



**TUGAS AKHIR - EE 184801**

**PEMETAAN RUANGAN BERBASIS LIDAR  
MENGUNAKAN *MOBILE ROBOT***

Zishwa Muhammad Jauhar Nafis  
NRP 07111540007001

Dosen Pembimbing  
Dr. Muhammad Rivai, ST., MT.

DEPARTEMEN TEKNIK ELEKTRO  
Fakultas Teknologi Elektro  
Institut Teknologi Sepuluh Nopember  
Surabaya 2019





**TUGAS AKHIR - EE 184801**

**PEMETAAN RUANGAN BERBASIS LIDAR  
MENGUNAKAN *MOBILE ROBOT***

Zishwa Muhammad Jauhar Nafis  
NRP 07111540007001

Dosen Pembimbing  
Dr. Muhammad Rivai, ST., MT.

DEPARTEMEN TEKNIK ELEKTRO  
Fakultas Teknologi Elektro  
Institut Teknologi Sepuluh Nopember  
Surabaya 2019





**FINAL PROJECT - EE 184801**

***INDOOR MAPPING MOBILE ROBOT BASED ON LIDAR***

Zishwa Muhammad Jauhar Nafis  
NRP 07111540007001

Supervisor  
Dr. Muhammad Rivai, ST., MT.

ELECTRICAL ENGINEERING DEPARTMENT  
Faculty of Electrical Technology  
Institut Teknologi Sepuluh Nopember  
Surabaya 2019

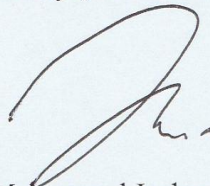


## PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul “Pemetaan Ruang Berbasis LIDAR Menggunakan *Mobile Robot*” adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 1 Juli 2019



Zishwa Muhammad Jauhar Nafis  
NRP 0711 15 4000 7001





# PEMETAAN RUANGAN BERBASIS LIDAR MENGUNAKAN *MOBILE ROBOT*

## TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan  
Untuk Memperoleh Gelar Sarjana Teknik  
Pada

Bidang Studi Elektronika  
Departemen Teknik Elektro  
Fakultas Teknologi Elektro  
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing



Dr. Muhammad Rivai, S.T., M.T.  
NIP. 196904261994031003





# PEMETAAN RUANGAN BERBASIS LIDAR MENGUNAKAN *MOBILE ROBOT*

Nama : Zishwa Muhammad Jauhar Nafis  
Pembimbing : Dr. Muhammad Rivai, ST., MT.

## ABSTRAK

Beberapa variabel dalam dunia robotika seperti lokalisasi, navigasi otomatis, dan juga pencarian objek akan lebih mudah diproses ketika sebuah peta ruangan pada lingkungan robot dapat diketahui. Selain itu peta dapat memberikan informasi bentuk suatu ruangan bagi manusia pada ruangan yang sulit dijangkau. Pada tugas akhir ini akan dibahas pembuatan peta pada sebuah ruangan tertentu dengan menggunakan Light Detection And Ranging (LIDAR) yang berputar sebesar 360°. LIDAR akan dibawa oleh sebuah *mobile robot* untuk menelusuri ruangan. Mobile robot akan bergerak secara manual ataupun otomatis dalam membentuk peta ruangan yang belum diketahui informasinya. Pergerakan robot akan dikontrol dengan menggunakan mikrokontroler arduino nano. Data LIDAR akan diintegrasikan dengan raspberry pi 3b yang kemudian akan dikirim ke laptop untuk divisualisasikan menjadi sebuah peta ruangan. Integrasi komputer dilakukan dalam *Robot Operating System (ROS)*. Pada tugas akhir ini digunakan *package* hector *Simultaneous localization and mapping (SLAM)* dalam melakukan pemetaan ruangan secara 2D serta tools untuk visualisasi Rviz. *Mobile robot* yang dirancang sudah berhasil mengimplementasikan hector SLAM dalam membentuk peta ruangan dan menampilkannya secara *realtime*. Pengujian telah dilakukan dengan menggunakan model ruangan seluas 2,4 m x 2,4 m dengan rata-rata error sebesar 6,45%. Serta pengujian pada ruangan pada gedung Departemen Elektronika B402 ITS dengan hasil peta pada mode manual dan otomatis dengan perbedaan error 0,49%.

Kata kunci : LIDAR, *Mobile Robot*, Pemetaan

.....*Halaman ini sengaja dikosongkan*.....

# **INDOOR MAPPING MOBILE ROBOT BASED ON LIDAR**

*Name* : Zishwa Muhammad Jauhar Nafis  
*Supervisor* : Dr. Muhammad Rivai, ST., MT.

## **ABSTRACT**

*Some variables at robotics such as localization, autonomous navigation, and mission will be easier to process when a room map in the robot environment can be known. The map can provide unknown room information for humans that it is difficult to reach. This final project will discuss room mapping using Light Detection And Ranging (LIDAR) which rotates at 360 °. LIDAR will be carried by a mobile robot to explore the room. Mobile robot will move manually or autonomous to map the room that its information is unknown. The movement of the robot will be controlled using an Arduino nano microcontroller. LIDAR data will be integrated with raspberry pi 3b and sent to a laptop for room mapping visualization. The computer system is based on the Robot Operating System (ROS). In this final assignment, a Hector Simultaneous localization and mapping (SLAM) package is used for 2D space mapping and Rviz visualization tools. The mobile robot designed has successfully implemented Hector SLAM for mapping a room and displaying it in realtime. In this research, a room model 2.4 m x 2.4 and Electrical Department building, B402 ITS are used for test. The average error test of room model is 6.45% and the test result for B402 room conclude that robot can map the room in manual and autonomous mode with a difference of error 0.49%.*

*Keywords: LIDAR, Mapping, Mobile Robot*

.....*Halaman ini sengaja dikosongkan*.....

## KATA PENGANTAR

Segala puji syukur kepada Allah SWT yang telah memberikan nikmat dan karunia-Nya kepada penulis sehingga penulis dapat menyelesaikan laporan tugas akhir dengan judul “**PEMETAAN RUANGAN BERBASIS LIDAR MENGGUNAKAN MOBILE ROBOT**”, sebagai salah satu persyaratan dalam menyelesaikan pendidikan program studi S1 di Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember.

Penulis mengucapkan banyak terima kasih kepada semua pihak yang telah membantu dan memberikan dukungan dalam penulisan dan penyusunan laporan tugas akhir ini. Oleh karena itu, penulis mengucapkan terima kasih yang tulus dan sebesar-besarnya kepada:

1. Dr. Muhammad Rivai, ST., MT. selaku dosen pembimbing yang telah membimbing dan memberikan saran serta arahan selama pengerjaan dan penulisan laporan tugas akhir.
2. Ir. Harris Pirngadi, MT., Fajar Budiman, ST., M.Sc., Ir. Totok Mujiono, Muhammad Attamimi, B.Eng., M.Eng., Ph.d., selaku dosen penguji yang telah memberikan masukan dan arahan selama melakukan revisi.
3. Kepala Departemen Teknik Elektro ITS, Dr. Eng. Ardyono Priyadi, ST., M.Eng. atas izin dan kesempatan yang diberikan kepada penulis untuk melaksanakan tugas akhir ini.
4. Orang tua serta seluruh keluarga yang memberikan dukungan baik moril maupun materiil.
5. Seluruh dosen bidang studi elektronika industri.
6. Teman-teman laboratorium B402 serta laboratorium B202.

Penulis berharap agar tugas akhir ini dapat memberikan manfaat kepada siapapun yang membacanya. Penulis juga menyadari bahwa masih banyak kekurangan dalam penulisan laporan tugas akhir ini. Oleh karena itu penulis menerima setiap kritik dan saran yang diberikan. Akhir kata penulis mengucapkan terima kasih yang sebesar-besarnya.

Surabaya, 1 Mei 2019

Penulis

.....*Halaman ini sengaja dikosongkan*.....



# DAFTAR ISI

PERNYATAAN KEASLIAN .....	<b>Kesalahan! Bookmark tidak ditentukan.</b>
ABSTRAK .....	v
ABSTRACT .....	vii
KATA PENGANTAR .....	ix
DAFTAR ISI .....	xi
DAFTAR GAMBAR .....	xv
DAFTAR TABEL.....	xvii
BAB I PENDAHULUAN .....	1
1.1 Latar Belakang .....	1
1.2 Perumusan Masalah .....	2
1.3 Tujuan Penelitian .....	2
1.4 Batasan Masalah .....	2
1.5 Metodologi Penelitian .....	2
1.6 Sistematika Penulisan .....	4
1.7 Relevansi .....	5
BAB II TINJAUAN PUSTAKA .....	7
2.1 <i>Mobile Robot</i> .....	7
2.2 <i>Light Detection and Ranging (LIDAR)</i> .....	8
2.3 YdLidar X4 .....	9
2.4 Raspberry Pi 3B .....	10
2.5 Mikrokontroler Arduino Nano .....	11
2.6 UBEC 5V 3A .....	13
2.7 Driver Motor DC .....	14
2.8 <i>Robot Operating System (ROS)</i> .....	15
2.8.1 Master.....	15
2.8.2 Node.....	16
2.8.3 Message.....	16
2.8.4 Topic .....	16
2.8.5 Publisher.....	16
2.8.6 Subscriber .....	16
2.8.7 Roscore.....	17
2.8.8 Rosrun .....	17
2.8.9 Roslaunch .....	17
2.9 <i>Simultaneous Localization and Mapping (SLAM)</i> .....	18
2.10 Occupancy Grid .....	18
2.11 <i>Scan Matching</i> .....	19

2.12	Tinjauan Pustaka .....	20
2.12.1	Research and Implementation of SLAM Based on LIDAR for Four-Wheeled Mobile Robot .....	20
2.12.2	Autonomous 2D SLAM and 3D Mapping of an Environment Using a Single 2D LIDAR and ROS .....	21
2.12.3	Obstacle Detection and Avoidance Algorithm for Autonomous Mobile Robot using 2D LiDAR .....	21
<b>BAB III PERANCANGAN SISTEM .....</b>		<b>22</b>
3.1.	Gambaran Umum Sistem .....	23
3.2.	Perancangan Perangkat Keras .....	25
3.2.1	Mekanik <i>Mobile Robot</i> .....	25
3.2.2	Catu Daya .....	26
3.2.3	Pemasangan YdLidar X4 .....	27
3.2.4	Driver Motor .....	28
3.2.5	Sistem Gerak Robot .....	29
3.2.6	Board Mikrokontroler Arduino Nano .....	31
3.3.	Perancangan Perangkat Lunak .....	32
3.4.1	Instalasi <i>Robot Operating System</i> .....	32
3.4.2	Pengambilan Data Sensor YdLidar X4 .....	34
3.4.3	Kontrol Motor DC .....	35
3.4.4	Konfigurasi ROS Master .....	36
3.4.5	Instalasi ROS <i>Package</i> Hector SLAM .....	36
3.4.6	Kontrol Manual Pergerakan Robot .....	38
3.4.7	Kontrol Otomatis Pergerakan Robot .....	39
3.4.	Implementasi Pemetaan Ruang Menggunakan LIDAR .....	42
<b>BAB IV PENGUJIAN DAN ANALISIS .....</b>		<b>45</b>
4.1	Pengujian Sensor YdLidar X4 .....	45
4.2	Pengujian Kontrol Motor DC .....	47
4.3	Pengujian Kecepatan Robot .....	51
4.4	Pengujian Otomatis Telusur Dinding Robot .....	53
4.5	Pengujian Posisi Robot dengan <i>Scan Matching</i> .....	53
4.6	Pengujian Peta <i>Occupancy Grid</i> .....	55
4.7	Pengaruh Kecepatan pada Kontrol Robot dan Pembuatan Peta ....	57
4.8	Pengujian Variasi Bentuk Geometri Ruang .....	59
4.9	Pengujian Pemetaan Ruang .....	60
4.9.1	Ruang buatan .....	60
4.9.2	Ruang Sesungguhnya .....	64
<b>BAB V PENUTUP .....</b>		<b>67</b>
5.1.	Kesimpulan .....	67

5.2. Saran .....	68
DAFTAR PUSTAKA .....	69
LAMPIRAN A .....	72
LAMPIRAN B .....	101
LAMPIRAN C .....	105
LAMPIRAN D .....	107
BIODATA PENULIS .....	109

.....*Halaman ini sengaja dikosongkan*.....

## DAFTAR GAMBAR

Gambar 2.1 Contoh Mobile Robot .....	7
Gambar 2.2 Cara kerja LIDAR ToF. ....	8
Gambar 2.3 Cara kerja LIDAR triangulasi . ....	9
Gambar 2.4 YdLidar X4.....	9
Gambar 2.5 Sistem Perangkat Keras Raspberry Pi 3b.....	10
Gambar 2.6 Raspberry Pi 3b .....	11
Gambar 2.7 Arduino Nano. ....	11
Gambar 2.8 Diagram Alur Perangkat Keras Arduino Nano .....	13
Gambar 2.9 UBEC 5V 3A.....	14
Gambar 2.10 Rangkaian H-Bridge .....	15
Gambar 2.11 Robot Operating System .....	17
Gambar 2.12 Peta dengan occupancy grid .....	19
Gambar 2. 13 Proses scan matching (a) Data koordinat biru akan dicocokkan dengan merah, (b) Hasil scan matching .....	20
Gambar 3.1 Skema Sistem Keseluruhan .....	23
Gambar 3.2 Blok Diagram Sistem.....	24
Gambar 3.3 Perancangan desain robot.....	25
Gambar 3.4 Alur catu daya motor dan Raspberry Pi .....	26
Gambar 3.5 Alur catu daya YdLidar dan Arduino .....	26
Gambar 3.6 Koneksi YdLidar X4 dengan Raspberry Pi.....	27
Gambar 3.7 Pin Adapter YdLidar X4 .....	27
Gambar 3.8 Sudut YdLidar X4 pada robot. ....	28
Gambar 3.9 Wiring Arduino dengan L298 .....	29
Gambar 3.10 Desain penggerak robot.....	30
Gambar 3.11 Desain skematik pada software eagle .....	31
Gambar 3.12 Desain PCB pada Eagle .....	32
Gambar 3.13 Hasil instalasi ROS pada laptop .....	33
Gambar 3.14 Hasil instalasi ROS pada Raspberry Pi 3b .....	33
Gambar 3.15 Blok Diagram PID kecepatan motor.....	36
Gambar 3.16 Alur diagram perintah manual.....	39
Gambar 3.17 Kontrol PID telusur ruangan. ....	40
Gambar 3.18 Pembagian sudut LIDAR .....	40

Gambar 3.19 Alur diagram pergerakan otomatis .....	41
Gambar 3.20 Alur diagram pembuatan peta dengan Hector SLAM .....	44
Gambar 4.1 Realisasi desain robot .....	45
Gambar 4.2. Proses Pengambilan Data Jarak .....	47
Gambar 4.3 Grafik respon pid motor 1.....	48
Gambar 4.4 Grafik Respon PID Motor 2.....	48
Gambar 4.5 Grafik respon pid motor 3.....	49
Gambar 4.6 Grafik PID motor 1 dengan beban .....	50
Gambar 4.7 Grafik PID motor 2 dengan beban. ....	50
Gambar 4.8 Grafik PID motor 3 dengan beban .....	51
Gambar 4.9. Proses Pengujian Kecepatan Robot.....	52
Gambar 4.10 Grafik hubungan konstanta dengan kecepatan .....	52
Gambar 4.11 Grafik PID telusur dinding. ....	53
Gambar 4.12. Proses pengambilan data posisi.....	54
Gambar 4.13. Pengujian Occupancy Grid .....	55
Gambar 4.14 Pengujian Pada Jarak 1 m.....	56
Gambar 4.15 Pengujian Pada Jarak 1,5 m .....	56
Gambar 4.16. Pengujian Pada Jarak 2 m.....	57
Gambar 4. 17 Variasi bentuk geometri ruangan (a) Hasil ruangan bentuk lingkaran (b) Hasil ruangan bentuk segitiga (c) Hasil ruangan bentuk persegi.....	60
Gambar 4.18 Desain ruangan percobaan 1 .....	61
Gambar 4.19. Hasil gambar pemetaan percobaan 1 .....	61
Gambar 4.20. Desain ruangan percobaan 2 .....	62
Gambar 4.21. Hasil gambar pemetaan percobaan 2 .....	63
Gambar 4.22 Hasil peta mode otomatis.....	65
Gambar 4.23 Hasil peta mode manual. ....	65
Gambar 4.24 Ruangn B402.....	66

## **DAFTAR TABEL**

Tabel 2.1 Spesifikasi YdLidar X4 .....	10
Tabel 2.2 Spesifikasi Arduino Nano .....	12
Tabel 3.1 Konfigurasi pin Arduino dengan motor driver. ....	29
Tabel 3.2 Logika kontrol arah motor. ....	35
Tabel 3.3 Parameter hector SLAM. ....	37
Tabel 3.4 Data karakter yang dikirim. ....	38
Tabel 4.1 Pengukuran jarak maksimal. ....	46
Tabel 4.2. Pengukuran jarak minimal. ....	46
Tabel 4.3 Parameter PID setiap motor. ....	47
Tabel 4.4 Hubungan konstanta dengan kecepatan robot.....	51
Tabel 4.5 Data pengujian posisi robot .....	54
Tabel 4.6 Hasil peta occupancy grid.....	57
Tabel 4.7 Hasil pengujian dengan variasi kecepatan. ....	58
Tabel 4.8 Waktu tempuh robot pada jarak 10 cm.....	59
Tabel 4.9 Hasil ukuran pemetaan percobaan 1.....	62
Tabel 4.10 Hasil ukuran pemetaan percobaan 2.....	63
Tabel 4. 11 Pengujian dengan ruangan asli.....	64

.....*Halaman ini sengaja dikosongkan*.....



# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Sebuah peta ruangan bisa dibangun dengan menggunakan bantuan robot. Informasi pemetaan pada ruangan tertentu sangat berguna baik untuk robot dalam melakukan tugasnya ataupun manusia yang ingin mengetahui suatu ruangan yang sulit untuk dijangkau [1]. Dengan adanya pemetaan ruangan ini didapatkan pengenalan tentang lingkungan sekitar [2], [3]. Sehingga akan memudahkan robot dalam melakukan tugas selanjutnya, seperti navigasi pada robot ataupun misi penentuan sebuah ruang. Oleh karena itu diperlukan robot yang dapat melakukan tugas pemetaan ruangan ini.

Untuk mendapatkan data dalam pembentukan peta ruangan dibutuhkan sensor yang memiliki jarak pembacaan yang jauh serta akurasi yang tinggi. LIDAR merupakan salah satu sensor optik yang memiliki pembacaan jarak yang jauh dan memiliki kerapatan data tentang sumbu koordinat dalam suatu ruangan. Sehingga LIDAR berpotensi untuk melakukan pembentukan peta pada suatu ruangan [2].

Untuk dapat melakukan pemetaan seluruh ruangan yang diinginkan maka LIDAR perlu menelusuri seluruh ruangan. *Mobile robot* merupakan salah satu jenis robot yang dapat digunakan dalam melakukan penelusuran ruangan [4]. Dengan demikian perlu dirancang sebuah *mobile robot* yang terintegrasi dengan sensor LIDAR.

Perkembangan framework robotika saat ini telah berkembang pesat, salah satunya adalah ROS (Robot Operating system). ROS menyediakan berbagai packages dan tools yang sangat berguna untuk pengembangan sebuah robot [5]. ROS bersifat *open source* sehingga sangat memungkinkan melakukan pengembangan pada *mobile robot* untuk mengolah data LIDAR dengan menggunakan framework ROS untuk mendapatkan visualisasi pemetaan yang lebih baik dan akurat. Dengan demikian, diharapkan robot ini dapat memberikan informasi tentang peta ruangan secara *realtime* dan peta dapat disimpan.

## 1.2 Perumusan Masalah

Rumusan masalah pada tugas akhir ini sebagai berikut:

1. Bagaimana cara membentuk peta ruangan menggunakan *mobile robot*.
2. Bagaimana pengolahan data LIDAR untuk pembuatan peta ruangan.

## 1.3 Tujuan Penelitian

Penelitian pada tugas akhir ini memiliki tujuan sebagai berikut:

1. Implementasi LIDAR untuk mendapatkan data posisi objek pada lingkungan robot dalam ruangan.
2. Robot dapat membentuk peta ruangan dengan LIDAR menggunakan beberapa *packages* dan *tools* pada ROS.

## 1.4 Batasan Masalah

Batasan masalah dalam tugas akhir ini sebagai berikut:

1. Lingkungan robot dalam melakukan pemetaan ruangan terhindar dari gangguan objek bergerak.
2. Pemetaan berbentuk peta dua dimensi.
3. Tinggi objek yang dihindari minimal sama dengan tinggi robot.

## 1.5 Metodologi Penelitian

Langkah-langkah yang dikerjakan pada tugas akhir ini adalah sebagai berikut.

1. Studi literatur

Studi literatur berisi kegiatan pengumpulan dan pengkajian dasar teori yang terpercaya untuk menunjang penulisan tugas akhir ini. Literatur dapat bersumber dari paper, jurnal, artikel, buku, maupun website yang bertaraf nasional dan internasional, serta dari hasil konsultasi dengan dosen pembimbing.

2. Observasi dan analisa masalah

Pada tahap ini dilakukan observasi terkait penerapan sensor LIDAR pada *mobile robot* untuk pemetaan ruangan serta komponen lainnya yang menunjang dalam pembuatan sistem. Pada tahap ini juga dilakukan analisa mengenai metode yang paling mudah diterapkan untuk pembuatan peta dengan menggunakan sensor LIDAR.

Observasi dan analisa masalah berasal dari *website*, *paper*, jurnal, buku, dan sumber-sumber yang dapat dipercaya lainnya. Berdasarkan referensi yang dibaca kebanyakan pemetaan ruangan dilakukan pada robot yang berbasis *Robot Operating System*.

### 3. Persiapan alat dan bahan

Tahap ini merupakan tahap pencarian informasi mengenai kebutuhan bahan untuk merancang *mobile robot* yang dapat melakukan tugas pemetaan ruangan ini. Informasi didapatkan dari studi literatur dan bimbingan dosen pembimbing. Kemudian dilakukan pengumpulan alat dan bahan yang dibutuhkan. Alat dan bahan yang diperlukan adalah seperangkat laptop, raspberry pi 3b, sensor LIDAR, motor dc, arduino nano, *driver* motor dc serta alat dan bahan pendukung lainnya.

### 4. Perancangan

Tahap perancangan ini meliputi perancangan perangkat keras dan perangkat lunak. Perangkat keras meliputi rangkaian elektronik dan desain mekanik robot. Sebelum mencetak rangkaian elektronik dalam satu PCB, setiap komponen dan modul diuji terlebih dahulu. Desain mekanik robot melanjutkan mekanik robot yang sudah ada dan selanjutnya ditambahkan beberapa mekanik pendukung. Perangkat lunak dirancang dengan pembuatan *source code* untuk pembacaan sensor, pembuatan peta, dan kontrol motor yang terintegrasi dalam *Robot Operating system*.

### 5. Pembuatan

Proses pembuatan diawali dengan pemasangan tiap komponen pada PCB. Pemasangan ini dilakukan dengan penyolderan komponen pada PCB. Selanjutnya pemasangan mekanik robot sesuai dengan desain yang telah ada serta melakukan *wiring* pada robot.

### 6. Pengujian sistem

Dalam tahap ini, akan dilakukan pengujian apakah sistem perangkat keras dan perangkat lunak yang dibuat dapat bekerja sesuai ekspektasi. Pengujian sistem dilakukan secara bertahap dan secara keseluruhan. Pengujian dilakukan terhadap perangkat keras dan perangkat lunak.

### 7. Analisa dan evaluasi

Analisa dilakukan terhadap hasil pengujian sehingga karakteristik perangkat lunak dan perangkat keras dapat diketahui. Analisa dilakukan pada respon pembacaan sensor dan pembuatan

peta ruangan yang dihasilkan. Apabila hasil yang didapatkan masih jauh dari yang diharapkan maka perlu dilakukan evaluasi pada sistem. Dari data evaluasi tersebut dilakukan perancangan dan pengujian kembali.

#### 8. Penyusunan laporan tugas akhir

Penyusunan laporan tugas akhir merupakan tahap akhir dari proses pengerjaan tugas akhir ini yang merupakan hasil akhir dan kesimpulan dari pembuatan dan pengujian sistem secara keseluruhan. Semua hal yang menyangkut pengerjaan tugas akhir dimuat dalam buku laporan tugas akhir.

### 1.6 Sistematika Penulisan

Dalam buku tugas akhir ini, pembahasan mengenai sistem yang dibuat terbagi menjadi lima bab dengan sistematika penulisan sebagai berikut:

- **BAB I: Pendahuluan**  
Bab ini meliputi penjelasan latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi, sistematika penulisan, dan relevansi.
- **BAB II: Tinjauan Pustaka**  
Bab ini berisi mengenai teori penunjang yang terkait dalam pengerjaan tugas akhir ini.
- **BAB III: Perancangan Sistem**  
Bab ini menjelaskan tentang perencanaan sistem yang meliputi perangkat keras dan perangkat lunak untuk pembuatan alat pada tugas akhir ini.
- **BAB IV: Pengujian Dan Analisis**  
Bab ini berisi tentang pengujian robot yang telah dibuat dan analisa hasil dari pengujian yang telah didapat.
- **BAB V: Penutup**  
Bab ini berisi tentang kesimpulan yang diperoleh dari alat yang telah dibuat serta saran untuk pengembangan selanjutnya.

## **1.7 Relevansi**

Pemetaan ruangan dengan menggunakan lidar dapat membantu sebuah robot dalam melakukan tugas-tugas berikutnya seperti lokalisasi atau navigasi. Selain itu, dapat menjadi bekal untuk mengetahui bentuk dari suatu ruangan yang sulit dijangkau oleh manusia. Pemetaan ruangan dilakukan dengan cara menjalankan *mobile robot* secara *wireless* dengan kontrol keyboard pada laptop atau menjalankan *mobile robot* secara otomatis menelusuri ruangan dengan mengikuti dinding. Peta ruangan dapat disimpan dan diakses lagi untuk keperluan selanjutnya.

.....*Halaman ini sengaja dikosongkan*.....

## BAB II

### TINJAUAN PUSTAKA

Dalam suatu penelitian dibutuhkan teori-teori yang sudah ada sebelumnya untuk dikaji dalam memperkuat argumen penulis. Teori tersebut digunakan untuk membantu penulis dan sebagai dasar dalam membuat suatu penelitian.

Pada bab ini terdapat teori dasar yang menjadi landasan untuk merumuskan dan menyelesaikan masalah yang akan dibahas pada penelitian ini. Selain itu, juga terdapat tinjauan pustaka tentang komponen yang akan digunakan untuk membuat robot pada penelitian ini.

#### 2.1 *Mobile Robot*

*Mobile robot* adalah sebuah robot dengan konstruksi dengan ciri khasnya adalah mempunyai aktuator berupa roda atau kaki untuk menggerakkan keseluruhan badan robot tersebut, sehingga robot tersebut dapat melakukan perpindahan posisi dari satu titik ke titik yang lain [6]. *Mobile robot* digunakan secara luas sebagai aktuator dalam berbagai bidang aplikasi dengan dilengkapi sensor untuk mengendalikan arah gerakan robot tersebut. *Mobile robot* banyak digunakan di berbagai bidang seperti industri, operasi, transportasi, perencanaan jalur dan pelacakan, serta militer [7].



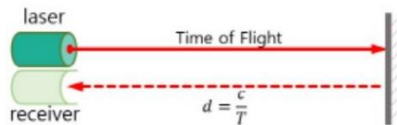
**Gambar 2.1** Contoh *Mobile Robot* [8]

## 2.2 Light Detection and Ranging (LIDAR)

Light Detection And Ranging (LIDAR) merupakan teknologi pengukur jarak dengan cara memantulkan cahaya laser dari suatu objek. LIDAR menggunakan cahaya tunggal dan koheren pada spektrum dan frekuensi tertentu melalui radiasi elektromagnetik. Sehingga pancarannya memiliki sudut pancaran yang kecil dan memiliki intensitas yang tinggi untuk dapat mencapai jarak yang jauh dan terarah dengan tepat pada suatu objek. LIDAR melakukan pengukuran secara berulang dan berurutan sehingga kondisi lingkungan dapat diketahui [3].

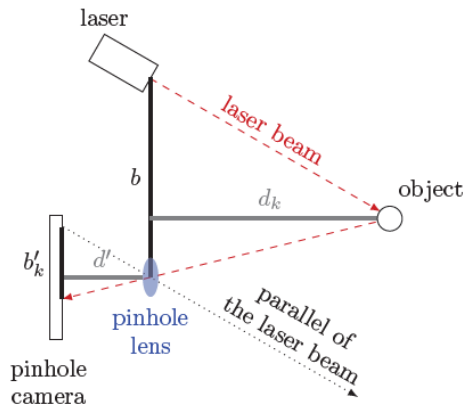
Berdasarkan metode pengukuran jaraknya LIDAR dibagi menjadi dua, yaitu LIDAR *Time of Flight* (ToF) dan LIDAR triangulasi. Cara kerja LIDAR ToF dengan memancarkan sinar laser terhadap objek dan kemudian pantulan dari sinar tersebut akan diterima oleh *receiver*. Waktu tempuh sejak sinar dipancarkan sampai diterima kembali akan menjadi pembagi dari kecepatan cahaya. Perbandingan antara kecepatan cahaya dan waktu akan diolah menjadi data jarak.

Sedangkan Cara kerja LIDAR triangulasi memanfaatkan perhitungan persamaan segitiga. Laser memancarkan sinyal laser inframerah yang kemudian dipantulkan oleh objek yang akan dideteksi. Cahaya melewati lensa lubang jarum dan mengenai sensor kamera CCD. Ini berarti bahwa jarak ke objek sebanding dengan sudut cahaya yang dipantulkan, dan dapat memperkirakan jarak aktual menggunakan konsep persamaan trigonometri segitiga [9]. Berdasarkan dimensinya ada beberapa jenis LIDAR, yaitu LIDAR dengan satu titik (1D), LIDAR 2D dan LIDAR 3D.



**Gambar 2.2** Cara kerja LIDAR ToF [10].





**Gambar 2.3** Cara kerja LIDAR triangulasi [9].

### 2.3 YdLidar X4

YdLidar X4 merupakan lidar yang digunakan sebagai pemindaian jarak dalam ruangan. Prinsip kerja dari sensor ini menggunakan metode segitiga trigonometri (triangulasi) dalam menentukan jarak yang diukur. YdLidar X4 menggunakan laser yang memenuhi standar FDA kelas 1. YdLidar X4 dilengkapi dengan motor dc yang berfungsi sebagai pemutar lidar. Sehingga lidar dapat melakukan pemindaian 360°. Hasil dari pemindaian tersebut berupa nilai sudut beserta jarak objek.



**Gambar 2.4** YdLidar X4 [11].

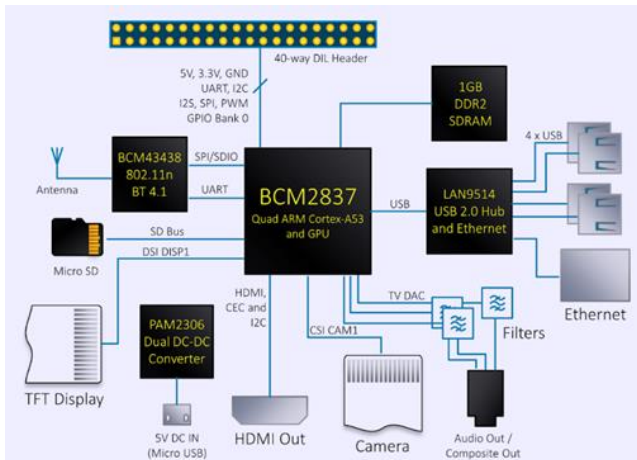
Spesifikasi dari sensor ini tercantum pada tabel 2.1

**Tabel 2.1** Spesifikasi YdLidar X4

Parameter	Nilai	Satuan
Supply Voltage	5	V
Starting current	450	mA
Working Current	350	mA
Sleep Current	300	mA
Frekuensi pemindaian	7	Hz
Frekuensi data jarak	5000	Hz
Jarak Jangkau	0.12 – 10	M
Jangkauan Sudut	0-360	Derajat
Resolusi Sudut	0.5	Derajat

## 2.4 Raspberry Pi 3B

Raspberry Pi 3 B merupakan generasi ketiga dari produk raspberry pi. *Single-board computer* yang memiliki sistem broadcom BCM2837 64-bit CPU quad core ARM Cortex-A53 dengan kecepatan 1.2 GHz. Raspberry pi 3b memiliki 40 pin I/O, 1 GB RAM, 4 USB port, wireless LAN 802.11, Bluetooth Low Energi 4.1, konektor HDMI dan konektor audio 3.5mm [12], [13].



**Gambar 2.5** Sistem Perangkat Keras Raspberry Pi 3b [13].

Dengan ukuran dimensi yang kecil dan kemampuan komputasi yang cukup memadai maka Raspberry pi ini banyak digunakan dalam pembuatan robot berbasis single-board computer. Operating system dari raspberry pi 3 B ini dibooting melalui micro sd card dan berjalan dengan bermacam-macam *Operating system*, seperti ubuntu mate, windows 10 IOT, Raspbian Stretch, Noob dan lain sebagainya.

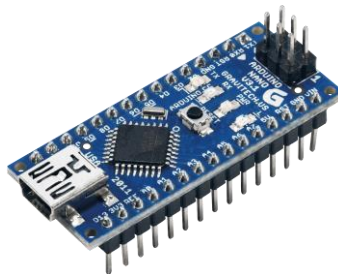


**Gambar 2.6** Raspberry Pi 3b [13].

Sebagaimana *personal computer*, raspberry pi 3 B ini mampu menampung pemasangan ROS pada *Operating System* yang berbasis linux seperti raspbian dan ubuntu mate.

## 2.5 Mikrokontroler Arduino Nano

Salah satu modul mikrokontroler ATmega yang terkenal adalah arduino. Arduino nano merupakan salah satu jenis board arduino berbasis ATmega328P yang berukuran kecil sehingga cocok untuk digunakan dalam proyek yang tidak membutuhkan banyak pin I/O [10]. Adapun spesifikasi dari arduino nano sendiri adalah sebagai pada tabel 2.2.

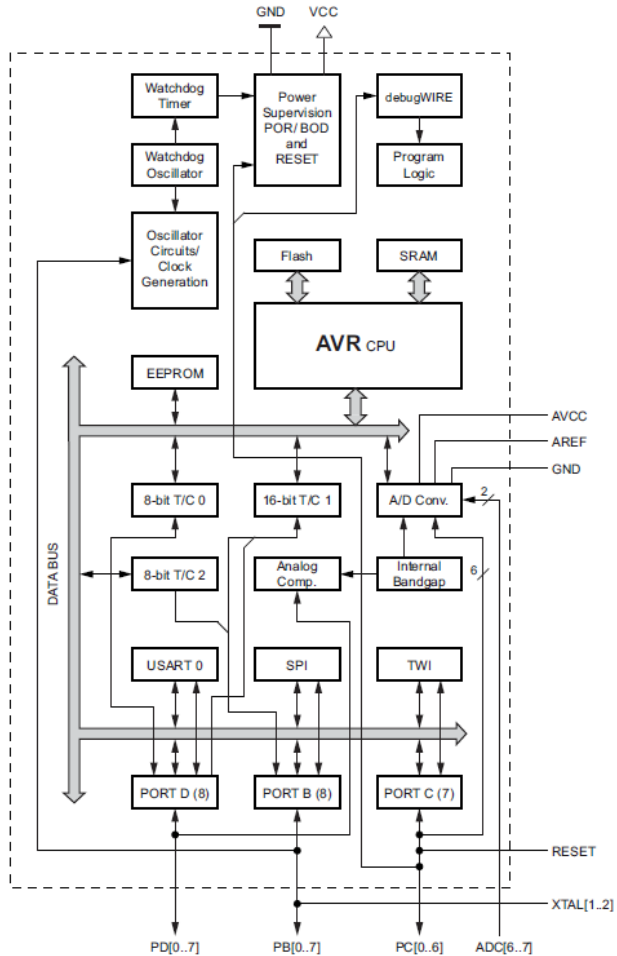


**Gambar 2.7** Arduino Nano [14].

**Tabel 2.2** Spesifikasi Arduino Nano [14].

<b>Spesifikasi Arduino Nano</b>	
Mikrokontroler	ATmega328p
Arsitektur	AVR
Tegangan operasi	5V
Flash memori	32 KB
SRAM	2 KB
Clock	16 MHz
Jumlah pin analog	8
EEPROM	1 KB
Arus DC per pin I/O	40 mA
Tegangan input	7 – 12 V
Jumlah pin digital	22
<i>PWM output</i>	6
Konsumsi daya	19 mA
Serial	D1-TX, D0-RX
I2C	A5-SCL, A4-SDA
SPI	D10-SS, D11-MOSI, D12-MISO, D13-SCK

Dengan adanya fitur komunikasi serial, arduino nano ini dapat berkomunikasi dengan device lain baik sesama mikrokontroler ataupun dengan mini PC raspberry pi. Dengan fitur ini pun, bisa menjadi sarana untuk saling berbagi tugas antar device sehingga tidak terlalu memberatkan komputasi yang dilakukan.

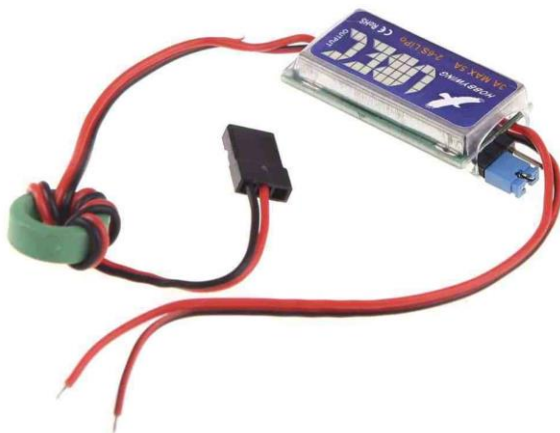


**Gambar 2.8** Diagram Alur Perangkat Keras Arduino Nano [15].

## 2.6 UBEC 5V 3A

UBEC (*Universal Battery Eliminated Circuit*) adalah perangkat elektronika yang berfungsi untuk menurunkan nilai tegangan dengan nilai yang diinginkan. UBEC ini berfungsi layaknya sebagai regulasi tegangan. Alat ini biasa di gunakan untuk menurunkan tegangan dari input 6V-23V

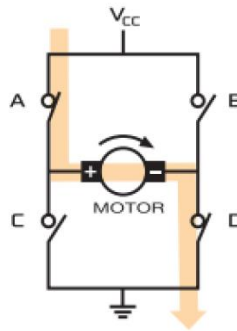
menjadi 5V dan 6V dengan memilih jumper yang terdapat pada unit UBEC. UBEC biasanya digunakan robot untuk memberikan masukan tegangan untuk beberapa komponen yang membutuhkan sumber tegangan sesuai dengan keluaran dari UBEC.



**Gambar 2.9** UBEC 5V 3A [16].

## 2.7 Driver Motor DC

Driver motor merupakan rangkaian untuk mengendalikan kecepatan dan arah putar motor. Rangkaian ini akan memperkuat arus yang akan masuk ke motor dari unit pengontrol yang dalam hal ini merupakan mikrokontroler sehingga arus yang masuk ke motor cukup besar untuk menggerakkan motor. Rangkaian driver motor yang sering dipakai adalah rangkaian konfigurasi jembatan H atau biasa disebut H-Bridge. Rangkaian ini dapat mengatur arah putaran motor [17].



**Gambar 2.10** Rangkaian *H-Bridge* [18].

Jika saklar A dan saklar D disambungkan dan saklar B dan saklar C tidak disambungkan, arus akan mengalir dari tegangan sumber ke sisi kiri motor. Hal ini akan membuat gerakan motor searah jarum jam. Sebaliknya, jika saklar yang tersambung adalah saklar B dan saklar C, arus akan mengalir dari tegangan sumber ke sisi kanan motor. Hal ini akan membuat motor bergerak berlawanan arah jarum jam [18].

## 2.8 Robot Operating System (ROS)

Perkembangan tentang teknologi robotika terus berkembang. Robot Operating System (ROS) merupakan salah satu *open source framework* di bidang robotika yang fleksibel untuk memprogram robot software. ROS adalah kumpulan tools, libraries, dan packages yang bertujuan untuk menyederhanakan tugas serta menciptakan perilaku robot yang kompleks dan kuat di berbagai platform robotika. Beberapa fitur pada ROS yang sering digunakan adalah rekognisi, pemetaan, navigasi, perencanaan pergerakan, visualisasi, komunikasi antar proses serta beberapa fitur untuk debugging [5], [19]. Beberapa istilah mendasar yang berada pada ROS adalah sebagaimana berikut:

### 2.8.1 Master

Master bertindak sebagai server nama untuk koneksi node-to-node dan komunikasi pesan. Perintah `roscore` digunakan untuk menjalankan master, ketika menjalankan master bisa mendaftarkan nama setiap node dan mendapatkan informasi pada node. Koneksi

antara node dan komunikasi pesan seperti topic dan service tidak mungkin dilakukan tanpa adanya master.

### **2.8.2 Node**

Sebuah node mengacu pada unit terkecil dari prosesor yang berjalan di ROS. ROS merekomendasikan membuat satu simpul tunggal untuk setiap tujuan, dan disarankan untuk mengembangkannya agar mudah digunakan kembali. Misalnya, dalam kasus robot mobil, program untuk mengoperasikan robot dipecah menjadi fungsi khusus. Setiap node digunakan untuk setiap fungsi seperti drive sensor, konversi data sensor, rekognisi halangan, drive motor, input encoder, dan navigasi.

### **2.8.3 Message**

Sebuah node mengirim atau menerima data antar node melalui sebuah message. Message sendiri merupakan variabel seperti integer, floating point, dan boolean.

### **2.8.4 Topic**

Topic dalam ROS sebagaimana topik dalam sebuah percakapan. Node publisher akan mendaftarkan topiknya dengan master dan kemudian mulai mengirimkan message pada suatu topik. Node subscriber yang ingin menerima informasi, akan meminta topik dari node publisher yang terkait dengan nama topik yang terdaftar di master. Berdasarkan informasi ini, node subscriber langsung terhubung ke node publisher untuk bertukar pesan sebagai topik.

### **2.8.5 Publisher**

Publisher merupakan pelaku transmisi pesan relatif yang terkait dengan topic. Node publisher mendaftarkan informasi dan topiknya sendiri dengan master, dan mengirim message ke node subscriber yang meminta topik yang sama. Publisher dideklarasikan di node dan dapat dideklarasikan beberapa kali dalam satu node.

### **2.8.6 Subscriber**

Subscriber merupakan penerima pesan relatif yang terkait dengan topik. node subscriber mendaftarkan informasi dan topiknya sendiri dengan master, dan menerima informasi publisher yang menerbitkan topik dari master. Berdasarkan informasi yang diterima dari publisher, node subscriber meminta koneksi langsung ke node



publsiher dan menerima pesan dari node publsiher yang terhubung. Subscriber dinyatakan dalam node dan dapat dinyatakan beberapa kali dalam satu node.

### **2.8.7 Roscore**

Roscore adalah perintah yang menjalankan master ROS. Jika beberapa komputer berada dalam jaringan yang sama, maka dapat dijalankan dari komputer lain di jaringan tersebut. Ketika menjalankan master ROS, alamat URI dan nomor port yang ditetapkan untuk variabel lingkungan ROS\_MASTER\_URI harus ditetapkan. Jika belum menetapkan variabel lingkungan, maka alamat IP lokal saat ini digunakan sebagai alamat URI dan nomor port 11311 digunakan yang merupakan nomor port default untuk master.

### **2.8.8 Rosrun**

Rosrun adalah perintah eksekusi dasar ROS. Ini digunakan untuk menjalankan satu node dalam paket. Node menggunakan variabel lingkungan ROS\_HOSTNAME yang disimpan di komputer di mana node berjalan sebagai alamat URI.

### **2.8.9 Roslaunch**

Kalau rosrun digunakan untuk menjalankan satu node, roslaunch sebaliknya. Dengan perintah ini dapat menjalankan beberapa node. Roslaunch menggunakan file .launch untuk menentukan node mana yang akan dieksekusi. File ini berbasis XML (*Extensible Markup Language*).



**Gambar 2.11** *Robot Operating System* [19].

Dalam aplikasi untuk pemetaan ruangan beberapa packages yang dapat digunakan yaitu cartographer, gmapping, hector SLAM, Octomap. Selain itu juga ada tools yang umum digunakan dalam visualisasi yaitu rviz [5].

## **2.9 *Simultaneous Localization and Mapping (SLAM)***

Pemetaan Ruang merupakan proses mengintegrasikan informasi yang dikumpulkan dengan sensor robot ke dalam representasi yang diberikan. Hal ini dapat dijelaskan dengan pertanyaan "Seperti apa lingkungan ini?" Aspek penting dalam pemetaan adalah representasi lingkungan dan interpretasi data sensor. Lokalisasi dan pemetaan simultan, atau SLAM, adalah proses pembuatan peta menggunakan robot atau kendaraan tanpa awak yang dapat bernavigasi pada lingkungan itu ketika menggunakan peta yang dihasilkannya. SLAM adalah teknik di balik pemetaan robot atau kartografi robot. Robot tersebut merencanakan suatu jalur di suatu daerah, tetapi pada saat yang sama, ia juga harus mencari tahu di mana dirinya berada di tempat itu. Proses SLAM menggunakan susunan komputasi, algoritma dan masukan sensor yang kompleks untuk bernavigasi di sekitar lingkungan yang sebelumnya tidak diketahui atau untuk merevisi peta lingkungan yang diketahui sebelumnya [20], [21].

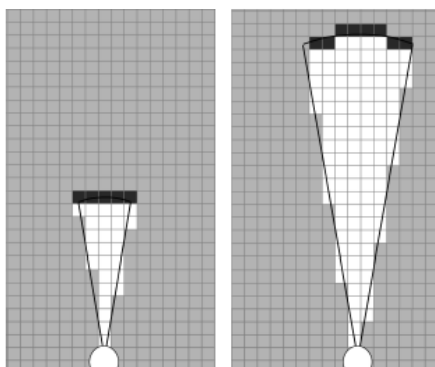
SLAM awalnya dikembangkan oleh Hugh Durrant-Whyte dan John J. Leonard berdasarkan karya sebelumnya oleh Smith, Self and Cheeseman-Durrant-Whyte dan Leonard. SLAM merupakan konsep yang dapat diterapkan dengan menggunakan berbagai algoritma yang berbeda.

## **2.10 *Occupancy Grid***

*Occupancy grid* digunakan untuk mewakili ruang kerja robot sebagai *grid* terpisah. Informasi tentang lingkungan dapat dikumpulkan dari sensor secara real time atau diambil dari pengetahuan sebelumnya. Sensor LIDAR, sensor jarak, dan kamera biasanya digunakan untuk menemukan halangan di lingkungan robot. *Occupancy grid* digunakan dalam pemetaan untuk mengintegrasikan informasi sensor dalam peta diskrit, dalam perencanaan lintasan untuk menemukan lintasan bebas halangan, dan untuk melokalisasi robot di lingkungan yang diketahui serta dapat membuat peta dengan berbagai ukuran dan resolusi agar sesuai dengan spesifikasi yang diinginkan [22].

*Occupancy grid* memiliki dua representasi, yaitu *Occupancy grid* biner dan *occupancy grid* probabilitas [23]. *Occupancy grid* biner menggunakan nilai benar untuk mewakili ruang kerja yang diduduki dan nilai salah untuk mewakili ruang kerja yang kosong. Setiap *grid* menunjukkan dimana halangan berada dan apakah robot dapat bergerak melalui ruang itu.

*Occupancy grid* probabilitas menggunakan nilai probabilitas untuk membuat representasi peta yang lebih rinci. Representasi ini adalah metode yang banyak digunakan untuk menggunakan *Occupancy grid*. Setiap sel dalam *Occupancy grid* memiliki nilai yang mewakili probabilitas hunian sel itu. Nilai mendekati 1 mewakili probabilitas tinggi bahwa terdapat halangan sel. Nilai mendekati 0 menunjukkan probabilitas bahwa sel tidak ditempati dan bebas halangan.

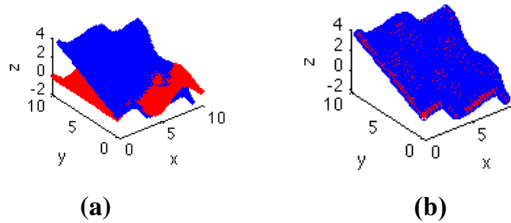


**Gambar 2.12** Peta dengan *occupancy grid* [22].

## 2.11 Scan Matching

*Scan matching* adalah proses menyelaraskan pindaian laser (LIDAR) satu sama lain atau dengan peta yang ada. Sensor lidar terbaru memiliki gangguan yang rendah dalam pengukuran jarak dan frekuensi pemindaian yang tinggi. Metode untuk menggunakan hasil pindaian lidar mungkin menghasilkan hasil yang sangat akurat karena alasan tersebut. Untuk banyak sistem robot, akurasi dan ketepatan pemindai lidar jauh lebih tinggi daripada data odometri roda [24]. Algoritma yang sering digunakan dalam scan matching ini adalah *Iterative Closest Point* (ICP).

ICP merupakan algoritma yang digunakan untuk meminimalkan perbedaan antara dua titik koordinat. ICP sering digunakan untuk merekonstruksi permukaan 2D atau 3D dari pemindaian yang berbeda. Hasil dari scan matching adalah transformasi translasi dan rotasi dari LIDAR [25].



**Gambar 2. 13.** Proses *scan matching* [26] **(a)** Data koordinat biru akan dicocokkan dengan merah, **(b)** Hasil *scan matching*.

## 2.12 Tinjauan Pustaka

### 2.12.1 Research and Implementation of SLAM Based on LIDAR for Four-Wheeled Mobile Robot [1]

Penelitian ini menggunakan *mobile robot* dengan roda sebanyak 4 menggunakan sensor RpLidar A1 dengan pengolahan data pada raspberry pi 3b dan mikorkontroller teensy 3.1. robot juga dilengkapi dengan sensor IMU GY-85 dan encoder untuk mendapatkan data posisi robot. Implementasi SLAM dilakukan pada ROS dengan menggunakan *package gmapping* sehingga memerlukan data odometri dari roda dalam melakukan pemetaan ruangan dan lokalisasi. *Gmapping* menggunakan *Rao-Black-well-ized particle filtering* (RBPF) untuk menentukan posisi dari robot. Dalam melakukan pemetaan robot diberi kontrol data titik pada *tools rviz* sehingga robot akan berhenti ketika titik-titik tersebut telah terpenuhi. Selama robot bergerak menelusuri ruangan yang tidak diketahui, data pemindaian LIDAR akan diolah untuk membuat peta dan ditampilkan dalam *rviz*.

### **2.12.2 Autonomous 2D SLAM and 3D Mapping of an Environment Using a Single 2D LIDAR and ROS [2]**

Penelitian ini menjelaskan algoritma yang melakukan rekonstruksi pemetaan 3D secara otomatis dengan menggunakan sensor LIDAR 2D serta implementasinya pada platform seluler menggunakan ROS. Sensor LIDAR 2D yang digunakan adalah Hokuyo URG-04LX-UG01. Pengambilan data 3D didapatkan dengan cara menambahkan servo pada robot ini. *Platform* robot yang digunakan adalah Amigobot. Sebelum membentuk informasi dalam bentuk 3D dilakukan proses pengolahan informasi 2D terlebih dahulu. ROS *package* yang digunakan dalam melakukan SLAM 2D ini adalah Google cartographer. Untuk membentuk peta 3D digunakan *package* ROS Octomap. Algoritma robot dalam melakukan pemetaan otomatis diimplementasikan dengan perpustakaan (SMACH) yang terinspirasi dari *google street view*.

### **2.12.3 Obstacle Detection and Avoidance Algorithm for Autonomous Mobile Robot using 2D LiDAR [3]**

Penelitian ini berfokus pada deteksi hambatan dan penghindaran rintangan berdasarkan sensor LIDAR 2D UTM-30LX. LIDAR pengukuran digunakan untuk mengidentifikasi struktur spasial hambatan. Metode yang diusulkan didasarkan pada segmentasi cloud titik data yang digunakan untuk mengurangi waktu pemrosesan, diikuti oleh pengelompokan. Kontribusi utama dari pekerjaan ini adalah pemetaan yang tepat dari struktur geometris hambatan menggunakan algoritma convex hull. Algoritma ini menggunakan metode sudut kutub dan struktur hambatan dalam hal peningkatan poligon efisiensi algoritma perencanaan jalur grafik visibilitas dan penghindaran kendala. Efektivitas model yang diusulkan diverifikasi di platform simulasi MATLAB.

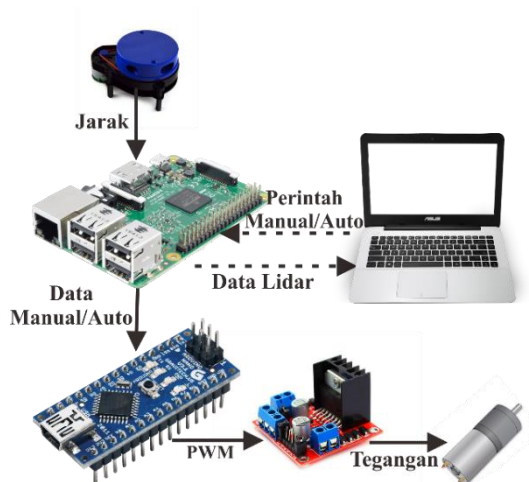
.....*Halaman ini sengaja dikosongkan*.....

## BAB III PERANCANGAN SISTEM

Pada bab ini akan dijelaskan mengenai perancangan robot secara keseluruhan, meliputi perancangan perangkat keras dan perancangan perangkat lunak. Perancangan perangkat lunak terdiri dari perancangan mekanik robot dan elektronik. Sedangkan perancangan perangkat lunak berhubungan dengan pemrograman untuk implementasi pembuatan peta ruangan dengan robot.

### 3.1. Gambaran Umum Sistem

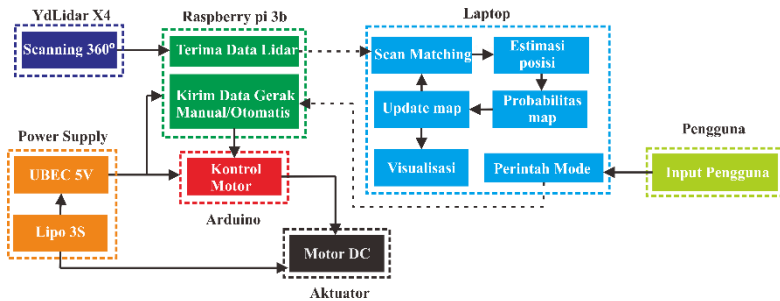
*Mobile robot* yang dibuat bertujuan untuk memberikan visualisasi bentuk ruangan ke dalam bentuk peta. Sensor yang digunakan adalah untuk pemetaan ruangan ini adalah LIDAR. Untuk mengakses sensor LIDAR digunakan sebuah *single board computer*. Data LIDAR dikirimkan ke laptop untuk diolah menjadi peta. Laptop dan *single board computer* ini terintegrasi dalam ROS. Untuk menjalankan robot digunakan kontrol manual dari pengguna atau robot berjalan otomatis dengan menelusuri ruangan. Perintah tersebut berasal dari *single board computer* dan dikirimkan ke mikrokontroller.



**Gambar 3.1** Skema sistem keseluruhan.

Jenis LIDAR yang digunakan untuk membuat peta ini adalah LIDAR yang dapat berputar 360° yaitu, YdLidar X4 keluaran EAI Team. *Single board computer* yang digunakan adalah Raspberry Pi 3b. Sensor LIDAR digunakan sebagai sensor jarak pada setiap objek yang berada pada lingkungan robot. Karena LIDAR sudah dilengkapi motor dc yang memutar sensor sebesar 360° maka jarak pada seluruh sudut dapat diketahui. Cara pengambilan data LIDAR menggunakan komunikasi serial. Setelah data LIDAR masuk pada Raspberry Pi 3b, kemudian data tersebut dikirimkan secara *wireless* ke laptop yang dan dilakukan perhitungan untuk membuat sebuah peta.

Data LIDAR pertama kali akan diproses pada proses scan matching, yaitu proses pencarian transformasi posisi dan rotasi hasil pemindaian data LIDAR tanpa menggunakan data odometri roda. Hasil dari scan matching ini berupa estimasi posisi dan orientasi robot. Setelah posisi dan orientasi dari robot diketahui dilakukan perhitungan probabilitas peta untuk menentukan nilai pada setiap peta grid. Nilai probabilitas dari setiap grid tersebut yang akan mempresentasikan peta yang akan ditampilkan.



**Gambar 3.2** Blok Diagram Sistem

Robot ini bergerak dengan 3 roda omni dan motor dc. Kontrol pergerakan robot ini menggunakan mikrokontroller Arduino Nano. Kecepatan motor dikontrol menggunakan PID. Dalam menelusuri ruangan yang akan dibentuk petanya robot ini dapat dikontrol secara manual ataupun secara otomatis.

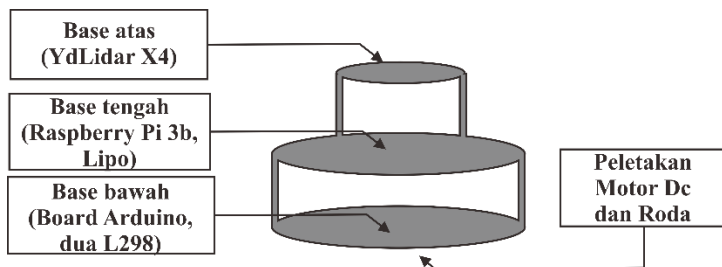


Kontrol manual dilakukan secara *wireless* dengan cara menekan tombol *keyboard* pada laptop untuk memberikan data pada raspberry pi 3b dengan bantuan *remote desktop* VNC. Dari Raspberry Pi 3b ini data akan dilanjutkan menuju Arduino Nano yang bertindak sebagai controller motor dc. Selain itu robot ini juga dapat berjalan secara otomatis menelusuri ruangan. Pilihan untuk otomatis atau manual tergantung dari input dari pengguna untuk menjalankan program manual atau otomatis. Ketika robot berjalan maka peta akan ditampilkan secara *realtime* pada laptop.

## 3.2. Perancangan Perangkat Keras

### 3.2.1 Mekanik *Mobile Robot*

Desain robot merupakan mekanik robot dari penelitian sebelumnya. Mekanik robot yang masih digunakan untuk penelitian ini adalah kerangka robot, motor dc serta roda omni. Selanjutnya dilakukan modifikasi pada robot untuk kebutuhan pada penelitian ini. Perancangan mekanik robot digunakan sebagai penentuan lokasi untuk setiap komponen yang dibutuhkan oleh robot. Dalam mekanik robot ini terdapat tiga *base*. *Base* pertama, yakni *base* paling bawah merupakan tempat komponen motor dc, driver motor dan board mikrokontroller Arduino Nano. *Base* kedua merupakan *base* bagian tengah tempat Raspberry Pi 3b serta baterai. Sedangkan *base* paling atas ditempati sensor YdLidar X4. Penempatan sensor LIDAR harus terbebas dari halangan yang berasal dari robot ini sendiri. Sehingga peletakan sensor lidar ini berada di tempat paling atas.

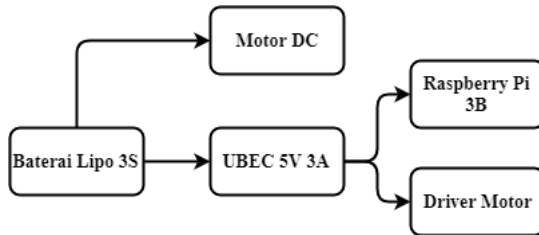


**Gambar 3.3** Perancangan desain robot.

Kerangka robot pada penelitian ini terbuat dari akrilik. Spacer besi digunakan sebagai pemisah antar *base* satu dengan lainnya. Roda robot terdiri dari 3 roda omni dengan penggerak motor dc 12v yang dilengkapi dengan encoder.

### 3.2.2 Catu Daya

Pada *mobile robot* ini digunakan sumber catu daya yang berasal dari baterai lipo 3 sel 11,1V mAh. Sedangkan untuk beberapa komponen seperti driver motor dan raspberry pi 3b membutuhkan tegangan sebesar 5V, maka dibutuhkan penurun tegangan dari tegangan baterai ke tegangan 5V. Untuk menurunkan tegangan tersebut digunakan UBEC 5V 3A. Sedangkan untuk catu daya dari motor sendiri langsung menggunakan baterai.



**Gambar 3.4** Alur catu daya motor dan Raspberry Pi

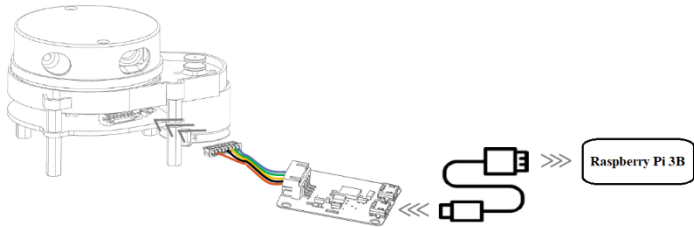
Arduino Nano dan sensor YdLidar X4 menggunakan sumber yang berasal dari usb serial yang dihubungkan ke port usb raspberry pi.



**Gambar 3.5** Alur catu daya YdLidar dan Arduino

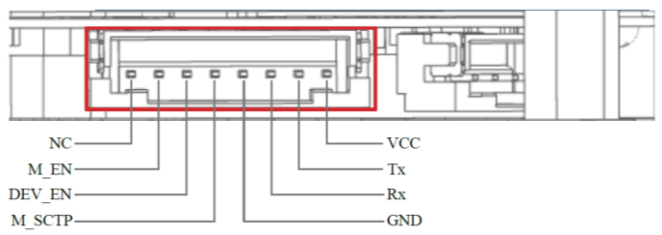
### 3.2.3 Pemasangan YdLidar X4

Berdasarkan datasheet dari YdLidar X4 ini, untuk koneksi dengan perangkat lain menggunakan usb adapter yang berkomunikasi secara serial. Adapun koneksi YdLidar X4 dengan Raspberry Pi 3B melalui usb adapter sebagaimana pada gambar 3.6. Catu daya untuk sensor LIDAR ini sendiri didapatkan langsung dari usb tersebut.



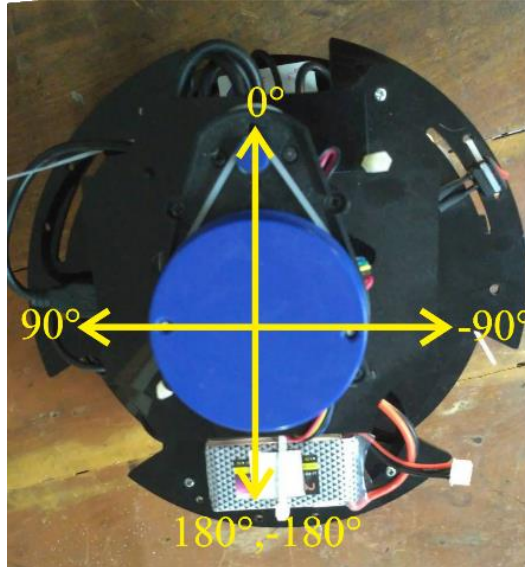
**Gambar 3.6** Koneksi YdLidar X4 dengan Raspberry Pi

Terdapat delapan pin pada adapter yang disambungkan ke sensor ydlidar X4. Delapan pin tersebut adalah VCC, TX, RX, GND, M\_EN, DEV\_EN, M\_SCTP dan NC. Pin M\_EN digunakan untuk mengaktifkan motor, pin DEV\_EN untuk mengaktifkan mulai proses pemindaian dan pin M\_SCTP digunakan sebagai kontrol kecepatan motor dc. Dengan adanya usb adapter yang telah disediakan oleh produsen, untuk akses perintah dari pin-pin tersebut bisa melalui komunikasi serial melalui usb adapter.



**Gambar 3.7** Pin adapter YdLidar X4

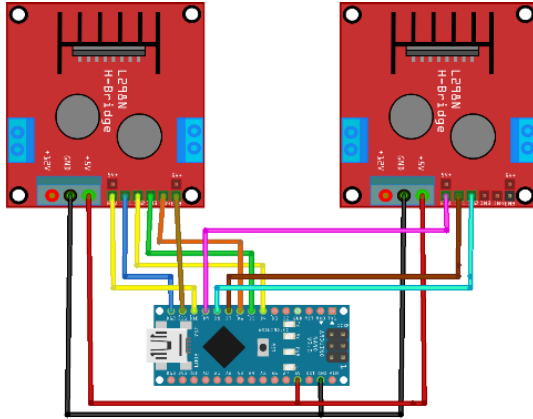
Berdasarkan datasheet dari sensor ini dilakukan konfigurasi sudut sensor LIDAR pada robot seperti gambar 3.8. Sudut  $0^{\circ}$  dipasang sebagai muka depan dari robot, sudut  $-90^{\circ}$  pada arah kanan, sudut  $90^{\circ}$  pada araha kiri dan sudut  $180^{\circ}$  untuk belakang robot.



**Gambar 3.8** Sudut YdLidar X4 pada robot.

### 3.2.4 Driver Motor

Roda pada robot ini terdapat tiga motor driver yang dilengkapi dengan masing masing penggerak motor dc. Modul motor driver yang digunakan pada robot ini adalah modul L298. Driver motor L298 merupakan rangkaian h-bridge yang mampu mengendalikan beban-beban induktif seperti relay, motor, dan selenoid. L298 terdiri dari transistor-transistor logik (TTL) dengan gerbang nand yang memudahkan dalam menentukan arah putaran suatu motor dc. Setiap modul driver motor l298 dapat mengontrol dua motor dc, sehingga pada robot ini digunakan dua buah motor driver l298.



**Gambar 3.9** Wiring Arduino dengan L298

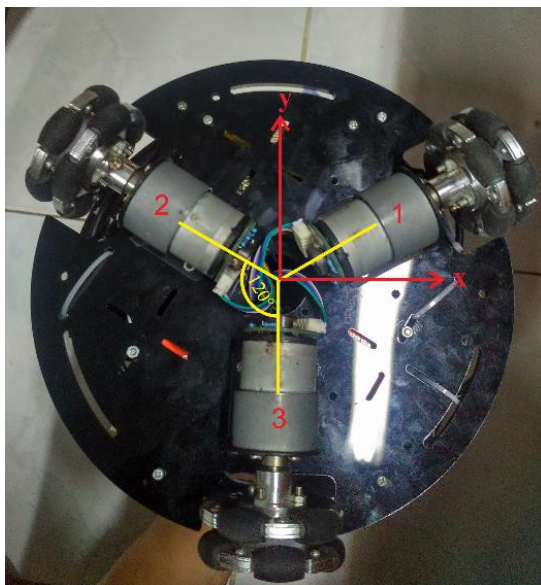
Konfigurasi antara pin Arduino dengan motor driver sebagaimana pada tabel 3.1.

**Tabel 3.1** Konfigurasi pin Arduino dengan motor driver.

Pin arduino	Pin motor driver
D12	IN1
D4	IN2
D10	EN1
D5	IN3
D6	IN4
D11	EN2
D7	IN5
D8	IN6
D9	EN3

### 3.2.5 Sistem Gerak Robot

Robot dirancang dengan menggunakan penggerak 3 roda omni dan motor dc 12v. Robot didesain menggunakan sistem penggerak roda omni dengan jarak antar roda 120°.



**Gambar 3.10** Desain penggerak robot

Dari gambar 3.10, dapat diturunkan kecepatan tiap-tiap roda terhadap sumbu x dan y robot sehingga bisa didapatkan  $V_x$  dan  $V_y$  robot. Penurunan kecepatan tiap roda dapat dijelaskan sebagai berikut. Untuk roda 1, didapatkan komponen persamaan kecepatan

$$V_{1x} = V_x \cos(120 + \alpha) \quad (3.1)$$

$$V_{1y} = V_y \sin(120 + \alpha) \quad (3.2)$$

$$\omega_1 = R\dot{\theta} \quad (3.3)$$

$$V_1 = V_x \cos(120 + \alpha) + V_y \sin(120 + \alpha) + R\dot{\theta} \quad (3.4)$$

Untuk roda 2, didapatkan komponen persamaan kecepatan

$$V_{2x} = V_x \cos(240 + \alpha) \quad (3.5)$$

$$V_{2y} = V_y \sin(240 + \alpha) \quad (3.6)$$

$$\omega_2 = R\dot{\theta} \quad (3.7)$$

$$V_2 = V_x \cos(240 + \alpha) + V_y \sin(240 + \alpha) + R\dot{\theta} \quad (3.8)$$

Untuk roda 3, didapatkan komponen persamaan kecepatan

$$V_{3x} = V_x \cos(0 + \alpha) \quad (3.9)$$

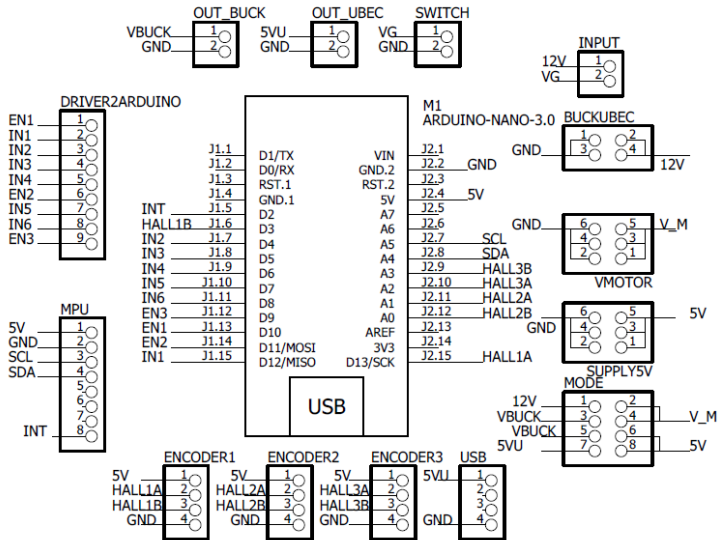
$$V_{3y} = V_y \sin(0 + \alpha) \quad (3.10)$$

$$\omega_3 = R\dot{\theta} \quad (3.11)$$

$$V_3 = V_x \cos(0 + \alpha) + V_y \sin(0 + \alpha) + R\dot{\theta} \quad (3.12)$$

### 3.2.6 Board Mikrokontroler Arduino Nano

Board ini didesain untuk merapikan *wiring* antar setiap komponen. Mulai dari manajemen catu daya serta pengaturan pin-pin arduino nano yang digunakan sebagai kontrol motor dc.



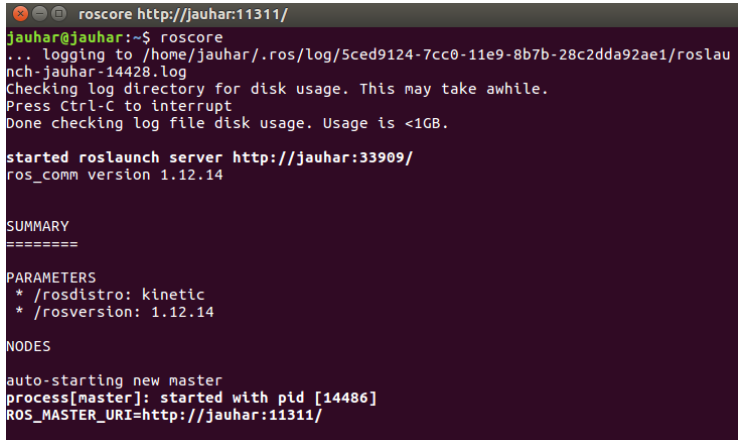
**Gambar 3.11** Desain skematik pada *software eagle*

Skematik board ini dibuat menggunakan *software eagle*. Setelah dibuat skematik dari setiap komponen dilakukan penataan komponen-komponen tersebut agar didapatkan desain yang efisien. Dalam melakukan desain PCB ini diatur sedemikian rupa sehingga ukuran dari board Arduino ini cukup untuk diletakkan pada base robot ini. Sehingga tidak terjadi dimensi yang berlebihan dari desain PCB ini.





melodic, kinetic, indigo, lunar dan lain lain. Dalam penelitian ini digunakan ROS versi kinetic. Untuk mengetahui apakah proses instalasi telah berhasil atau belum, maka dicoba dengan menjalankan command roscore pada terminal.



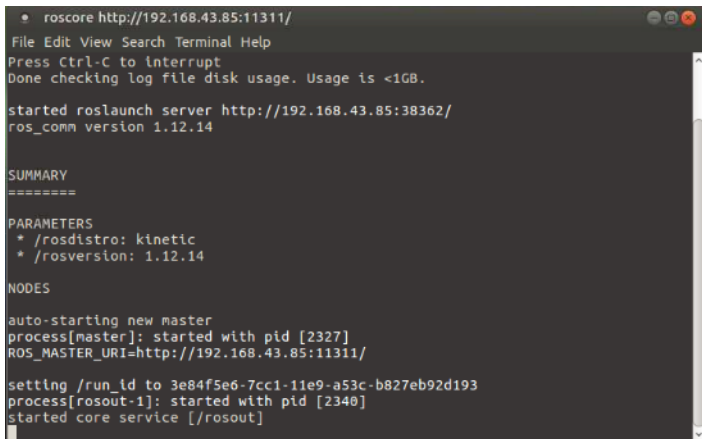
```
roscout http://jauhar:11311/
jauhar@jauhar:~$ roscout
... logging to /home/jauhar/.ros/log/5ced9124-7cc0-11e9-8b7b-28cdda92ae1/roslau
nch-jauhar-14428.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://jauhar:33909/
ros_comm version 1.12.14

SUMMARY
=====
PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.14

NODES
auto-starting new master
process[roscout]: started with pid [14486]
ROS_MASTER_URI=http://jauhar:11311/
```

**Gambar 3.13** Hasil instalasi ROS pada laptop



```
roscout http://192.168.43.85:11311/
File Edit View Search Terminal Help
Press Ctrl-C to Interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.43.85:38362/
ros_comm version 1.12.14

SUMMARY
=====
PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.14

NODES
auto-starting new master
process[roscout]: started with pid [2327]
ROS_MASTER_URI=http://192.168.43.85:11311/

setting /run_id to 3e84f5e6-7cc1-11e9-a53c-b827eb92d193
process[roscout-1]: started with pid [2340]
started core service [/roscout]
```

**Gambar 3.14** Hasil instalasi ROS pada Raspberry Pi 3b

### 3.4.2 Pengambilan Data Sensor YdLidar X4

Pengambilan data jarak dan sudut dari sensor LIDAR didapatkan dengan cara komunikasi serial antara YdLidar x4 dengan Raspberry Pi 3b. Baudrate yang digunakan adalah 128000 dengan 8 data bit, 1 stop bit dan no parity. Berdasarkan datasheet YdLidar X4 untuk mengambil data dan mengendalikan LIDAR terdapat beberapa macam *command* yang harus dikirimkan melalui serial. Pemrograman pengambilan data lidar menggunakan perpustakaan dari produsen, yang kemudian disesuaikan parameter yang berhubungan dengan versi YdLidar X4. Penyesuaian parameter diatur pada file .launch. Berikut pengaturan parameter yang diaplikasikan dalam file lidar.launch pada raspberry pi 3b.

```
<launch>
  <node name="ydlidar_node" pkg="ydlidar"
type="ydlidar_node" output="screen" respawn="false" >
    <param name="port" type="string"
value="/dev/ydlidar"/>
    <param name="baudrate" type="int"
value="128000"/>
    <param name="frame_id" type="string"
value="laser"/>
    <param name="low_exposure" type="bool"
value="false"/>
    <param name="resolution_fixed" type="bool"
value="true"/>
    <param name="auto_reconnect" type="bool"
value="true"/>
    <param name="reversion" type="bool"
value="false"/>
    <param name="angle_min" type="double" value="-
180" />
    <param name="angle_max" type="double"
value="180" />
    <param name="range_min" type="double"
value="0.12" />
```

```

        <param name="range_max" type="double"
value="10.0" />
        <param name="ignore_array" type="string" value=""
/>
        <param name="samp_rate" type="int"
value="10"/>
        <param name="frequency" type="double"
value="7"/>
    </node>
</launch>

```

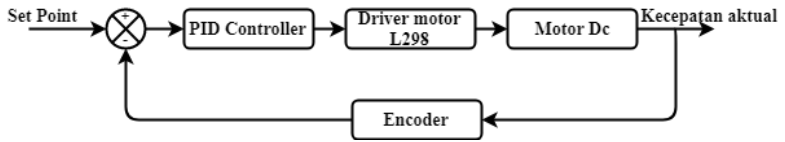
### 3.4.3 Kontrol Motor DC

Pada penelitian ini kecepatan motor dikontrol menggunakan kontrol PID. Keluaran sinyal kontrol berupa sinyal *pulse width modulation* (PWM). Untuk membangkitkan sinyal PWM, digunakan fungsi `analogWrite()` pada arduino nano. Pin yang digunakan untuk mengeluarkan sinyal PWM ini seperti pada tabel 3.1 diatas. Arah putaran motor dikontrol dengan memberikan sinyal keluaran yang sesuai dengan logika pada tabel 3.2.

**Tabel 3.2** Logika kontrol arah motor.

Motor 1		Motor 2		Motor 3		Keterangan
0	0	0	0	0	0	Berhenti
0	1	0	1	0	1	Berputar searah jarum jam
1	0	1	0	1	0	Berputar berlawanan jarum jam
1	1	1	1	1	1	Berhenti

Implementasi kontrol PID untuk kecepatan motor dc berdasarkan pada blok diagram gambar 3.15



**Gambar 3.15** Blok Diagram PID kecepatan motor

### 3.4.4 Konfigurasi ROS Master

Dalam robot operating system sangat memungkinkan komunikasi antar komputer melalui jaringan nirkabel. Komunikasi tersebut dapat dilakukan dengan ketentuan ada komputer yang dijadikan master dan dalam satu jaringan yang sama. Dalam robot ini yang bertindak sebagai sebagai master adalah Raspberry Pi 3b dan jaringan yang digunakan adalah hotspot dari *smartphone*. Konfigurasi pertama untuk melakukan komunikasi adalah mencatat alamat ip pada setiap komputer. Dengan memberikan comand `ifconfig` pada terminal maka alamat ip dari masing-masing komputer dapat dilihat. Adapun alamat ip dari raspberry pi 3b adalah 192.168.43.85 sedangkan alamat ip dari laptop sendiri adalah 192.168.43.200. setelah alamat ip dari masing-masing komputer diketahui dilakukan konfigurasi pada file `.bashrc`. Berikut syntax untuk melakukan konfigurasi tersebut.

- Konfigurasi ros master pada raspberry pi 3b  
`export ROS_MASTER_URI=http://192.168.43.85`  
`export ROS_IP=192.168.43.85`
- Konfigurasi ros node pada laptop  
`export ROS_MASTER_URI=http://192.168.43.85`  
`export ROS_IP=192.168.43.200`

### 3.4.5 Instalasi ROS Package Hector SLAM

ROS bersifat open source, sehingga banyak package-package yang telah dikembangkan oleh pengembang lain yang dapat dijadikan dasar untuk pengembangan lebih lanjut. Hector SLAM merupakan salah satu package yang dapat digunakan untuk aplikasi pemetaan ruangan berbasis LIDAR tanpa

menggunakan odometri roda. Hector SLAM dikembangkan oleh Universitas Teknik Darmstadt Jerman.

Proses instalasi package hector SLAM dengan mengikuti langkah-langkah yang ada pada website ros.org. Setelah proses instalasi selesai dilakukan pengaturan parameter untuk pemetaan. Parameter pemetaan yang harus dimasukkan pada hector ada pada tabel 3.3.

**Tabel 3.3** Parameter hector SLAM.

Parameter	Keterangan
Map resolution	Resolusi dari peta dalam satuan meter.
Map size	Ukuran dari peta dengan bentuk persegi.
Map update distance thresh	Ambang batas jarak yang harus dilewati untuk melakukan pembaruan peta
Map update angle thresh	Ambang batas sudut yang dilalui untuk melakukan pembaruan peta
Update factor free	Probabilitas untuk pembaruan grid yang bebas halangan
Update factor occupied	Probabilitas untuk pembaruan grid yang mengenai halangan
Laser min dist	Jarak minimal lidar
Laser max dist	Jarak maksimal lidar
Scan subscriber queue size	Besar antrian dalam meminta data pemindaian

Pengaturan file .launch untuk parameter pemetaan berada pada folder hector\_mapping. Penamaan file tersebut adalah konfigurasi\_peta.launch, adapun isi dari file tersebut adalah sebagai berikut

```
<!-- Map size / start point -->
<param name="map_resolution" value="0.020"/>
<param name="map_size" value="$(arg map_size)"/>
<param name="map_start_x" value="0.5"/>
```

```

<param name="map_start_y" value="0.5" />
<param name="map_multi_res_levels" value="3" />
<!-- Map update parameters -->
<param name="update_factor_free" value="0.4"/>
<param name="update_factor_occupied" value="0.9" />
<param name="map_update_distance_thresh"
value="0.40"/>
<param name="map_update_angle_thresh" value="0.06" />

```

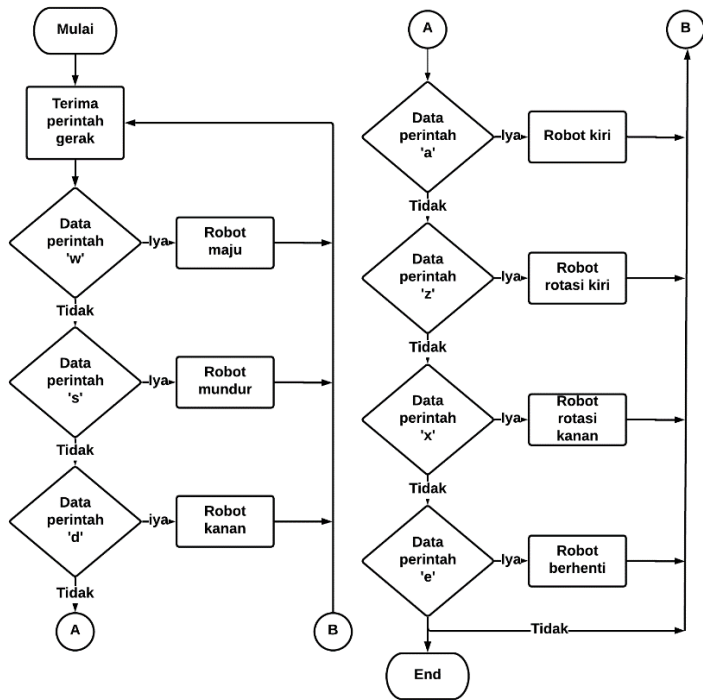
### 3.4.6 Kontrol Manual Pergerakan Robot

Untuk melakukan pemetaan ruangan secara manual robot dikontrol menggunakan perintah keyboard dari Raspberry Pi 3b ke Arduino Nano. Perintah tersebut dikirim secara serial. Kontrol manual ini bertujuan untuk mengoreksi hasil pemetaan ruangan tidak ada yang terlewatkan. Perintah yang dikirim dalam bentuk karakter. Daftar karakter yang dikirim adalah seperti yang tercantum pada tabel 3.4

**Tabel 3.4.** Data karakter yang dikirim.

Karakter yang dikirim	Perintah
w	Robot maju
s	Robot mundur
d	Robot kanan
a	Robot kiri
z	Robot rotasi kiri
x	Robot rotasi kanan
e	Robot berhenti

Alur diagram dari pengiriman perintah pada robot seperti pada gambar 3.16.



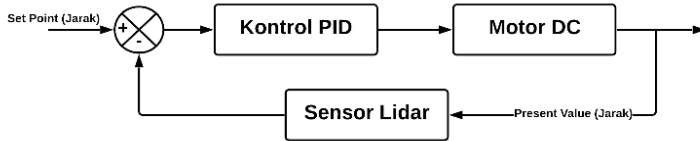
**Gambar 3.16** Alur diagram perintah manual.

Dalam Raspberry Pi 3b dilakukan pembacaan karakter *keyboard* yang ditekan dengan fungsi `getch()` dan kemudian dikirimkan secara serial dengan perpustakaan serial dan fungsi `serial.write()`. Selanjutnya data tersebut akan dibaca oleh Arduino Nano sebagai bentuk perintah untuk menggerakkan motor dc. Data tersebut diolah dalam fungsi `jalan()` dengan parameter data yang diterima dengan pemanfaatan *switch-case*.

### 3.4.7 Kontrol Otomatis Pergerakan Robot

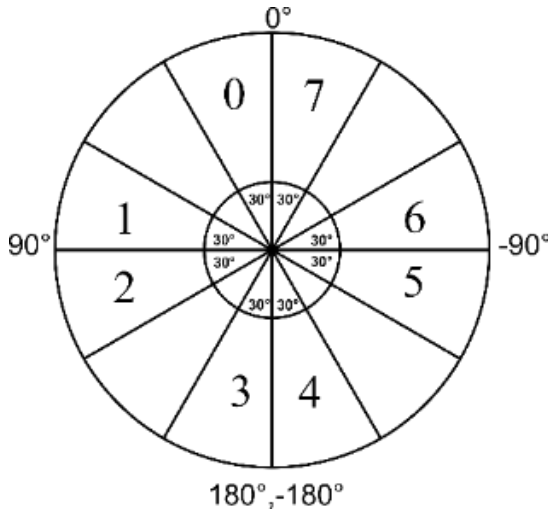
Metode yang digunakan dalam melakukan pergerakan robot secara otomatis adalah telusur dinding sebelah kanan. Dalam menjaga jarak antara robot dengan dinding agar selalu sesuai dengan

nilai jarak yang diinginkan digunakan kontrol PID. Data jarak didapatkan dari LIDAR yang dikirim secara serial ke Arduino Nano. Data tersebut kemudian diolah pada arduino nano.



**Gambar 3.17** Kontrol PID telusur ruangan.

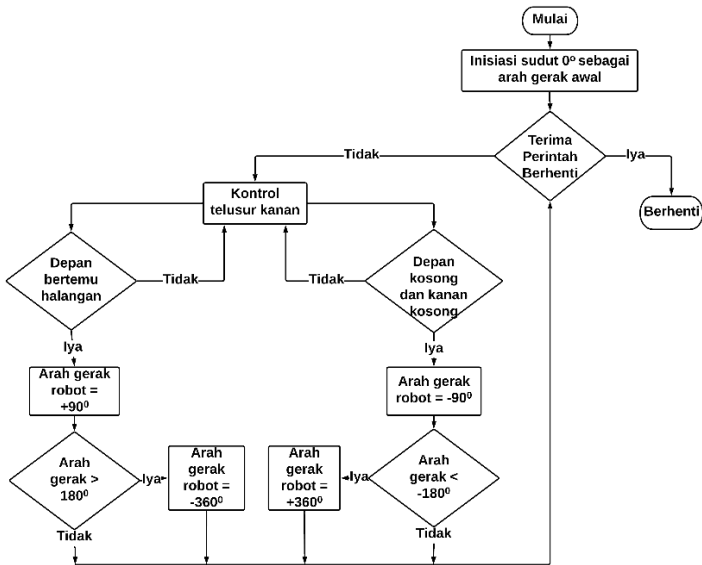
Untuk mengetahui jarak pada sisi kanan depan kiri dan belakang dilakukan pembagian sudut pada pembacaan LIDAR. Sudut dibagi menjadi dua belas bagian, dengan setiap bagian sebesar 30°. Dari 12 bagian sudut tersebut digunakan 8 bagian dan diberi indeks pada 8 bagian tersebut sebagaimana gambar 3.18.



**Gambar 3.18** Pembagian sudut LIDAR



Terdapat bagian sudut yang tidak digunakan yang berfungsi sebagai pemisah agar memudahkan dalam menentukan jarak depan kanan kiri dan belakang. Untuk memanfaatkan desain roda omni ini pergerakan robot dalam melakukan telusur ruangan digunakan tanpa belok. Robot bergerak hanya pada sumbu vertikal dan horizontal saja. Robot dapat bergerak pada sumbu robot  $0^\circ$ ,  $90^\circ$ ,  $-90^\circ$  dan  $180^\circ$  dalam menelusuri ruangan. Algoritma telusur ruangan secara keseluruhan sebagaimana terdapat pada gambar 3.19.



**Gambar 3.19** Alur diagram pergerakan otomatis

Untuk inisiasi arah gerak robot pertama kali adalah pada sudut  $0^\circ$ . Robot akan berjalan ketika mendapati perintah jalan dari pengguna dan dapat berhenti sewaktu-waktu pengguna ingin menghentikan robot. Untuk mengubah orientasi sudut robot ini terdapat beberapa syarat. Ketika jarak depan robot sudah lebih kecil dari ambang batas yang ditentukan maka robot akan berubah

orientasi arah gerak dengan menambahkan sudut sebesar 90°. Sedangkan ketika robot sudah kehilangan acuan dinding kanan, maka robot akan mengurangi orientasi arah gerak sebesar 90°.

### 3.4. Implementasi Pemetaan Ruang Menggunakan LIDAR

Terdapat beberapa metode yang digunakan untuk pembuatan peta 2D yang terintegrasi dalam ROS seperti google carthographer, gmapping dan hector slam. Pembuatan peta pada penelitian kali ini menggunakan ros package hector SLAM. Algoritma hector SLAM ini menempatkan posisi robot berdasarkan scan matching dan tidak menggunakan odometri roda yang merupakan metode umum dalam algoritma SLAM lainnya. LIDAR bertugas melakukan scan matching untuk menemukan robot dengan cepat dan akurat karena tingkat pembaruan yang tinggi. Algoritma Hector menggunakan metode minimisasi gaussian-newton yang dianggap sebagai pembaruan untuk metode newton. Data jarak dari LIDAR akan digunakan untuk menentukan halangan yang ditemui robot.

Halangan tersebut akan direpresantikan dalam pemetaan *occupancy grid*. Pada hector SLAM dalam membentuk menentukan hunian dalam grid digunakan probabilitas log-odd yang mengikuti aturan probabilitas bayes. Probabilitas bayes merupakan perhitungan probabilitas rekrusif yang memperhatikan nilai dari perhitungan yang sebelumnya. Jika  $p(m_{x,y})$  probabilitas hunian grid pada kordinat x dan y,  $p(z)$  merupakan probabilitas pengukuran LIDAR [22]. Maka persamaan probabilitas berdasarkan bayes adalah

$$p(m_{x,y}|z) = \frac{p(z|m_{x,y})p(m_{x,y})}{p(z)} \quad (3.1)$$

Persamaan bayesian untuk probabilitas hunian grid sama saat adanya halangan dan tidak adalah

$$p(m_{x,y} = 1|z) = \frac{p(z|m_{x,y}=1)p(m_{x,y}=1)}{p(z)} \quad (3.2)$$

$$p(m_{x,y} = 0|z) = \frac{p(z|m_{x,y}=0)p(m_{x,y}=0)}{p(z)} \quad (3.3)$$

Untuk menjadikan persamaan tersebut menjadi probabilitas log odd digunakan rumus odd terlebih dahulu untuk menghitung perbandingan dari probabilitas dari peta tersebut

$$odd = \frac{P \text{ terjadi}}{P \text{ tidak terjadi}} \quad (3.4)$$

$$odd = \frac{p(m_{x,y}=1|z)}{p(m_{x,y}=0|z)} \quad (3.5)$$

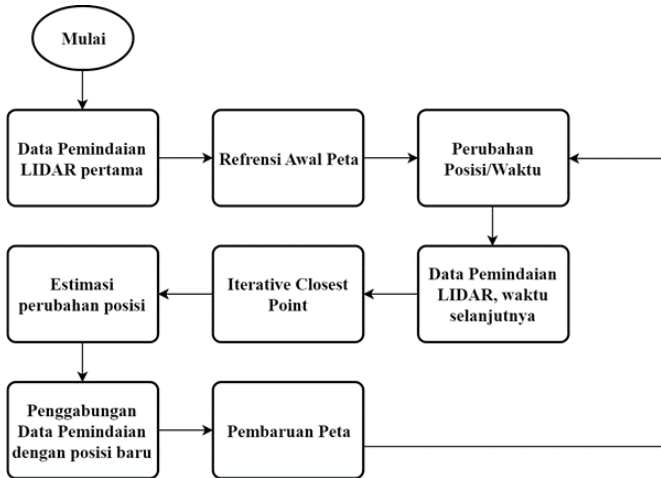
Dari persamaan (3.2) (3.3) dan (3.5) maka didapatkan persamaan sebagai berikut

$$\begin{aligned} \log Odd &= \log \frac{p(m_{x,y} = 1|z)}{p(m_{x,y} = 0|z)} = \log \frac{p(z|m_{x,y} = 1)p(m_{x,y} = 1)}{p(z|m_{x,y} = 0)p(m_{x,y} = 0)} \\ &= \log \frac{p(z|m_{x,y} = 1)}{p(z|m_{x,y} = 0)} + \log \frac{p(m_{x,y}=1)}{p(m_{x,y}=0)} \end{aligned} \quad (3.6)$$

$$\log odd^+ = \log odd \text{ pengukuran} + \log odd^- \quad (3.7)$$

Jika diketahui posisi robot  $\psi = (x, y, \theta)$  maka untuk mendapatkan kordinat dari halangan yang terdeteksi oleh objek pada jarak  $d$  adalah

$$\begin{bmatrix} x_{objek} \\ y_{objek} \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} d \\ 0 \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix} \quad (3.8)$$

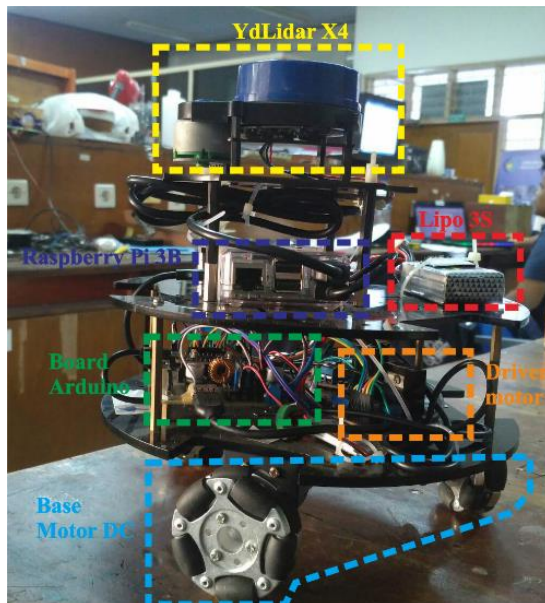


**Gambar 3.20** Alur diagram pembuatan peta dengan Hector SLAM [27].

## BAB IV

### PENGUJIAN DAN ANALISIS

Pada bab ini membahas tentang pengujian perancangan sebagaimana yang telah dibuat pada bab 3. Pengujian dilakukan untuk memberikan analisa dan evaluasi terhadap robot yang telah dibuat. Berdasarkan perancangan perangkat keras yang ada bab 3 dihasilkan realisasi desain dengan dimensi 24 cm x 24 cm x 25 cm.



**Gambar 4.1** Realisasi desain robot

#### 4.1 Pengujian Sensor YdLidar X4

Pada pengujian ini dilakukan pengukuran jarak jangkauan minimal hingga maksimal LIDAR terhadap objek. Pengujian dilakukan dengan mengukur jarak objek yang menjadi target dari sensor. Pengujian jarak ini berdasarkan rentang jarak yang dapat diukur yang tercantum dalam datasheet sensor ini. Pada tabel 4.1 ditunjukkan data hasil pembacaan jarak LIDAR dengan rentang jarak pengujian antara 0,5 – 12 m dengan rata-rata error pembacaan adalah 1,104%. Dengan error yang dihasilkan

pada pengujian ini, sensor ini masih cukup bagus untuk digunakan dalam pembuatan peta ruangan.

Ketika lidar mengukur jarak yang lebih besar dari 10,5 m sensor ini tidak dapat membaca jarak dari objek. Sesuai dengan datasheet, sensor ini memiliki pengukuran jarak maksimal  $\pm 10$  m. Pada tabel 4.2 ditunjukkan data hasil pembacaan jarak minimal objek yang dapat dibaca oleh LIDAR, dengan rentang jarak pengujian antara 0 - 0,24 m dengan rata-rata error yang pembacaan adalah 0,59%. Dengan demikian didapatkan hasil bahwa jarak minimal LIDAR dapat mengukur jarak objek adalah 0,12 m. Hasil ini sesuai dengan datasheet sensor ini.

**Tabel 4.1** Pengukuran jarak maksimal.

Jarak (m)	Jarak terbaca (m)	Error (%)
0,5	0,510	2
1,5	1,518	1,2
2,5	2,509	0,36
3,5	3,517	0,48
4,5	4,531	0,68
5,5	5,525	0,45
6,5	6,561	0,93
7,5	7,606	1,4
8,5	8,600	1,17
9,5	9,679	1,88
10,5	10,662	1,54
11,5	0	-
12,5	0	-
Rata-rata error		1,104

**Tabel 4.2.** Pengukuran jarak minimal.

Jarak (m)	Jarak terbaca (m)	Error (%)
0 - 0,11	0	-
0,12	0,121	0,83
0,16	0,161	0,63
0,20	0,201	0,50
0,24	0,241	0,42
Rata-rata Error		0,59



**Gambar 4.2.** Proses pengambilan data jarak.

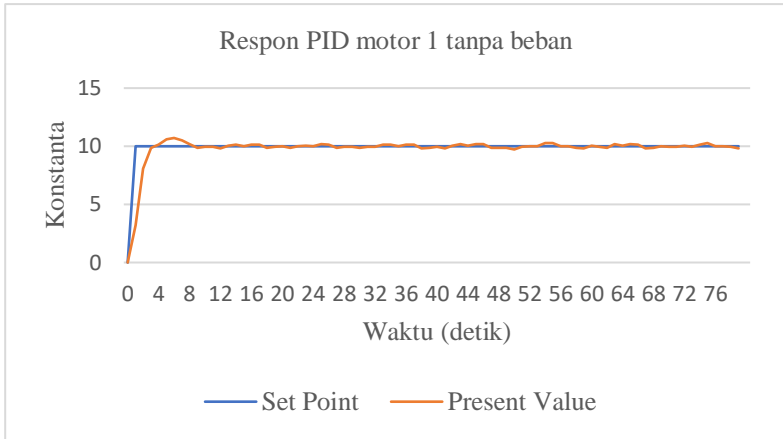
## 4.2 Pengujian Kontrol Motor DC

Pada pengujian ini dilakukan *tuning* secara manual nilai dari parameter kontrol PID untuk mendapatkan respon yang bagus. Dari beberapa percobaan nilai kp, ki, dan kd didapatkan nilai parameter pada tabel 4.3.

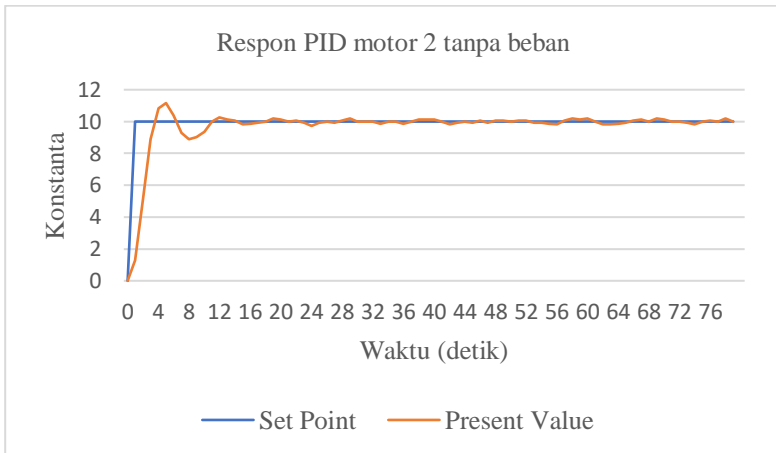
**Tabel 4.3** Parameter PID setiap motor.

Parameter PID								
Motor 1			Motor 2			Motor 3		
KP	KI	KD	KP	KI	KD	KP	KI	KD
5	2,8	2,5	4	2,8	2,5	4	2,8	2,5

Data respon PID direpresentasikan dengan bantuan grafik. Berikut gambar grafik dari masing-masing respon pada setiap motor.

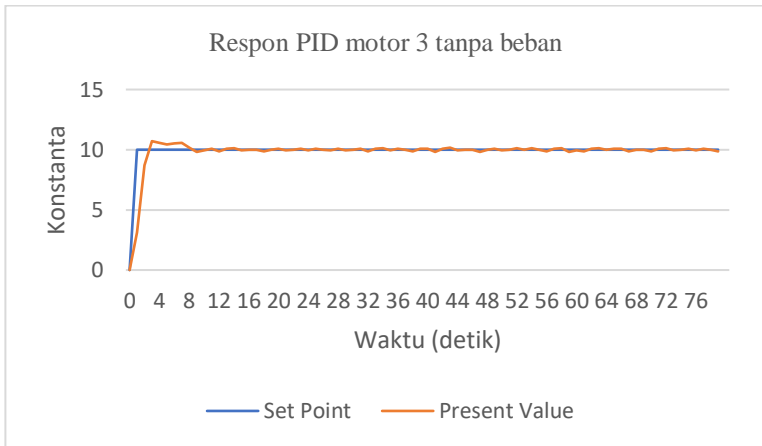


**Gambar 4.3** Grafik respon PID motor 1.



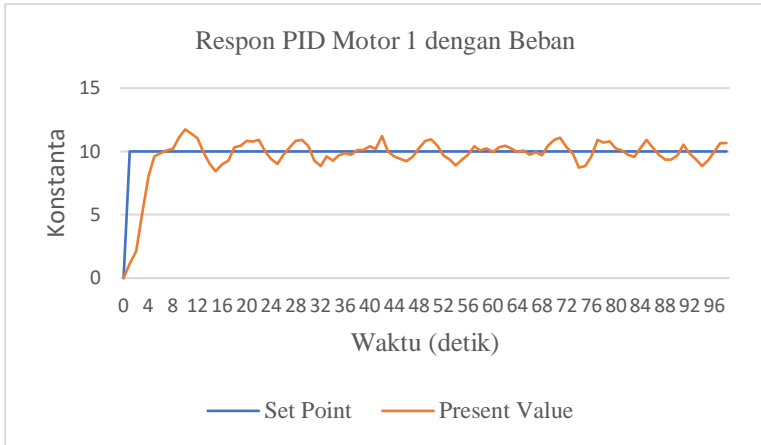
**Gambar 4.4** Grafik respon PID motor 2.



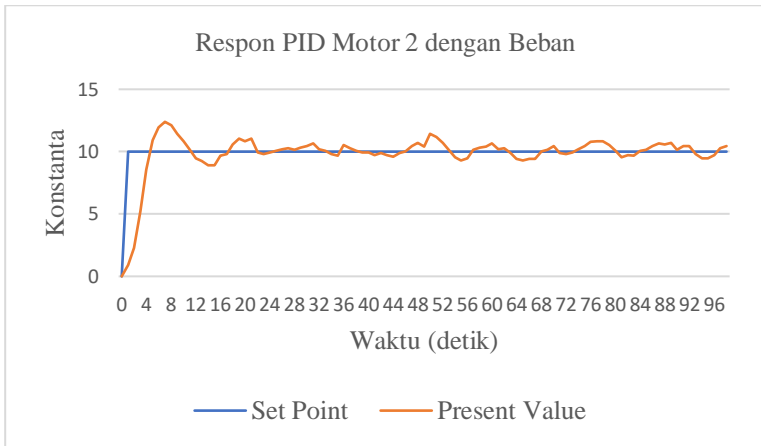


**Gambar 4.5** Grafik respon PID motor 3.

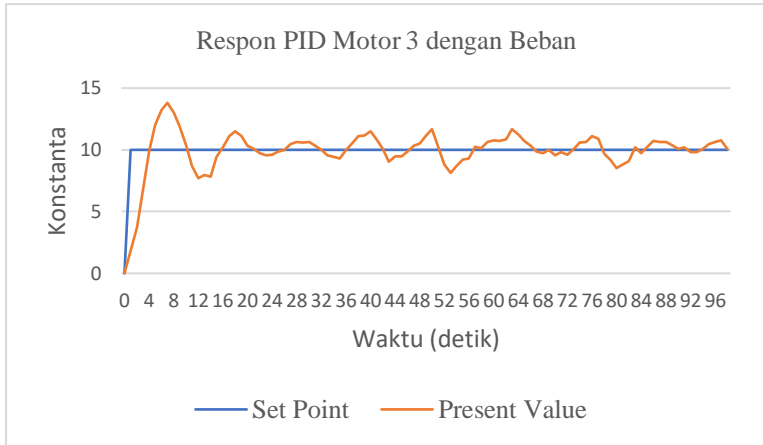
Berdasarkan grafik yang dihasilkan, terlihat bahwa kontrol PID motor tanpa beban sudah cukup baik. Perbedaan respon pada motor sangat dipengaruhi oleh karakteristik dari setiap motor. Pada motor 2 terlihat bahwa respon motor masih mengalami *overshoot*. Setelah mendapatkan parameter PID pada saat motor tanpa beban maka dilakukan pengujian PID dengan beban. Berdasarkan grafik PID motor dengan beban terlihat bahwa respon masih mengalami osilasi sehingga pergerakan robot masih belum bisa stabil lurus.



**Gambar 4.6** Grafik PID motor 1 dengan beban.



**Gambar 4.7** Grafik PID motor 2 dengan beban.



**Gambar 4.8** Grafik PID motor 3 dengan beban.

### 4.3 Pengujian Kecepatan Robot

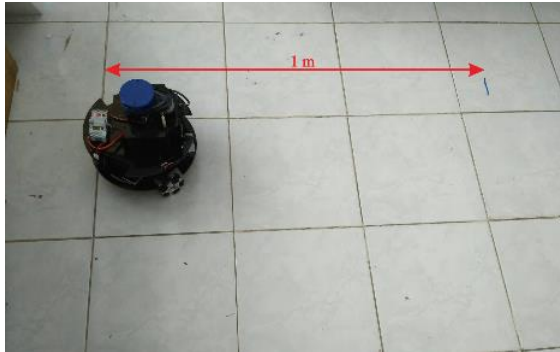
Pengujian selanjutnya ada bertujuan untuk mencari hubungan antara konstanta kecepatan yang diinputkan dalam program dengan kecepatan robot yang sebenarnya. Pengujian ini dilakukan dengan cara menjalankan robot sejauh 1 meter dan mencatat waktu yang ditempuh selama jarak 1 meter tersebut. Hasil dari pengujian ini tercatat pada tabel 4.4

**Tabel 4.4** Hubungan konstanta dengan kecepatan robot.

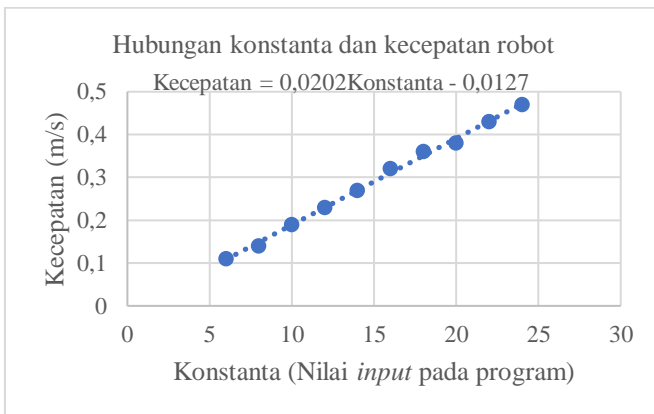
Konstanta (K)	Waktu (detik)	Kecepatan (m/s)
6	9,26	0,11
8	6,96	0,14
10	5,14	0,19
12	4,34	0,23
14	3,68	0,27
16	3,17	0,32
18	2,74	0,36
20	2,61	0,38
22	2,30	0,43
24	2,13	0,47

Data pada tabel 4.4 ditampilkan dalam grafik pada gambar 4.9. Dari grafik tersebut terlihat hubungan linear antara konstanta dengan kecepatan sebenarnya. Serta didapatkan hubungan konstanta dalam program dengan kecepatan robot yang sebenarnya sesuai dengan persamaan linear.

$$\text{Kecepatan robot} = 0,0202K - 0,0127 \quad (4.1)$$



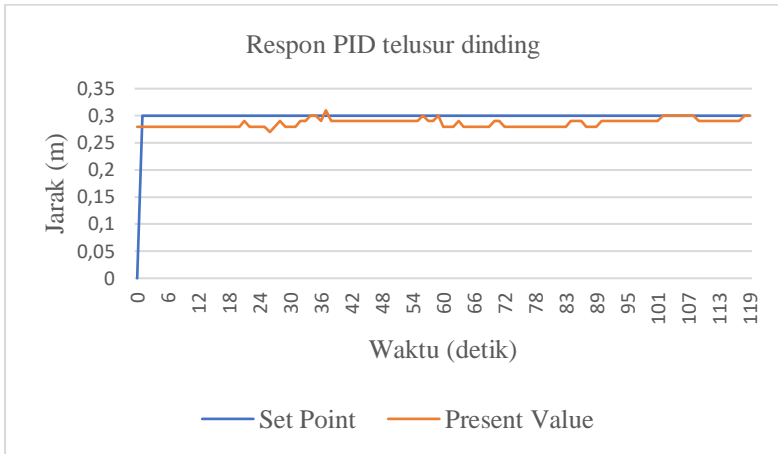
**Gambar 4.9.** Proses pengujian kecepatan robot



**Gambar 4.10** Grafik hubungan konstanta dengan kecepatan

#### 4.4 Pengujian Otomatis Telusur Dinding Robot

Pada pengujian ini dilakukan respon dari robot untuk mempertahankan posisi berdasarkan *set point* jarak dengan kontrol PID yang digunakan. Parameter PID didapatkan dengan metode *tuning* secara manual. Dari proses *tuning* tersebut didapatkan parameter PID dengan  $k_p = 20$ ,  $k_i = 0$ , dan  $k_d = 1000$ .



**Gambar 4.11** Grafik PID telusur dinding.

Pergerakan robot dalam menelusuri dinding sudah cukup bagus meskipun masih terdapat *error steady state*. Sesuai dengan tujuan kontrol ini agar robot tidak terlalu jauh atau terlalu dekat dengan meminimalisir perubahan pergerakan robot. Karena perubahan yang drastis akan mengakibatkan kesalahan dalam pembuatan peta.

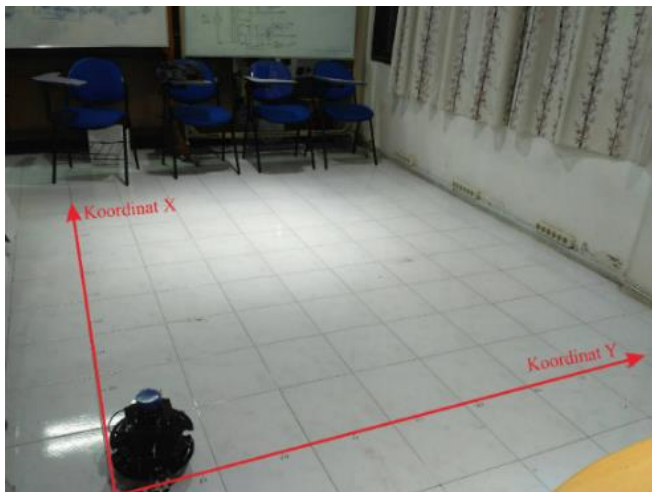
#### 4.5 Pengujian Posisi Robot dengan *Scan Matching*

Hector slam dalam menentukan posisi robot tanpa menggunakan data odometri roda dari suatu robot melainkan menggunakan *scan matching*. Pada pengujian ini dilakukan perbandingan posisi robot dengan scan matching dan posisi robot yang sebenarnya pada sumbu xy. Pengujian dilakukan dengan cara menjalankan robot dengan kontrol keyboard sejauh jarak yang ditentukan, kemudian dilakukan analisa.

**Tabel 4.5** Data pengujian posisi robot

Posisi sebenarnya (m)		Posisi yang terukur (m)		Error (%)	
X	Y	X	Y	X	Y
0,25	0,25	0,26	0,23	4	8
0,50	0,50	0,52	0,42	4	16
0,75	0,75	0,76	0,73	1,3	2,6
1,00	1,00	0,98	0,96	2	4
1,25	1,25	1,23	1,23	1,6	1,6
1,50	1,50	1,42	1,45	5,3	3,3
1,75	1,75	1,71	1,70	2,2	2,8
2,00	2,00	1,98	1,95	1	2,5
Rata-rata error				2,69	5,11

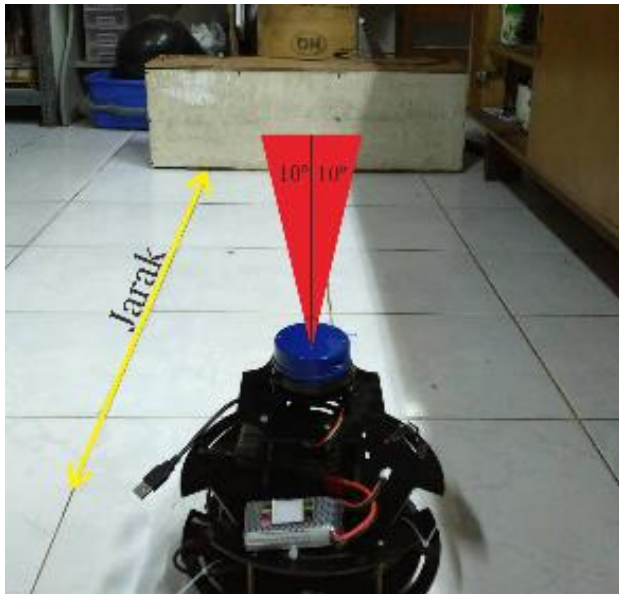
Berdasarkan data yang ada pada tabel 4.4. didapatkan rata-rata error pada sumbu-x sebesar 2,69 % dan rata-rata error pada sumbu-y sebesar 5,11 %. Karena penentuan posisi pada Hector SLAM menggunakan metode *scanning matching* maka error dari pembacaan sensor LIDAR akan mempengaruhi posisi robot. Data posisi ini akan digunakan sebagai pengolahan data peta.



**Gambar 4.12.** Proses pengambilan data posisi

#### 4.6 Pengujian Peta *Occupancy Grid*

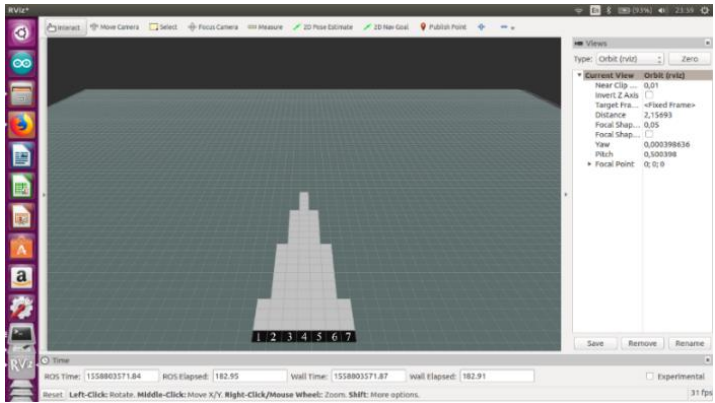
Pengujian ini dilakukan untuk mengetahui pembentukan peta *occupancy grid* apakah sudah benar atau masih terdapat kesalahan. Pengujian dilakukan dengan cara melakukan pemindaian pada LIDAR sebesar  $20^\circ$  dengan jarak uji robot dengan halangan sebesar 1 m, 1,5 m dan 2 m. Panjang dari halangan sendiri adalah 0,9 m. Resolusi yang diberikan untuk setiap grid adalah 5 cm. Ketika sensor LIDAR mengenai halangan maka grid akan berwarna hitam dan warna abu-abu untuk daerah yang tanpa halangan. Dan daerah selain itu merupakan grid yang tidak diketahui.



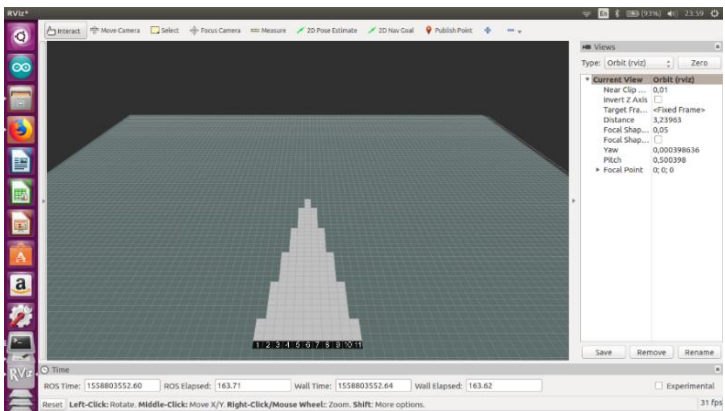
**Gambar 4.13.** Pengujian *Occupancy Grid*

Secara teori panjang halangan yang terukur oleh LIDAR sesuai dengan persamaan trigonometri segitiga, yaitu

$$\text{Panjang halangan terukur} = 2 \times \text{jarak robot ke halangan} \times \tan 10^\circ$$

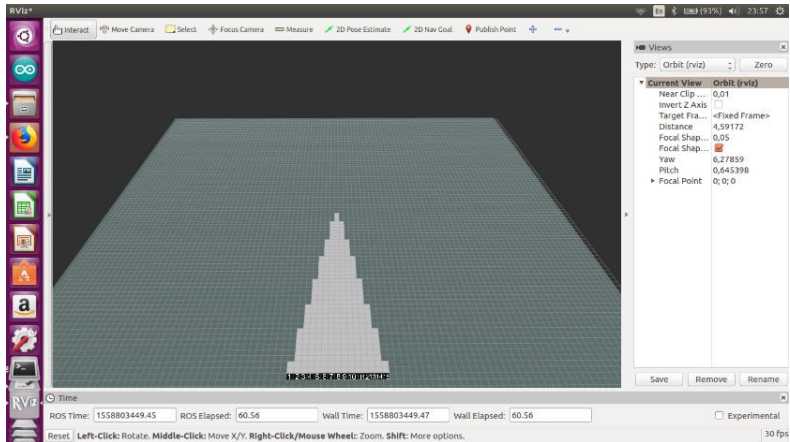


**Gambar 4.14** Pengujian Pada Jarak 1 m



**Gambar 4.15** Pengujian Pada Jarak 1,5 m





**Gambar 4.16.** Pengujian Pada Jarak 2 m

Berdasarkan gambar yang dihasilkan dari pengujian dibuatlah tabel sebagai berikut:

**Tabel 4.6** Hasil peta *occupancy grid*

Jarak	Jumlah grid hitam	Panjang halangan secara teori	Panjang halangan pada peta	Error (%)
1	7	0,352	0,35	0,568
1,5	11	0,528	0,55	4,166
2	15	0,705	0,75	6,382
Rata-rata error				3,705

#### 4.7 Pengaruh Kecepatan pada Kontrol Robot dan Pembuatan Peta

Pada pengujian ini dilakukan percobaan dengan variasi besarnya kecepatan pada robot. Setelah dilakukan percobaan tersebut didapatkan data sebagaimana pada tabel 4.7.

**Tabel 4.7** Hasil pengujian dengan variasi kecepatan.

Konstanta Kecepatan	Pemetaan Ruangan	Respon Gerak Robot
8	Berhasil	Tidak terjadi tabrakan dan menelusuri ruangan dengan baik
10	Berhasil	Tidak terjadi tabrakan dan menelusuri ruangan dengan baik
12	Berhasil	Tidak terjadi tabrakan dan menelusuri ruangan dengan baik
14	Berhasil	Terjadi tabrakan dan kontrol telusur ruangan sedikit kacau
16	Berhasil	Terjadi tabrakan dan kontrol telusur ruangan sedikit kacau
18	Berhasil	Terjadi tabrakan dan kontrol telusur ruangan bertambah kacau
20	Berhasil	Terjadi tabrakan dan kontrol telusur ruangan bertambah kacau
22	Gagal	Terjadi tabrakan dan kontrol telusur ruangan sangat kacau
24	Gagal	Terjadi tabrakan dan kontrol telusur ruangan sangat kacau

Berdasarkan data pada tabel 4.7. pengaruh kecepatan akan berdampak pada kontrol pergerakan robot dalam menelusuri ruangan. Ketika pergerakan robot sudah tidak beraturan maka data pemindaian LIDAR yang akan diolah menjadi peta menjadi tidak beraturan juga.

Frekuensi pemindaian sensor YdLidar X4 adalah 7 Hz, maka periode dari sensor tersebut adalah 0,14 detik sehingga waktu yang digunakan pengambilan data lidar tidak boleh lebih cepat dari periode sensor ini. Pada hector SLAM pembaruan data dilakukan ketika sudah melebihi ambang batas yang ditentukan oleh pengguna. Pada tugas akhir ini ambang batas tersebut sebesar 10 cm. Berdasarkan perhitungan kecepatan robot maka didapatkan waktu tempuh pada masing-masing kecepatan pada jarak 10 cm sebagaimana tabel 4.8.

**Tabel 4.8** Waktu tempuh robot pada jarak 10 cm.

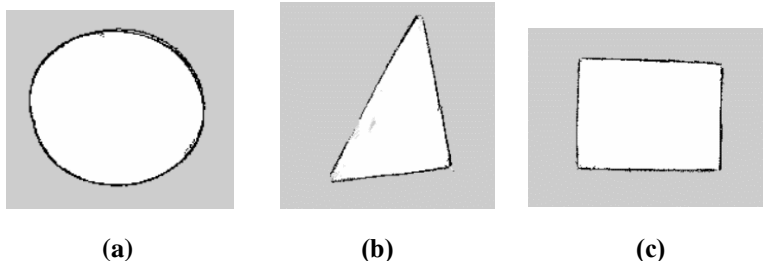
Konstanta Kecepatan	Waktu tempuh (detik)	Melembi Periode LIDAR
8	0,71	Tidak
10	0,52	Tidak
12	0,43	Tidak
14	0,37	Tidak
16	0,31	Tidak
18	0,27	Tidak
20	0,26	Tidak
22	0,23	Tidak
24	0,21	Tidak

Berdasarkan perhitungan maka seharusnya ketika robot bergerak pada kecepatan maksimal maka frekuensi pemindaian dari sensor LIDAR masih mampu dalam melakukan pemetaan ruangan. Pada pengujian ini tidak didapatkan kecepatan yang melebihi dari frekuensi LIDAR dikarenakan kecepatan robot telah saturasi pada konstanta 24.

#### **4.8 Pengujian Variasi Bentuk Geometri Ruang**

Pada pengujian ini bertujuan untuk mengetahui apakah robot mampu bergerak pada berbagai bentuk variasi ruangan serta memberikan informasi tentang bentuk ruangan. Pengujian pertama dilakukan dengan bentuk geometri dari ruangan adalah lingkaran dengan diameter 118 cm. Pada pengujian ini bentuk ruangan yang dihasilkan sesuai dengan ruangan aslinya. Namun pergerakan robot tidak bergerak dengan menelusuri mengikuti bentuk dinding dari ruangan, dikarenakan algoritma yang digunakan untuk otomatis ini robot hanya bergerak pada arah vertikal dan horizontal. Pada pengujian selanjutnya dengan variasi bentuk geometri berupa segitiga dengan ukuran 192 cm x 158 cm x 122,5 cm. Pada pengujian robot berhasil membentuk peta sesuai dengan aslinya dan robot dapat bergerak pada ruangan tersebut tanpa menabrak dinding. Pada pengujian selanjutnya berupa ruangan berbentuk persegi dengan ukuran 158 cm x 122,5 cm. Pada ruangan persegi tersebut robot bergerak tanpa

menabrak dinding serta mampu memberikan informasi dari bentuk ruangan tersebut.



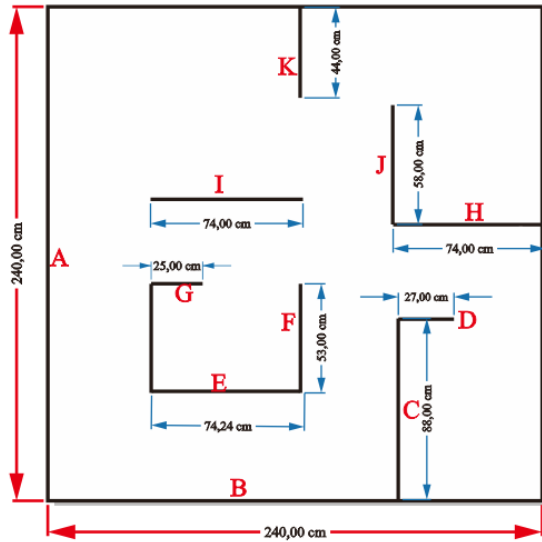
**Gambar 4.17** Variasi bentuk geometri ruangan (a) Hasil ruangan bentuk lingkaran (b) Hasil ruangan bentuk segitiga (c) Hasil ruangan bentuk persegi.

## 4.9 Pengujian Pemetaan Ruangan

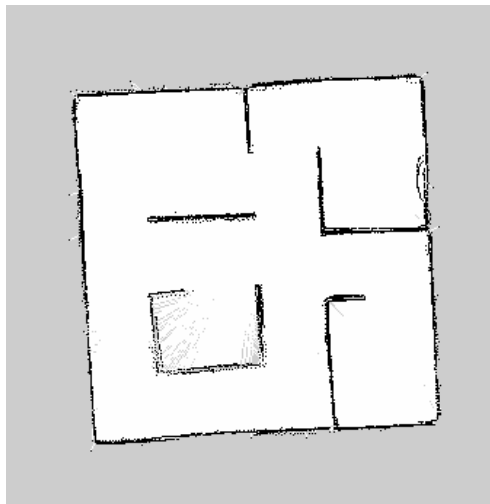
Pengujian dilakukan dengan menggunakan ruangan buatan yang memiliki ukuran 2,4 m x 2,4 m dan selanjutnya dilakukan uji coba pada ruangan sesungguhnya dengan ukuran 12 m x 8,4 m. Robot dalam melakukan tugas memetakan ruangan robot dikontrol secara manual melalui keyboard pada laptop dan pergerakan otomatis. Robot dijalankan untuk menelusuri seluruh lorong-lorong ruangan yang ada. Data LIDAR dikirim ke laptop untuk kemudian diolah dan ditampilkan dalam tools rviz. Semua proses data yang dilakukan selama pemetaan ruangan ini disimpan dalam rosbag, yang kemudian dilakukan analisa terhadap peta yang telah dibentuk.

### 4.9.1 Ruangan buatan

Pada percobaan ini robot dijalankan otomatis dalam menelusuri seluruh ruangan sesuai dengan desain pada gambar 4.17 dan 4.18.



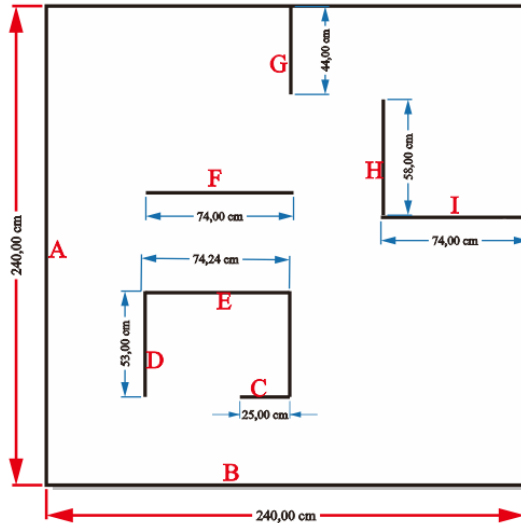
**Gambar 4.18** Desain ruangan percobaan 1



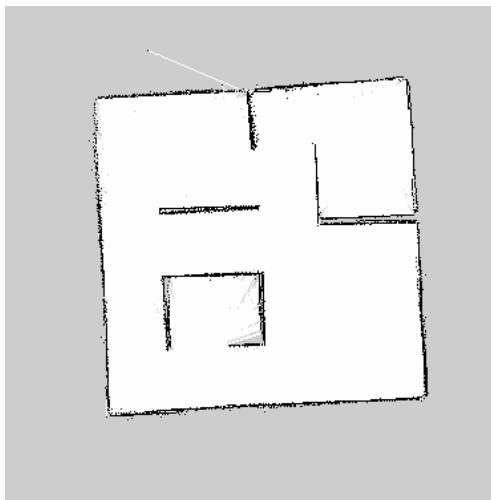
**Gambar 4.19** Hasil gambar pemetaan percobaan 1

**Tabel 4.9** Hasil ukuran pemetaan percobaan 1.

Indeks dinding	Panjang dinding sebenarnya (m)	Panjang dinding yang terukur (m)	Error (%)
A	2,4	2,36	1,66
B	2,4	2,36	1,66
C	0,88	0,78	11,36
D	0,27	0,29	7,41
E	0,74	0,71	4,05
F	0,53	0,62	16,98
G	0,25	0,28	12,00
H	0,74	0,64	13,51
I	0,74	0,73	1,35
J	0,58	0,57	1,72
K	0,44	0,39	11,36
Rata-rata error (%)			7,55



**Gambar 4.20** Desain ruangan percobaan 2



**Gambar 4.21** Hasil gambar pemetaan percobaan 2

**Tabel 4.10** Hasil ukuran pemetaan percobaan 2.

Indeks dinding	Panjang dinding sebenarnya (m)	Panjang dinding yang terukur (m)	Error (%)
A	2,4	2,32	3,33
B	2,4	2,38	0,83
C	0,25	0,26	4,00
D	0,53	0,55	3,77
E	0,74	0,88	18,91
F	0,74	0,75	1,35
G	0,44	0,43	2,27
H	0,58	0,65	12,06
I	0,74	0,73	1,35
Rata-rata error (%)			5,32

Pada percobaan 1 dihasilkan rata-rata error sebesar 7,55 % dan pada percobaan 2 dihasilkan error sebesar 5,32% maka

dihasilkan error rata-rata dari kedua percobaan tersebut sebesar 6,43% pada hasil ukuran peta yang dihasilkan.

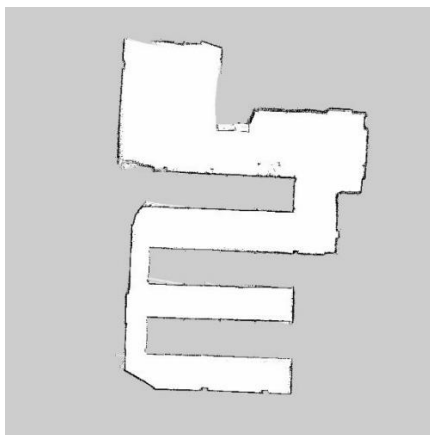
#### 4.9.2 Ruangannya

Pada pengujian ini robot dijalankan pada mode manual dan otomatis. Pengujian ini bertujuan untuk mengetahui apakah robot dapat melakukan pemetaan dengan ruangan pada mode manual dan otomatis. Selain itu pengujian ini juga dapat digunakan sebagai evaluasi untuk penerapan pada ruangan yang sesungguhnya. Pengujian dilakukan di gedung Departemen Teknik Elektro ITS Laboratorium Elektronika Industri B402.

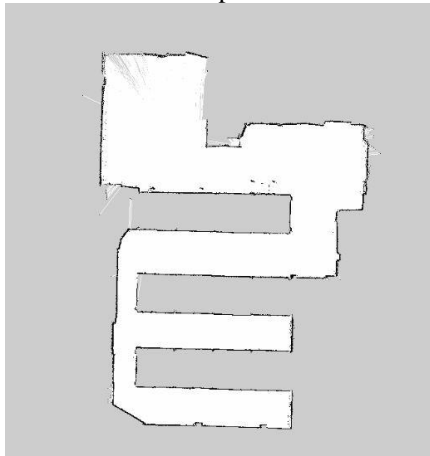
**Tabel 4.11** Pengujian dengan ruangan asli.

Indeks dinding	Panjang dinding sebenarnya (m)	Panjang dinding yang terukur (m)		Error (%)	
		Manual	Otomatis	Manual	Otomatis
A	1,16	1,17	1,13	0,86	2,59
B	4,61	4,52	4,44	1,95	3,69
C	1,15	1,14	1,16	0,87	0,87
D	4,84	4,81	4,79	0,62	1,03
E	1,22	1,21	1,21	0,82	0,82
F	4,84	4,61	4,54	4,75	6,20
G	4,84	4,77	4,78	1,45	1,24
H	1,10	1,08	1,13	1,82	2,73
I	4,84	4,79	4,79	1,03	1,03
J	1,25	1,32	1,15	5,06	8,00
K	0,71	0,81	0,79	14,08	11,27
L	4,84	4,68	4,91	3,31	1,45
M	6,26	6,16	6,21	1,60	0,80
N	2,06	2,05	1,97	0,49	4,37
O	1,25	1,24	1,31	0,80	4,80
P	0,71	0,78	0,77	9,86	8,45
Q	2,63	2,64	2,66	0,38	1,14
R	3,85	3,71	3,58	3,64	7,01
S	0,73	0,69	0,63	5,48	13,70
T	1,24	1,04	1,18	16,13	4,84
U	5,80	5,91	5,72	1,90	1,38
V	4,34	4,01	4,26	7,60	1,84
W	3,00	2,69	2,45	10,33	18,33
X	3,24	3,22	3,32	00,62	2,47
Rata-rata error (%)				4,00	4,59





**Gambar 4.22** Hasil peta mode otomatis.



**Gambar 4.23** Hasil peta mode manual.



**Gambar 4.24** Ruangn B402

Dari percobaan data pergerakan robot manual dan otomatis didapatkan masing masing error pengukuran 4,00 % dan 4,59 %. Berdasarkan data error tersebut perbedaan error yang didapatkan tidak begitu jauh. Robot dapat membentuk peta dalam mode manual dan otomatis. Namun pada mode otomatis akan mengalami kekurangan dalam pembuatan peta apabila robot terus menelusuri dinding yang sama.

## **BAB V**

### **PENUTUP**

#### **5.1. Kesimpulan**

Pada penelitian ini telah dilakukan perancangan *mobile robot* yang mampu memberikan informasi tentang pemetaan ruangan menggunakan LIDAR. Algoritma yang digunakan adalah Hector SLAM yang merupakan salah satu dari *package* ROS. Sistem gerak pada robot menggunakan 3 roda omni. Robot dapat bergerak secara manual atau otomatis dalam menelusuri ruangan. Pada pengujian pembacaan jarak LIDAR data yang dihasilkan memiliki error yang kecil. Jangkauan jarak yang dapat dibaca sensor dapat digunakan untuk memetakan ruangan yang luas. Namun sensor ini memiliki keterbatasan pada jarak minimal. Pada pengujian kontrol kecepatan motor ketika motor mendapatkan beban, respon pid yang dihasilkan masih mengalami osilasi. Pergerakan robot yang dilakukan secara otomatis dapat mengikuti dinding dan menjaga jarak robot dengan dinding sesuai dengan set point dengan kontrol pid. Robot bergerak pada arah horizontal dan vertikal tanpa mengubah orientasi arah. Kontrol pergerakan robot dapat dipengaruhi oleh kecepatan robot, dimana semakin cepat robot bergerak maka kontrol akan semakin jelek. Dan hal ini berdampak pada pembuatan peta sehingga pembuatan peta bisa gagal.

Pengujian algoritma Hector SLAM dalam menentukan posisi robot dengan menggunakan *scan matching* didapatkan rata-rata error pada sumbu-x sebesar 2,6% dan sumbu-y sebesar 5,1%. Pembentukan peta pada penelitian ini menggunakan pemetaan yang berdasarkan peta *occupancy grid* yang dapat membentuk peta melalui nilai probabilitas dari *grid*. Pada pengujian Pembentukan peta ruangan buatan didapatkan rata-rata error sebesar 6,45%. Dan pada pengujian pemetaan ruangan laboratorium B402 robot dapat membentuk peta pada mode manual dan otomatis dengan perbedaan error 0,49%. Sehingga dapat disimpulkan bahwa dalam melakukan pemetaan ruangan baik manual ataupun otomatis dapat memberikan informasi yang tidak jauh berbeda.

## 5.2. Saran

Metode yang digunakan dalam pergerakan robot untuk menelusuri ruangan pada mode otomatis masih terlalu sederhana sehingga perlu dilakukan pembaruan pada algoritma tersebut. Misalkan saja dapat digunakan metode *frontier based* ataupun metode yang lainnya.

## DAFTAR PUSTAKA

- [1] D. Shen, Y. Huang, Y. Wang, and C. Zhao, “Research and Implementation of SLAM Based on LIDAR for Four-Wheeled Mobile Robot,” p. 5.
- [2] M. G. Ocando, N. Certad, S. Alvarado, and A. Terrones, “Autonomous 2D SLAM and 3D mapping of an environment using a single 2D LIDAR and ROS,” in *2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)*, Curitiba, 2017, pp. 1–6.
- [3] D. Ghorpade, A. D. Thakare, and S. Doiphode, “Obstacle Detection and Avoidance Algorithm for Autonomous Mobile Robot using 2D LiDAR,” in *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, PUNE, India, 2017, pp. 1–6.
- [4] M. Emharraf, M. Rahmoun, M. Saber, and M. Azizi, “Mobile robot: Simultaneous localization and mapping of unknown indoor environment,” in *2015 International Conference on Electrical and Information Technologies (ICEIT)*, Marrakech, Morocco, 2015, pp. 1–6.
- [5] Y. Pyo, H. Cho, R. Jung, and T. Lim, *ROS Robot Programming*. Seoul: ROBOTIS Co.,Ltd., 2017.
- [6] D. I. Pratiwi, “Rancang Bangun Deteksi Jalur Pipa Terpendam Menggunakan Mobile Robot dengan Metal Detector,” vol. 6, no. 1, p. 7, 2017.
- [7] S. Karakaya, G. Küçükyıldız, C. Toprak, and H. Ocak, “Development of a human tracking indoor mobile robot platform,” in *Proceedings of the 16th International Conference on Mechatronics - Mechatronika 2014*, 2014, pp. 683–687.
- [8] “SUMMIT-XL | Robotnik.” [Online]. Available: <https://www.robotnik.eu/mobile-robots/summit-xl/>. [Accessed: 26-May-2019].
- [9] A. Alhashimi, D. Varagnolo, and T. Gustafsson, “Statistical Modeling and Calibration of Triangulation Lidars:,” in *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics*, Lisbon, Portugal, 2016, pp. 308–317.
- [10] R. T. • 7 M. Ago, “Low Cost Solid-State 2D LiDAR,” *Steemit*, 08-Dec-2018. [Online]. Available: <https://steemit.com/technology/@rnjena/low-cost-solid-state-2d-lidar>. [Accessed: 02-Jul-2019].

- [11] “YDLIDAR X4 Datasheet.” EAI Team, 2015.
- [12] M. Rivai, F. Budiman, D. Purwanto, and J. Simamora, “Meat Freshness Identification System Using Gas Sensor Array And Color Sensor In Conjunction With Neural Network Pattern Recognition,” *Vol.*, no. 12, p. 13, 2018.
- [13] “Buy a Raspberry Pi 3 Model B – Raspberry Pi.” [Online]. Available: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. [Accessed: 26-May-2019].
- [14] “Arduino - ArduinoNano.” [Online]. Available: <https://www.arduino.cc/en/Guide/ArduinoNano>. [Accessed: 23-Nov-2018].
- [15] Atmel, “ATmega328P Datasheet.” Atmel Corporation, 2015.
- [16] “Hobbywing 5V/6V 3A Switch Mode Ultimate UBEC.” [Online]. Available: <http://buaya-instrument.com/hobbywing-5v6v-3a-switch-mode-ultimate-ubec-uhw3a.html>. [Accessed: 02-Jul-2019].
- [17] “Motor Drivers vs. Motor Controllers - Tutorial Australia.” [Online]. Available: <https://core-electronics.com.au/tutorials/motor-drivers-vs-motor-controllers.html>. [Accessed: 26-May-2019].
- [18] Y. Prakoso, “Desain Dan Implementasi Pengukuran Posisi Bola Menggunakan Kamera 360 Derajat Pada Robot Sepak Bola,” Institut Teknologi Sepuluh Nopember, Surabaya, 2017.
- [19] “ROS.org | About ROS.”
- [20] J. D. Gigliero, “Lidar Basics for Mapping Applications,” p. 11.
- [21] “Robotic Mapping: Simultaneous Localization and Mapping (SLAM),” *GIS Lounge*, 15-Jan-2013. [Online]. Available: <https://www.gislounge.com/robotic-mapping-simultaneous-localization-and-mapping-slam/>. [Accessed: 23-Nov-2018].
- [22] C. Stachniss, *Robotic Mapping and Exploration*, vol. 55. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.
- [23] “Occupancy Grids - MATLAB & Simulink - MathWorks Switzerland.” [Online]. Available: <https://ch.mathworks.com/help/robotics/ug/occupancy-grids.html>. [Accessed: 19-May-2019].
- [24] N. Yu and B. Zhang, “An Improved Hector SLAM Algorithm based on Information Fusion for Mobile Robot,” in *2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, 2018, pp. 279–284.
- [25] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf, “A flexible and scalable SLAM system with full 3D motion estimation,” in *2011*

- IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2011, pp. 155–160.
- [26] “Iterative Closest Point - File Exchange - MATLAB Central.” [Online]. Available: <https://ch.mathworks.com/matlabcentral/fileexchange/27804>. [Accessed: 26-May-2019].
- [27] Y. Wu and Z. Ding, “Research on Laser Navigation Mapping and Path Planning of Tracked Mobile Robot Based on Hector SLAM,” in *2018 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, 2018, vol. 3, pp. 59–65.

.....*Halaman ini sengaja dikosongkan*.....



## LAMPIRAN A

### A. Program Arduino

```
#include "PinChangeInterrupt.h"
```

```
#define IN1 12
```

```
#define IN2 4
```

```
#define IN3 5
```

```
#define IN4 6
```

```
#define IN5 7
```

```
#define IN6 8
```

```
#define EN2 11
```

```
#define EN1 10
```

```
#define EN3 9
```

```
#define encod1A 13
```

```
#define encod1B 3
```

```
#define encod2A A0
```

```
#define encod2B A1
```

```
#define encod3A A2
```

```
#define encod3B A3
```

```
#define kp1 4
```

```
#define kp2 4
```

```
#define kp3 5
```

```
#define ki1 2.8
```

```
#define ki2 2.8
```

```
#define ki3 2.8
```

```
#define kd1 2.5
```

```
#define kd2 2.5
```

```
#define kd3 2.5
```

```
#define kpw 20
```

```

#define kdw 2000
#define jumlah_jarak 8
const byte numChars = 64;
char receivedChars[numChars];
char temp[jumlah_jarak+1][8];

boolean newData = false;

volatile long int countM1 = 0;
volatile long int countM2 = 0;
volatile long int countM3 = 0;

char data;

unsigned int waktuNow;
unsigned int waktuPrev;
double VM1;
double VM2;
double VM3;

double VM1_f;
double VM2_f;
double VM3_f;

bool cwM1, cwM2, cwM3;

double er1,er2,er3;
double le1,le2,le3;
double P_out1,P_out2,P_out3;
double I_out1,I_out2,I_out3;
double D_out1,D_out2,D_out3;
double out1,out2,out3;
double per1,per2,per3;

float sp = 0.30;

```

```
float error;
float prev_error;
float ang;
float thd_dep = 0.30;
float vel = 8;
float velx = 0;
float max_ang = 5;

String penanda;
float jarak[jumlah_jarak];

bool mlaku = false;
bool tahan = false;
int muka = 0;
int hitung = 0;
int state = 0;
int pos = 0;

void setup() {
  Serial.begin(115200);

  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  pinMode(IN3,OUTPUT);
  pinMode(IN4,OUTPUT);
  pinMode(IN5,OUTPUT);
  pinMode(IN6,OUTPUT);

  pinMode(EN1,OUTPUT);
  pinMode(EN2,OUTPUT);
  pinMode(EN3,OUTPUT);

  pinMode(encod1A,INPUT);
  pinMode(encod1B,INPUT);
  pinMode(encod2A,INPUT);
  pinMode(encod2B,INPUT);
```

```
pinMode(encod3A,INPUT);
pinMode(encod3B,INPUT);

digitalWrite(IN1,LOW);
digitalWrite(IN2,LOW);
digitalWrite(IN3,LOW);
digitalWrite(IN4,LOW);
digitalWrite(IN5,LOW);
digitalWrite(IN6,LOW);

attachPinChangeInterrupt(digitalPinToPCINT(encod1A),motor1interrupt
A,CHANGE);

attachPinChangeInterrupt(digitalPinToPCINT(encod1B),motor1interrupt
B,CHANGE);

attachPinChangeInterrupt(digitalPinToPCINT(encod2A),motor2interrupt
A,CHANGE);

attachPinChangeInterrupt(digitalPinToPCINT(encod2B),motor2interrupt
B,CHANGE);

attachPinChangeInterrupt(digitalPinToPCINT(encod3A),motor3interrupt
A,CHANGE);

attachPinChangeInterrupt(digitalPinToPCINT(encod3B),motor3interrupt
B,CHANGE);

}

void loop()
{
```

```

waktuNow = millis();
if((waktuNow -waktuPrev)>=100)
{
    waktuPrev = waktuNow;
    VM1_f = ((double)countM1/1550)*100;
    VM2_f = ((double)countM2/1550)*100;
    VM3_f = ((double)countM3/1550)*100;

    recvWithStartEndMarkers();
    parsing_data();
    if(penanda == "a")mlaku = true;
    else if(penanda == "s")mlaku = false;

    if(mlaku)
    {
        //Serial.println(pos);
        if(penanda == "y")hitung++;
        if(hitung>10)
        {
            zigzag_ka(8);
        }
    }
    else if(!mlaku)
    {
        mandek();
        reset_data();
    }
    //jalan(data);
    //gerak(0,-8,0);
    countM1 = 0;
    countM2 = 0;
    countM3 = 0;
}
//terima_data();
}

```

```

void zigzag_ka(float kec)
{
    switch(pos)
    {
        case 0:// kode awalan 4
        //Serial.println(jarak[6]);
        error = jarak[6] - sp;
        ang = kpw*error + kdw*(error - prev_error);

        if(ang >= max_ang)ang = max_ang;
        if(ang <= -max_ang)ang = -max_ang;

        prev_error = error;

        if(jarak[0] < thd_dep || jarak[7] < thd_dep)
        {
            pos = 1;
        }
        else if((jarak[0] > thd_dep || jarak[7] > thd_dep)&&jarak[6]>0.40)
        {
            pos = 42;//41
            ang = 0;
        }
        omni_kecepatan_pergerakan_roda(0,kec,-ang,0);
        break;
        case 41:
        mandek();
        delay(100);
        pos = 42;
        break;
        case 42:
        if(jarak[5]>0.40)
        {
            pos = 43;

```

```

}
omni_kecepatan_pergerakan_roda(0,kec,0,0);
break;
case 43:
omni_kecepatan_pergerakan_roda(kec,0,0,0);
if(jarak[4]<0.40)pos = 3;
break;
case 1:
error = jarak[0] - sp;
ang = kpw*error + kdw*(error - prev_error);

if(ang >= max_ang)ang = max_ang;
if(ang <= -max_ang)ang = -max_ang;

prev_error = error;
if(jarak[1] < thd_dep || jarak[2] < thd_dep)
{
    pos = 2;
}
else if((jarak[1] > thd_dep || jarak[2] > thd_dep) && jarak[0]>0.40)
{
    pos = 12;//11
    ang = 0;
}
omni_kecepatan_pergerakan_roda(-kec,0,-ang,0);
break;

case 11:
mandek();
delay(100);
pos = 12;
break;
case 12:
if(jarak[7]>0.40)
{
    pos = 13;

```

```

}
omni_kecepatan_pergerakan_roda(-kec,0,0,0);
break;
case 13:
omni_kecepatan_pergerakan_roda(0,kec,0,0);
if(jarak[6]<0.40)pos = 0;
break;

case 2:
error = jarak[2] - sp;
ang = kpw*error + kdw*(error - prev_error);

if(ang >= max_ang)ang = max_ang;
if(ang <= -max_ang)ang = -max_ang;

prev_error = error;
if(jarak[3] < thd_dep || jarak[4] < thd_dep)
{
    pos = 3;
}
else if((jarak[3] > thd_dep || jarak[4] > thd_dep) && jarak[2]>0.40)
{
    pos = 22;//21
    ang = 0;
}
omni_kecepatan_pergerakan_roda(0,-kec,-ang,0);
break;
case 21:
mandek();
delay(100);
pos = 22;
break;
case 22:
if(jarak[1]>0.40)
{
    pos = 23;

```



```

}
omni_kecepatan_pergerakan_roda(0,-kec,0,0);
break;
case 23:
omni_kecepatan_pergerakan_roda(-kec,0,0,0);
if(jarak[0]<0.40)pos = 1;
break;

case 3:
error = jarak[4] - sp;
ang = kpw*error + kdw*(error - prev_error);

if(ang >= max_ang)ang = max_ang;
if(ang <= -max_ang)ang = -max_ang;

prev_error = error;
if(jarak[5] < thd_dep || jarak[6] < thd_dep)
{
    pos = 0;
}
else if((jarak[5] > thd_dep || jarak[6] > thd_dep) && jarak[4]>0.40)
{
    pos = 32;//31
    ang = 0;
}
omni_kecepatan_pergerakan_roda(kec,0,-ang,0);
break;
case 31:
mandek();
delay(100);
pos = 32;
break;
case 32:
if(jarak[3]>0.40)
{
    pos = 33;

```

```

    }
    omni_kecepatan_pergerakan_roda(kec,0,0,0);
    break;
    case 33:
    omni_kecepatan_pergerakan_roda(0,-kec,0,0);
    if(jarak[2]<0.40)pos = 2;
    break;
    }
}
void recvWithStartEndMarkers() {
    static boolean recvInProgress = false;
    static byte ndx = 0;
    char startMarker = '<';
    char endMarker = '>';
    char rc;

    while (Serial.available() > 0 && newData == false) {
        rc = Serial.read();

        if (recvInProgress == true) {
            if (rc != endMarker) {
                receivedChars[ndx] = rc;
                ndx++;
                if (ndx >= numChars) {
                    ndx = numChars - 1;
                }
            }
        }
        else {
            receivedChars[ndx] = '\0'; // terminate the string
            recvInProgress = false;
            ndx = 0;
            newData = true;
        }
    }

    else if (rc == startMarker) {

```

```

        recvInProgress = true;
    }
}

void parsing_data() {
    int data = 0;
    int data_ke = 0;
    int index = 0;
    if (newData == true)
    {
        for(int i = 0; i < numChars ; i++)
        {
            if(receivedChars[i] == '\n')
            {
                index = 0;
                data_ke++;
            }
            else
            {
                temp[data_ke][index] = receivedChars[i];
                index++;
            }
        }

        penanda = temp[0];

        for(int i = 0; i < jumlah_jarak; i++)
        {
            jarak[i] = atof(temp[i+1]);
        }
        newData = false;
    }
}

void terima_data()

```

```

{
  if(Serial.available(>0)
  {
    data = Serial.read();
  }
}

float sudut_roda[3];
float omni_kecepatan_motor[3];
void omni_kecepatan_pergerakan_roda(float kecepatan_x, float
kecepatan_y, float kecepatan_w, float sudut_rai)
{
  int i;

  int omni_alpha = 120;

  for(i=0;i<3;i++)
  {
    sudut_roda[i] = (i*omni_alpha - sudut_rai + 150)*PI/180
//KONVERSI KE RADIAN
    omni_kecepatan_motor[i] = -(-sinf(sudut_roda[i])*kecepatan_x +
cosf(sudut_roda[i])*kecepatan_y + kecepatan_w/4);
  }

gerak(omni_kecepatan_motor[0],omni_kecepatan_motor[1],omni_kecep
atan_motor[2]);
}

void gerak(double M2, double M1, double M3)
{

  cwM1 = M1<0 ? true : false;
  cwM2 = M2<0 ? true : false;
  cwM3 = M3<0 ? true : false;

  er1 = abs(M1) - abs(VM1_f);

```

```

per1 += er1;
P_out1 = er1*kp1;

if(per1>255)per1 = 255;
if(per1<0)per1 = 0;
I_out1 = ki1*per1;
D_out1 = kd1*(er1 - le1);
out1 = P_out1 + I_out1 + D_out1;

er2 = abs(M2) - abs(VM2_f);
P_out2 = er2*kp2;
per2 += er2;
if(per2>255)per2 = 255;
if(per2<0)per2 = 0;
I_out2 = ki2*per2;
D_out2 = kd2*(er2_f-le2);
out2 = P_out2 + I_out2 + D_out2;

er3 = abs(M3) - abs(VM3_f);
P_out3 = er3*kp3;
per3 += er3;
if(per3>255)per3 = 255;
if(per3<0)per3 = 0;
I_out3 = ki3*per3;
D_out3 = kd3*(er3-le3);
out3 = P_out3 + I_out3 +D_out3;

le1 = VM1_f;
le2 = VM2_f;
le3 = VM3_f;

int pid1 = constrain(out1,0,255);
int pid2 = constrain(out2,0,255);
int pid3 = constrain(out3,0,255);

```

```

if(cwM1)
{
    digitalWrite(IN1,LOW);
    digitalWrite(IN2,HIGH);
    analogWrite(EN1,abs(pid1));
}
else
{
    digitalWrite(IN1,HIGH);//ENCODMIN
    digitalWrite(IN2,LOW);
    analogWrite(EN1,abs(pid1));
}

if(cwM2)
{
    digitalWrite(IN3,LOW);//ENCODMIN
    digitalWrite(IN4,HIGH);
    analogWrite(EN2,abs(pid2));
}
else
{
    digitalWrite(IN3,HIGH);
    digitalWrite(IN4,LOW);
    analogWrite(EN2,abs(pid2));
}

if(cwM3)
{
    digitalWrite(IN5,LOW);
    digitalWrite(IN6,HIGH);
    analogWrite(EN3,abs(pid3));
}
else
{
    digitalWrite(IN5,HIGH);
    digitalWrite(IN6,LOW);//ENCODMIN

```

```
    analogWrite(EN3,abs(pid3));  
  }  
}
```

```
void reset_data()
```

```
{  
  er1 = 0;  
  er2 = 0;  
  er3 = 0;  
  
  le1 = 0;  
  le2 = 0;  
  le3 = 0;  
  
  P_out1 = 0;  
  P_out2 = 0;  
  P_out3 = 0;  
  
  D_out1 = 0;  
  D_out2 = 0;  
  D_out3 = 0;  
  
  out1 = 0;  
  out2 = 0;  
  out3 = 0;  
}
```

```
void mandek()
```

```
{  
  digitalWrite(IN1,LOW);  
  digitalWrite(IN2,LOW);  
  digitalWrite(IN3,LOW);  
  digitalWrite(IN4,LOW);  
  digitalWrite(IN5,LOW);  
  digitalWrite(IN6,LOW);  
}
```

```
}
```

```
void motor1interruptA()
```

```
{  
  if( digitalRead(encod1B) == 0 ) {  
    if ( digitalRead(encod1A) == 0 ) {  
      countM1--;  
    } else {  
      countM1++;  
    }  
  } else {  
    if ( digitalRead(encod1A) == 0 ) {  
      countM1++;  
    } else {  
      countM1--;  
    }  
  }  
}
```

```
void motor1interruptB()
```

```
{  
  if ( digitalRead(encod1A) == 0 ) {  
    if ( digitalRead(encod1B) == 0 ) {  
      countM1++;  
    } else {  
      countM1--;  
    }  
  } else {  
    if ( digitalRead(encod1B) == 0 ) {  
      countM1--;  
    } else {  
      countM1++;  
    }  
  }  
}
```



```

//MOTOR2
void motor2interruptA()
{
  if( digitalRead(encod2B) == 0 ) {
    if ( digitalRead(encod2A) == 0 ) {
      countM2--;
    } else {
      countM2++;
    }
  } else {
    if ( digitalRead(encod2A) == 0 ) {
      countM2++;
    } else {
      countM2--;
    }
  }
}

```

```

void motor2interruptB()
{
  if ( digitalRead(encod2A) == 0 ) {
    if ( digitalRead(encod2B) == 0 ) {
      countM2++;
    } else {
      countM2--;
    }
  } else {
    if ( digitalRead(encod2B) == 0 ) {
      countM2--;
    } else {
      countM2++;
    }
  }
}

```

```

//MOTOR3

```

```

void motor3interruptA()
{
  if( digitalRead(encod3B) == 0 ) {
    if ( digitalRead(encod3A) == 0 ) {
      countM3--;
    } else {
      countM3++;
    }
  }else {
    if ( digitalRead(encod3A) == 0 ) {
      countM3++;
    } else {
      countM3--;
    }
  }
}

```

```

void motor3interruptB()
{
  if ( digitalRead(encod3A) == 0 ) {
    if ( digitalRead(encod3B) == 0 ) {
      countM3++;
    } else {
      countM3--;
    }
  } else {
    if ( digitalRead(encod3B) == 0 ) {
      countM3--;
    } else {
      countM3++;
    }
  }
}

```

## B. Program pada Laptop

```
<?xml version="1.0"?>

<launch>
  <arg name="tf_map_scanmatch_transform_frame_name"
default="scanmatcher_frame"/>
  <arg name="base_frame" default="base_link"/>
  <arg name="odom_frame" default="base_link"/>
  <arg name="pub_map_odom_transform" default="false"/>
  <arg name="scan_subscriber_queue_size" default="10"/>
  <arg name="scan_topic" default="scan"/>
  <arg name="map_size" default="1024"/>

  <node pkg="hector_mapping" type="hector_mapping"
name="hector_mapping" output="screen">

    <!-- Frame names -->
    <param name="map_frame" value="map" />
    <param name="base_frame" value="$(arg base_frame)" />
    <param name="odom_frame" value="$(arg odom_frame)" />

    <!-- Tf use -->
    <param name="use_tf_scan_transformation" value="true"/>
    <param name="use_tf_pose_start_estimate" value="false"/>
    <param name="pub_map_odom_transform" value="$(arg
pub_map_odom_transform)"/>

    <!-- Map size / start point -->
    <param name="map_resolution" value="0.02"/>
    <param name="map_size" value="$(arg map_size)"/>
    <param name="map_start_x" value="0.5"/>
    <param name="map_start_y" value="0.5" />
    <param name="map_multi_res_levels" value="3" />

    <!-- Map update parameters -->
    <param name="update_factor_free" value="0.4"/>
    <param name="update_factor_occupied" value="0.9" />
    <param name="map_update_distance_thresh" value="0.40"/>
    <param name="map_update_angle_thresh" value="0.06" />
```

```

<param name="laser_z_min_value" value = "-1.0" />
<param name="laser_z_max_value" value = "1.0" />

<!-- Advertising config -->
<param name="advertise_map_service" value="true"/>

<param name="scan_subscriber_queue_size" value="$(arg
scan_subscriber_queue_size)"/>
<param name="scan_topic" value="$(arg scan_topic)"/>

<!-- Debug parameters -->
<!--
<param name="output_timing" value="false"/>
<param name="pub_drawings" value="true"/>
<param name="pub_debug_output" value="true"/>
-->
<param name="tf_map_scanmatch_transform_frame_name"
value="$(arg tf_map_scanmatch_transform_frame_name)" />
</node>

<node pkg="tf" type="static_transform_publisher"
name="base_to_laser_broadcaster" args="0 0 0 0 0 base_link laser
50" />

</launch>

```

### C. Program Raspberry Pi 3b

```

#include "ros/ros.h"
#include "sensor_msgs/LaserScan.h"
#include <serial/serial.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <termios.h>
#include <iostream>

#define RAD2DEG(x) ((x)*180./M_PI)

```

```
#define depan1    0
#define deserki   1
#define deki      2
#define belka     3
#define belserka  4
#define bel1      5
#define bel2      6
#define belserki  7
#define belki     8
#define deka      9
#define deserka  10
#define depan2   11
```

```
#define JUMLAH_DAERAH 12
```

```
serial::Serial ser;
```

```
bool jalan = false;
```

```
bool obs = false;
```

```
float sudut;
```

```
float rata_jarak[JUMLAH_DAERAH];
```

```
float pembagi_jarak[JUMLAH_DAERAH];
```

```
float simpan[80];
```

```
float simpan1[80];
```

```
float simpan2[80];
```

```
float simpan3[80];
```

```
float simpan4[80];
```

```
float simpan5[80];
```

```
float simpan6[80];
```

```
float simpan7[80];
```

```
float simpan8[80];
```

```
float simpan9[80];
```

```
float simpan10[80];
```

```

float simpan11[80];

char key(' ');
char data_kirim;
int cnt = 0;
bool hold[JUMLAH_DAERAH] = {false,false,false};
bool daerah[JUMLAH_DAERAH];
int sensor[JUMLAH_DAERAH];
int muka;
float thd_dinding = 0.3;
char temp[64];

//float jarak;

void tentukan_aksi();
float cari_terkecil(float arr[],int n);

int getch(void)
{
    int ch;
    struct termios oldt;
    struct termios newt;

    tcgetattr(STDIN_FILENO, &oldt);
    newt = oldt;

    newt.c_lflag &= ~(ICANON | ECHO);
    newt.c_iflag |= IGNBRK;
    newt.c_iflag &= ~(INLCR | ICRNL | IXON | IXOFF);
    newt.c_lflag &= ~(ICANON | ECHO | ECHOK | ECHOE | ECHONL |
ISIG | IEXTEN);
    newt.c_cc[VMIN] = 1;
    newt.c_cc[VTIME] = 0;
    tcsetattr(fileno(stdin), TCSANOW, &newt);

    ch = getchar();

```

```

tcsetattr(STDIN_FILENO, TCSANOW, &oldt);

return ch;
}

void scanCallback(const sensor_msgs::LaserScan::ConstPtr& scan)
{
    int count = scan->scan_time / scan->time_increment;

    int index[JUMLAH_DAERAH] = {0,0,0,0,0,0,0,0,0,0};

    //printf("[YDLIDAR INFO]: I heard a laser scan %s[%d]:\n", scan-
>header.frame_id.c_str(), count);
    //printf("[YDLIDAR INFO]: angle_range : [%f, %f]\n",
RAD2DEG(scan->angle_min), RAD2DEG(scan->angle_max));

    for(int i = 0; i < count; i++) {
        float degree = RAD2DEG(scan->angle_min + scan-
>angle_increment * i);
        float jarak = scan->ranges[i];

        if(jarak!=0 && (degree < 30 && degree > 0))//depan1
        {
            simpan[index[depan1]] = jarak ;
            index[depan1]++;
        }
        // else if(jarak!=0 && (degree > 30 && degree < 60))//depan serong
kiri
        // {
        //     simpan1[index[deserki]] = jarak ;
        //     index[deserki]++;
        // }
        else if(jarak!=0 && (degree > 60 && degree < 90))//depan kiri
        {
            simpan2[index[deki]] = jarak ;

```

```

    index[deki]++;
}
else if(jarak!=0 && (degree > 90 && degree < 120))//belakang
kanan
{
    simpan3[index[belka]] = jarak ;
    index[belka]++;
}
// else if(jarak!=0 && (degree > 120 && degree < 150))//belakang
serong kanan
// {
//     simpan4[index[belserka]] = jarak ;
//     index[belserka]++;
// }
else if(jarak!=0 && (degree > 150 && degree < 180))//belakang 1
{
    simpan5[index[bel1]] = jarak ;
    index[bel1]++;
}
else if(jarak!=0 && (degree > -180 && degree < -150))//belakang 2
{
    simpan6[index[bel2]] = jarak ;
    index[bel2]++;
}
// else if(jarak!=0 && (degree > -150 && degree < -120))//belakang
serong kiri
// {
//     simpan7[index[belserki]] = jarak ;
//     index[belserki]++;
// }
else if(jarak!=0 && (degree > -120 && degree < -90))//belakang kiri
{
    simpan8[index[belki]] = jarak ;
    index[belki]++;
}
else if(jarak!=0 && (degree > -90 && degree < -60))//depan kanan

```



```

    {
        simpan9[index[deka]] = jarak ;
        index[deka]++;
    }
    // else if(jarak!=0 && (degree > -60 && degree < -30))//depan
serong kanan
    // {
    //     simpan10[index[deserka]] = jarak ;
    //     index[deserki]++;
    // }
    else if(jarak!=0 && (degree > -30 && degree < 0))//depan 2
    {
        simpan11[index[depan2]] = jarak ;
        index[depan2]++;
    }
}

//sprintf(temp, "<y\n%0.2f\n%0.2f\n%0.2f\n%0.2f\n>",rata_jarak[DEPA
N],rata_jarak[KANAN],rata_jarak[KIRI],rata_jarak[BELAKANG]);
//ser.write(temp);

float j_depan1 = cari_terkecil(simpan,index[depan1]);
//float j_deserki = cari_terkecil(simpan1,index[deserki]);
float j_deki = cari_terkecil(simpan2,index[deki]);
float j_belka = cari_terkecil(simpan3,index[belka]);
//float j_belserka = cari_terkecil(simpan4,index[belserka]);
float j_bel1 = cari_terkecil(simpan5,index[bel1]);

float j_bel2 = cari_terkecil(simpan6,index[bel2]);
//float j_belserki = cari_terkecil(simpan7,index[belserki]);
float j_belki = cari_terkecil(simpan8,index[belki]);
float j_deka = cari_terkecil(simpan9,index[deka]);
//float j_deserka = cari_terkecil(simpan10,index[deserka]);
float j_depan2 = cari_terkecil(simpan11,index[depan2]);

```



```

}
catch(serial::IOException& e)
{
  ROS_ERROR_STREAM("gagal menemukan port");
  return -1;
}

if(ser.isOpen())
{
  ROS_INFO_STREAM("Inisialisasi port");
}
else
{
  return -1;
}

ros::Duration(2).sleep();
printf("ROBOT JALAN\n");
sprintf(temp, "<a\n%0.2f\n%0.2f\n%0.2f\n%0.2f\n>", 0.0, 0.0, 0.0, 0.0);
ser.write(temp);
while (ros::ok())
{
  ros::spinOnce();
}

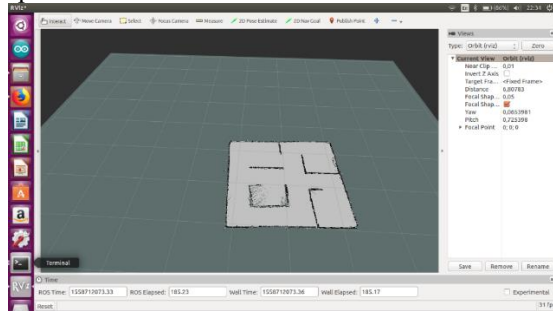
sprintf(temp, "<s\n%0.2f\n%0.2f\n%0.2f\n%0.2f\n>", 0.0, 0.0, 0.0, 0.0);
ser.write(temp);
printf("ROBOT BERHENTI\n");
return 0;}

```

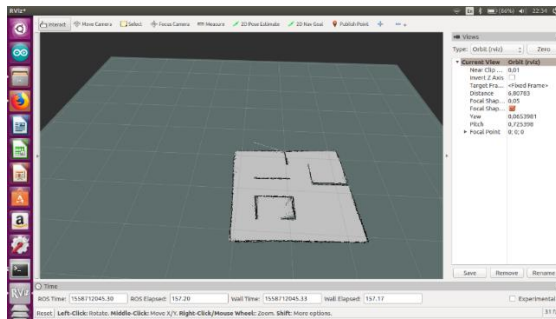


# LAMPIRAN B

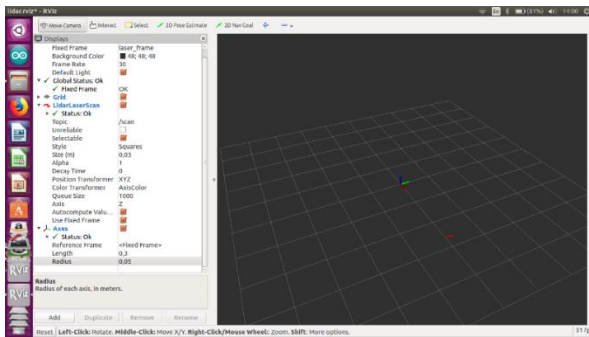
## A. Tampilan pada *tools* RVIZ



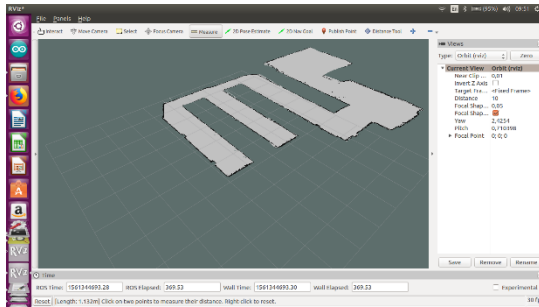
Gambar peta percobaan 1



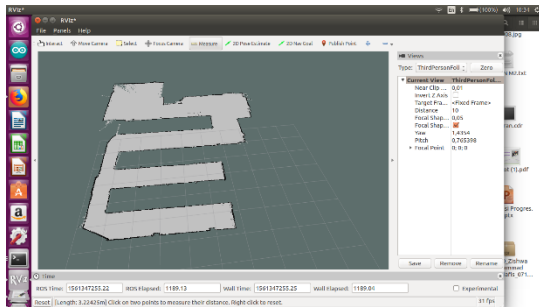
Gambar peta percobaan 2



Gambar pengukuran jarak

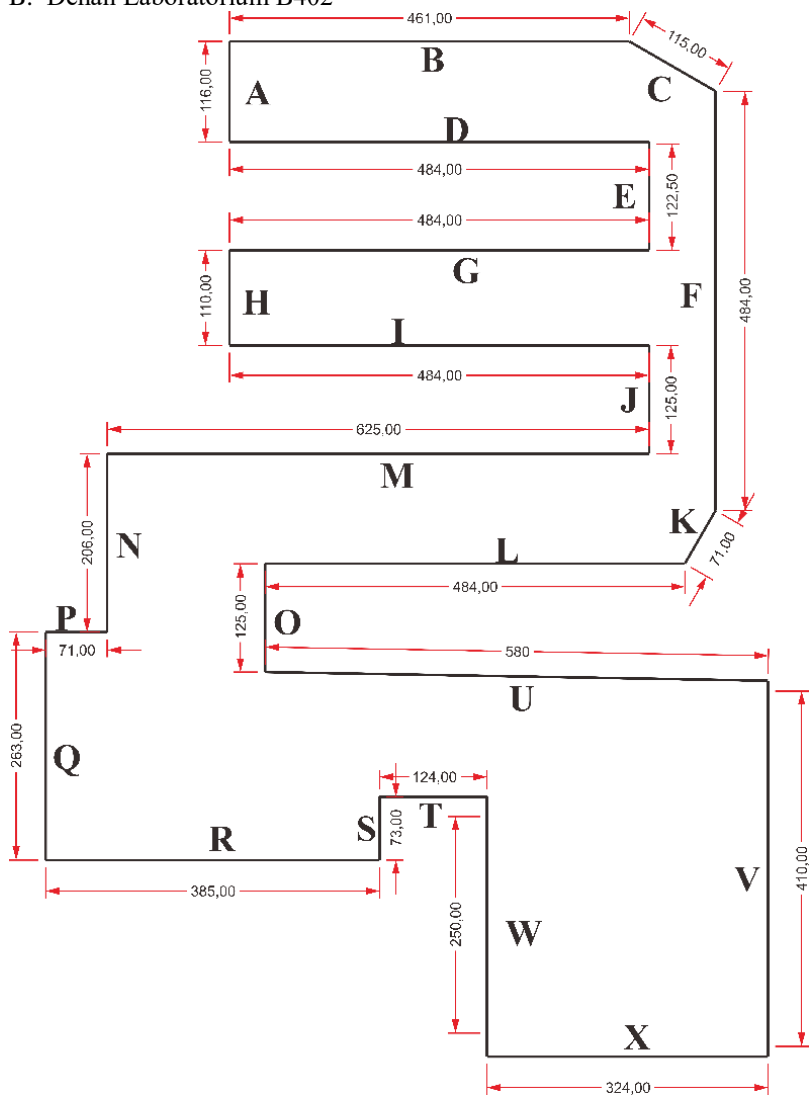


Gambar peta dengan robot otomatis

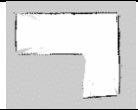


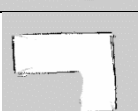


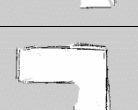
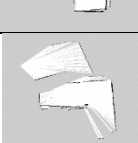


Gambar peta dengan robot manual

## B. Denah Laboratorium B402



C. Hasil Pemetaan Pengujian Variasi Kecepatan

Konstanta Kecepatan	Hasil Peta
8	
10	
12	
14	
16	
18	
20	
22	



## LAMPIRAN C

### A. Dokumentasi kegiatan





## LAMPIRAN D

### A. Datasheet YdLidar X4

#### Product parameters

CHART 1 YDLIDAR X4PRODUCT PARAMETERS

Item	Min.	Typical Value	Max.	Unit	Remark
Range Frequency	-	5000	-	Hz	5000 range sampling per second
Scanning Frequency	6	-	12	Hz	Configurable by software
Range	0.12	-	>10	m	Indoor
Scanning Angle	-	0-360	-	Deg	-
Range resolution	-	<0.5	-	mm	Range<2m
		< 1% of actual distance			Range>2m
Angle resolution	0.48	0.50	0.52	Deg	Scanning Frequency 7Hz
Working life	-	1500	-	h	Continuous working life

#### Electrical parameters

CHART 2 YDLIDAR X4 ELECTRICAL PARAMETERS

Item	Min.	Typical Value	Max.	Unit	Remark
Supply voltage	4.8	5	5.2	V	Excessive voltage can damage the device. Low voltage can affect performance.
Voltage ripple	0	50	100	mV	High ripple affects performance and can even cause Lidar to fail to range
Starting current	400	450	480	mA	High current at startup
Sleep current	280	300	340	mA	System sleeps, motor does not rotate
Working current	330	350	380	mA	System work, motor rotates

#### Interface definition

The X4 provides a PH2.0-8P female connector. The interface has functional

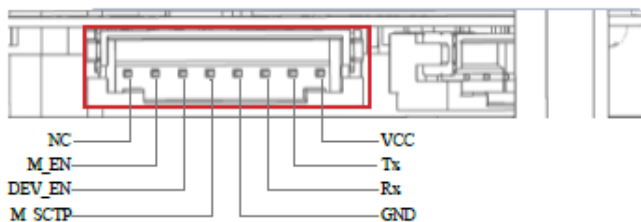


FIG 3 YDLIDAR X4 INTERFACES

CHART 3 YDLIDAR X4 INTERFACE DEFINITION

Pin	Type	Description	Default	Range	Remark
VCC	Power supply	voltage positive	5V	4.8V~5.2V	-
Tx	Output	System serial output	-	-	Data Stream: Lidar → Peripherals
Rx	Input	System serial port input	-	-	Data Stream: Peripherals → Lidar
GND	Power Supply	voltage negative	0V	0V	-
M_EN	Input	Motor enable control	3.3V	0V~3.3V	-
DEV_EN	Input	Ranging enable control	3.3V	0V~3.3V	-
M_SCTP	Input	Motor speed control	1.8V	0V~3.3V	Voltage speed regulation or PWM speed regulation
NC	-	Reserved pin	-	-	-

### Data communication

The X4 communicates using a 3.3V level serial port (UART). The user can connect the external system and the product through the physical interface on the product. And in accordance with the communication protocol of the system to obtain real-time scanning point cloud data, device information, device status, and set the device work mode. The communication parameters are as follows:

## BIODATA PENULIS



Zishwa Muhammad Jauhar Nafis, Lahir 7 Oktober 1996 di desa Banyubang kecamatan Solokuro Kabupaten Lamongan Jawa Timur. Sempat mengenyam pendidikan formal di MI Nurul Hidayah Banyubang sejak tahun 2003 sampai 2009, dilanjutkan ke MTs 16 Ma'arif Nurul Hidayah Banyubang pada tahun 2009 sampai 2012. Pada jenjang Madrasah Aliyah dilanjutkan di Pondok Pesantren Matholi'ul Anwar dan juga sekolah formal disana. Selama 3 tahun disana, memperdalam ilmu agama dan juga IPA. Setelah 3 tahun di pesantren ini melanjutkan ke jenjang yang lebih tinggi, ke dunia perkuliahan melalui Program Beasiswa Santri Berprestasi (PBSB) yang diadakan oleh Kementerian Agama RI. Penulis diterima di S1-Teknik Elektro ITS. Pada saat kuliah penulis aktif di organisasi CSSMoRA ITS serta Unit Kegiatan Mahasiswa seperti Robotika dan Rebana. Mulai tertarik pada bidang pemrograman dan robotika saat tahun pertama perkuliahan.