_slam with different robot systems/setups/postprocessing scenarios.
  </description>

  <!-- One maintainer tag required, multiple allowed, one person per tag -->
  <maintainer email="meyer@fsr.tu-darmstadt.de">Johannes Meyer</maintainer>

  <!-- One license tag required, multiple allowed, one license per tag -->
  <!-- Commonly used license strings: -->
  <!-- BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
  <license>BSD</license>

  <!-- Url tags are optional, but mutiple are allowed, one per tag -->
  <!-- Optional attribute type can be: website, bugtracker, or repository -->
  <url type="website">http://ros.org/wiki/hector_slam_launch</url>

  <!-- Author tags are optional, mutiple are allowed, one per tag -->
  <!-- Authors do not have to be maintainers, but could be -->
  <author email="kohlbrecher@sim.tu-darmstadt.de">Stefan Kohlbrecher</author>

<buildtool_depend>catkin</buildtool_depend>
<run_depend>hector_mapping</run_depend>
<run_depend>hector_map_server</run_depend>
<run_depend>hector_trajectory_server</run_depend>
<run_depend>hector_geotiff</run_depend>
<run_depend>hector_geotiff_plugins</run_depend>

<!-- The export tag contains other, unspecified, tags -->
<export>
  <!-- You can specify that this package is a metapackage here: -->
  <!-- <metapackage/> -->

  <!-- Other tools can request additional information be placed here -->

</export>
</package>
```

File cmakelist.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(hector_slam_launch)

## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
## is used, also find other catkin packages
find_package(catkin REQUIRED)
```

```
catkin_package(
# INCLUDE_DIRS include
# LIBRARIES hector_slam_launch
# CATKIN_DEPENDS hector_map_tools hector_nav_msgs nav_msgs pluginlib roscpp std_msgs
# DEPENDS
)

#####
## Build ##
#####

## Specify additional locations of header files
## Your package locations should be listed before other locations
# include_directories(include)
include_directories(
    ${catkin_INCLUDE_DIRS}
)
```

```
install(DIRECTORY launch/
    DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}/launch/
)

install(DIRECTORY rviz_cfg/
    DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}/rviz_cfg/
)

#####
## Testing ##
#####
```

Referensi : [GitHub - NickL77/RPLidar_Hector_SLAM: Hector SLAM without odometry data on ROS with the RPLidar A1](https://github.com/NickL77/RPLidar_Hector_SLAM)

- **LAMPIRAN 4 : Kode Membaca Area Navigasi**

```
import yaml
import numpy as np
from PIL import Image
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt
import pylab as pl
import pandas as pd
```

Melakukan *Import Library* yang diperlukan.

```
def read_yaml(yaml_file):
    with open(yaml_file, 'r') as f:
        return yaml.safe_load(f)
def read_pgm(pgm_file):
    image = Image.open(pgm_file)
    return np.array(image)
def extract_points(pgm_data, is_black=True):
    condition = (pgm_data == 0) if is_black else (pgm_data != 0)
    return np.column_stack(np.where(condition))
def to_world_coordinates(points, resolution, origin):
    wx = points[:, 1] * resolution + origin[0]
    wy = -(points[:, 0] * resolution + origin[1])
    return np.column_stack((wx, wy))
def cluster_points(points, eps=0.21, min_samples=3):
    return DBSCAN(eps=eps, min_samples=min_samples).fit(points)
```

Melakukan Ekstraksi Point untuk jalur Navigasi.

- *read_yaml(yaml_file)*: Membaca metadata dari file .yaml peta (berisi resolusi dan origin).
- *read_pgm(pgm_file)*: Membaca gambar peta .pgm dan mengubah ke *array numpy*.
- *extract_points(...)*: Mengekstrak titik-titik berwarna hitam (dinding) atau putih (jalur kosong) dari peta.
- *to_world_coordinates(...)*: Mengubah titik dari koordinat piksel ke koordinat dunia berdasarkan resolusi dan origin dari peta.
- *cluster_points(...)*: Mengelompokkan titik dinding menggunakan algoritma DBSCAN berdasarkan jarak (eps) dan jumlah minimal tetangga (*min_samples*).

```
yaml_file = r"..\path\to\mapping_data.bag.yaml"
pgm_file = r"..\path\to\mapping_data.bag.pgm"

metadata = read_yaml(yaml_file)
resolution = metadata['resolution']
origin = metadata['origin']
```

```

pgm_data = read_pgm(pgm_file)
white_points = extract_points(pgm_data, is_black=False)

world_black_points = to_world_coordinates(black_points, resolution,
                                             origin)
world_white_points = to_world_coordinates(white_points, resolution,
                                             origin)

obstacle_clustering = cluster_points(world_black_points)
clustered_obstacle_points=world_black_points[obstacle_clustering.labels_
                                              != -1]

white_clustering=cluster_points(world_white_points,           eps=0.21,
                                 min_samples=5)
clustered_white_points=world_white_points[white_clustering.labels_ != -
                                           1]

x_range = (-1.0, 12.5)
y_range = (-4.0, 2.0)

```

Implementasi Utama dan Visualisasi.

- Memuat file yaml dan pgm.
- Mengubah data peta ke *world coordinates*.
- Melakukan *Clustering* titik yang dianggap *obstacle* dan area navigasi.
- Menentukan Batasan area navigasi dengan range x dan y..

```

mask = (
    (clustered_white_points[:, 0] >= x_range[0]) &
    (clustered_white_points[:, 0] <= x_range[1]) &
    (clustered_white_points[:, 1] >= y_range[0]) &
    (clustered_white_points[:, 1] <= y_range[1])
)
filtered_white_world_points = white_world_points[mask]

path = []
path_set = set()

path_set = set()
for point in filtered_white_world_points:
    distances = np.linalg.norm(clustered_obstacle_points -
                                point, axis=1)
    if np.all(distances > 0.2): # threshold aman
        path_set.add(tuple(point))

path = np.array(list(path_set))

```

Filtering dan penyusunan jalur navigasi.

- Mengekstrak titik putih sebagai area navigasi.
- Memfilter titik berdasarkan area tertentu.

- Mengecek apakah titik putih cukup jauh dari dinding dengan jarak > 0.5m.
- Titik yang memenuhi syarat ditambahkan ke *path_set*.

```

if path_array.size > 0:
    white_clustering = cluster_points(path_array, eps=0.21,
                                         min_samples=3)
    clustered_path = path_array[white_clustering.labels_ != -1]
else:
    clustered_path = np.array([])

if clustered_path.size > 0:
    df_path = pd.DataFrame(clustered_path, columns=['x', 'y'])
    csv_path =
        r"C:\Users\Fernando\Documents\Telkom_University\S1_Teknik_Tele
        komunikasi\TIFA\SVM\FIX\FIX_KEKNYA\FIX_BET\path_points_Nav2.cs
        v"
    df_path.to_csv(csv_path, index=False)
    print(f"CSV file saved: {csv_path}")
else:
    print("No navigation path points found.")

if path_array.size > 0:
    white_clustering = cluster_points(path_array, eps=0.21,
                                         min_samples=3)
    labels = white_clustering.labels_

    clustered_path = path_array[labels != -1]
    clustered_labels = labels[labels != -1]

    mask_range = (
        (clustered_path[:, 0] >= x_range[0]) & (clustered_path[:, 0] <= x_range[1]) &
        (clustered_path[:, 1] >= y_range[0]) & (clustered_path[:, 1] <= y_range[1])
    )
    clustered_path_in_range = clustered_path[mask_range]
    clustered_labels_in_range = clustered_labels[mask_range]

    unique_labels, counts = np.unique(clustered_labels_in_range,
                                      return_counts=True)
    largest_cluster_label = unique_labels[np.argmax(counts)]

    final_path_points =
        clustered_path_in_range[clustered_labels_in_range ==
        largest_cluster_label]

    df_final_path = pd.DataFrame(final_path_points, columns=['x',
        'y'])
    final_csv_path =
        r"C:\Users\Fernando\Documents\Telkom_University\S1_Teknik_Tele
        komunikasi\TIFA\SVM\FIX\FIX_KEKNYA\FIX_BET\final_path_largest_
        cluster.csv"
    df_final_path.to_csv(final_csv_path, index=False)
    print(f"CSV file saved (largest cluster only):
        {final_csv_path}")

else:
    print("Tidak ada titik path yang bisa di-cluster.")

```

Output dan Penyimpanan.

- Menyimpan hasil ke file CSV.
- Jika jalur tersedia digambarkan di peta dengan warna biru.

Visualisasi akhir.

```
● ● ●

1 import yaml
2 import numpy as np
3 from PIL import Image
4 from sklearn.cluster import DBSCAN
5 import matplotlib.pyplot as plt
6 import pandas as pd
7
8 def read_yaml(yaml_file):
9     with open(yaml_file, 'r') as f:
10         return yaml.safe_load(f)
11
12 def read_pgm(pgm_file):
13     image = Image.open(pgm_file)
14     return np.array(image)
15
16 def extract_points(pgm_data, is_black=True):
17     condition = (pgm_data == 0) if is_black else (pgm_data != 0)
18     return np.column_stack(np.where(condition))
19
20 def to_world_coordinates(points, resolution, origin):
21     wx = points[:, 1] * resolution + origin[0]
22     wy = -(points[:, 0] * resolution + origin[1])
23     return np.column_stack((wx, wy))
24
25 def cluster_points(points, eps=0.21, min_samples=3):
26     return DBSCAN(eps=eps, min_samples=min_samples).fit(points)
```

```
● ● ●

1 yaml_file = r"C:\Users\Fernando\Documents\Telkom_University\S1_Teknik_Telekomunikasi\TIFA\SVM\FIX\~telcaf_new_07052025.yaml"
2 pgm_file = r"C:\Users\Fernando\Documents\Telkom_University\S1_Teknik_Telekomunikasi\TIFA\SVM\FIX\~telcaf_new_07052025.pgm"
3
4 metadata = read_yaml(yaml_file)
5 resolution = metadata['resolution']
6 origin = metadata['origin']
7
8 pgm_data = read_pgm(pgm_file)
9 black_points = extract_points(pgm_data, is_black=True)
10 white_points = extract_points(pgm_data, is_black=False)
11
12 world_black_points = to_world_coordinates(black_points, resolution, origin)
13 world_white_points = to_world_coordinates(white_points, resolution, origin)
14
15 obstacle_clustering = cluster_points(world_black_points)
16 clustered_obstacle_points = world_black_points[obstacle_clustering.labels_ != -1]
17
18
19 white_clustering = cluster_points(world_white_points, eps=0.21, min_samples=5)
20 clustered_white_points = world_white_points[white_clustering.labels_ != -1]
21
22 x_range = (-1.0, 12.5)
23 y_range = (-4.0, 2.0)
```

```

1  mask = (
2      (clustered_white_points[:, 0] >= x_range[0]) & (clustered_white_points[:, 0] <= x_range[1]) &
3      (clustered_white_points[:, 1] >= y_range[0]) & (clustered_white_points[:, 1] <= y_range[1])
4  )
5 filtered_white_world_points = clustered_white_points[mask]
6
7 path_set = set()
8 for point in filtered_white_world_points:
9     distances = np.linalg.norm(clustered_obstacle_points - point, axis=1)
10    if np.all(distances > 0.2): # threshold aman
11        path_set.add(tuple(point))
12
13 path = np.array(list(path_set))
14
15 df_black = pd.DataFrame(clustered_obstacle_points, columns=['x', 'y'])
16 csv_black = r'C:\Users\Fernando\Documents\Telkom_University\S1_Teknik_Telekomunikasi\TIFA\SVM\FIX\FIX_KENYA\FIX_BET\black_Nav2.csv'
17 df_black.to_csv(csv_black, index=False)
18 print(f"CSV file saved: {csv_black}")
19
20 if path.size > 0:
21     df_path = pd.DataFrame(path, columns=['x', 'y'])
22     csv_path = r'C:\Users\Fernando\Documents\Telkom_University\S1_Teknik_Telekomunikasi\TIFA\SVM\FIX\FIX_KENYA\FIX_BET\path_points_Nav2.csv'
23     df_path.to_csv(csv_path, index=False)
24     print(f"CSV file saved: {csv_path}")
25 else:
26     print("No navigation path points found.")

```

```

1 plt.figure(figsize=(10, 8))
2 plt.scatter(world_black_points[:, 0], world_black_points[:, 1], c=obstacle_clustering.labels_, cmap='tab20', s=1, alpha=0.2, label="Raw Obstacle")
3 plt.scatter(clustered_obstacle_points[:, 0], clustered_obstacle_points[:, 1], color='red', s=2, label="Clustered Obstacle")
4 plt.scatter(path[:, 0], path[:, 1], color='blue', s=6, label="Navigable Path")
5 plt.title("Path Planning with DBSCAN-based Clustering")
6 plt.xlabel("X (meter)")
7 plt.ylabel("Y (meter)")
8 plt.legend()
9 plt.axis('equal')
10 plt.grid(True)
11 plt.show()
12
13 print(f"Jumlah titik navigasi: {len(path)}")

```

- Kode Navigasi

```

#!/usr/bin/env python3
import rospy
import numpy as np
import pandas as pd
from geometry_msgs.msg import PoseStamped
from sensor_msgs.msg import LaserScan
from geometry_msgs.msg import Point
from visualization_msgs.msg import Marker
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
from scipy.spatial import KDTree
from std_msgs.msg import Float32MultiArray
import heapq
from scipy.spatial import KDTree
from geometry_msgs.msg import Point
from visualization_msgs.msg import Marker
from std_msgs.msg import Float32MultiArray

```

Import Library yang diperlukan.

```

navigation_data = pd.read_csv(r"path_points_Nav.csv")
navigation_points = navigation_data[['x', 'y']].values
def get_clusters(points):
    radius = 2.5
    nbrs = NearestNeighbors(radius=radius).fit(points)
    density = np.array([

```

```

        len(nbtrs.radius_neighbors([pt], return_distance=False)[0])
        for pt in points
    ]).reshape(-1, 1)
X_enhanced = np.hstack((points, density))
dbSCAN = DBSCAN(eps=2, min_samples=6)
labels = dbSCAN.fit_predict(X_enhanced)
cluster_sizes = pd.Series(labels).value_counts().to_dict()
return labels, cluster_sizes

def cluster_weight_func(i, current_pose):
    point = centerline_points[i]
    distance = np.linalg.norm(point - current_pose[:2])
    cid = labels[i]
    if cid != -1 and distance > 3.0:
        return cluster_sizes.get(cid, 1)
    else:
        return 1

```

Membaca file area navigasi dan mengelompokan titik navigasi berdasarkan kepadatan

menggunakan *DBSCAN*, lalu memberikan bobot pada cluster (besar = prioritas)

```

def compute_centerline(points, k=30):
    knn = NearestNeighbors(n_neighbors=k)
    knn.fit(points)
    centerline = []
    for p in points:
        neighbors = knn.kneighbors([p],
                                   return_distance=False)[0]
        centroid = np.mean(points[neighbors], axis=0)
        centerline.append(centroid)
    return np.array(centerline)

```

Perhitungan titik *centerline*.

- Fungsi ini menghitung *centerline* dari Kumpulan titik jalur navigasi dengan mencari *centroid* dari KNN (*K-Nearest Neighbors*) terdekat tiap titik.
- Digunakan untuk memperhalus dan membuat jalur navigasi lebih *centered* di antara area bebas hambatan.

```

def bfs(points, cluster_weight_func, start_idx, goal_idx, dists_all,
       indices_all, current_pose, blocked_indices=set()):
    queue = [(0, start_idx, [start_idx])]

```

```

visited = set()
while queue:
    cost, current_idx, path = heapq.heappop(queue)
    if current_idx == goal_idx:
        return [points[i] for i in path]
    if current_idx in visited or current_idx in blocked_indices:
        continue
    visited.add(current_idx)
    dists = dists_all[current_idx]
    neighbors = indices_all[current_idx]
    for dist, neighbor in zip(dists, neighbors):
        if neighbor not in visited and neighbor not in
            blocked_indices:
            weight = cluster_weight_func(neighbor, current_pose)
            new_cost = cost + dist / weight
            new_path = path + [neighbor]
            heapq.heappush(queue, (new_cost, neighbor, new_path))
print("[WARN] No path found!")
return []

```

Path finding dengan BFS (*Breadth-First Search*) berbasis KNN.

- Fungsi ini mencari jalur dari titik awal ke titik tujuan berdasarkan antrian, dengan pendekatan *Breadth-First Search* (BFS) menggunakan hasil KNN (*K-Nearest Neighbors*) dan Memilih rute berdasarkan jarak terpendek.
- Jalur yang dihasilkan bersifat *graph-based path* dengan asumsi titik-titik saling terhubung jika masuk KNN (*K-Nearest Neighbors*) terdekat.

```

def update_path():
    global smoothed_path, current_target_idx,
    last_known_pose_for_path
    if current_pose is None:
        # print("[WARN] Current pose not set yet!")
        return

    last_known_pose_for_path = np.copy(current_pose)
    # print(f"[INFO] Updating path using pose:
    {last_known_pose_for_path}")

    blocked_indices = set()
    if len(obstacles) > 0:
        obs_tree = KDTree(obstacles)
        for i, point in enumerate(centerline_points):
            if len(obs_tree.query_ball_point(point, r=0.6)) > 0:
                blocked_indices.add(i)
    # print(f"[INFO] {len(blocked_indices)} blocked points
    identified")

    start_idx = tree_center.query(last_known_pose_for_path[:2])[1]
    goal_idx = tree_center.query(goal_point)[1]

    corrected_path = bfs(
        centerline_points,
        cluster_weight_func,
        start_idx,

```

```

        goal_idx,
        dists_all,
        indices_all,
        current_pose=current_pose,
        blocked_indices=blocked_indices
    )

smoothed_path.clear()
prev_point = None
for i in range(0, len(corrected_path), 5):
    sample = corrected_path[i:i+5]
    if len(sample) == 0:
        continue
    avg_x = np.mean(np.array(sample)[:, 0]).round(2)
    avg_y = np.mean(np.array(sample)[:, 1]).round(2)
    current_point = [avg_x, avg_y, 0.0]
    if prev_point is None or
       np.linalg.norm(np.array(current_point) - np.array(prev_point)) > 0.3:
        smoothed_path.append(current_point)
        prev_point = current_point
smoothed_path[:] = np.array(smoothed_path)
# print(f"[INFO] Smoothed path has {len(smoothed_path)} points")
current_target_idx = 1

publish_marker_path()
replan_path_pub.publish(Float32MultiArray(data=[0]))
publish_target_point()

```

KNN+BFS dan *Smoothing* jalur dan menyimpan ke csv.

- Membagi hasil jalur ke dalam segmen kecil (5 titik per segmen).
- Menentukan rute dari fungsi `bfs_and_knn()`
- Setiap segmen diambil rata-rata titiknya untuk menghasilkan jalur yang lebih mulus (*smoothed*).

```

def is_path_blocked():
    if len(smoothed_path) == 0:
        return False
    for point in smoothed_path[3:15]:
        for obs in obstacles:
            if np.linalg.norm(np.array(point[:2]) - obs) < 0.3:
                return True
    return False

def scan_callback(msg):
    global obstacles
    if current_pose is None:
        return

    angle = msg.angle_min
    local_obstacles = []
    for r in msg.ranges:
        if msg.range_min < r < min(msg.range_max, 2.0):
            x = r * np.cos(angle) + current_pose[0]
            y = r * np.sin(angle) + current_pose[1]
            local_obstacles.append(np.array([x, y]))

```

```

        angle += msg.angle_increment
        obstacles = local_obstacles

        if is_path_blocked():
            rospy.loginfo("Obstacle detected! Replanning path...")
            replan_path_pub.publish(Float32MultiArray(data=[1]))
            update_path()

```

Menetapkan radius pemindaian *RPLiDAR* sejauh 2 meter dan memeriksa apakah ada *obstacle* yang dekat dengan titik destinasi dalam radius 0.3 meter.

```

def pose_callback(msg):
    global current_pose, current_target_idx
    current_pose = np.array([msg.pose.position.x, msg.pose.position.y,
                           0.0])

    if len(smoothed_path) == 0:
        return

    target_point = smoothed_path[current_target_idx]
    dist_to_target = np.linalg.norm(current_pose[:2]-target_point[:2])

    if dist_to_target < 0.5:
        if current_target_idx + 1 < len(smoothed_path):
            current_target_idx += 1
            rospy.loginfo(f'Reached point {current_target_idx}, sending
next point...')
            publish_target_point()
        else:
            rospy.loginfo("Goal reached!")

```

Memeriksa posisi robot untuk memastikan apakah robot sudah tiba di titik destinasi atau belum.

```

centerline_points = compute_centerline(navigation_points, k=30)
tree_center = KDTree(centerline_points)
knn = NearestNeighbors(n_neighbors=5)
knn.fit(centerline_points)
dists_all, indices_all = knn.kneighbors(return_distance=True)

labels, cluster_sizes = get_clusters(centerline_points)

current_pose = None
last_known_pose_for_path = None
goal_point = np.array([8.0, -4.0])
obstacles = []
smoothed_path = []
current_target_idx = 1

```

```

marker_pub = None
target_pub = None
replan_path_pub = None
if __name__ == "__main__":
    rospy.init_node("dynamic_path_with_dbSCAN")

    rospy.Subscriber("/slam_out_pose", PoseStamped, pose_callback)
    rospy.Subscriber("/scan", LaserScan, scan_callback)

    marker_pub = rospy.Publisher("/planned_path_marker", Marker,
                                queue_size=10)
    target_pub = rospy.Publisher("/target_point", Float32MultiArray,
                                queue_size=10)
    replan_path_pub = rospy.Publisher("/replan_path", Float32MultiArray,
                                    queue_size=10)

while current_pose is None and not rospy.is_shutdown():
    rospy.loginfo("Waiting for initial pose...")
    rospy.sleep(0.5)

rospy.loginfo("Initial pose received. Generating path...")
update_path()

rospy.spin()

```

Program utama untuk memanggil fungsi lainnya serta menetapkan posisi awal untuk perencanaan pertama

```
● ● ●
1 #!/usr/bin/env python3
2 import rospy
3 import numpy as np
4 import pandas as pd
5 from geometry_msgs.msg import PoseStamped
6 from sensor_msgs.msg import LaserScan
7 from geometry_msgs.msg import Point
8 from visualization_msgs.msg import Marker
9 from sklearn.cluster import DBSCAN
10 from sklearn.neighbors import NearestNeighbors
11 from scipy.spatial import KDTree
12 from std_msgs.msg import Float32MultiArray
13 import heapq
14
15 navigation_data = pd.read_csv(r"path_points_Nav.csv")
16 navigation_points = navigation_data[['x', 'y']].values
17
18 def get_clusters(points):
19     radius = 0.3
20     nbrs = NearestNeighbors(radius=radius).fit(points)
21     density = np.array([
22         len(nbrs.radius_neighbors([pt], return_distance=False)[0])
23         for pt in points
24     ]).reshape(-1, 1)
25     X_enhanced = np.hstack((points, density))
26     dbscan = DBSCAN(eps=2, min_samples=6)
27     labels = dbscan.fit_predict(X_enhanced)
28     cluster_sizes = pd.Series(labels).value_counts().to_dict()
29     return labels, cluster_sizes
30
31 labels, cluster_sizes = get_clusters(navigation_points)
```

```

1  def cluster_weight_func(i):
2      cid = labels[i]
3      return cluster_sizes.get(cid, 1) if cid != -1 else 1
4
5  def compute_centerline(points, k=30):
6      knn = NearestNeighbors(n_neighbors=k)
7      knn.fit(points)
8      centerline = []
9      for p in points:
10          neighbors = knn.kneighbors([p], return_distance=False)[0]
11          centroid = np.mean(points[neighbors], axis=0)
12          centerline.append(centroid)
13      return np.array(centerline)
14
15 def bfs(points, cluster_weight_func, start_idx, goal_idx, dists_all, indices_all, blocked_indices=set()):
16     queue = [(0, start_idx, [start_idx])]
17     visited = set()
18     while queue:
19         cost, current_idx, path = heapq.heappop(queue)
20         if current_idx == goal_idx:
21             return [points[i] for i in path]
22         if current_idx in visited or current_idx in blocked_indices:
23             continue
24         visited.add(current_idx)
25         dists = dists_all[current_idx]
26         neighbors = indices_all[current_idx]
27         for dist, neighbor in zip(dists, neighbors):
28             if neighbor not in visited and neighbor not in blocked_indices:
29                 weight = cluster_weight_func(neighbor)
30                 new_cost = cost + dist / weight
31                 new_path = path + [neighbor]
32                 heapq.heappush(queue, (new_cost, neighbor, new_path))
33     print("[WARN] No path found!")
34     return []
35
36 centerline_points = compute_centerline(navigation_points, k=30)
37 tree_center = KDTree(centerline_points)
38 knn = NearestNeighbors(n_neighbors=5)
39 knn.fit(centerline_points)
40 dists_all, indices_all = knn.kneighbors(return_distance=True)
41
42 current_pose = None
43 last_known_pose_for_path = None
44 goal_point = np.array([6.0, 2.0])
45 obstacles = []
46 smoothed_path = []
47 current_target_idx = 1
48
49 marker_pub = None
50 target_pub = None
51 replan_path_pub = None
52
53 def update_path():
54     global smoothed_path, current_target_idx, last_known_pose_for_path
55     if current_pose is None:
56         # print("[WARN] Current pose not set yet!")
57         return
58
59     last_known_pose_for_path = np.copy(current_pose)
60     # print(f"[INFO] Updating path using pose: {last_known_pose_for_path}")
61
62     blocked_indices = set()
63     if len(obstacles) > 0:
64         obs_tree = KDTree(obstacles)
65         for i, point in enumerate(centerline_points):
66             if len(obs_tree.query_ball_point(point, r=0.3)) > 0:
67                 blocked_indices.add(i)
68     # print(f"[INFO] {len(blocked_indices)} blocked points identified")
69
70     start_idx = tree_center.query(last_known_pose_for_path[:2])[1]
71     goal_idx = tree_center.query(goal_point)[1]
72
73     corrected_path = bfs(
74         centerline_points,
75         cluster_weight_func,
76         start_idx,
77         goal_idx,
78         dists_all,
79         indices_all,
80         blocked_indices=blocked_indices
81     )
82
83     smoothed_path.clear()
84     prev_point = None
85     for i in range(0, len(corrected_path), 5):
86         sample = corrected_path[i:i+5]
87         if len(sample) == 0:
88             continue
89         avg_x = np.mean(np.array(sample)[:, 0]).round(2)
90         avg_y = np.mean(np.array(sample)[:, 1]).round(2)
91         current_point = [avg_x, avg_y, 0.0]
92         if prev_point is None or np.linalg.norm(np.array(current_point) - np.array(prev_point)) > 0.3:
93             smoothed_path.append(current_point)
94         prev_point = current_point
95     smoothed_path[:] = np.array(smoothed_path)
96     # print(f"[INFO] Smoothed path has {len(smoothed_path)} points")
97     current_target_idx = 1
98
99     publish_marker_path()
100    replan_path_pub.publish(Float32MultiArray(data=[0]))
101    publish_target_point()

```

```

1 def is_path_blocked():
2     if len(smoothed_path) == 0:
3         return False
4     for point in smoothed_path[:10]:
5         for obs in obstacles:
6             if np.linalg.norm(np.array(point[:2]) - obs) < 0.3:
7                 return True
8     return False
9
10 def pose_callback(msg):
11     global current_pose, current_target_idx
12     current_pose = np.array([msg.pose.position.x, msg.pose.position.y, 0.0])
13
14     if len(smoothed_path) == 0:
15         return
16
17     target_point = smoothed_path[current_target_idx]
18     dist_to_target = np.linalg.norm(current_pose[:2] - target_point[:2])
19
20     if dist_to_target < 0.5:
21         if current_target_idx + 1 < len(smoothed_path):
22             current_target_idx += 1
23             rospy.loginfo(f'Reached point {current_target_idx}, sending next point...')
24             publish_target_point()
25         else:
26             rospy.loginfo("Goal reached!")
27
28 def scan_callback(msg):
29     global obstacles
30     if current_pose is None:
31         return
32
33     angle = msg.angle_min
34     local_obstacles = []
35     for r in msg.ranges:
36         if msg.range_min < r < min(msg.range_max, 2.0):
37             x = r * np.cos(angle) + current_pose[0]
38             y = r * np.sin(angle) + current_pose[1]
39             local_obstacles.append(np.array([x, y]))
40         angle += msg.angle_increment
41     obstacles = local_obstacles
42
43     if is_path_blocked():
44         rospy.loginfo("Obstacle detected! Replanning path...")
45         replan_path_pub.publish(Float32MultiArray(data=[1]))
46         update_path()

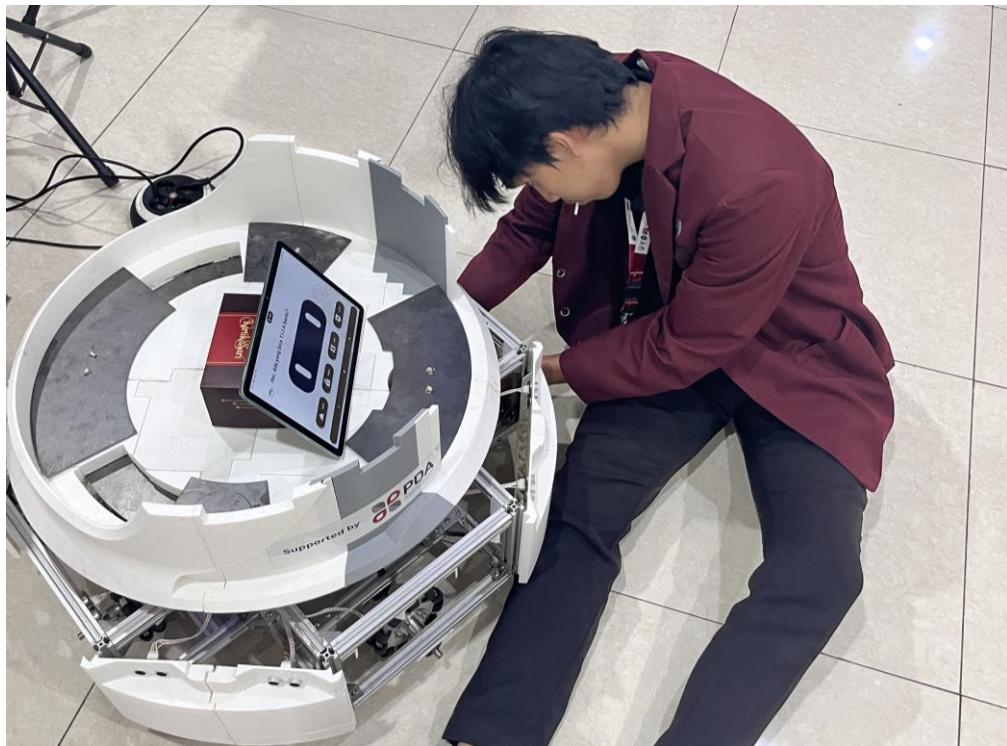
```

```
● ● ●
```

```
1  if __name__ == "__main__":
2      rospy.init_node("dynamic_path_with_dbSCAN")
3
4      rospy.Subscriber("/slam_out_pose", PoseStamped, pose_callback)
5      rospy.Subscriber("/scan", LaserScan, scan_callback)
6
7      marker_pub = rospy.Publisher("/planned_path_marker", Marker, queue_size=10)
8      target_pub = rospy.Publisher("/target_point", Float32MultiArray, queue_size=10)
9      replan_path_pub = rospy.Publisher("/replan_path", Float32MultiArray, queue_size=10)
10
11     while current_pose is None and not rospy.is_shutdown():
12         rospy.loginfo("Waiting for initial pose...")
13         rospy.sleep(0.5)
14
15     rospy.loginfo("Initial pose received. Generating path...")
16     update_path()
17
18     rospy.spin()
19
```

- **LAMPIRAN 5 : Dokumentasi dan Foto Produk**







- **LAMPIRAN 6 : Video Dokumentasi Pengujian**

<https://drive.google.com/drive/folders/1kUH9PtRZrQCoVTurp2ouVG-U8pJby465?usp=sharing>

- **LAMPIRAN 7 : Timeline Revisi Dokumen**

No	Tanggal	Catatan Bimbingan	Pembimbing
1	01-OCT-24	Konsultasi judul yang tepat untuk Tugas Akhir	AHMAD TRI HANURANTO
2	23-NOV-24	Konsultasi terkait penulisan pada Bab 1	AHMAD TRI HANURANTO
3	09-DEC-24	Revisi judul yang kurang tepat dan penambahan spesifikasi produk	SONY SUMARYO
4	16-DEC-24	Revisi Bab 3 terkait jadwal dan anggaran biaya yang digunakan	AHMAD TRI HANURANTO
5	02-JAN-25	Revisi Bab 3 terkait Blok diagram sistem yang masih kurang tepat	AHMAD TRI HANURANTO
6	06-JAN-25	Revisi Bab 3 terkait batasan dan analisa	AHMAD TRI

		masalah yang masih kurang tepat	HANURANTO
7	12-MAR-25	Diskusi terkait algoritma Machine Learning yang tepat untuk navigasi robot	AHMAD TRI HANURANTO
8	14-MAR-25	Revisi terkait blok diagram Machine Learning yang kurang spesifik	SONY SUMARYO
9	19-MAR-25	Diskusi dan Menyampaikan progres terkait perkembangan robot pengantar makanan	AHMAD TRI HANURANTO
10	09-APR-25	Revisi terkait penggunaan Machine Learning yang kurang tepat untuk navigasi robot	AHMAD TRI HANURANTO
11	16-APR-25	Diskusi progres terkait Machine Learning yang sudah siap untuk melakukan pengujian langsung ke robot pengantar makanan	AHMAD TRI HANURANTO
12	21-MAY-25	Revisi Bab 5 terkait pengujian data yang kurang tepat	AHMAD TRI HANURANTO
13	26-MAY-25	Konsultasi melalui Zoom terkait metode pengujian yang tepat untuk robot pengantar makanan pada Bab 5	SONY SUMARYO
14	28-MAY-25	Menyampaikan sedikit progres Bab 5 dan Konsultasi untuk pengujian selanjutnya	AHMAD TRI HANURANTO
15	02-JUN-25	Menyampaikan Progres Bab 5 terkait pengujian dan revisi untuk pengujian yang masih bisa untuk ditambah lagi	AHMAD TRI HANURANTO
16	04-JUN-25	Revisi Bab 5 terkait pengujian untuk berbagai kondisi pada lingkungan Tel U Coffee	AHMAD TRI HANURANTO