TUGAS AKHIR – EE 184801

AUTONOMOUS DOCKING SYSTEM UNTUK *MOBILE ROBOT BERBASIS* LIDAR PADA STASIUN PENGISIAN DAYA NIRKABEL

Rizky Reza Pahlevi
NRP 07111640000063

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.
Fajar Budiman, S.T., M.Sc.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – EE 184801

AUTONOMOUS DOCKING SYSTEM UNTUK *MOBILE ROBOT* BERBASIS LIDAR PADA STASIUN PENGISIAN DAYA NIRKABEL

Rizky Reza Pahlevi
NRP 07111640000063

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.
Fajar Budiman, S.T., M.Sc.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

FINAL PROJECT – EE 184801

LIDAR BASED MOBILE ROBOT AUTONOMOUS DOCKING SYSTEM ON WIRELESS CHARGING STATION

Rizky Reza Pahlevi
NRP 07111640000063

Supervisor
Dr. Muhammad Rivai, ST., MT.
Fajar Budiman, S.T., M.Sc.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa Tugas akhir saya dengan judul "*Autonomous Docking System* untuk *Mobile Robot* Berbasis LIDAR pada Stasiun Pengisian Daya Nirkabel" merupakan hasil karya intelektual mandiri, dan diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan ataupun menjiplak karya pihak lain.

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

Surabaya, 12 Juli 2020

Rizky Reza Pahlevi
NRP 0711 16 40000 063

AUTONOMOUS DOCKING SYSTEM UNTUK MOBILE ROBOT BERBASIS LIDAR PADA STASIUN PENGISIAN DAYA NIRKABEL

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I

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

SURABAYA
JULI, 2020

AUTONOMOUS DOCKING SYSTEM UNTUK MOBILE ROBOT BERBASIS LIDAR PADA STASIUN PENGISIAN DAYA NIRKABEL

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing II

Fajar Budiman, S.T., M.Sc.
NIP. 198607072014041001

SURABAYA
JULI, 2020

Autonomous Docking System untuk *Mobile Robot* berbasis LIDAR pada Stasiun Pengisian Daya Nirkabel

Nama : Rizky Reza Pahlevi
Pembimbing : Dr. Muhammad Rivai, ST., MT.
Fajar Budiman, S.T., M.Sc.

ABSTRAK

Salah satu pengembangan *autonomous mobile robot* adalah sebagai robot pengawas dan keamanan. Selain dapat mengerjakan tugas utamanya secara *autonomous*, robot dikembangkan lagi supaya dapat memonitor kondisi diri secara *realtime*. Salah satunya adalah mengamati kondisi baterai kemudian memutuskan apakah pengisian baterai diperlukan atau tidak. Setelah memutuskan untuk mengisi baterai robot akan menuju stasiun pengisian baterai secara *autonomous*. Permasalahan yang dihadapi adalah bagaimana cara robot mendeteksi lokasi stasiun pengisian dengan tepat. Penelitian ini dilakukan untuk membuat sebuah sistem *autodocking* pada *autonomous mobile robot* dengan menggunakan sensor *Light Detection and Ranging* (LIDAR) untuk menuju lokasi dari stasiun pengisian. *Mobile robot* yang akan dirancang berupa *Unmanned Ground Vehicle* (UGV) roda empat. Data dari sensor diakuisisi dengan mikrokontroler Teensy 3.6 dan diolah lebih lanjut menggunakan Raspberry Pi 4. Metode yang digunakan untuk mendekati stasiun pengisian adalah *Self Localization and Mapping* (SLAM). SLAM dapat ditemukan dalam bentuk *package* pada *Robot Operating System* yang dipasang pada Raspberry Pi 4. Output yang diharapkan adalah *mobile robot* dapat berjalan menuju stasiun pengisian dan melakukan *autodocking* secara *autonomous* setelah sistem diterapkan. Kecepatan optimal pemetaan adalah 0.6m/s untuk linier dan 0.3rad/s. Hasil dari lokalisasi AMCL memberikan rata-rata error koordinat x 0.029m dan koordinat y 0.0086m. Tingkat keberhasilan navigasi docking yang telah dibuat adalah 100% dengan rata-rata error koordinat x = 0.027m, y = 0.108m dan orientasi = 6.967 derajat.

Kata kunci: AMCL, Hector SLAM, LIDAR, Mobile Robot, ROS, UGV

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

LIDAR based Mobile Robot Autonomous Docking System on Wireless Charging Station

Nama : Rizky Reza Pahlevi
Pembimbing : Dr. Muhammad Rivai, ST., MT.
Fajar Budiman, S.T., M.Sc.

ABSTRACT

One of the autonomous mobile robot developments is as a surveillance and security robot. Besides carry out its main tasks autonomously, robots have been developed so that they can monitor themselves in real time. For example, The robot can observing the condition of the battery then deciding whether or not charging the battery is needed. After deciding to charge the robot the battery will go to the battery charging station autonomously. The problem faced is how the robot detects the location of the charging station appropriately. This research was conducted to create an autodocking system on autonomous mobile robots by using the Light Detection and Ranging (LIDAR) sensor to find a way to the location of the charging station. Mobile robot that will be designed in the form of four-wheeled Unmanned Ground Vehicle (UGV). In addition to using LIDAR. Data from the sensor was acquired with a Teensy 3.6 micro controller and further processed using Raspberry Pi 4. The method used to approach filling stations is Self Localization and Mapping (SLAM). SLAM can be found as a package in the Robot Operating System installed on Raspberry Pi 4. The expected output is that the mobile robot can move to the charging station and autodocking autonomously after the system is implemented. The system have been tested Mapping optimal speed is 0.6 m / s for linear and 0.3rad / s. The results of AMCL localization give an average error of coordinates x 0.029m and coordinates y 0.0086m. The success rate of docking navigation that has been made is 100% with an average coordinate error x = 0.027m, y = 0.108m and orientation = 6.967 degrees.

Keywords: AMCL, Hector SLAM, LIDAR, Mobile Robot, ROS, UGV

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

KATA PENGANTAR

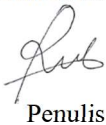
Segala puji syukur kepada Allah SWT yang telah memberikan nikmat dan karunia-Nya kepada penulis sehingga penulis dapat menyelesaikan laporan tugas akhir dengan judul "***Autonomous Docking Syste*m untuk *Mobile Robot* berbasis LIDAR pada Stasiun Pengisian Daya Nirkabel**", sebagai salah satu persyaratan dalam menyelesaikan pendidikan program studi S1 di Departemen Teknik Elektro, Fakultas Teknologi Elektro dan Informatika Cerdas, 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. dan Fajar Budiman, ST., M.Sc., selaku dosen pembimbing yang telah membimbing dan memberikan saran serta arahan selama pengerjaan dan penulisan laporan tugas akhir.
2. Ronny Mardiyanto, ST., MT., Ph.D, Ir. Tasripan, MT., Muhammad Attamimi, B.Eng., M.Eng., Ph.D., Ir. Harris Pirngadi, MT., selaku dosen penguji yang telah memberikan masukan dan arahan selama melakukan revisi.
3. Kepala Departemen Teknik Elektro ITS, Dedet Candra Riawan, ST., M.Eng., Ph.D atas izin dan kesempatan yang diberikan kepada penulis untuk melaksanakan tugas akhir ini.
4. Seluruh dosen dan tenaga pendidik Departemen Teknik Elektro.
5. Orang tua serta seluruh keluarga yang memberikan dukungan baik moril maupun materiil.
6. Teman-teman laboratorium B402 & B202 dan Angkatan E56.

Penulis berharap agar dengan adanya tugas akhir ini dapat bermanfaat kepada pembacanya. Penulis juga menyadari bahwa masih banyak kekurangan dalam penulisan laporan tugas akhir ini. Oleh karena itu kritik saran sangat terbuka luas untuk diterima. Akhir kata penulis mengucapkan terima kasih yang sebesar-besarnya.

Surabaya, 12 Juli 2020



Penulis

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

DAFTAR ISI

HALAMAN JUDUL

PERNYATAAN KEASLIAN

LEMBAR PENGESAHAN

ABSTRAK ..... **i**

ABSTRACT ..... **iii**

KATA PENGANTAR ..... **v**

DAFTAR ISI ..... **vii**

DAFTAR GAMBAR ..... **xi**

DAFTAR TABEL ..... **xiii**

BAB 1 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 3

1.6 Sistematika Penulisan 4

1.7 Relevansi 5

BAB 2 TINJAUAN PUSTAKA **7**

2.1 Skid Steer Mobile Robot (SSMR) 7

2.1.1 Model Kinematika SSMR 8

2.2 Light Detection and Ranging (LIDAR) 9

2.3 YDLIDAR X4 10

2.4 Mikrokontroler Teensy 3.6 11

2.5 Raspberry Pi 4 13

2.6 Driver Motor DC 15

2.7 Zero Crossing Detector 16
2.8 Robot Operating System (ROS) 17
2.9 Master 17
2.10 Node 17
2.11 Package 17
2.12 Message 18
2.13 Topic 18
2.14 Service 18
2.15 rosbuild & catkin 18
2.16 rosrun & roslaunch 18
2.17 bag 19
2.18 rosserial 19
2.19 Simultaneous Localization and Mapping (SLAM) 19
2.20 Hector SLAM 20
2.21 Occupancy Grid 21
2.22 Scan Matching 22
2.23 Adaptive Monte Carlo Localization 25
2.24 Tinjauan Pustaka 25
2.24.1 Sensor Guided Docking of Autonomous Mobile Robot for Battery Recharging 25
2.24.2 Vision-Based Autonomous Docking and Re-charging System for Mobile Robot in Warehouse Environment 25
2.24.3 Large-Scale Outdoor SLAM Based on 2D Lidar 26
2.24.4 Mobile Robot Navigation Using 2D LIDAR 26
BAB 3 PERANCANGAN SISTEM 27
3.1 Gambaran Umum Sistem 27

3.2 Perancangan Perangkat Keras 28
3.2.1 Platform mobile robot 28
3.2.2 Pemasangan sensor 29
3.2.3 Catu Daya 29
3.2.4 Pemasangan Motor 30
3.2.5 Kinematika mobile robot 32
3.2.6 Board ekstensi Teensy 3.6 34
3.2.7 Stasiun Pengisian 38
3.3 Perancangan Perangkat Lunak 38
3.3.1 Alur Data Sensor 38
3.3.2 Pengambilan Data Sensor 39
3.3.3 Kontrol Motor DC 40
3.3.4 Perancangan Sistem ROS 41
3.3.5 Pembuatan Peta 42
3.3.6 Lokalisasi Robot 42
3.3.7 Navigasi Robot 43
BAB 4 PENGUJIAN DAN ANALISIS 45
4.1 Pengujian Sensor LIDAR 45
4.1.1 Akurasi Sensor 45
4.1.2 Terhadap Berbagai bahan 48
4.1.3 Terhadap Berbagai Intensitas Cahaya 50
4.2 Pengujian Kontrol Motor DC 52
4.3 Pengujian Pemetaan SLAM 57
4.4 Pengujian Lokalisasi SLAM 59
4.5 Pengujian Navigasi Docking 61
4.6 Pengujian Navigasi Docking dengan Halangan Baru 65

BAB 5 PENUTUP **69**
5.1 Kesimpulan 69
5.2 Saran 69
DAFTAR PUSTAKA **71**
LAMPIRAN A **75**
LAMPIRAN B **101**
LAMPIRAN C **113**
BIODATA PENULIS **127**

DAFTAR GAMBAR

Gambar 2.1 Contoh SSMR: Pioneer P3AT 7
Gambar 2.2 Kinematika SSMR 8
Gambar 2.3 Ilustrasi perhitungan jarak TOF LIDAR 9
Gambar 2.4 Cara Kerja YDLIDAR X4 10
Gambar 2.5 Pinout Teensy 3.6 12
Gambar 2.6 Raspberry pi 4 14
Gambar 2.7 Pinout Raspberry Pi 4 14
Gambar 2.8 Cara kerja H-bridge motor 15
Gambar 2.9 Driver motor cytron MDD 10A 16
Gambar 2.10 Contoh rangkaian ZCD LM 311 16
Gambar 2.11 Robot operating system 19
Gambar 2.12 Occupancy Grid Map 21
Gambar 2.13 Ilustrasi Hector scan matching 23
Gambar 2.14 Ilustrasi Hector scan matching 2 24
Gambar 3.1. Gambaran umum sistem 27
Gambar 3.2 Rancangan mobile robot 28
Gambar 3.3 Peletakan sensor YDLIDAR X4 29
Gambar 3.4 Pembagian catu daya 30
Gambar 3.5 Diagram pemasangan motor 31
Gambar 3.6 Skematik pemasangan motor 31
Gambar 3.7 Kinematika mobile robot 33
Gambar 3.8 Skematik rangkaian board ekstensi Teensy 3.6 35
Gambar 3.9 Skematik rangkaian power supply -5V 36
Gambar 3.10 Skematik rangkaian ZCD 36
Gambar 3.11 Desain board ekstensi Teensy 3.6 37
Gambar 3.12 Desain stasiun pengisian 38
Gambar 3.13 Alur data sensor 39
Gambar 3.14 Diagram alir penghitung pulsa 40
Gambar 3.15 Kontrol motor DC 41
Gambar 3.16 Perancangan sistem ROS 41
Gambar 3.17 Diagram alir pemetaan 42
Gambar 3.18 Diagram Alir Lokalisasi 43
Gambar 3.19 Diagram alir navigasi move_base 44
Gambar 4.1 Realisasi Mobile Robot 45
Gambar 4.2 Pengukuran Akurasi LIDAR 46

Gambar 4.3 Grafik pengaruh intensitas cahaya terhadap pembacaan LIDAR (indoor)..... 50
Gambar 4.4 Pengujian Luminasi outdoor cahaya matahari secara langsung..... 52
Gambar 4.5 Grafik kontrol kecepatan motor tanpa beban..... 54
Gambar 4.6 Grafik kontrol kecepatan motor dengan beban..... 55
Gambar 4.7 Pengujian kontrol motor terhadap kemiringan..... 56
Gambar 4.8 Denah ruangan pengujian indoor..... 57
Gambar 4.9 Hasil pemetaan dengan berbagai kecepatan..... 58
Gambar 4.10 Hasil pemetaan dari tempat awal yang berbeda..... 59
Gambar 4.11 Lokasi pengujian lokalisasi AMCL..... 60
Gambar 4.12 Pengujian navigasi docking..... 61
Gambar 4.13 Lokasi docking dan mulai navigasi docking..... 62
Gambar 4.14 Lokasi awal dan tujuan pengujian navigasi halangan baru..... 65
Gambar 4.15 Peletakan halangan baru..... 66

DAFTAR TABEL

Tabel 2.1 Spesifikasi YDLIDAR X4 11
Tabel 2.2 Spesifikasi Teensy 3.6 12
Tabel 2.3 Spesifikasi Raspberry Pi 4 13
Tabel 4.1 Pengukuran akurasi LIDAR 47
Tabel 4.2 Pembacaan LIDAR terhadap berbagai bahan 49
Tabel 4.3 Data pengujian intensitas cahaya outdoor 51
Tabel 4.4 Percobaan hubungan kecepatan dengan perpindahan 53
Tabel 4.5 Percobaan konversi kecepatan linier ke set poin pulsa 53
Tabel 4.6 Pengujian kontrol kecepatan terhadap kemiringan 56
Tabel 4.7 Pengujian pengaruh kecepatan terhadap hasil pemetaan 58
Tabel 4.8 Pengujian pengaruh perbedaan tempat mulai terhadap pemetaan 59
Tabel 4.9 Pengujian lokalisasi 60
Tabel 4.10 Hasil pengujian navigasi docking 64
Tabel 4.11 Percobaan Navigasi Halangan Baru 67

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

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Seiring perkembangan teknologi kecerdasan buatan, mulai dikembangkan *mobile robot* cerdas yang digunakan pada berbagai bidang, antara lain pabrik kimia, tambang batu bara, gas dan minyak dan lain sebagainya [1]. Banyak pekerjaan manusia yang telah tergantikan akibat adanya kecerdasan buatan. Hal ini sangat menguntungkan mengingat terdapat beberapa pekerjaan yang dilaksanakan di tempat yang berbahaya dan memperburuk kondisi kesehatan dalam jangka panjang. Contohnya yaitu dalam industri gas dan senyawa kimia diperlukan pemeriksaan kebocoran gas secara berkala sepanjang pipa saluran gas. Apabila gas tersebut terhirup dalam jumlah banyak dan terakumulasi akan berdampak buruk untuk kesehatan. Dengan adanya *autonomous* mobile robot resiko bahaya kesehatan dapat dikurangi [2]. Akan tetapi pekerjaan yang mempunyai waktu operasi yang lama dan dilakukan secara berkala seperti contoh di atas diperlukan manajemen sumber daya yang baik. Oleh karena itu dibuat sistem *autodocking* pada *mobile robot* supaya robot dapat berjalan ke stasiun pengisian terdekat pada saat daya baterai rendah untuk melakukan pengisian. Dengan adanya sistem ini robot dapat berjalan secara *autonomous* sepenuhnya.

Beberapa riset telah dilakukan dengan metode pendekatan yang berbeda. Pada riset sebelumnya, *autodocking* dilakukan menggunakan sensor infra merah yang di pancarkan oleh stasiun pengisian dan dideteksi oleh robot [3]. Metode ini memiliki kekurangan yaitu memiliki keandalan yang rendah karena keterbatasan derajat kebebasan yang menyebabkan robot tidak berhasil menuju stasiun pengisian. Kemudian robot melakukan *autodocking* dengan metode deteksi pola khusus sebagai tanda adanya stasiun menggunakan kamera [4], metode ini terbatas pada pencahayaan yang ada di lingkungan sekitar robot. Selain itu juga dilakukan *autodocking* yang menggunakan suara dengan frekuensi 900Hz-1100Hz sebagai pemandu robot untuk menuju stasiun pengisian [5]. Metode ini memiliki kekurangan yaitu jarak efektif kurang dari 100cm.

Kekurangan dari riset sebelumnya dapat diperbaiki dengan menggunakan sensor *Light Detection and Ranging* (LIDAR). LIDAR memiliki jarak pengukuran yang jauh dan tidak terlalu terpengaruh kondisi pencahayaan lingkungan. Penggunaan LIDAR dapat memungkinkan robot untuk melakukan *Simultaneous Localization and Mapping* (SLAM). SLAM adalah serangkaian algoritma robot untuk mendapatkan peta dari lingkungan sekitarnya sekaligus menentukan lokasi robot pada peta tersebut [6]. Oleh karena itu dengan mengaplikasikan SLAM robot akan dapat mengetahui dimana posisi robot terhadap stasiun pengisian. Kemudian robot melakukan navigasi menuju stasiun pengisian saat baterai rendah. SLAM dapat dilakukan dengan menggunakan Robot Operating System (ROS) yang dipasangkan pada Raspberry Pi 4.

1.2 Perumusan Masalah

Permasalahan yang dibahas dalam penelitian ini adalah:

1. Metode robot melakukan *autodocking*.
2. Metode pemetaan ruangan
3. Metode lokalisasi robot

1.3 Tujuan Penelitian

Tujuan yang ingin dicapai dari penelitian ini adalah:

1. Implementasi SLAM sebagai metode robot untuk melakukan *autodocking*.
2. Implementasi metode Hector SLAM untuk pemetaan
3. Implementasi metode AMCL untuk melakukan lokalisasi

1.4 Batasan Masalah

Berikut adalah batasan masalah dalam penelitian ini:

1. Tidak terdapat objek atau halangan dinamis pada lingkungan percobaan.
2. Kondisi jalan yang akan dilalui robot rata.
3. Peletakan halangan tidak menutupi stasiun pengisian atau terdapat ruang yang cukup untuk dilewati robot
4. Peta telah diketahui sebelum melakukan navigasi

1.5 Metodologi Penelitian

Berikut langkah-langkah yang dikerjakan pada penelitian ini:

1. Studi Literatur

Studi literatur adalah kegiatan yang dilakukan untuk mengkaji dasar teori yang berhubungan dengan penelitian yang akan dilakukan. Sumber literasi berasal dari buku, jurnal, artikel, *papper*, *datasheet* yang sudah distandarisasi nasional atau internasional.

2. Observasi dan Analisa Masalah

Observasi dilakukan pada hal-hal yang berkaitan dengan pembuatan sistem *autodocking* menggunakan LIDAR. Mulai dari metode, desain sampai komponen yang menunjang pembuatan sistem. Observasi juga dilakukan pada penelitian yang telah dilakukan sebelumnya sebagai referensi pembuatan sistem. Hasil observasi kemudian dianalisa sehingga sebisa mungkin dapat membuat sistem yang lebih baik.

3. Perancangan Sistem

Perancangan dibagai menjadi dua bagian yaitu perancangan perangkat lunak dan perangkat keras. Perancangan perangkat keras dilakukan terlebih dahulu meliputi: perancangan mekanik robot, papan *Printed Circuit Board* (PCB), dan stasiun pengisian. Sedangkan perancangan perangkat lunak adalah hal-hal yang berkaitan dengan pengolahan dan pembuatan algoritma. Mulai dari akuisisi dan interpretasi data tiap sensor serta algoritma robot untuk melakukan *autodocking*.

4. Persiapan Alat dan Bahan

Perancangan yang telah dibuat, alat dan bahan didaftar kemudian dilakukan pencarian. Alat dan bahan bisa didapatkan dari barang pribadi, peminjaman dari laboratorium, peminjaman fasilitas ITS, pembelian secara offline dan online. Alat dan bahan yang dibutuhkan antara lain Laptop, Raspberry Pi 4, motor, driver motor, lidar, baterai, kompas, mikrokontroler, aluminium, akrilik dan komponen kecil lainnya.

5. Pembuatan dan perakitan

Pada tahap pembuatan bahan-bahan yang tidak dapat diolah sendiri atau harus menggunakan jasa dikerjakan terlebih dahulu. Terutama pada pemotongan aluminium dan akrilik menggunakan jasa potong laser. Selain itu PCB yang telah didesain dikirimkan ke penyedia jasa cetak PCB. Kemudian bahan lainnya dikerjakan sembari menunggu pesanan selesai. Setelah semua bahan selesai perakitan robot dan pengkabelan dilakukan bersamaan.

6. Pengujian Sistem

Pada tahap ini dilakukan pengujian terhadap perangkat keras maupun perangkat lunak. Pengujian dilakukan untuk mengetahui apakah sistem yang telah dipasangkan sudah bekerja sesuai ekspektasi atau tidak. Pengujian dilakukan bertahap dari pengujian tiap sensor hingga pengujian keseluruhan sistem.

7. Analisa dan Evaluasi Sistem

Data dari hasil pengujian sistem dianalisa untuk mengetahui karakteristik sistem yang telah dipasangkan. Analisa dilakukan pada akurasi sensor, lingkungan yang sesuai dan posisi akhir robot setelah melakukan *autodocking*. Apabila hasil yang didapatkan masih jauh dari harapan maka akan dilakukan evaluasi sistem dan pengujian kembali.

8. Penyusunan Laporan Tugas Akhir

Penyusunan laporan tugas akhir adalah tahap akhir dari serangkaian tahap pengerjaan penelitian. Semua hal yang berkaitan dengan pengerjaan tugas akhir dimuat dalam buku laporan tugas akhir.

1.6 Sistematika Penulisan

Semua pembahasan selama pengerjaan tugas akhir *autodocking mobile robot* menggunakan LIDAR dimasukkan ke dalam laporan tugas akhir. Adapun sistematika penulisan laporan antara lain:

- 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 dan data spesifikasi modul yang terkait dalam pengerjaan tugas akhir ini.

- BAB III: Perancangan Sistem

Bab ini menjelaskan tentang perencanaan sistem perangkat keras dan perangkat lunak yang digunakan pada tugas akhir ini.

- BAB IV: Pengujian Dan Analisis

Bab ini berisi tentang pengujian sistem mulai dari sensor hingga keseluruhan sistem dan analisa hasil dari pengujian yang telah dilakukan.

- BAB V: Penutup

Bab ini berisi tentang kesimpulan yang diperoleh dari alat yang telah dibuat serta saran untuk pengembangan penelitian selanjutnya.

1.7 Relevansi

Autodocking adalah sistem yang sangat berguna untuk *autonomous mobile robot* yang memiliki waktu kerja yang lama. Dengan adanya SLAM, robot juga dapat melakukan pemetaan dan lokalisasi sehingga robot dapat digunakan untuk keperluan lain seperti robot pengawas misalnya. Pendeteksian tempat stasiun pengisian memanfaatkan fitur bentuk stasiun pengisian yang dibuat sedemikian rupa supaya dapat dideteksi oleh LIDAR.

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

BAB 2
TINJAUAN PUSTAKA

Dalam bab ini dijelaskan mengenai teori-teori dasar dan metode yang digunakan sistem untuk melakukan *autodocking* dengan sensor LIDAR. Adapun tinjauan pustaka tentang komponen yang digunakan dalam sistem ini. Pengkajian dasar teori dan tinjauan pustaka dilakukan sebagai landasan teori untuk menyelesaikan masalah yang dibahas pada tugas akhir ini.

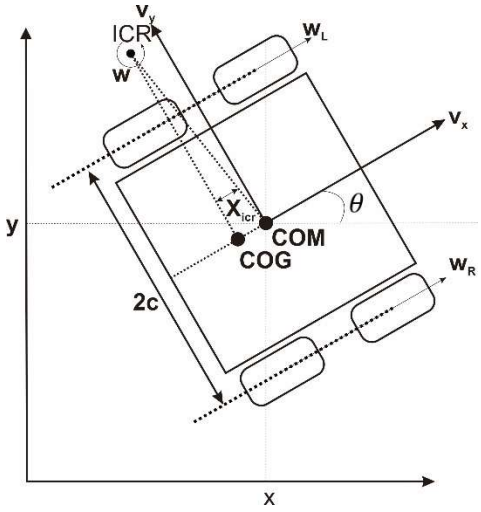
2.1 Skid Steer Mobile Robot (SSMR)

Mobile robot didefinisikan sebagai robot yang dapat bergerak dari suatu tempat ke tempat lain secara *autonomous,* tanpa bantuan operator dari luar (manusia) [1]. *Mobile robot* yang bergerak di darat dibagi lagi menjadi dua berkaki dan beroda. Untuk *mobile robot beroda* dibagi lagi menjadi banyak macam berdasarkan strukturnya. Antara lain *Differential Drive, Bicycle Drive, Tricycle Drive, Ackerman Drive, Synchronous Drive, Omnidirection Drive* [2]. Adapun tipe *Skid Steer Mobile Robot* (SSMR) yang memiliki bentuk seperti *Ackerman Steer* dengan roda depan yang tidak bisa berputar pada porosnya. Jenis ini memiliki kelebihan desain mekanik yang sederhana, memiliki kemampuan manuver yang tinggi, dan cocok digunakan untuk medan yang kasar [3]. Gambar 2.1 adalah salah satu contoh *mobile robot* jenis SSMR keluaran perusahaan Pioneer.

Gambar 2.1 Contoh SSMR: Pioneer P3AT [4]

2.1.1 Model Kinematika SSMR

Gambar 2.2 adalah model kinematika SSMR [5]:



Gambar 2.2 Kinematika SSMR

$$\dot{q} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & x_{ICR} sin\ \theta \\ \sin\theta & -x_{ICR} cos\ \theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ \omega \end{bmatrix} \quad (2.1)$$

$\dot{q}$ adalah vektor kecepatan umum yang di dalamnya ada variabel $\dot{X}$ dan $\dot{Y}$ yang merupakan koordinat kecepatan umum lateral dan longitudinal. Serta terdapat $\dot{\theta}$ kecepatan sudut umum yang bernilai sama dengan $\dot{\omega}$ karena berada dalam gerak planar. x_{ICR} adalah koordinat *instantaneous center rotation* (ICR). ICR merupakan titik imajiner yang menjadi pusat perputaran sebuah robot. ICR terletak di sepanjang sumbu x_l(sumbu x titik pusat massa). v_x adalah kecepatan longitudinal robot dan ω adalah kecepatan sudut robot.

Persamaan di atas masih belum dapat diimplementasikan sebagai desain kontrol sebuah robot. Oleh karena itu digunakan persamaan berikut:

$$\omega_R = \frac{v_x + \omega}{R} \quad (2.2)$$

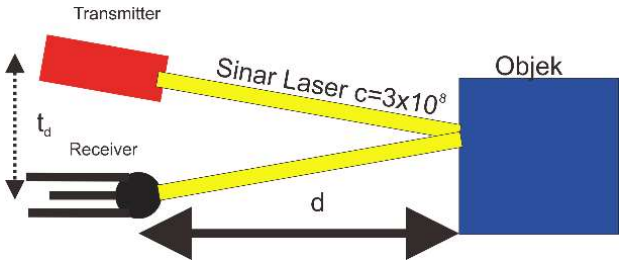
$$\omega_L = \frac{v_x - \omega}{R} \quad (2.3)$$

ω_R dan ω_l kecepatan motor kanan dan kiri. c adalah jarak antara roda robot dihitung dari titik tengah roda. Pada persamaan ini diasumsikan kecepatan motor kanan depan dan kanan kiri sama. Hal serupa berlaku untuk motor kiri.

2.2 Light Detection and Ranging (LIDAR)

LIDAR adalah sebuah radar yang memancarkan gelombang elektromagnetik yang berada pada pita frekuensi cahaya tampak [6]. LIDAR memancarkan cahaya ke seluruh ruangan kemudian mendeteksi pantulan cahayanya. Benda yang dikenai dapat berupa benda padat, cair atau gas. LIDAR yang banyak digunakan adalah pengukuran jarak menggunakan *Time of Flight* (TOF). LIDAR TOF menggunakan laser sebagai sumber cahayanya sehingga dapat mendeteksi objek yang jauh. Selain itu dengan ditambahkan mekanisme berputar, LIDAR dapat membentuk peta 2D atau 3D. Teknologi ini sangat berguna untuk pengembangan navigasi dan pemetaan *autonomous*.

LIDAR dibagi menjadi dua berdasarkan cara mengarahkan cahayanya antara lain LIDAR mekanik dan *Solid-state* LIDAR. LIDAR mekanik menggunakan optik kualitas tinggi dan sebuah mekanisme berputar supaya memiliki *Field of View* (FOV) sampai dengan 360 derajat. LIDAR jenis ini memiliki kelemahan *Signal to Noise Ratio* (SNR) yang tinggi karena aspek mekanik. Sedangkan *Solid-State* LIDAR adalah LIDAR yang tidak mempunyai mekanisme berputar dan memiliki FOV yang terbatas. Akan tetapi dengan menggabungkan beberapa data *Solid-State* LIDAR akan mendapatkan data yang menyerupai LIDAR mekanik dengan harga yang lebih murah [7]. Gambar 2.3 merupakan cara menghitung jarak yang didapatkan dari sensor TOF LIDAR.



Gambar 2.3 Ilustrasi perhitungan jarak TOF LIDAR

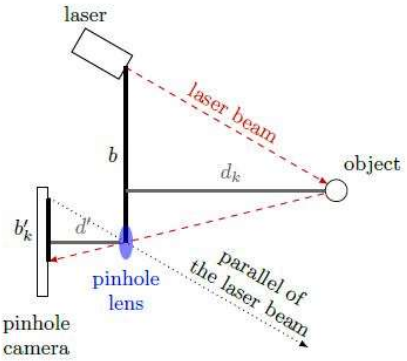
TOF LIDAR dapat mengetahui jarak d dengan menghitung waktu tempuh pantulan cahaya t_d mulai saat waktu dipancarkan dikali dengan kecepatan cahaya c. Jarak dihitung dengan persamaan berikut:

$$d = \frac{c.t_d}{2} \tag{2.4}$$

2.3 YDLIDAR X4

YDLIDAR X4 adalah produk LIDAR buatan Shenzen EAI Technology. YDLIDAR X4 dilengkapi motor yang digunakan sebagai mekanisme putar sehingga memiliki FOV 360 derajat. Output data dari YDLIDAR X4 adalah jarak baca dan sudut pembacaan yang mana dapat digunakan sebagai data *point cloud* dari lingkungan sekitar. Prinsip kerja dari LIDAR ini adalah dengan menggunakan teknik Triangulasi. Laser mengemisikan sinyal infra merah yang kemudian dipantulkan oleh objek yang dikenai. Cahaya pantulan akan masuk ke dalam lensa lubang jarum dan kemudian mengenai sensor kamera CCD. Segitiga yang terbentuk oleh (b, d_k) dan (b'_k, d') adalah sebangun. Dapat juga diartikan bahwa jarak baca dan sudut pantul tidak proporsional. Jarak d_k dapat diestimasi setelah sensor kamera membaca jarak b'_k menggunakan konsep kesebangunan segitiga [8]. Konsep tersebut digambarkan dalam gambar 2.4.

$$\frac{b}{b'_k} = \frac{d_k}{d'} \tag{2.5}$$



Gambar 2.4 Cara Kerja YDLIDAR X4 [9]

Tabel 2.1 Spesifikasi YDLIDAR X4 [10]

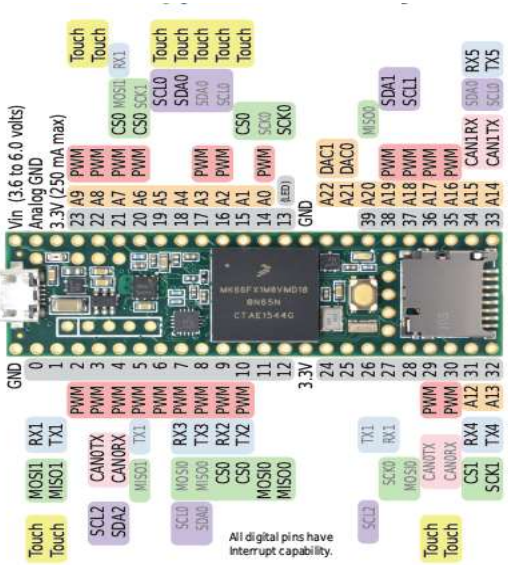
Parameter	Min	*Typical*	Max	Satuan	Keterangan
Frekuensi pengukuran jarak	-	5000	-	Hz	5000 kali per detik
Frekuensi Motor	6	-	12	Hz	PWM atau pengatur tegangan
Jarak Pengukuran	0.12	-	>10	m	80% *reflectivity*
Sudut Pengukuran		0~360		Derajat	
Error Mutlak		2		cm	Jarak ≤ 0.5m
Error Relatif	-	1.5%	-	-	0.5m<Jarak≤6m
	-	2.0%	-	-	6m<Jarak≤8m
Resolusi Sudut	0.48	0.5	0.52	Derajat	Jarak ≤ 0.5m
Masa Hidup		1500		jam	Kerja terus menerus

2.4 Mikrokontroler Teensy 3.6

Teensy 3.6 adalah modul mikrokontroler yang dilengkapi dengan prosesor 32bit ARM Cortex-M4. Dengan ukuran yang cukup kecil 62.3mm x 18mm Teensy sudah dilengkapi banyak fitur sehingga mudah digunakan. Terdapat *bootloader* yang sudah terpasang di dalam Teensy sehingga program dapat langsung diunggah menggunakan kabel USB. Kecepatan *clock* Teensy 3.6 yang mencapai 180Mhz sangat mencukupi kebutuhan pemrosesan data. Terdapat 58 pin I/O dengan toleransi sinyal 3.3V. Selain itu Teensy 3.6 dapat diprogram menggunakan Arduino IDE dengan cara menginstall perangkat lunak tambahan Teensyduino. Teensy 3.6 digunakan sebagai perantara antara sensor-sensor dan Raspberry Pi 4. Tabel 2.2 adalah spesifikasi Teensy 3.6 [11]. Sedangkan gambar 2.5 adalah pin keluarannya.

Tabel 2.2 Spesifikasi Teensy 3.6

Parameter	Nilai	Keterangan
Processor	MK66FX1M0VMD18 Cortex M4	180Mhz - 240Mhz
Flash Memory	1024	kbytes
RAM	256	kbytes
EEPROM	4096	Bytes
DMA	32	Channel
Digital I/O	58	Pin
ADC	25	Pin
DAC	2	Pin
TIMER	19	Pin
Komunikasi	USB, Serial, SPI, I2C, CAN BUS, Digital Audio, SD Card, Ethernet	-



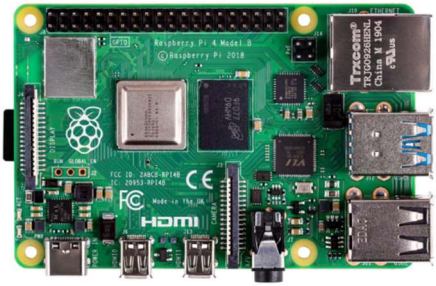
Gambar 2.5 Pinout Teensy 3.6 [12]

2.5 Raspberry Pi 4

Raspberry Pi 4 adalah generasi terbaru dari Raspberry Pi yang dirilis tahun 2019. Banyak peningkatan yang dimiliki Raspberry Pi 4 dibandingkan dengan pendahulunya Raspberry Pi 3B. Seri ini dibekali dengan prosesor Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC dengan clock 1.5Ghz. Tersedia juga beberapa varian RAM 1,2 atau 4GB. Dengan spesifikasi dan ukuran yang cukup kecil mini komputer ini dapat menjalankan program yang cukup berat seperti SLAM. Spesifikasi Raspberry Pi 4 dituliskan dalam tabel 2.3 [13] dan gambar 2.6 adalah papan Raspberry Pi 4 tampak dari atas.

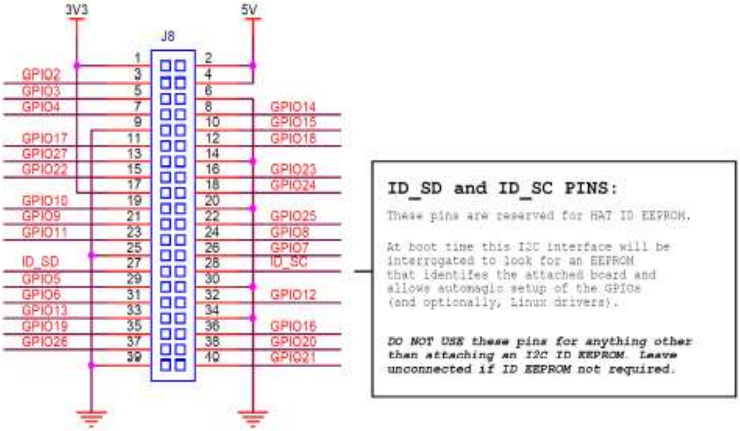
Tabel 2.3 Spesifikasi Raspberry Pi 4 [13]

Processor	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memory	1GB, 2GB or 4GB LPDDR4
Connectivity:	2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 ports 2 × USB 2.0 ports.
GPIO:	Standard 40-pin GPIO header
Video & sound	2 × micro HDMI ports (up to 4Kp60 supported) 2-lane MIPI DSI display port 2-lane MIPI CSI camera port 4-pole stereo audio and composite video port
Multimedia:	H.265 (4Kp60 decode); H.264 (1080p60 decode, 1080p30 encode); OpenGL ES, 3.0 graphics
Input power	5V DC via USB-C connector (minimum 3A1) 5V DC via GPIO header (minimum 3A1)



Gambar 2.6 Raspberry pi 4

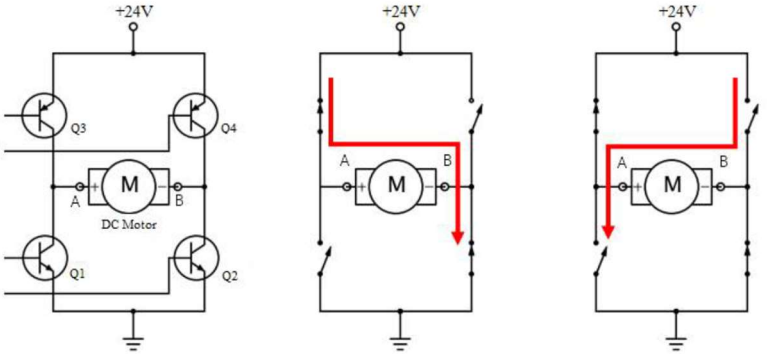
Raspberry Pi 4 dapat dijalankan menggunakan *Operating System* ubuntu dan Raspbian yang di-*install* pada *SD Card*. Seperti *Personal Computer* lainnya Raspberry Pi 4 juga mampu untuk melakukan pemasangan ROS. Raspberry pi 4 memiliki 40 pin keluaran yang mana digambarkan pada gambar 2.7.



Gambar 2.7 Pinout Raspberry Pi 4 [13]

2.6 Driver Motor DC

Driver motor adalah sebuah modul yang digunakan untuk mengatur kecepatan dan arah putaran motor. Pengaturan kecepatan motor oleh mikrokontroler dilakukan dengan mengatur *duty cycle* dari *pulse width modulation* (PWM). Sedangkan pengaturan arah putaran motor dilakukan dengan rangkaian *H-bridge*. Cara kerja dari rangkaian tersebut adalah dengan membuka dan menutup arus rangkaian menggunakan transistor. Gambar 2.8 merupakan penjelasan cara kerja rangkaian *H-bridge*.



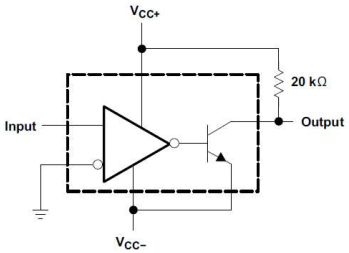
Gambar 2.8 Cara kerja H-bridge motor [14]

Terdapat banyak manufaktur yang membuat rangkaian driver motor menyesuaikan dengan kebutuhan pengguna. Salah satunya adalah modul keluaran Cytron yang memiliki kapasitas arus maksimal 10A dan terdapat 2 kanal output motor. Selain itu rangkaian ini sudah menggunakan komponen *solid-state* sepenuhnya dan tidak menggunakan relay mekanik. Oleh karena itu modul ini memiliki waktu respon yang lebih cepat dan menghindari efek kerusakan penggunaan relay mekanik [MDD. Terdapat 5 pinout untuk mengatur arah dan kecepatan motor, antara lain pin ground, PWM1, direksi1, PWM2, dan direksi2. Driver Cytron MDD 10 ditunjukan pada gambar 2.9.

Gambar 2.9 Driver motor cytron MDD 10A [15]

2.7 Zero Crossing Detector

Sinyal yang dihasilkan oleh encoder motor JLN4B02 adalah sinyal sinusoidal dengan tegangan puncak 1Vpp. Supaya sinyal dapat diterjemahkan oleh mikrokontroler, dibutuhkan rangkaian *Zero Crossing Detector* (ZCD). ZCD adalah rangkaian yang menghasilkan sinyal output saat sinyal AC yang masuk ke dalam input melewati tegangan 0V. Rangkaian ZCD terdiri dari sebuah op-amp yang dikonfigurasi sebagai rangkaian komparator. Gamabr 2.10 adalah contoh rangkaian ZCD menggunakan IC LM311.



Gambar 2.10 Contoh rangkaian ZCD LM 311 [16]

Terdapat beberapa pertimbangan dalam mendesain rangkaian ZCD. Rentang input tegangan, *minimum overdrive voltage, output drive current,* dan waktu respon. Spesifikasi yang dimiliki LM311 cocok digunakan untuk mendeteksi sinyal encoder yang memiliki frekuensi tinggi. Karena LM 311 memiliki waktu respon yang sangat cepat yaitu 165ns atau setara 6Mhz.

2.8 Robot Operating System (ROS)

Robot Operating System (ROS) adalah sebuah meta *operating system* yang bersifat *open source* untuk robot. ROS menyediakan *tools* dan *libraries* untuk mengambil, menulis, mem-*build* dan menjalankan *script code* antar beberapa komputer. ROS juga dapat disebut sebagai platform perangkat lunak yang menyediakan berbagai pengembangan *environment* khusus untuk aplikasi program pembuatan robot. Tujuan adanya ROS adalah untuk membuat sebuah *environment* dimana pengguna seluruh dunia bisa berkolaborasi dalam pembuatan perangkat lunak robot. Dengan kata lain ROS membantu pembuat robot agar dapat menggunakan riset-riset yang telah dilakukan sebelumnya [17].

2.9 Master

Master adalah server yang mengatur koneksi dan komunikasi pesan antar node-node. Master dapat dibuat dengan memanggil perintah roslaunch. Jadi tanpa adanya master ros tidak dapat melakukan komunikasi antar node. Ketika master dijalankan secara *default* ROS_MASTER_URI akan membuat alamat URI yang berisi alamat IP lokal komputer dengan *port* 11311.

2.10 Node

Node adalah bagian terkecil dari ROS. Node berisi program-program yang dapat melakukan pertukaran pesan menggunakan Topic dan Services. Setiap node berjalan secara mandiri dan paralel terhadap node lain. Pada saat awal dijalankan Node mendaftarkan informasi berupa nama, jenis pesan, alamat URI dan nomor port dari node tersebut.

2.11 Package

Semua aplikasi ROS dikembangkan dalam bentuk suatu package. Package berisi semua konfigurasi file untuk menjalankan package lain atau node. Dalam package juga terdapat semua file yang dibutuhkan untuk menjalankan package tersebut termasuk *depedency libraries,* dataset, dan *file* konfigurasi. Adapun Metapackage yang berisi package yang sering digunakan. Terdapat 10 metapackage bawaan dari ROS antara lain AMCL, DWA, EKF, map_server dan lain-lain.

2.12 Message

Message adalah pesan yang dikirimkan antar node. Pesan yang dikirimkan dapat berisi nilai int, float, array, char, dan tipe data yang lain. Nested message structure adalah pesan yang berisi jajaran pesan atau message array.

2.13 Topic

Suatu node dapat berhubungan dengan node lain dengan topic yang telah ditentukan oleh pengguna. Saat node bertindak sebagai publisher node mengirimkan Message ke dalam sebuah Topic. Kemudian ada Node yang bertindak sebagai Subscriber yaitu penerima pesan berdasarkan topic yang diinginkan. Topic mempermudah komunikasi antar node-node.

2.14 Service

Service adalah komunikasi dua arah yang bersifat sinkron antara service server dan service client. Service client bertugas sebagai peminta sebuah service untuk melakukan tugas tertentu sedangkan service server merespon permintaan dari client.

2.15 rosbuild & catkin

rosbuild adalah perintah untuk mem-*build* sistem ROS. Penggunaan rosbuild mulai tergantikan dengan adanya perintah catkin. Meskipun masih ada beberapa versi yang menggunakan, rosbuild tidak disarankan oleh pengembang ROS. Catkin dapat mem-*build* sistem dengan berbagai macam platform dengan menggunakan CMake (*Cross Platform Make*). CMake diatur dalam file bernama CMakeList.txt.

2.16 rosrun & roslaunch

rosrun adalah perintah untuk menjalankan sebuah node dalam sebuah package. Dalam node tersebut sudah terdapat variabel *environment* ROS_HOSTNAME sebagai alamat URI dengan nomor Port yang unik. roslaunch untuk menjalankan beberapa node sekaligus. roslaunch mengeksekusi berkas program dengan ekstensi “*.launch”.

2.17 bag

Data yang diperoleh dari pesan ROS dapat disimpan dalam berkas dengan ekstensi "*.bag". rosbag dapat memuat kembali data yang telah disimpan. Kegunaan dari perintah ini adalah ketika robot membutuhkan data *environment* yang telah didapatkan sebelumnya seperti pemetaan robot tidak perlu melakukan pengambilan data ulang.

2.18 rosserial

rosserial adalah sebuah package yang berfungsi untuk mengubah pesan komunikasi ROS ke dalam bentuk data serial. Rosserial biasanya digunakan untuk komunikasi antara komputer dengan mikrokontroler. Dalam rosserial ada juga rosserial server dan rosserial client yang bekerja seperti service. Keduanya dapat melakukan pengiriman dan penerimaan data.

Gambar 2.11 Robot operating system [17]

ROS juga menyediakan program SLAM dalam beberapa jenis package antara lain HectorSLAM, Cartographer, dan gmapping. Hasil dari pemetaan tersebut kemudian divisualisasikan menggunakan program rviz.

2.19 Simultaneous Localization and Mapping (SLAM)

SLAM adalah sebuah sistem bagaimana sebuah robot dapat melakukan navigasi di lingkungan yang tidak diketahui. Untuk dapat melakukan navigasi robot harus mengetahui peta lingkungannya atau posisinya dalam peta tersebut. Tetapi kedua data tersebut tidak diberikan kepada robot sehingga robot harus melakukan lokalisasi dan pemetaan secara bersamaan [18]. SLAM terdiri dari dua proses yaitu lokalisasi dan pemetaan.

Pemetaan adalah sebuah tugas yang diberikan pada robot untuk mengeksplorasi dan merepresentasikan data mengenai ruang yang dilewatinya. Terdapat beberapa macam bentuk peta antara lain *Metric Maps, Topological Maps, Topometric Maps, semi metric topological maps,* dan *measurement maps*.

Lokalisasi adalah penentuan posisi dalam sebuah peta. Terdapat dua jenis lokalisasi yaitu *global localization dan local localization*. *Global localization* adalah penentuan posisi robot terhadap *frame* global tanpa mengetahui terlebih dahulu posisi awalnya. Dengan kondisi ini robot tidak memiliki data *feedback* dari odometri dan data yang didapatkan dari sensor menjadi ambigu. Permasalahan ini dapat diselesaikan dengan menggunakan Monte Carlo Localization. Sedangkan *local localization* hampir sama dengan lokalisasi sebelumnya akan tetapi diberikan data posisi awal. Algoritma pencarian posisi bergantung pada parameter model dari lingkungan kemudian difilter menggunakan filter bayes.

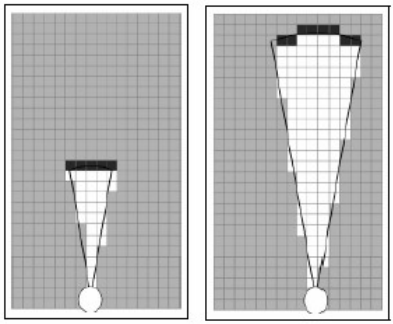
2.20 Hector SLAM

Hector SLAM adalah salah satu algoritma SLAM yang bersifat *open source* dan tersedia dalam *package* ROS. Hector SLAM dibuat oleh team Hector dari Darmstadt bersama dengan tim peneliti internasional lainnya. Awalnya algoritma ini digunakan pada robot UGV untuk tugas *Urban Search and Rescue*. Dalam package ini tersedia program mulai dari pengolahan data sensor untuk SLAM 2D dan 3D, perencanaan eksplorasi, hingga kontroler robot untuk melakukan navigasi [19][20]. Pada penelitian ini Hector SLAM digunakan untuk membangun peta 2D. Peta direpresentasikan dalam bentuk *Occupancy Grid*.

Dalam pembuatan peta 2D, robot harus mengetahui perpindahan robot di dalam peta. Hector SLAM mendeteksi perpindahan robot menggunakan metode *Scan Matching*. Inilah yang menjadi kelebihan dari Hector SLAM, dimana *package* ini tidak membutuhkan data odometri dari encoder motor sehingga dapat mengatasi masalah medan yang tidak rata.

2.21 Occupancy Grid

Occupancy Grid adalah salah satu jenis representasi peta yang menggambarkan lingkungan dalam grid yang terpisah. Informasi lingkungan bisa didapatkan dari data sensor jarak dan kamera. Terdapat dua representasi Occupancy Grid antara lain Occupancy grid biner dan Occupancy grid probability [21]. Gambar 2.12 adalah contoh hasil pembuatan peta Occupancy Grid.



Gambar 2.12 Occupancy Grid Map [21]

Occupancy grid biner menggambarkan lingkungan ke dua nilai 0 dan 1. 0 untuk lingkungan yang belum telusuri atau tidak ada halangan. Nilai 1 untuk koordinat lingkungan yang terdapat halangan. Sedangkan Occupancy Grid Probability merepresentasikan peta lebih detail dengan mencacah nilai 0 sampai 1. Apabila mendekati 0 berarti tidak terdapat halangan atau belum terjamah. Sebaliknya jika mendekati 1 maka semakin pasti bahwa pada grid tersebut terdapat halangan.

Untuk memperbarui nilai probabilitas setiap piksel pada peta *occupancy grid map* digunakan persamaan log odd, diterapkan aturan bayes.

$$p(m_{x,y}|z) = \frac{p(z|m_{x,y})p(m_{x,y})}{p(z)} \tag{2.6}$$

$p(m_{x,y}|z)$ adalah nilai posterior dari piksel $m_{x,y}$ dengan diberikan pengukuran z. $p(z|m_{x,y})$ adalah *measurement model* dari pengukuran yang

telah ditentukan sebelumnya (contoh : *likelyhood model*). Sehingga untuk merepresentasikan ada dan tidaknya halangan pada suatu piksel peta digunakan persamaan sebagai berikut :

$$p(m_{x,y} = 1|z) = \frac{p(z|m_{x,y} = 1)p(m_{x,y}=1)}{p(z)} \quad (2.7)$$

$$p(m_{x,y} = 0|z) = \frac{p(z|m_{x,y} = 0)p(m_{x,y}=0)}{p(z)} \quad (2.8)$$

Untuk mempermudah perhitungan, diperkenalkan persamaan log odd yaitu persamaan yang menunjukkan perbandingan antara kemungkinan peluang terjadi dan tidak terjadi.

$$odd = \frac{p(ada\ halangan)}{p(tidak\ ada\ halangan)} \quad (2.9)$$

Dengan memasukkan persamaan 2.7 dan 2.8 pada persamaan 2.9 akan dihasilkan persamaan:

$$odd = \frac{p(z|m_{x,y} = 1)p(m_{x,y}=1)}{p(z|m_{x,y} = 0)p(m_{x,y}=0)} \quad (2.10)$$

$$\log odd = \frac{\log p(z|m_{x,y} = 1)p(m_{x,y}=1)}{\log p(z|m_{x,y} = 0)p(m_{x,y}=0)} \quad (2.11)$$

$$\log odd = \frac{\log p(z|m_{x,y} = 1)}{\log p(z|m_{x,y} = 0)} + \frac{\log\ (m_{x,y}=1)}{\log p(m_{x,y}=0)} \quad (2.12)$$

$$\log odd^{+} = \log odd\ pengukuran + \log odd^{-} \quad (2.13)$$

Persamaan 2.13 digunakan sebagai persamaan untuk memperbarui nilai probabilitas keberadaan halangan pada piksel peta.

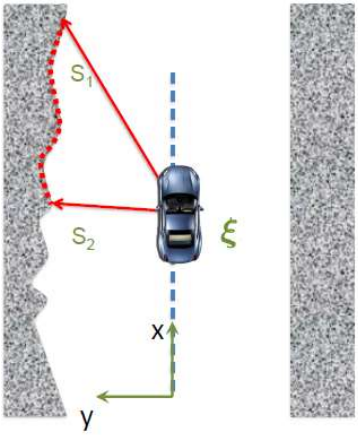
2.22 Scan Matching

Scan matching adalah sebuah solusi dari bagaimana cara untuk mendapatkan translasi dan rotasi dari sebuah robot dengan diberikan dua set data scan LIDAR. Hasil dari perhitung Scan Matching dapat digunakan sebagai odometri yang jauh lebih akurat dibandingkan dengan encoder motor [20]. Terdapat banyak pendekatan dari Scan Matching ini, teknik yang banyak digunakan adalah Iterative Closest Point (ICP).

ICP terdiri dari tiga langkah, asosiasi, transformasi dan evaluasi error. Keseluruhan langkah dilakukan hingga mendapatkan keselarasan yang diinginkan. Dari dua set data scan, data scan pertama adalah target transformasi. Untuk mencari besarnya transformasi antara scan lama dan scan baru dapat menggunakan metode tetangga terdekat. Titik scan dari yang baru diasosiasikan dengan scan target kemudian dihubungkan dengan garis. Langkah selanjutnya adalah pencarian transformasi dengan meminimalisir rata-rata kuadrat jarak antara kedua titik yang telah terhubung. Lalu langkah ketiga adalah mengevaluasi error untuk menentukan apakah perlu dilakukan pengulangan keseluruhan proses.

Scan matching yang terdapat dalam Hector SLAM melakukan pencarian titik akhir sinar pada peta menggunakan pendekatan Gauss-Newton. Dengan metode ini tidak perlu melakukan asosiasi data antara titik akhir sinar atau pencarian posisi secara menyeluruh. $\xi = (p_x, p_y, \psi)^T$ adalah pose robot [20]. n adalah jumlah *scan* M adalah nilai peta pada koordinat yang diberikan S_i. Gambar 2.13 adalah posisi robot pada waktu awal.

$$\xi^* = argmin_\xi \sum_{i=1}^{n} [1 - M(S_i(\xi))]^2 \tag{2.14}$$



Gambar 2.13 Ilustrasi Hector scan matching [21]

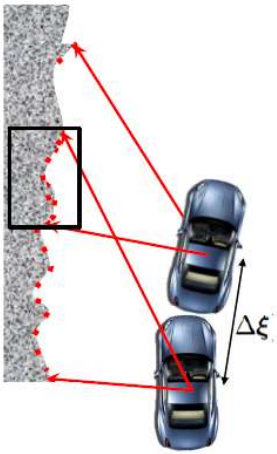
Pada persamaan di atas nilai yang dicari adalah nilai terkecil dari sigma jadi dapat dituliskan:

$$\sum_{i=1}^{n}[1 - M(S_i(\xi + \Delta\xi))]^2 \rightarrow 0 \quad (2.15)$$

$$\sum_{i=1}^{n}[1 - M(S_i(\xi) - \nabla M\big(S_i(\xi)\big)\frac{\partial S_i(\xi)}{\partial \xi}\Delta\xi))]^2 \rightarrow 0 \quad (2.16)$$

Persamaan 2.15 diubah ke persamaan 2.16 menggunakan ekspansi taylor dari fungsi M. Solusi dari persamaan 2.16 adalah $\Delta\xi$ yang dapat dicari menggunakan persamaan Gauss Newton. Hasil evaluasi dari persamaan tersebut memberikan $\Delta\xi$ yang memperkecil fungsi objektif. Gambar 2.14 adalah gambar penyesuaian dua hasil *scan* dari dua posisi robot.

$$\Delta\xi = H^{-1} \sum_{i=1}^{n}\left[\nabla M\big(S_i(\xi)\big)\frac{\partial S_i(\xi)}{\partial \xi}\Delta\xi\right]^T [1 - M(S_i(\xi))] \quad (2.17)$$



Gambar 2.14 Ilustrasi Hector scan matching 2 [21]

2.23 Adaptive Monte Carlo Localization

Monte Carlo Localization (MCL) adalah salah satu teknik lokalisasi robot dalam sebuah peta menggunakan particle filter [22]. Kemudian dilakukan pengembangan terhadap MCL supaya lebih efisien dengan membuat jumlah sampel atau partikel menjadi adaptif sehingga menjadi Adaptive MCL atau AMCL. Salah satu teknik yang digunakan untuk mengubah jumlah partikel dengan Kullback Leibler Distance atau KLD-Sampling.

2.24 Tinjauan Pustaka

Adapun tinjauan pustaka mengenai penelitian yang berhubungan dengan *autonomous mobile robot*. Tinjauan pustaka ini bertujuan untuk mengulas penelitian sebelumnya dan dijadikan bahan refleksi untuk penelitian yang akan dilakukan.

2.24.1 Sensor Guided Docking of Autonomous Mobile Robot for Battery Recharging [23]

Penelitian ini adalah tentang membuat sebuah sistem *autonomous docking* yang dipandu dengan sensor infra merah dan sensor ultrasonik. Penelitian menggunakan robot dengan bentuk platform *differential drive* 2 roda dan 1 roda bebas. Mikrokontroler yang digunakan adalah atmega 16. Robot akan melakukan *docking* ketika kondisi baterai kurang dari 40%. Robot mencari stasiun pengisian dengan algoritma pencarian acak. Yang dimaksud dengan pencarian acak adalah ketika robot sudah dalam kondisi harus di-*charge* kembali robot akan berputar 360 derajat untuk mencari objek dengan jarak kurang dari 200cm. sistem ini memiliki kekurangan tidak bisa membedakan antara stasiun pengisian dengan objek lain.

2.24.2 Vision-Based Autonomous Docking and Re-charging System for Mobile Robot in Warehouse Environment [24]

Pada penelitian ini dilakukan pembuatan sistem *autonomous docking* menggunakan pengolahan citra dengan sensor kamera kinect. Metode pendeteksian stasiun pengisian yang digunakan adalah dengan mendeteksi *augmented reality* (AR) code yang ditempelkan pada stasiun pengisian. Ketika pengisian baterai akan dilakukan robot akan mencari objek (stasiun pengisian) yang terdapat AR code kemudian melakukan navigasi ke arah objek tersebut. Sistem ini diimplementasikan pada robot

tipe *differential drive* dengan tambahan 4 roda bebas sebagai penopang. Sistem ini sudah dapat membedakan gambar AR code meskipun dalam kondisi banyak *noise* pada gambar. Akan tetapi perubahan pencahayaan dapat mengurangi atau membuat kesalahan interpretasi AR code sehingga perlu dilakukan penelitian lebih lanjut.

2.24.3 Large-Scale Outdoor SLAM Based on 2D Lidar [25]

Penelitian ini fokus pada pembuatan SLAM yang dapat digunakan untuk outdoor menggunakan LIDAR Velodyne V16. Kontribusi dari penelitian ini adalah akurasi dan keandalan metode scan matching yang meningkat setelah diterapkannya algoritma Correlative Scan Matching (CSM). Kemudian untuk pendeteksian Loop Closure digunakan AdaBoost yang sangat efisien dalam menyeleksi loop closure palsu. Pada bagian back-end SLAM digunakan algoritma graph optimization berbasis *incremental smoothing and mapping* (iSAM). Hasil dari penelitian ini sudah bisa diterapkan pada kendaraan dalam ukuran besar yaitu mobil golf.

2.24.4 Mobile Robot Navigation Using 2D LIDAR [26]

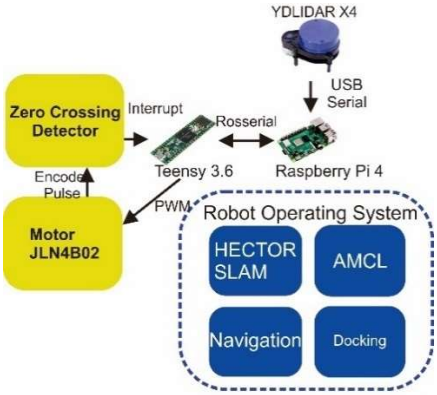
Kontribusi yang dihasilkan dari penelitian ini adalah navigasi menggunakan 2D LIDAR dengan cara mencari pembacaan sudut yang tidak terdapat penghalang atau objek di depannya. Setelah mengetahui sudut-sudut tersebut kemudian dihitung panjang busur yang terbentuk dan kemudian dipilih busur yang terlebar. Algoritma ini menggunakan 5 elemen filter gauss untuk memperhalus pembacaan jarak ketika ada penghalang. Sensor yang digunakan adalah sensor SICK LMS 151. Platform robot yang digunakan *differential drive* 2 roda ditambah 1 roda bebas.

BAB 3
PERANCANGAN SISTEM

Bab ini berisi penjelasan tentang perancangan keseluruhan sistem mulai dari perancangan perangkat keras dan perangkat lunak yang digunakan selama penelitian dilakukan. Bab ini bertujuan untuk memberikan gambaran umum dan perancangan sistem secara mendetail dari sistem yang akan dibuat

3.1 Gambaran Umum Sistem

Sistem yang akan dibuat adalah sistem *autonomous docking* pada *mobile robot*. Sensor utama yang digunakan adalah YDLIDAR X4. Data dari LIDAR diambil oleh Raspberry Pi 4 dan data sensor lain didapat dari mikrokontroler Teensy 3.6. Raspberry Pi 4 bertindak sebagai pengolah data *Top Level* sedangkan Teensy 3.6 sebagai pengolah data *Low Level* yang lebih dekat dengan perangkat keras. Platform dari robot adalah model *skid steering mobile robot*. Sistem mencakup lokalisasi, pemetaan, navigasi docking dan mendeteksi stasiun pengisian. Sebelum melakukan navigasi docking robot melakukan pemetaan terlebih dahulu. Kemudian lokalisasi dan navigasi dapat dilakukan. Gambaran sistem secara umum dijelaskan pada gambar 3.1

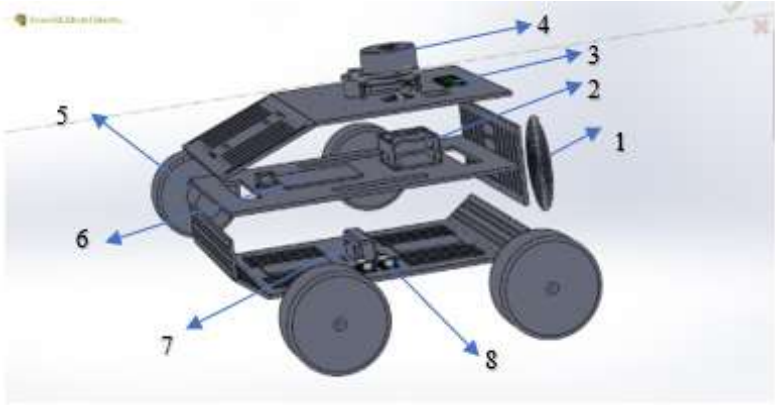


Gambar 3.1. Gambaran umum sistem

3.2 Perancangan Perangkat Keras

3.2.1 Platform mobile robot

Model platform dari *mobile robot* yang akan dibuat adalah *skid steering mobile robot* (SSMR). Pertimbangan pemilihan platform SSMR karena kelebihan desain mekaniknya yang cocok untuk berbagai medan [3]. Bahan dasar yang digunakan adalah aluminium 1.5mm dan akrilik 5mm untuk bagian luarnya. Sedangkan bagian dalam menggunakan akrilik 3mm supaya beban robot lebih ringan. Selain itu desain aluminium dibuat berlubang untuk mengurangi beban. Semua modul berada di dalam robot kecuali LIDAR. Setiap bagian rancangan *mobile robot* dijelaskan pada gambar 3.2.



Gambar 3.2 Rancangan *mobile robot*

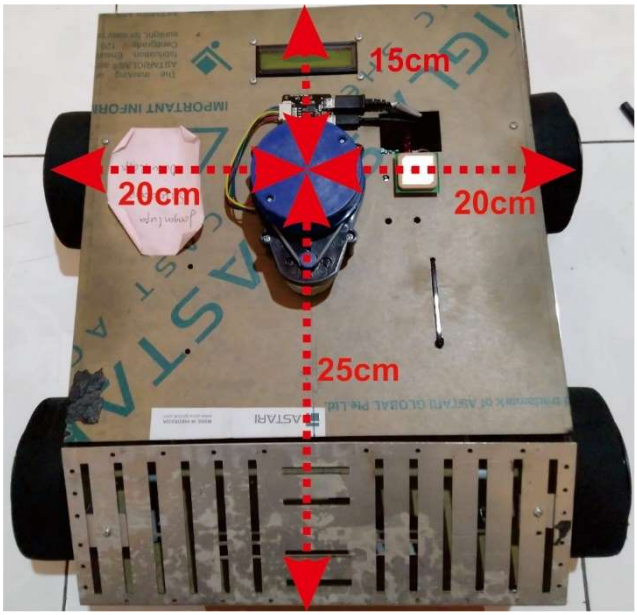
Keterangan:

1. *Wireless charger receiver coil*
2. Raspberry Pi 4
3. LCD 16x2
4. YDLIDAR X4
5. Motor DC
6. PCB ekstensi Teensy 3.6
7. Baterai 2200mah
8. *Buck Converter* LM 2596

Platform robot dibagi menjadi 3 bagian bawah tengah dan atas. Bagian bawah menjadi tempat roda, motor, driver motor, dan baterai. Bagian tengah terdapat papan PCB ekstensi Teensy 3.6, Raspberry Pi 4, dan modul YDLIDAR X4. Sedangkan bagian atas diletakkan LCD 16x2 sebagai antar muka, dan Sensor YDLIDAR X4

3.2.2 Pemasangan sensor

Sensor YDLIDAR X4 diletakkan di bagian paling atas robot bertujuan supaya tidak ada bagian robot yang menghalangi pembacaan sensor. LIDAR diletakkan di tengah untuk memudahkan penentuan transformasi koordinat antar *frame* laser dan *frame base* robot di ROS. YDLIDAR X4 dipasang pada *mobile robot* seperti pada gambar 3.3.

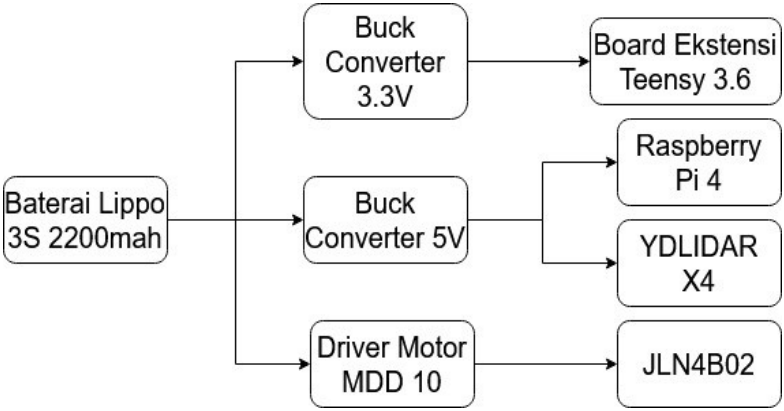


Gambar 3.3 Peletakan sensor YDLIDAR X4

3.2.3 Catu Daya

Catu daya utama dari *mobile robot* adalah Baterai lithium polimer 3 sel dengan kapasitas 2200mah dan tegangan maksimal 12.6V. Catu daya ini terlalu besar untuk memberikan *supply* ke mikrokontroler dan

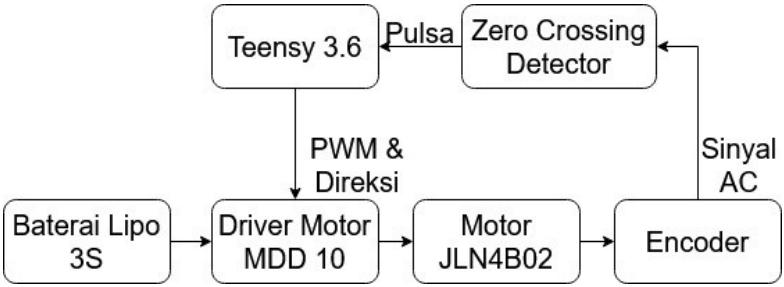
komputer. Oleh karena itu digunakan dua buck converter yang digunakan untuk menurunkan tegangan. Buck converter yang pertama di set pada tegangan 3.3V untuk memberikan tegangan pada sensor dan mikrokontroler. Sedangkan yang kedua diset pada tegangan 5V sebagai sumber catu daya Raspberry Pi 4 dan YDLIDAR X4. Gambar 3.4 menunjukkan blok diagram dari pembagian catu daya.



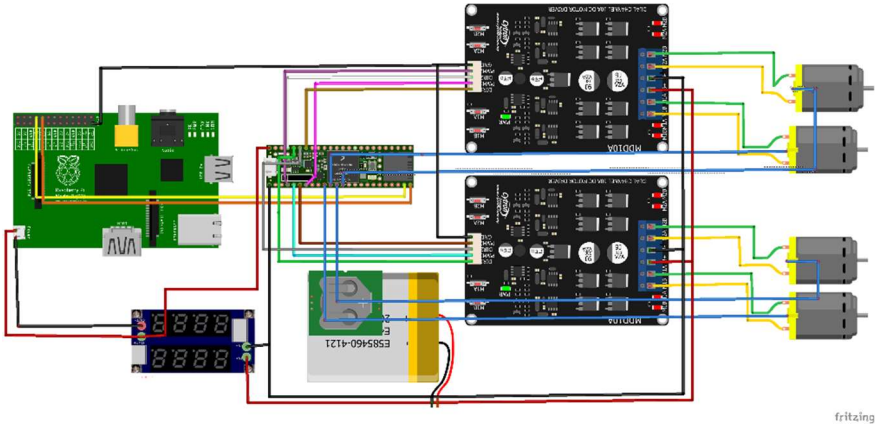
Gambar 3.4 Pembagian catu daya

3.2.4 Pemasangan Motor

Pemasangan motor dibagi menjadi 3 bagian antara lain motor, driver motor dan encoder motor. Motor yang digunakan adalah motor Sankyo JLN4B02. Motor ini mempunyai keluaran pin sejumlah 4 antara lain: 12V, GND, GND, *Signal*. Sinyal yang dihasilkan oleh encoder adalah sinyal sinusoidal dengan tegangan 1Vpp. Output dari sinyal encoder diolah terlebih dahulu menggunakan Zero Crossing Detector supaya sinyal dapat diterima Teensy 3.6. Pemasangan motor digambarkan dalam blok diagram gambar 3.5. Sedangkan gambar 3.6 adalah rangkaian skematik dari pemasangan motor.



Gambar 3.5 Diagram pemasangan motor



Gambar 3.6 Skematik pemasangan motor

Teensy mengatur kecepatan motor menggunakan pin PWM dan mengatur arah putaran motor menggunakan pin direksi. Motor Driver MDD 10 membutuhkan 5 pin input untuk menggerakkan 2 motor dengan maksimal arus 10A. Berikut adalah pemetaan pin pada pemasangan motor. Pin dipetakan pada tabel 3.1.

Tabel 3.1 Pemetaan pin komponen motor

Pin Teensy 3.6	Pin Tujuan	Keterangan
2	PWM1	Driver MDD10(1) motor 1
3	PWM2	Driver MDD10(1) motor 2
4	PWM1	Driver MDD10(2) motor 3
5	PWM2	Driver MDD10(2) motor 4
23	DIR1	Driver MDD10(1) motor 1
22	DIR2	Driver MDD10(1) motor 2
21	DIR1	Driver MDD10(2) motor 3
20	DIR2	Driver MDD10(2) motor 4
11	Output ZCD1	Sinyal Encoder Motor 1
10	Output ZCD2	Sinyal Encoder Motor 2
9	Output ZCD3	Sinyal Encoder Motor 3
8	Output ZCD4	Sinyal Encoder Motor 4

3.2.5 Kinematika mobile robot

Desain mekanik robot dikerjakan dengan perangkat lunak SolidWorks 2017. Dimensi dari *mobile robot* secara keseluruhan adalah 37.5x38x22cm seperti pada gambar 3.7. Radius roda *mobile robot* 11.5cm dinotasikan sebagai R. Sehingga apabila diturunkan dari persamaan tiap roda

$$v_{1x} = \omega_R * \mathrm{R} \tag{3.1}$$

$$v_{2x} = \omega_R * \mathrm{R} \tag{3.2}$$

$$v_{3x} = \omega_L * \mathrm{R} \tag{3.3}$$

$$v_{4x} = \omega_L * \mathrm{R} \tag{3.4}$$

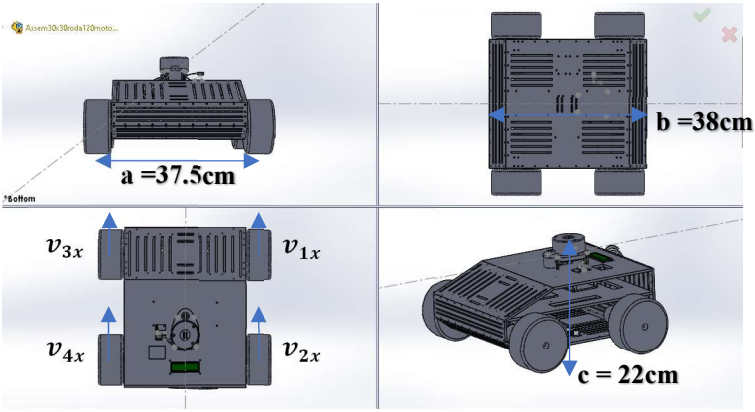
Apabila persamaan kinematika SSMR dimasukkan dengan nilai c yang sudah diketahui 37.5cm dan R = 11.5cm

$$v_{1x} = \frac{v_x + \omega * 37.5}{11.5} * 11.5 \quad (3.5)$$

$$v_{2x} = \frac{v_x + \omega * 37.5}{11.5} * 11.5 \quad (3.6)$$

$$v_{3x} = \frac{v_x - \omega * 37.5}{11.5} * 11.5 \quad (3.7)$$

$$v_{4x} = \frac{v_x - \omega * 37.5}{11.5} * 11.5 \quad (3.8)$$



Gambar 3.7 Kinematika mobile robot

Adapun *inverse kinematic mobile robot* yaitu mengubah pembacaan kecepatan tiap roda menjadi kecepatan global robot. Kecepatan robot dari *inverse kinematic* dinotasikan V_{RIV}. Kecepatan ini dibagi lagi menjadi dua macam kecepatan yaitu kecepatan sumbu x atau V_{XIV} dan kecepatan sumbu y atau V_{YIV}. θ adalah orientasi robot saat ini.

$$V_{RIV} = \frac{(v_{1x} + v_{2x} + v_{3x} + v_{4x})}{4} \quad (3.9)$$

$$V_{XIV} = V_{RIV} * cos\theta \quad (3.10)$$

$$V_{YIV} = V_{RIV} * sin\theta \quad (3.11)$$

Kemudian kecepatan robot dapat diintegralkan menjadi posisi robot saat ini. Berikut adalah persamaan untuk mendapatkan posisi robot

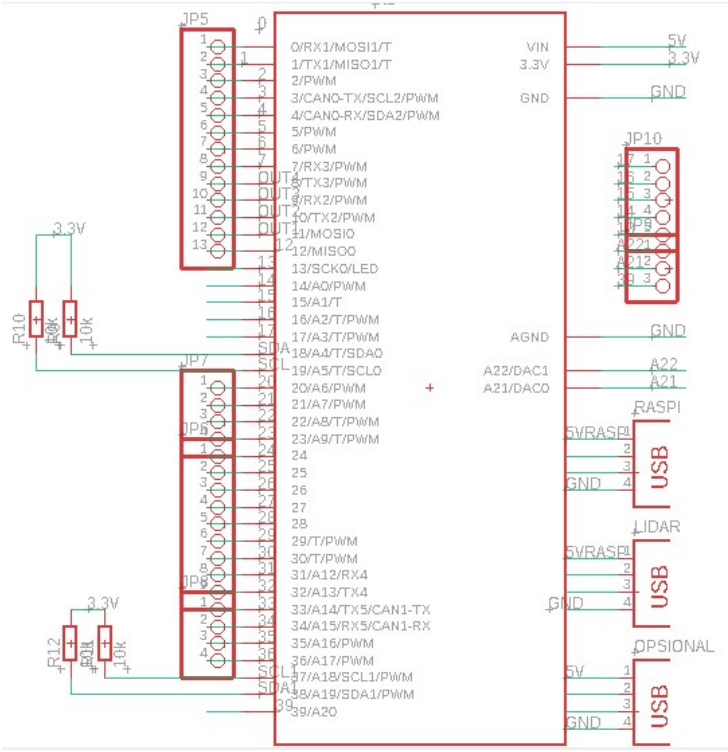
$$pos_x = \int V_{XIV} \tag{3.12}$$

$$pos_y = \int V_{YIV} \tag{3.13}$$

$$pos_{total} = \sqrt[2]{{pos_x}^2 + {pos_y}^2} \tag{3.14}$$

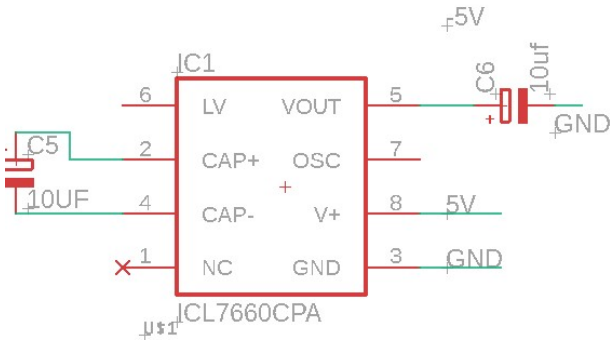
3.2.6 Board ekstensi Teensy 3.6

Board ekstensi didesain menggunakan perangkat lunak Eagle CAD versi 9.5.1 *education*. Board ekstensi digunakan untuk mempersiapkan pin-pin yang akan digunakan pada Teensy 3.6 pada penelitian ini. Selain itu komponen-komponen elektronik lain seperti ZCD dan pengatur tegangan diletakkan pada papan tersebut guna mengurangi penggunaan kabel. Dalam merancang sebuah *board* terdapat dua berkas yaitu *schematic* dan *board*. *Schematic* adalah berkas yang memuat koneksi antar komponen. Berkas *board* digunakan untuk menyimpan penjaluran antar komponen. Rangkaian skematik dari papan ekstensi Teensy 3.6 digambarkan pada gambar 3.8.

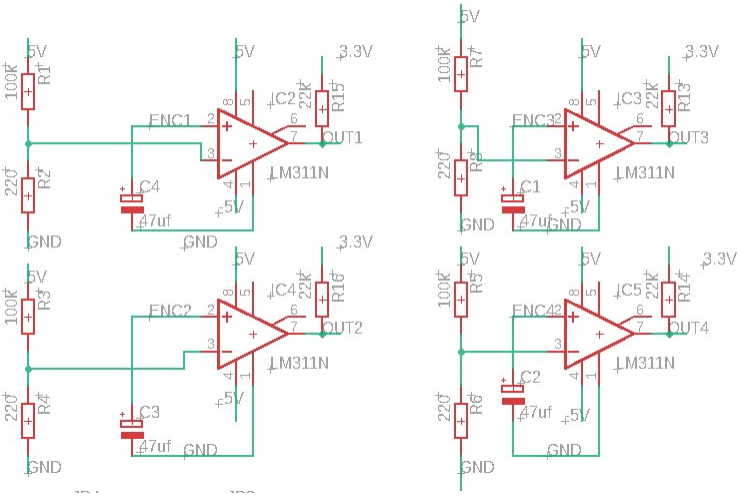


Gambar 3.8 Skematik rangkaian board ekstensi Teensy 3.6

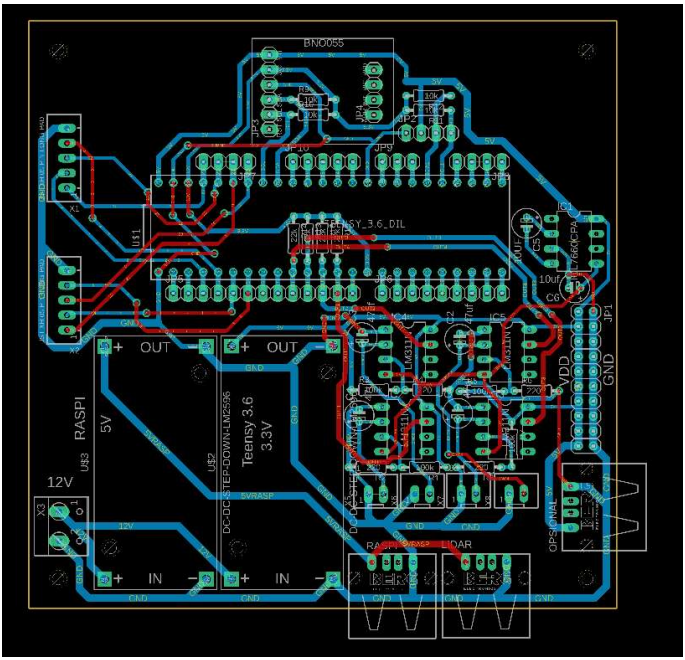
Rangkaian ZCD menggunakan IC Op Amp LM311N membutuhkan input tegangan negatif sehingga diperlukan komponen ICL 7660CPA untuk mengubah tegangan +5V menjadi -5V. Rangkaian catu daya -5V digambarkan pada gambar 3.9. +5V. Pada rangkaian ZCD dibuat batas minimal tegangan untuk memotong sinyal *noise* dengan tegangan puncak kurang dari 10mV. Batas 10mV disambungkan dengan input *inverting*. Sedangkan sinyal encoder disambungkan ke input *non-inverting*. Masing-masing motor membutuhkan 1 ZCD sehingga nantinya akan ada 4 ZCD seperti pada gambar 3.10 yang outputnya akan terhubung ke pin *interrupt* Teensy 3.6 yaitu pin 11 hingga pin 8. Papan yang akan dicetak nantinya akan berbentuk seperti gambar 3.11.



Gambar 3.9 Skematik rangkaian power supply -5V



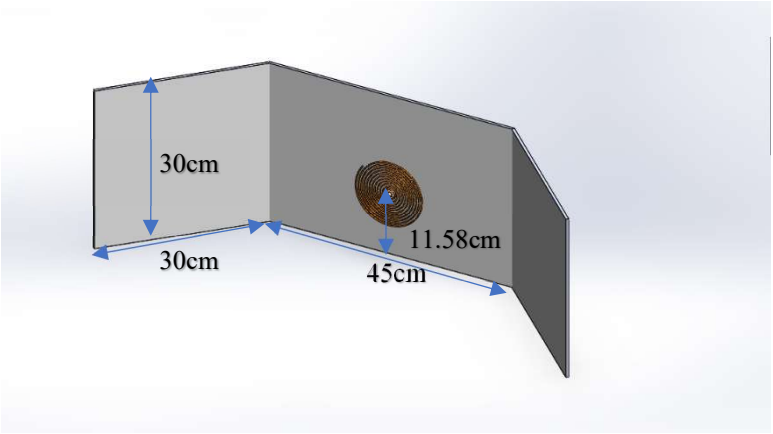
Gambar 3.10 Skematik rangkaian ZCD



Gambar 3.11 Desain board ekstensi Teensy 3.6

3.2.7 Stasiun Pengisian

Stasiun Pengisian di desain dengan SolidWork 2017. Bahan dasar yang akan digunakan adalah akrilik. Ukuran dari stasiun pengisian disesuaikan dengan platform robot. Gambar 3.12 adalah hasil desain stasiun pengisian

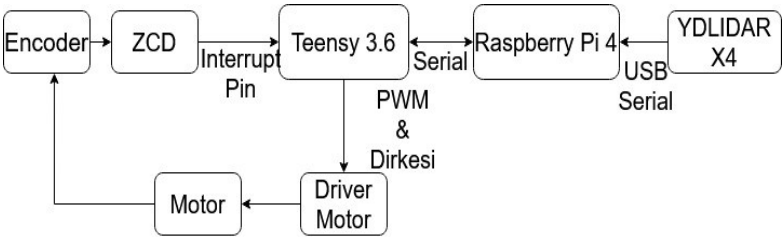


Gambar 3.12 Desain stasiun pengisian

3.3 Perancangan Perangkat Lunak

3.3.1 Alur Data Sensor

Data sensor odometri diakses dengan menggunakan mikrokontroler. Data tersebut kemudian dikirim ke Raspberry Pi 4 melalui protokol komunikasi serial. Mikrokontroler menggunakan rosserial supaya data yang dikirimkan tersedia dalam bentuk topik. Sedangkan LIDAR diakses langsung oleh Raspberry Pi 4 melalui kanal serial. Raspberry Pi 4 menyediakan 5 kanal serial melalui pin dan USB. Data yang diterima dari mikrokontroler melewati kanal serial S0 dengan *baudrate* 57600, kanal AMA0 digunakan untuk kabel data USB dari LIDAR. Hasil pengolahan data tersebut berupa kecepatan linier dan kecepatan sudut orientasi sebagai input kontrol *mobile robot*. Perancangan alur data dari sistem digambarkan pada gambar 3.13.



Gambar 3.13 Alur data sensor

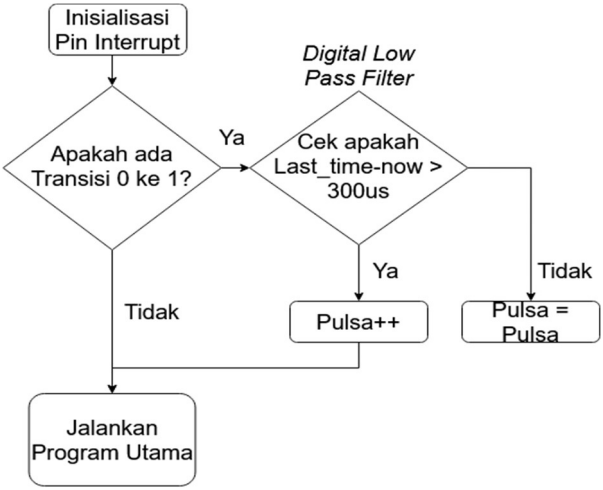
3.3.2 Pengambilan Data Sensor

Pengambilan data sensor dibagi menjadi beberapa bagian antara lain sebagai berikut:

3.3.2.1 Data Sensor Odometri

Sensor odometri yang dihasilkan oleh motor yang digunakan tersedia dalam bentuk sinyal AC. Oleh karena itu diperlukan ZCD untuk mengubah sinyal ac menjadi bentuk pulsa. Ambang batas perubahan pulsa pada rangkaian ZCD ditentukan oleh input *inverting*. Rangkaian dibuat memiliki ambang batas 10mV untuk menghindari *noise* pada sinyal encoder. Output di *pull up* ke 3.3V karena menyesuaikan tegangan toleransi pin digital mikrokontroler.

Kemudian ditambahkan *low pass filter* digital pada mikro kontroler supaya pembacaan encoder semakin baik. Pembacaan pulsa dilakukan dengan mengaktifkan *interrupt* saat terjadi transisi dari 0 ke 1. Hasil pembacaan pulsa dijadikan variabel kontrol kecepatan. Diagram alir dari filter seperti pada gambar 3.14.



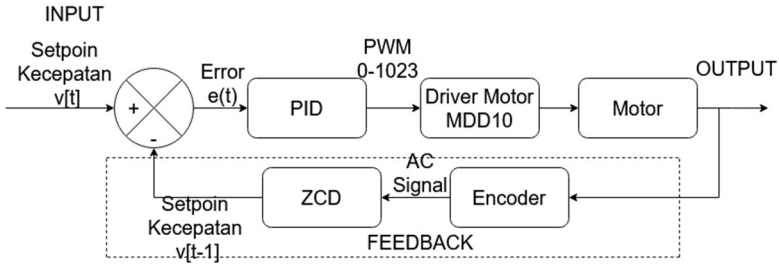
Gambar 3.14 Diagram alir penghitung pulsa

3.3.2.2 Data Sensor LIDAR

Modul YDLIDAR X4 terhubung dengan Raspberry Pi 4 melalui kabel USB. Data yang dikirimkan oleh LIDAR adalah jarak dan sudut pembacaan. YDLIDAR menyediakan dua node ROS untuk mengenkripsi data yang dikirim oleh LIDAR. ydlidar_node adalah node untuk menginisialisasi parameter LIDAR. Sedangkan ydlidar_client adalah node yang digunakan untuk menerjemahkan data mentah LIDAR.

3.3.3 Kontrol Motor DC

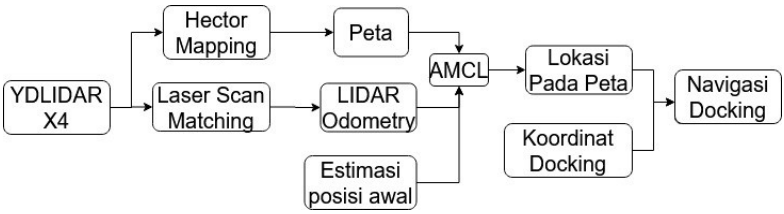
Kecepatan motor DC diatur menggunakan kontrol PID. Kontrol PID adalah kontrol yang terdiri dari *proportional, integral* dan *derrivative*. Input dari kontrol ini adalah set poin dan umpan balik dari pembacaan sensor. Dalam hal ini set poin yang diberikan adalah kecepatan motor dan umpan balik didapatkan dari pembacaan pulsa encoder melalui ZCD. Output dari kontrol adalah nilai PWM motor yang berkisar antara 0-1023. Kontrol kecepatan motor memerlukan waktu respon yang tinggi supaya hasil dari kontrol memenuhi harapan. Oleh karena itu waktu sampling dibuat 50ms. Diagram blok dari kontrol motor DC menggunakan PID digambarkan pada gambar 3.15.



Gambar 3.15 Kontrol motor DC

3.3.4 Perancangan Sistem ROS

Semua proses pengolahan data pada Raspberry Pi 4 dirancang menggunakan ROS. Sistem ROS secara umum digambarkan dalam diagram blok seperti pada gambar 3.16.



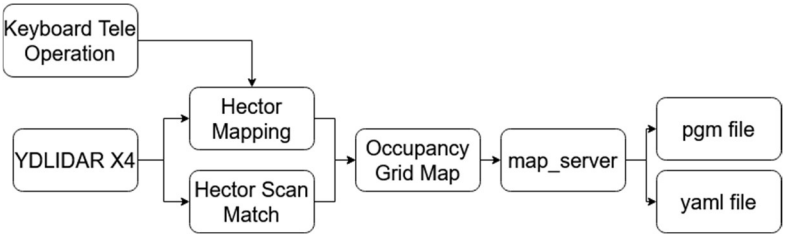
Gambar 3.16 Perancangan sistem ROS

Data dari LIDAR awalnya digunakan untuk membuat peta dari lingkungan yang akan diujikan. Kemudian peta tersebut disimpan. Setelah data peta telah didapatkan, data scan dari YD LIDAR X4 diolah pada laser scan matching untuk menghasilkan data odometri LIDAR. Estimasi posisi awal perlu diberikan pada AMCL supaya persebaran partikel dari AMCL tidak konvergen ke arah yang salah. Persebaran partikel dari AMCL menunjukkan kemungkinan posisi robot dalam peta. Partikel tersebut akan menjadi konvergen seiring berjalannya robot. Data perpindahan robot didapatkan dari odometri LIDAR. Setelah didapatkan estimasi posisi robot pada peta, dikirimkan koordinat docking sebagai

koordinat tujuan navigasi robot. Perancangan sistem ROS dipecah lagi menjadi tiga bahasan antara lain Pembuatan peta, Lokalisasi robot dan Navigasi Robot

3.3.5 Pembuatan Peta

Pembuatan peta dilakukan dengan menggunakan hector mapping dan kemudian hasil dari pemetaan disimpan. Peta disimpan menggunakan sebuah node bernama map_server. Hasil dari penyimpanan peta ada dua yaitu berkas dengan ekstensi pgm dan yaml. Berkas pgm berisi gambar peta yang telah dibangun, sementara yaml berisi informasi umum dari peta tersebut misalnya ukuran panjang dan lebar peta. Gambar 3.17 merupakan diagram alir dari program pemetaan.



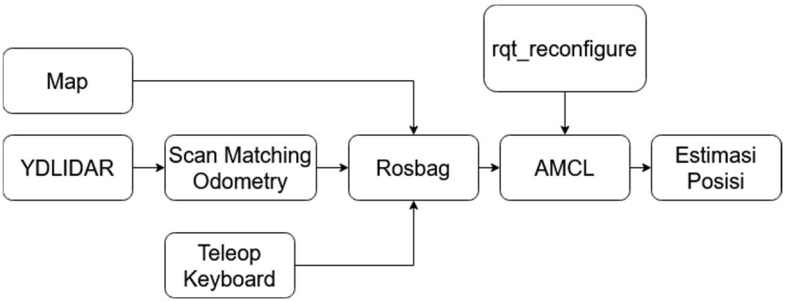
Gambar 3.17 Diagram alir pemetaan

Pada program pembuatan peta, robot digerakkan secara manual menggunakan keyboard komputer. Penentuan kecepatan sangat berpengaruh pada hasil peta yang akan dibangun. Apabila kecepatan terlalu cepat maka akan menghasilkan peta yang tumpang tindih. Oleh karena itu akan dilakukan pengujian kecepatan maksimal dari robot supaya tidak terjadi tumpang tindih.

3.3.6 Lokalisasi Robot

Untuk menentukan posisi robot dalam peta digunakan package AMCL. AMCL memerlukan inisialisasi estimasi posisi dan orientasi awal dari robot pada peta. Setelah inisialisasi diberikan partikel akan menyebar di sekitar estimasi posisi tersebut. Partikel yang menyebar akan mendekat dan menjadi posisi robot saat ini seiring perpindahan robot dari hasil scan matching. Partikel tersebut sewaktu-waktu dapat menyebar kembali apabila data perpindahan robot menjadi tidak meyakinkan akibat

perubahan posisi yang terlalu cepat. Hal ini dapat diatur dalam parameter-parameter amcl. Oleh karena itu perlu dilakukan pencarian nilai parameter yang paling baik supaya estimasi posisi robot tidak menyebar lagi.



Gambar 3.18 Diagram Alir Lokalisasi

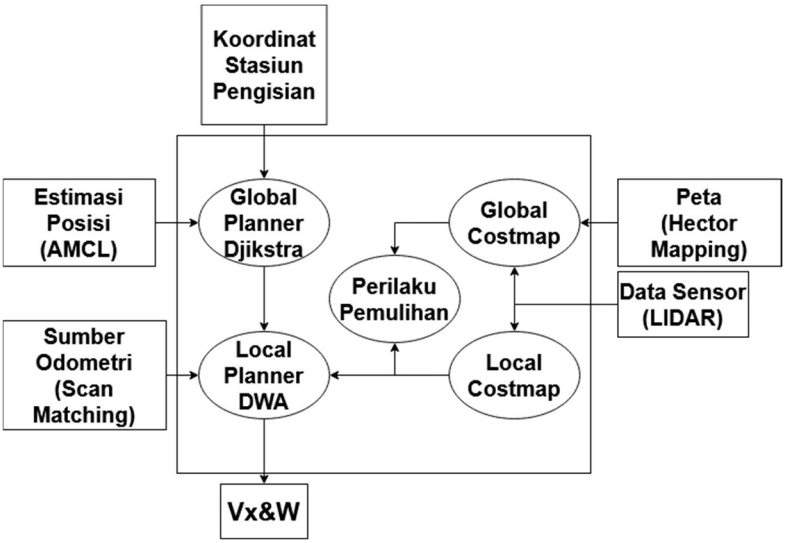
Seperti halnya pada gambar 3.18 data dari pergerakan dan *scan* robot akan disimpan dalam rosbag. Kemudian data tersebut akan dimuat ulang sebagai data simulasi. Untuk mengubah parameter dari AMCL digunakan *graphical user interface* yang disediakan oleh ROS yaitu rqt_reconfigure.

3.3.7 Navigasi Robot

Navigasi robot dijalankan oleh program move_base. Dalam program move_base terdapat beberapa node untuk keperluan navigasi sebuah robot. Terdapat node global planner, local planner, local costmap dan global costmap. Keseluruhan node bekerja bersama menghasilkan perintah kecepatan linier dan kecepatan sudut robot seperti pada gambar 3.19.

Global costmap dan Local costmap adalah node dari package costmap2D. Global costmap menampilkan nilai cost pada peta yang telah disimpan sebelumnya. Nilai cost dari global costmap bersifat statis untuk menandakan objek halangan yang tidak bergerak. Sedangkan local costmap menghasilkan nilai cost yang berubah-ubah sesuai dengan objek halangan yang terlihat oleh scan laser. Local costmap biasanya digunakan untuk mendeteksi halangan yang sebelumnya tidak ada dalam peta yang

tersedia dan halangan yang bergerak. Costmap memiliki parameter yang disebut *inflation radius*. *Inflation radius* adalah jarak maksimal jalur terhadap penghalang yang bisa dilalui robot.



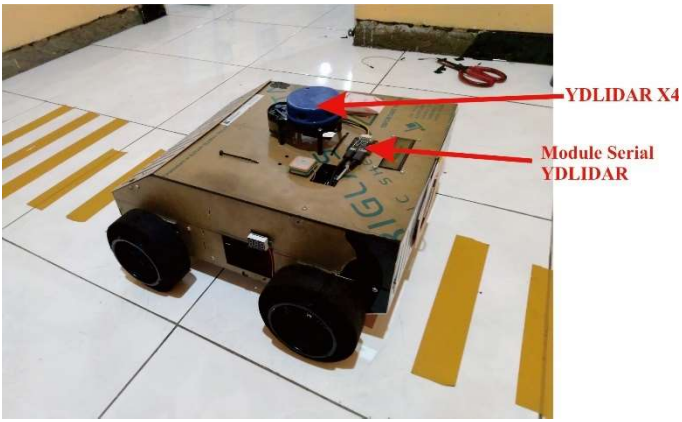
Gambar 3.19 Diagram alir navigasi move_base

Move_base memiliki dua perencanaan jalur, yaitu global dan local planner. Pada penelitian ini perencanaan jalur global yang digunakan adalah algoritma Djikstra. Sedangkan local planner menggunakan *Dynamic Window Approach*.

Saat ini metode pencarian parameter yang paling optimal untuk navigasi adalah secara eksperimental [27]. Parameter diubah-ubah sama seperti lokalisasi yaitu menggunakan rqt_reconfigure.

BAB 4
PENGUJIAN DAN ANALISIS

Pada bab ini dibahas tentang pengujian terhadap sistem yang telah dirancang sebagaimana telah termuat dalam bab sebelumnya. Kemudian dilakukan analisa dan evaluasi terhadap sistem berdasarkan data yang didapatkan. Gambar 4.1 adalah hasil realisasi desain *mobile robot*.



Gambar 4.1 Realisasi Mobile Robot

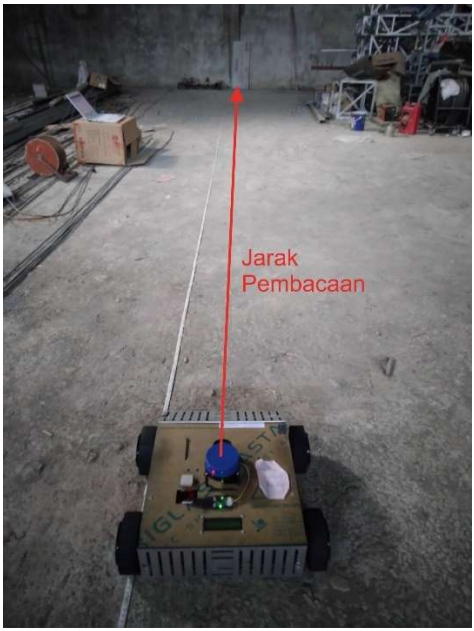
4.1 Pengujian Sensor LIDAR

Pengujian terhadap sensor LIDAR dilakukan untuk mengetahui performa dari LIDAR yang digunakan. Selain itu pengujian juga ditujukan untuk mengetahui kapabilitas dari LIDAR sehingga penggunaan dari LIDAR dapat disesuaikan. Terdapat beberapa macam pengujian terhadap sensor LIDAR.

4.1.1 Akurasi Sensor

Pengujian akurasi sensor dilakukan pada jarak pembacaan mulai dari robot menempel dengan objek yang akan diukur atau 0.25m sampai jarak melebihi jarak maksimal pembacaan 0.12m. Rata-rata cahaya intensitas ruangan 6 lumen. Pengukuran dilakukan dengan cara meletakkan objek pada dinding sehingga dapat diasumsikan objek berdiri

tegak. Kemudian robot digeser setiap jarak 0.5m. Cara pengambilan data pembacaan jarak ditunjukkan pada gambar 4.2.



Gambar 4.2 Pengukuran Akurasi LIDAR

Tabel 4.1 Pengukuran akurasi LIDAR

Jarak (m)	Pembacaan (m)	Error (%)
0.250	0.255	2.000
0.350	0.366	4.571
0.850	0.855	0.588
1.350	1.345	0.370
1.850	1.830	1.081
2.350	2.315	1.489
2.850	2.750	3.509
3.350	3.210	4.179
3.850	3.680	4.416
4.350	4.138	4.874
4.850	4.605	5.052
5.350	5.032	5.944
5.850	5.513	5.761
6.350	5.905	7.008
6.850	6.380	6.861
7.350	6.802	7.456
7.850	7.250	7.643
8.350	7.670	8.144
8.850	8.080	8.701
9.350	8.505	9.037
9.850	8.960	9.036
10.350	9.300	10.145
10.850	9.630	11.244
11.350	10.020	11.718
11.850	10.400	12.236
12.350	10.850	12.146
Rata-rata Error (%)		6.354

Berdasarkan data tabel 4.1 YDLIDAR memiliki error yang kecil pada jarak pengukuran menempel hingga 2.35 meter. Error bertambah dalam jumlah yang besar ketika pembacaan di atas 2.35 dan bertambah kurang lebih 1% setiap pertambahan jarak 1 meter. Hal tersebut

merupakan faktor kekurangan dari sensor yang digunakan. Meskipun pada *datasheet* YDLIDAR X4 memiliki jangkauan pembacaan sampai 12m akan tetapi pembacaan yang bagus didapatkan pada jarak kurang dari 2.35 meter.

4.1.2 Terhadap Berbagai bahan

Tujuan dari pengujian terhadap berbagai bahan adalah untuk mengetahui apakah perbedaan bahan objek mempengaruhi hasil pembacaan LIDAR. Selain itu hasil dari pengujian dapat digunakan untuk memutuskan bahan dasar dari stasiun pengisian. Pengujian dilakukan pada jarak 0.2 hingga 2.6m dengan bahan uji kayu, akrilik dan plat besi. Data dituliskan kedalam tabel 4.2.

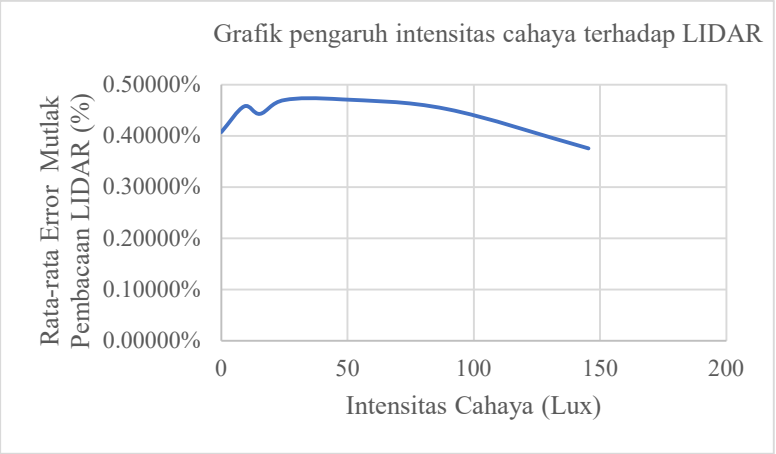
Berdasarkan tabel perhitungan error, error rata-rata dari pembacaan LIDAR paling besar adalah pada pembacaan kayu. Hal tersebut dikarenakan bahan dari kayu yang diujikan memiliki tekstur yang tidak rata. Berbeda dengan pembacaan akrilik dan plat besi yang memiliki rata-rata error di bawah 1% karena permukaannya yang rata. Jadi dapat disimpulkan bahwa perbedaan bahan tidak berpengaruh besar terhadap hasil pembacaan sensor melainkan tekstur dari bahan lebih besar pengaruhnya. Keputusan untuk bahan dasar dari stasiun pengisian digunakan akrilik karena meskipun plat besi memiliki rata-rata error yang lebih rendah karena berbahan dasar logam dapat mempengaruhi *wireless charger* yang bekerja dengan prinsip elektro magnetik.

Tabel 4.2 Pembacaan LIDAR terhadap berbagai bahan

Jarak (m)	Kayu		Akrilik		Besi	
	Pembacaan (m)	Error (%)	Pembacaan (m)	Error (%)	Pembacaan (m)	Error (%)
0.24	0.239	1.809	0.23925	1.809	0.2352	0.085
0.25	0.239	4.320	0.250	0.000	0.243	2.720
0.35	0.338	3.357	0.339	3.071	0.343	2.000
0.45	0.441	2.000	0.439	2.389	0.436	3.111
0.55	0.540	1.818	0.535	2.727	0.541	1.636
0.65	0.642	1.231	0.640	1.538	0.634	2.462
0.75	0.740	1.333	0.738	1.600	0.747	0.400
0.85	0.843	0.824	0.840	1.176	0.850	0.000
0.95	0.941	0.947	0.945	0.526	0.950	0.000
1.05	1.038	1.143	1.037	1.238	1.047	0.286
1.15	1.137	1.130	1.139	0.957	1.148	0.174
1.25	1.238	0.960	1.244	0.480	1.249	0.080
1.35	1.338	0.889	1.340	0.741	1.348	0.148
1.45	1.438	0.828	1.443	0.483	1.447	0.207
1.55	1.539	0.710	1.543	0.452	1.546	0.258
1.65	1.639	0.667	1.640	0.606	1.649	0.061
1.75	1.735	0.857	1.746	0.229	1.746	0.229
1.85	1.835	0.811	1.843	0.378	1.850	0.000
1.95	1.934	0.821	1.949	0.051	1.955	0.256
2.05	2.037	0.634	2.045	0.244	2.057	0.341
2.15	2.135	0.698	2.144	0.279	2.152	0.093
2.25	2.232	0.800	2.251	0.044	2.248	0.089
2.35	2.333	0.723	2.350	0.000	2.347	0.128
2.45	2.430	0.816	2.451	0.041	2.448	0.082
Error rata2 (%)	1.26		0.88		0.62	

4.1.3 Terhadap Berbagai Intensitas Cahaya

Pengujian Terhadap berbagai intensitas cahaya dilakukan di *indoor* dan *outdoor*. Pengujian *indoor* dilakukan dengan mengganti besarnya daya lampu yang digunakan. Terdapat 6 macam daya lampu dan kondisi mati. Intensitas cahaya dicatat menggunakan aplikasi pada telepon pintar "Lux Light Meter". Satuan intensitas cahaya yang digunakan adalah Lux yang mana merupakan intensitas cahaya pada suatu objek. Dalam hal ini intensitas cahaya pada objek yang akan diukur jaraknya. Grafik pada gambar 4.3 menunjukkan pengaruh dari intensitas cahaya terhadap hasil pembacaan jarak pada ruangan indoor.



Gambar 4.3 Grafik pengaruh intensitas cahaya terhadap pembacaan LIDAR (indoor)

Error rata-rata pada percobaan indoor memiliki tren cenderung turun setelah intensitas cahaya berada pada 40 Lux ke atas. Jadi dapat disimpulkan bahwa pada rentan intensitas cahaya 0-150 Lux 150 Lux paling baik.

Percobaan outdoor dilakukan di dua tempat, yaitu tempat yang terkena sinar matahari secara langsung dan tempat yang tidak terkena secara langsung. Percobaan dilakukan mulai pukul 06:00 hingga 08:30.

Tabel 4.3 Data pengujian intensitas cahaya outdoor

Waktu	luminasi (Lux)	Jarak (m)	Pembacaan (m)	Error (%)	Sinar Matahari secara langsung?
6:00	510	3.93	3.77	4.07	Tidak
6:06	620	3.93	3.77	4.07	Tidak
6:08	718	3.93	3.77	4.07	Tidak
6:13	872	3.93	3.77	4.07	Tidak
6:15	919	3.93	3.77	4.07	Tidak
6:15	951	3.93	3.77	4.07	Tidak
6:17	1007	3.93	3.77	4.07	Tidak
6:22	1188	3.93	3.77	4.07	Tidak
6:35	1342	3.93	3.77	4.07	Tidak
6:48	1423	3.93	3.77	4.07	Tidak
7:00	1512	3.93	3.77	4.07	Tidak
7:09	1614	3.93	3.77	4.07	Tidak
7:10	1710	3.93	3.77	4.07	Tidak
7:21	1812	3.93	3.77	4.07	Tidak
7:35	1816	3.93	3.77	4.07	Tidak
7:46	1895	3.93	3.77	4.07	Tidak
8:08	2612	1.20	0.00	100.00	Ya
8:30	3600	1.20	0.00	100.00	Ya

Berdasarkan data tabel 4.3, LIDAR memiliki akurasi yang stabil dengan error 4.07% pada intensitas cahaya kurang dari 2000 Lux. Pada intensitas cahaya lebih dari 2000 Lux LIDAR tidak dapat lagi mendeteksi adanya objek. Hal ini dikarenakan sinar laser yang dipancarkan LIDAR tersamarkan atau kalah dengan sinar matahari sehingga tidak ada pembacaan yang terdeteksi. Dengan demikian dapat disimpulkan YDLIDAR dapat digunakan indoor maupun outdoor dengan syarat intensitas cahaya kurang dari 2000 Lux. Gambar 4.4 adalah gambar pengambilan data pada tempat outdoor.

Gambar 4.4 Pengujian Luminasi outdoor cahaya matahari secara langsung

4.2 Pengujian Kontrol Motor DC

Motor JLN4B02 tidak memiliki *datasheet* yang menjelaskan hubungan antara pulsa yang dihasilkan encoder dengan kecepatan putaran roda. Oleh karena itu perlu dilakukan percobaan untuk mengetahui hal tersebut. Percobaan dilakukan dengan menjalankan robot secara lurus kemudian menghitung Pos_{tot} yang dihasilkan dari kinematika terbalik robot. Hubungan kecepatan dan perpindahan dituliskan pada tabel 4.4.

Tabel 4.4 Percobaan hubungan kecepatan dengan perpindahan

Pos total (pulsa)	Jarak Tempuh (m)	Pulsa : Jarak Tempuh
772.50	0.77	1003.25
1544.75	1.54	1003.08
2308.50	2.30	1003.70
3088.00	3.10	996.13
3858.75	3.85	1002.27
4632.25	4.63	1000.49

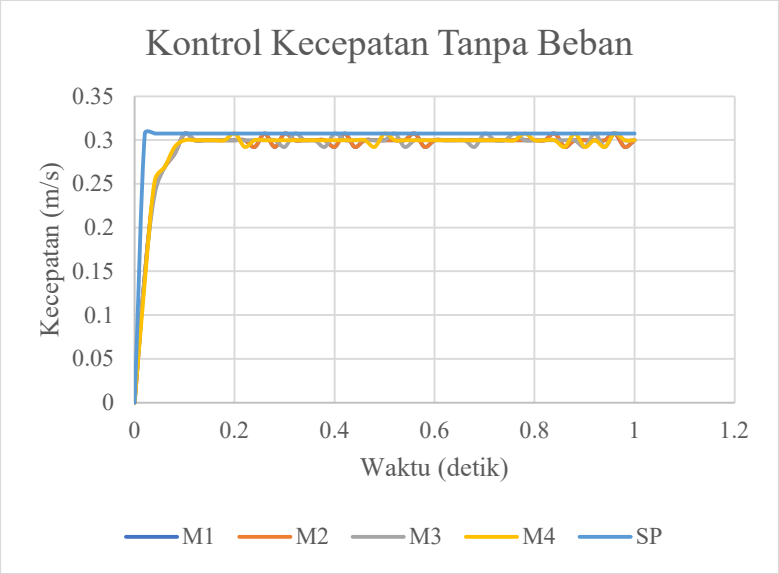
Berdasarkan tabel di atas dapat diperkirakan setiap 1 meter perpindahan menghasilkan 1000 pulsa motor. Setelah mengetahui hubungan pulsa dengan jarak tempuh dilakukan pencarian hubungan antara kecepatan input dengan kecepatan yang sesungguhnya agar input yang diberikan sesuai dengan kecepatan asli robot. Percobaan dengan cara memberikan input pulsa motor selama 5 detik kemudian mencatat jarak tempuh robot. Tabel 4.5 menunjukkan perbandingan antara kecepatan input dengan kecepatan robot sesungguhnya.

Tabel 4.5 Percobaan konversi kecepatan linier ke set poin pulsa

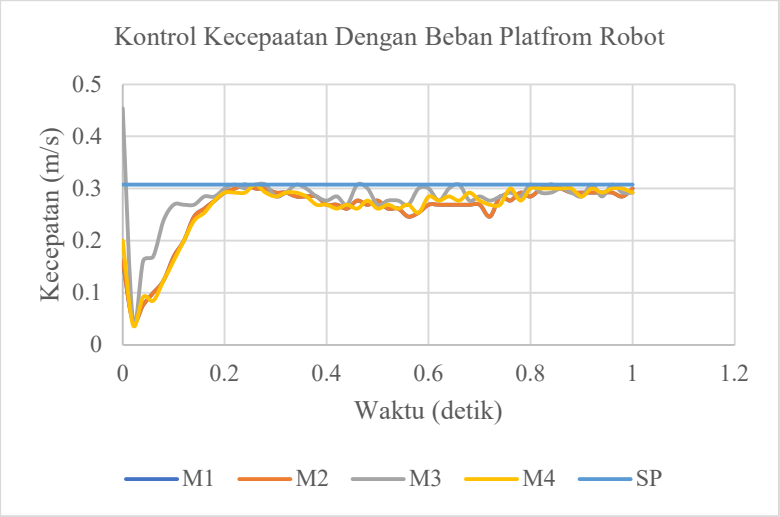
Input Set Poin Kecepatan (meter/ detik)	Jarak tempuh 5 detik		Kecepatan asli (meter/detik)	Input Kecepatan :Kecepatan Asli
	pulsa	meter		
0.200	772.500	0.773	0.155	1.294
0.400	1544.75	1.545	0.309	1.294
0.600	2308.50	2.309	0.462	1.299
0.800	3088.00	3.088	0.618	1.295
1.000	3858.75	3.859	0.772	1.296
1.200	4632.25	4.632	0.926	1.295

Perhitungan jarak tempuh dalam meter didapatkan berdasarkan perbandingan sebelumnya. Jadi supaya set poin dari motor dapat menerima kecepatan linier robot dalam satuan meter, input kecepatan harus dikalikan sebuah konstanta bernilai 1.2959.

Kontrol yang digunakan untuk mengatur kecepatan motor DC adalah kontrol PID. Metode yang digunakan untuk menentukan parameter PID adalah metode empiris. Langkah awal metode ini adalah menentukan nilai P hingga kecepatan mengalami osilasi. Kemudian dilanjutkan konstanta D untuk menghilangkan osilasi sinyal. Terakhir penentuan nilai I untuk menghilangkan error *steady state*-nya. Didapatkan nilai PID untuk setiap motor adalah P =4, I=50, D=0.15. Berikut adalah respon kontrol PID dalam kondisi tanpa beban dan dengan beban (platform robot). Pengujian dilakukan pada bidang lantai. Gambar 4.5 menunjukkan grafik hasil pembacaan kecepatan setiap motor dc dengan kondisi tanpa beban. Sedangkan gambar 4.6 dengan beban robot itu sendiri.



Gambar 4.5 Grafik kontrol kecepatan motor tanpa beban



Gambar 4.6 Grafik kontrol kecepatan motor dengan beban

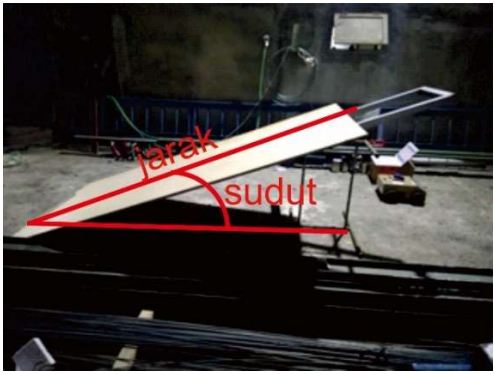
Berdasarkan grafik pengujian di atas, dapat disimpulkan bahwa kontrol PID sudah cukup stabil meskipun ditambah dengan beban platform robot.

Pengujian juga dilakukan pada bidang dengan beberapa macam kemiringan. Data kemiringan diambil menggunakan aplikasi pada telepon pintar "Clinometer". Variabel yang diamati adalah jarak tempuh robot yang dilakukan selama beberapa kali percobaan. Selang waktu pengambilan data selama 5 detik dan kecepatan di set pada nilai 0.377m/detik sehingga jarak tempuh yang seharusnya dicapai oleh robot adalah 188.5cm.

Tabel 4.6 Pengujian kontrol kecepatan terhadap kemiringan

	Sudut (derajat)				
Percobaan	0	5	10	15	20
1	155.0	140	149.0	124.0	110.0
2	156.0	140	146.0	123.0	109.0
3	155.0	140	132.0	119.0	110.0
Rata-rata jarak tempuh (cm)	155.3	140	142.3	122.0	109.7
Error (188.5-Jarak) (cm)	33.2	48.5	46.2	66.5	78.8
Error (188.5-Jarak) (%)	17.61	25.72	24.51	35.28	41.80

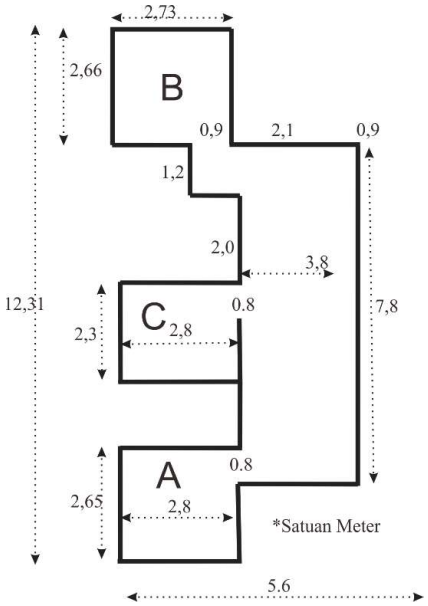
Berdasarkan tabel 4.6 semakin tinggi kemiringan robot jarak tempuh yang dicapai robot juga semakin menjauhi nilai jarak tempuh yang seharusnya. Hal ini dikarenakan kontrol PID memerlukan waktu respon untuk menuju nilai set poin. Selain itu faktor selip juga mempengaruhi hasil percobaan. Selip mulai terjadi pada sudut 15 dan 20 derajat.



Gambar 4.7 Pengujian kontrol motor terhadap kemiringan

4.3 Pengujian Pemetaan SLAM

Hasil pemetaan robot diujikan terhadap beberapa variabel antara lain kecepatan linier dan anguler yang berbeda dan tempat dimulainya pemetaan. Percobaan kecepatan linier dan sudut bertujuan untuk mencari kecepatan maksimal robot dalam membangun sebuah peta. Percobaan dilakukan dengan membandingkan panjang dan lebar hasil pemetaan dengan pengukuran yang sebenarnya. Pengukuran sebenarnya yang dihasilkan adalah panjang 5.6m dan lebar 12.31m. Gambar 4.8 adalah denah peta dari ruangan yang akan dibuat petanya



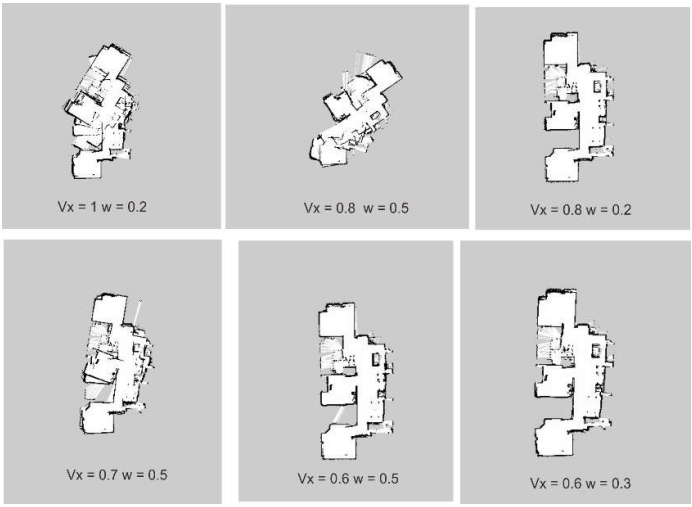
Gambar 4.8 Denah ruangan pengujian indoor

Tabel 4.7 adalah tabel data dari hasil pemetaan dengan kecepatan berbeda:

Tabel 4.7 Pengujian pengaruh kecepatan terhadap hasil pemetaan

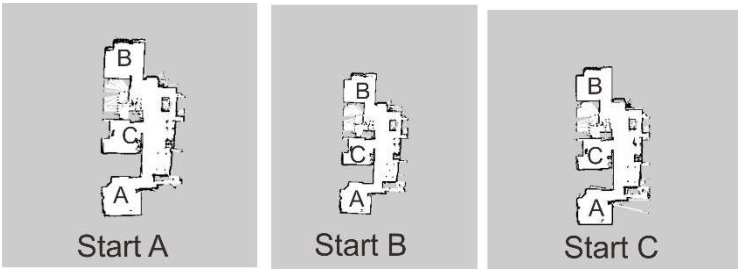
Kecepatan Linier (m/s)	Kecepatan Sudut (rad/s)	Panjang (m)	Lebar (m)	Error Panjang (m)	Error Lebar (m)	Error Panjang (%)	Error Lebar (%)
0.6	0.3	5.47	11.81	-0.13	-0.5	2.32	3.67
0.6	0.5	5.48	11.81	-0.12	-0.5	2.14	3.67
0.7	0.5	5.97	11.85	0.37	-0.54	6.61	3.34
0.8	0.2	5.51	11.8	-0.09	-0.51	1.61	3.75
1	0.2	7.06	12.84	1.46	0.53	26.07	4.73

Dari data di atas dapat disimpulkan bahwa jika semakin besar kecepatan linier dan sudut semakin besar error yang dihasilkan. Pada kecepatan linier di bawah 0.7 error yang dihasilkan masih stabil. Hal ini dikarenakan pembacaan odometri *scan matching* tidak dapat mengimbangi laju perpindahan robot. sehingga data *scan matching* tidak valid dan menyebabkan hasil pemetaan tumpang tindih. Hasil pemetaan dengan variasi kecepatan ditunjukkan pada gambar 4.9.



Gambar 4.9 Hasil pemetaan dengan berbagai kecepatan

Percobaan kedua yaitu pemetaan yang dimulai dari ruangan yang berbeda. Kecepatan ditentukan berdasarkan hasil pemetaan paling stabil sebelumnya yaitu kecepatan linier $V_x = 0.6$ dan kecepatan sudut w = 0.3. Percobaan dimulai dari 3 ruangan yang berbeda. Hasil pemetaan dari berbagai tempat mulai pemetaan ditunjukkan pada gambar 4.10.



Gambar 4.10 Hasil pemetaan dari tempat awal yang berbeda

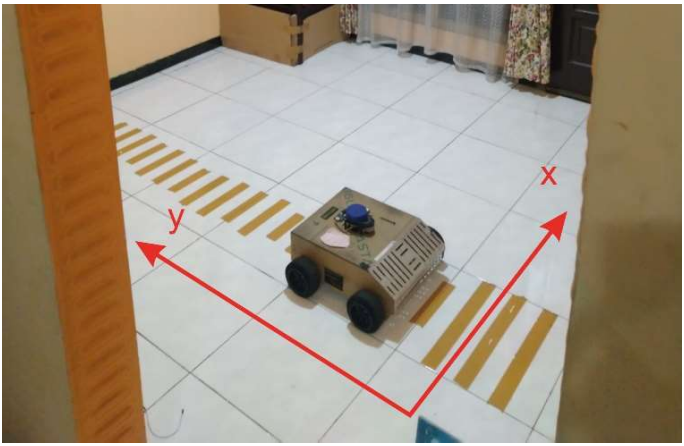
Tabel 4.8 Pengujian pengaruh perbedaan tempat mulai terhadap pemetaan

Start	Panjang (m)	Lebar (m)	Error Panjang (m)	Error Lebar (m)	Error Panjang (%)	Error Lebar (%)
A	5.47	11.81	-0.13	-0.5	2.321	3.670
B	5.53	11.8	-0.07	-0.51	1.250	3.752
C	5.52	11.82	-0.08	-0.49	1.429	3.589

Berdasarkan gambar dan tabel 4.8 tempat dimulainya pemetaan tidak mempengaruhi hasil dari pemetaan keseluruhan ruangan.

4.4 Pengujian Lokalisasi SLAM

Pengujian lokalisasi dilakukan dengan menggerakkan robot ke beberapa titik koordinat. Kemudian membandingkan hasil estimasi posisi dan sudut AMCL dengan koordinat sebenarnya. Pengujian dilakukan di salah satu ruangan yaitu ruangan B. Ruangan B memiliki dimensi 2.66mx2.73m. Penentuan koordinat pembacaan posisi lokalisasi ditunjukkan pada gambar 4.11.



Gambar 4.11 Lokasi pengujian lokalisasi AMCL

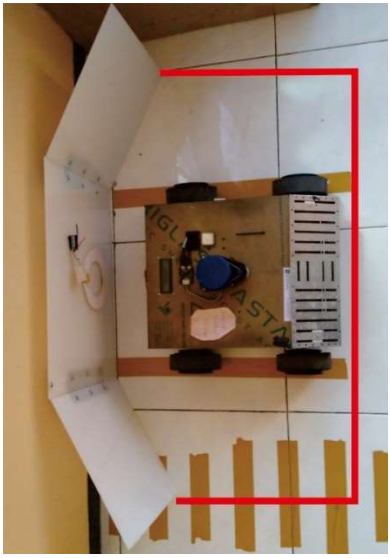
Pengujian lokalisasi dilakukan setelah estimasi posisi menjadi konvergen. Pengujian lokalisasi menggunakan AMCL menghasilkan pembacaan posisi yang sangat akurat yaitu dengan rata-rata error koordinat x 0.03m dan koordinat y 0.009m seperti pada tabel 4.9.

Tabel 4.9 Pengujian lokalisasi

	Koordinat AMCL (m)		Koordinat Asli (m)		Error (m)	
Titik	X	Y	X	Y	X	Y
1	0.534	0.015	0.566	0.01	0.032	-0.005
2	1.256	0.03	1.28	0.02	0.024	-0.01
3	2.165	0.044	2.22	0.04	0.055	-0.004
4	-0.002	0.726	0	0.76	0.002	0.034
5	0.865	0.885	0.89	0.91	0.025	0.025
6	1.659	0.888	1.7	0.9	0.041	0.012
				Rata-rata Error Mutlak	0.030	0.009

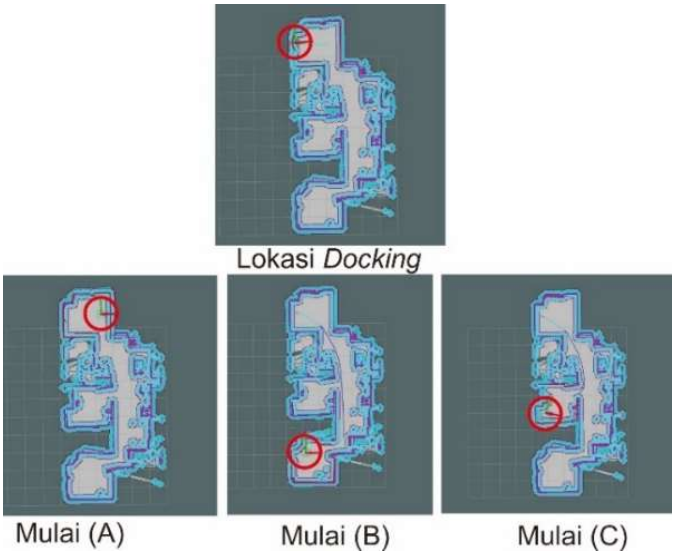
4.5 Pengujian Navigasi Docking

Pengujian navigasi *docking* dilakukan dengan mencoba mengirimkan perintah *docking* dari beberapa titik mulai yang berbeda. Tingkat keberhasilan *docking* dicatat sebagai tingkat kehandalan sistem. Robot dinyatakan berhasil menuju tempat *docking* apabila berhenti di dalam area kotak merah seperti pada gambar 4.12. Selain tingkat keberhasilan akurasi lokasi berhenti robot juga dicatat dengan perhitungan koordinat. Koordinat didapatkan dari pembacaan topik hasil lokalisasi AMCL.



Gambar 4.12 Pengujian navigasi docking

Navigasi robot menggunakan *package* ROS yaitu move_base. Di dalam *package* tersebut dijalankan program *global planner* dan *local planner*. *Global planner* yang digunakan dalam penelitian ini adalah algoritma Djikstra dan *Local planner Dynamic Window Approach.* Untuk mencari parameter navigasi yang tepat perlu dilakukan percobaan kepada sistem secara langsung.



Gambar 4.13 Lokasi docking dan mulai navigasi docking

Gambar 4.13 menunjukkan tempat awal dan tujuan setiap pengujian. Terdapat 9 kali percobaan, percobaan 1 sampai 3 dimulai dari titik A dan percobaan 4 hingga 6 dimulai dari titik B dan sisanya dari titik C. Dari 9 kali percobaan sistem 9 kali berhasil. Akan tetapi pada beberapa percobaan terdapat error orientasi pada posisi akhir yang cukup besar. Hal ini disebabkan karena kecepatan yang dihasilkan DWA terlalu cepat sehingga respon kontrol posisi dan orientasi mengalami keterlambatan. Apabila kecepatan diturunkan robot tidak bisa berjalan karena *noise* yang dihasilkan motor pada kecepatan rendah menjadi besar sehingga menyebabkan mikrokontroler salah membaca *noise* tersebut sebagai pulsa input. Kelemahan dari motor JLN4B02 ini mempersulit untuk menemukan parameter yang tepat. Jika berjalan pada kecepatan rendah motor tidak dapat berjalan karena pada kecepatan yang terlalu rendah yang terbaca hanya *noise* tanpa adanya gerakan robot. Akan tetapi jika kecepatan diset terlalu cepat respon kontrol gerak dari robot terlambat. Sehingga dengan seting terbaik menghasilkan data pada tabel 4.5. Solusi untuk menyelesaikan permasalahan ini dengan menggunakan motor

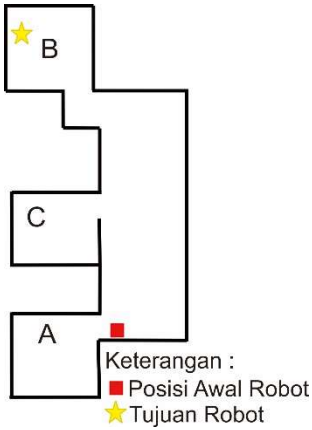
dengan encoder tipe incremental dan memperbesar kumparan stasiun pengecasan. Berdasarkan data tersebut prosentase keberhasilan robot mendekati tempat docking adalah 100% dengan rata-rata error koordinat x = 0.027m, y = 0.108m dan orientasi = 6.967 derajat. Data pengujian navgiasi ditunjukkan pada tabel 4.10.

Tabel 4.10 Hasil pengujian navigasi docking

Koordinat Mulai			Koordinat Berhenti			Koordinat Stasiun Pengisian			Error (m)			Waktu Tempuh (Detik)	Keberhasilan
X	Y	θ	X	Y	θ	X	Y	θ	X	Y	θ		
1.350	5.760	5.667	-0.630	5.770	2.893	-0.640	5.800	2.187	-0.010	0.030	-0.706	20	Ya
1.310	5.710	2.546	-0.620	5.770	-4.512	-0.640	5.800	2.187	-0.020	0.030	6.699	20	Ya
1.260	5.690	4.743	-0.630	5.780	-1.852	-0.640	5.800	2.187	-0.010	0.020	4.039	19	Ya
-0.070	-0.130	-5.898	-0.590	5.530	19.210	-0.640	5.800	2.187	-0.050	0.270	-17.023	136	Ya
-0.270	-0.200	-4.281	-0.610	5.760	-5.252	-0.640	5.800	2.187	-0.030	0.040	7.439	173	Ya
-0.250	-15.000	-7.052	-0.600	5.630	15.921	-0.640	5.800	2.187	-0.040	0.170	-13.734	189	Ya
0.170	-2.940	-3.356	-0.600	5.570	2.997	-0.640	5.800	2.187	-0.040	0.230	-0.809	140	Ya
0.180	-2.870	-3.703	-0.620	5.750	-3.859	-0.640	5.800	2.187	-0.020	0.050	6.046	119	Ya
0.350	-2.820	0.579	-0.620	5.930	-4.021	-0.640	5.800	2.187	-0.020	-0.130	6.209	173	Ya

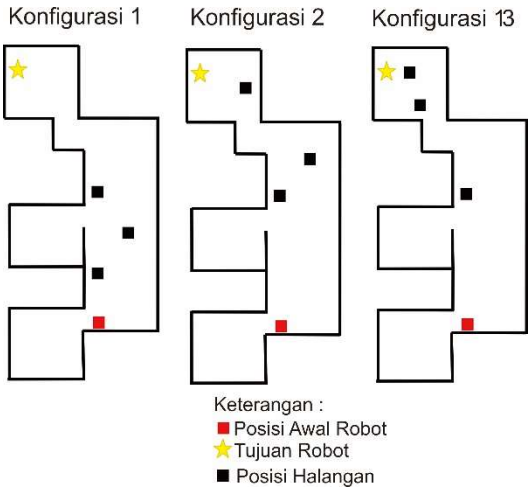
4.6 Pengujian Navigasi Docking dengan Halangan Baru

Tujuan dilakukan pengujian ini adalah menguji kehandalan sistem navigasi docking yang telah dirancang. Pengujian dilakukan dengan menambahkan halangan yang sebelumnya belum diketahui atau belum terpetakan. Halangan dalam pengujian ini bersifat statis. Berikut adalah posisi awal dan tujuan robot. Lokasi awal dan tujuan pengujian digambarkan pada gambar 4.14.



Gambar 4.14 Lokasi awal dan tujuan pengujian navigasi halangan baru

Pengujian dilakukan sebanyak 3 kali dengan 3 variasi peletakan 3 buah halangan. Halangan yang digunakan berupa kotak kardus dengan ukuran yang sama dan memiliki tinggi yang cukup untuk dideteksi LIDAR. Peletakan halangan dibuat menghalangi jalur terpendek robot akan tetapi masih terdapat ruang yang cukup sehingga robot masih bisa melewatinya. Pada pengujian ini dicatat waktu tempuh, jumlah benturan robot terhadap halangan dan keberhasilan robot berjalan menuju stasiun pengisian. Gambar 4.15 adalah gambar peletakan posisi halangan pada peta.



Gambar 4.15 Peletakan halangan baru

Pada seluruh percobaan konfigurasi 1 dan 2 robot berhasil sampai pada lokasi tujuan. Akan tetapi pada percobaan konfigurasi 3 robot tidak dapat menuju stasiun pengisian karena robot tidak dapat membuat perencanaan jalur yang memungkinkan karena peletakan halangan yang terlalu dekat. Apabila hal tersebut terjadi robot akan memasuki program perilaku pemulihan yaitu robot berputar di tempat kemudian memuat ulang program global dan local costmap. Perilaku pemulihan ini dilakukan berulang kali hingga robot menemukan jalur yang memungkinkan. Pada percobaan konfigurasi ketiga, robot tetap tidak dapat menemukan jalur yang dapat dilewati robot sehingga robot hanya berputar-putar ditempat. Dari percobaan ini dapat disimpulkan bahwa sistem navigasi hanya bisa menghindari halangan baru apabila terdapat jalur yang lebar untuk dilalui robot. Apabila jalur terlalu sempit robot selalu gagal membuat perencanaan jalur baru. Tabel 4.11 adalah tabel data pengujian setiap percobaan.

Tabel 4.11 Percobaan Navigasi Halangan Baru

Percobaan	Tabrakan	Waktu Tempuh(detik)	Berhasil
1	1	173	Ya
2	3	194	Ya
3	4	240	Ya
4	1	166	Ya
5	1	193	Ya
6	0	166	Ya
7	4	-	Tidak
8	1	-	Tidak
8	3	-	Tidak

Terjadinya tabrakan pada percobaan ini karena penentuan jarak minimal robot terhadap dinding atau radius inflasi ditentukan berdasarkan jarak titik tengah robot terhadap bagian robot terluar sehingga robot terlalu dekat dengan halangan ketika berputar. Apabila radius inflasi diperbesar robot tidak dapat melewati celah yang sempit. Meskipun sistem navigasi yang telah dibuat bisa melewati halangan baru jalur yang dilewati robot masih kurang efisien sehingga waktu tempuh robot menjadi sangat lama. Hal ini dikarenakan penentuan rentan waktu simulasi atau sim_time dalam program DWA terlalu singkat. Sim_time diset pada 0.1 detik karena apabila rentan waktu tersebut diperbesar kemungkinan robot akan menabrak akan semakin besar. Hal ini disebabkan karena kontrol pergerakan robot tidak selalu sesuai dengan simulasi gerakan yang dihasilkan oleh DWA.

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

BAB 5
PENUTUP

5.1 Kesimpulan

Pada penelitian ini telah dibuat sistem *autonomous docking* yang diterapkan pada *mobile robot* jenis SSMR. Pengujian pembacaan LIDAR terhadap intensitas cahaya menghasilkan kesimpulan bahwa YDLIDAR hanya dapat membaca hingga 2000 Lux. Kontrol PID motor bekerja dengan baik pada saat tanpa beban, ketika beban platform robot diterapkan kontrol mengalami sedikit osilasi. Pengujian kontrol kecepatan motor juga dilakukan pada beberapa kemiringan dan setiap kemiringan bertambah error juga semakin besar. Motor JLN4B02 kurang cocok digunakan beban berat. Pemetaan terbaik dihasilkan pada kecepatan linier 0.6m/detik dan kecepatan anguler 0.3rad/detik. Kecepatan tersebut berlaku dengan tidak melakukan pergerakan berputar ditempat karena algoritma *scan matching* tidak bisa mengimbangi gerakan tersebut. Pengujian lokalisasi AMCL dilakukan pada ruangan B menghasilkan estimasi posisi yang cukup akurat dengan rata-rata error mutlak untuk sumbu x 0.03m dan sumbu y 0.009m. Tingkat keberhasilan navigasi docking yang telah dibuat adalah 100% dengan rata-rata error koordinat x = 0.027m, y = 0.108m dan orientasi = 6.967 derajat. Sistem navigasi dapat mengantisipasi halangan baru bila terdapat celah jalur yang cukup lebar.

5.2 Saran

Metode yang telah dibuat masih memerlukan data pemetaan dari lingkungan, pemberian koordinat estimasi dari robot dan estimasi koordinat pengisian. Pengembangan penelitian ini dapat dilakukan dengan pemetaan secara otomatis menggunakan algoritma eksplorasi dan deteksi bentuk dari stasiun pengisian. Penambahan sensor-sensor lain seperti IMU dan GPS dapat diimplementasikan menggunakan *sensor fusion* untuk meningkatkan akurasi dari pemetaan dan lokalisasi robot.

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

DAFTAR PUSTAKA

[1] H. R. Rasam, "Review on Land-Based Wheeled Robots," *MATEC Web of Conferences*, vol. 53, p. 01058, 2016, doi: 10.1051/matecconf/20165301058.

[2] G. Klančar, A. Zdešar, S. Blažič, and I. Škrjanc, Eds., *Wheeled mobile robotics: from fundamentals towards autonomous systems*. Oxford: Butterworth-Heinemann, an imprint of Elsevier, 2017.

[3] T. Wang, Y. Wu, J. Liang, C. Han, J. Chen, and Q. Zhao, "Analysis and Experimental Kinematics of a Skid-Steering Wheeled Robot Based on a Laser Scanner Sensor," *Sensors*, vol. 15, no. 5, pp. 9681–9702, Apr. 2015, doi: 10.3390/s150509681.

[4] Adept Technology, "Pioneer P3-AT." Columbia Drive, 2011.

[5] Elektrik Mühendisleri Odası, Ed., *2011 7th International Conference on Electrical and Electronics Engineering (ELECO 2011): Bursa, Turkey, 1 - 4 December 2011*. Piscataway, NJ: IEEE, 2011.

[6] "Lidar," in *GaN Transistors for Efficient Power Conversion*, Chichester, UK: John Wiley & Sons, Ltd, 2019, pp. 281–299.

[7] Texas Instrument, "An Introduction to Automotive LIDAR." Texas Instrument, 2020.

[8] 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, doi: 10.5220/0005965803080317.

[9] T. Mashrik, H. Zunair, and M. F. Karin, "Design and Implementation of Security Patrol Robot Using Android Application," in *2017 Asia Modelling Symposium (AMS)*, Kota Kinabalu, Dec. 2017, pp. 77–82, doi: 10.1109/AMS.2017.20.

[10] EAI Team, "Datasheet YDLIDAR X4." EAI Team, 2017 2015.

[11] Freescale Semiconductor, Inc, "Kinetis K66 Sub-Family." Freescale Semiconductor .Inc, 2016.

[12] "Teensy and Teensy++ Pinouts, for C language and Arduino Software." https://www.pjrc.com/teensy/pinout.html (accessed Jul. 07, 2020).

[13] Raspberry Pi Trading Ltd., "Raspberry Pi 4 Product Databrief." Raspberry Pi Trading Ltd., 2019.

[14] T. Özer, S. Kivrak, and Y. Oğuz, "H Bridge DC Motor Driver Design and Implementation with Using dsPIC30f4011,"

International Journal of Innovative Research in Science, Engineering and Technology, vol. 6, pp. 75–83, May 2017.
[15] Cytron Technologies, "MDD 10A Dual Channel 10A DC Motor Driver." Cytorn Technologies Sdn. Bhd., 2013.
[16] Texas Instrument, "LM111, LM211, LM311 Differential Comparator." Texas Instrument, 2017.
[17] YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, and TaeHoon Lim, *ROS Robot Programming From basic to practical programming and robot application*. Republic of Korea: ROBOTIS Co.,Ltd., 2017.
[18] K. Tiwari and N. Young Chong, "Simultaneous Localization and Mapping (SLAM)," in *Multi-robot Exploration for Environmental Monitoring*, Elsevier, 2020, pp. 31–38.
[19] S. Kohlbrecher, J. Meyer, T. Graber, K. Petersen, U. Klingauf, and O. von Stryk, "Hector Open Source Modules for Autonomous Mapping and Navigation with Rescue Robots," in *RoboCup 2013: Robot World Cup XVII*, vol. 8371, S. Behnke, M. Veloso, A. Visser, and R. Xiong, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 624–631.
[20] 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*, Kyoto, Japan, Nov. 2011, pp. 155–160, doi: 10.1109/SSRR.2011.6106777.
[21] Paril Jain, "Simultaneous Localization and Mapping," 2016.
[22] B. Zhang, J. Liu, and H. Chen, "AMCL based map fusion for multi-robot SLAM with heterogenous sensors," in *2013 IEEE International Conference on Information and Automation (ICIA)*, Yinchuan, China, Aug. 2013, pp. 822–827, doi: 10.1109/ICInfA.2013.6720407.
[23] "Sensor Guided Docking of Autonomous Mobile Robot for Battery Recharging," *IJRTE*, vol. 8, no. 4, pp. 3812–3815, Nov. 2019, doi: 10.35940/ijrte.D8176.118419.
[24] F. Guangrui and W. Geng, "Vision-based autonomous docking and re-charging system for mobile robot in warehouse environment," in *2017 2nd International Conference on Robotics and Automation Engineering (ICRAE)*, Shanghai, Dec. 2017, pp. 79–83, doi: 10.1109/ICRAE.2017.8291357.

[25] R. Ren, H. Fu, and M. Wu, “Large-Scale Outdoor SLAM Based on 2D Lidar,” *Electronics*, vol. 8, no. 6, p. 613, May 2019, doi: 10.3390/electronics8060613.

[26] G. Csaba, L. Somlyai, and Z. Vamossy, “Mobil robot navigation using 2D LIDAR,” in *2018 IEEE 16th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, Kosice, Feb. 2018, pp. 000143–000148, doi: 10.1109/SAMI.2018.8324002.

[27] K. Zheng, “ROS Navigation Tuning Guide,” *arXiv:1706.09068 [cs]*, Apr. 2019, Accessed: Jun. 04, 2020. [Online]. Available: http://arxiv.org/abs/1706.09068.

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

LAMPIRAN A

A. Program Arduino

```
#include <Wire.h>
#include <EEPROM.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>
#include <TinyGPS.h>
#define OUTPUT_READABLE_YAWPITCHROLL
#define USE_TEENSY_HW_SERIAL
#define gpsPort Serial1
#define rad 57,2958

Adafruit_BNO055 bno = Adafruit_BNO055(55, 0x28);
TinyGPS gps;

IntervalTimer TimerMPU;
IntervalTimer TimerPID;
IntervalTimer TimerPrint;
IntervalTimer TimerRosserial;

void send_data_ros();

void pin_setup();
void bn055_setup();
void bn055_getdata();
void displayCalStatus();
void displaySensorOffsets(const adafruit_bno055_offsets_t
&calibData);
adafruit_bno055_offsets_t calibrationData;
float gps_latitude, gps_longitude;
void motor_setup();
void set_pid();
void pid_kecepatan();
void cetak_encoder1();
void cetak_encoder2();
void cetak_encoder3();
void cetak_encoder4();
```

```
  void cetak_encoder_pasti(); //cek disc encoder
  uint32_t timer;
  uint32_t first_time[5];
  uint32_t last_time[5];
  uint32_t timer_rpm = 0;
   // inisiasi motor
  const int8_t motor1 = 5;  //kanan depan
  const int8_t motor2 = 2;  //kanan belakang
  const int8_t motor3 = 3;  //kiri depan
  const int8_t motor4 = 4;  //kiri belakang

  const int8_t arah1 = 20; //kanan depan
  const int8_t arah2 = 23; //kanan depan
  const int8_t arah3 = 22; //kanan depan
  const int8_t arah4 = 21; //kanan depan
  int logic_arah[4];
  float pulsa[5];                                    //Encoder
  double kp[5], ki[5], kd[5];                        //Parameter PID
  double  setpoint[5], kecepatan[5], pwm_kecepatan[5];
//Variabel untuk PID(hanya Positif)
  double  input_kecepatan[5], display_kecepatan[5], input_ori;
//Input dan Output positif dan negatif dalam bentuk RPM
  double error[5], lasterror[5], sumerror[5], Derr[5];
//Variabel PID
  double rps[5], vlinier[2], wangular, ori;          ///1 kanan 0
kiri
  double kecepatan_x_robot, kecepatan_y_robot,
kecepatan_sudut_robot_inv; //Variabel Invers Kinematic
  double pos_x, pos_y,pos_tot, pos_ori, pos_x_set, pos_y_set,
pos_ori_set;
  double data_vx, data_vy, data_wz, c=0.4, konv_v_rpm =
1.295851914, save_vx, save_wz;          //data remote
  double kecepatan_motor_kanan, kecepatan_motor_kiri;
  long pulsa_simpan[5];
  float kecepatan_robot;

  float saveyaw, savepitch, saveroll;
  float ypr[3];          // [yaw, pitch, roll]   yaw/pitch/roll container and
gravity vector
```

```
  float yaw_rad;

  unsigned long timer_motor;
  unsigned long interval_mpu = 100000;            //Interval Interrupt
Timer encoder
  unsigned long interval_pid = 50000;             //Interval Intterupt
Timer PID
  unsigned long interval_print = 100000;              //Interval
Intterupt Timer PID
  unsigned long interval_rosserial = 100000;              //Interval
Intterupt Timer PID

  int hitung;
  int batas_hitung = 50;
  double save_kecepatan[5][505];
  int kecepatan_semua;
  String buff;
  const int led = 13;
  int jalan = 0;
  bool kondisi = false;
  bool enable_compass = false;
//  bool state_publish = false;
  int state_reset = 0;
  uint8_t mode_jalan = 0;

  ////////////////////////////////  ROS /////////////////
  #include <ros.h>
  #include <std_msgs/String.h>
  #include <std_msgs/Float32.h>
  #include <std_msgs/Int8.h>
  #include <geometry_msgs/Twist.h>
  #include <sensor_msgs/Imu.h>
  #include <tf/tf.h>
  #include <tf/transform_broadcaster.h>

  ros::NodeHandle  nh;
  geometry_msgs::TransformStamped t;
  tf::TransformBroadcaster broadcaster;
  sensor_msgs::Imu imu_msg;
```

```
  char base_link[] = "/base_footprint2";
  char odom[] = "/odom_encoder";

  void messageCb( const std_msgs::Int8& terima_data){
    mode_jalan = terima_data.data;
  }
  void messageCb_reset( const std_msgs::Int8& terima_data){
       state_reset = terima_data.data;
       pos_x = 0;
       pos_y = 0;
       pos_tot = 0;
       yaw_rad = 0;
  }
  void message_remote(const geometry_msgs::Twist& remote)
  {
    data_vx   = remote.linear.x*100;
    data_wz   = remote.angular.z;
  }
  ros::Subscriber<geometry_msgs::Twist>
set_kecepatan_remote("cmd_vel", message_remote);
  ros::Subscriber<std_msgs::Int8> reset_arduino("reset",
messageCb_reset);

  std_msgs::Float32 pos_x_alamat;
  std_msgs::Float32 pos_y_alamat;
  std_msgs::Float32 ori_w_alamat;
  std_msgs::Float32 kirim_compass;

  ros::Publisher pos_x_arduino("pos_x_arduino", &pos_x_alamat);
  ros::Publisher pos_y_arduino("pos_y_arduino", &pos_y_alamat);
  ros::Publisher ori_w_arduino("ori_w_arduino", &ori_w_alamat);
  ros::Publisher compass("compass", &kirim_compass);
  ros::Publisher imubno("imu", &imu_msg);


  ////////////////////////////////////////////////// ROS ////////////////////////////

  void setup() {
```

```
  //  //ROS SETUP//
  nh.initNode();
  nh.advertise(pos_x_arduino);
  nh.advertise(pos_y_arduino);
  nh.advertise(ori_w_arduino);
  nh.advertise(compass);
  nh.advertise(imubno);

  nh.subscribe(set_kecepatan_remote);
  nh.subscribe(reset_arduino);
  TimerRosserial.begin(send_data_ros, interval_rosserial);
  //  ////////////

  Serial.begin(57600);
  Wire.setSDA(18);
  Wire.setSCL(19);
  Wire.begin();

  pin_setup();
  motor_setup();
  bn055_setup();
  Serial.println("Calibration OK");
  delay(2000);
  set_pid();

  TimerMPU.begin(bn055_getdata, interval_mpu);
  TimerPrint.begin(debug, interval_print);
  TimerPID.begin(pid_kecepatan, interval_pid);

  timer = micros();
  timer_motor = millis();
  timer_rpm = millis();
  kecepatan_robot = 10;
  enable_compass = false;
}

void loop() {
  if (Serial.available() > 0) {
```

```
    kondisi = true;
   }
  }
  void debug()
  {
   //Program Debug Kontrol PID
   if (kondisi == true)
   {
     for(int cetakin = 0; cetakin<batas_hitung+1;cetakin++)
     {

     }
   }
   if(hitung>batas_hitung)
   {
    hitung = batas_hitung+1;
    kecepatan_robot = 0;
    enable_compass = false;
   }
   else if(hitung == batas_hitung+1)
   {
    kecepatan_robot = 0;
    enable_compass = false;
   }
   else
   {

   }
  ////////////////////////////////////////////////////////
   for(int f=0; f<5; f++)
   {
    rps[f] = 0;
    pulsa_simpan[f] = 0;
   }
  }

////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
void blink_led()
```

```
 {
  digitalWrite(led, HIGH);   // set the LED on
  delay(20);                 // wait for a second
  digitalWrite(led, LOW);    // set the LED off
  delay(20);
 }
 void pin_setup()
 {
   pinMode(led, OUTPUT);
   analogWriteResolution(10);
   analogWriteFrequency(motor1, 500);   //gak berani frekuensi
tinggi kalau kabel kecil
   analogWriteFrequency(motor2, 500);
   analogWriteFrequency(motor3, 500);
   analogWriteFrequency(motor4, 500);

   pinMode(11, INPUT_PULLUP);
   pinMode(10, INPUT_PULLUP);
   pinMode(9, INPUT_PULLUP);
   pinMode(8, INPUT_PULLUP);
   pinMode(7, INPUT_PULLUP);

   attachInterrupt(digitalPinToInterrupt(11), cetak_encoder1,
RISING);
   attachInterrupt(digitalPinToInterrupt(9), cetak_encoder2,
RISING);
   attachInterrupt(digitalPinToInterrupt(10), cetak_encoder3,
RISING);
   attachInterrupt(digitalPinToInterrupt(8), cetak_encoder4,
RISING);
   attachInterrupt(digitalPinToInterrupt(7), cetak_encoder_pasti,
RISING);
 }
 ////////////////////////////////////////////////////////
 ////////////////////////////////////////////////////////
 void cetak_encoder1()
 {
  first_time[0] = micros();
  if(first_time[0]-last_time[0]>300)
```

```
  {
   pulsa[0]++;
  }
  last_time[0] = first_time[0];
}
void cetak_encoder2()
{
  first_time[1] = micros();
  if(first_time[1]-last_time[1]>300)
  {
   pulsa[1]++;
  }
  last_time[1] = first_time[1];
}
void cetak_encoder3()
{
  first_time[2] = micros();
  if(first_time[2]-last_time[2]>300)
  {
   pulsa[2]++;
  }
  last_time[2] = first_time[2];
}
void cetak_encoder4()
{
  first_time[3] = micros();
  if(first_time[3]-last_time[3]>300)
  {
   pulsa[3]++;
  }
  last_time[3] = first_time[3];
}
void cetak_encoder_pasti()        //untuk encoder disc
{
  first_time[4] = micros();
  if(first_time[4]-last_time[4]>300)
  {
   pulsa[4]++;
  }
```

```
  last_time[4] = first_time[4];
}
void reset_pulsa()
{
  for(int x = 0; x<5; x++)
    {
      pulsa_simpan[x] += pulsa[x];
      pulsa[x] = 0;
    }
}
void print_pulsa()
{
  for(int x = 0; x<4; x++)
    {
      Serial.print("Pulsa ke - ");
      Serial.print(x);
      Serial.print(" = ");
      Serial.println(pulsa[x]);
    }
}
double hitung_kecepatan()
{
      for(int x = 0; x<5; x++)
        {
           if(logic_arah[x] == 1)
           {
              display_kecepatan[x] = -1*pulsa[x];
           }
           else
           {
              display_kecepatan[x] = 1*pulsa[x];
           }
            kecepatan[x] = pulsa[x];
//          print_pulsa();
        }
       reset_pulsa();
}
//////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////
```

```
double save_kecepatan_linier, save_kecepatan_anguler;
void forward_kinematic()
{
    double kecepatan_linier, kecepatan_anguler;
//    data_vx = 20;
//    data_wz = 1;
    double R = 0.115;
    kecepatan_linier = data_vx*konv_v_rpm; //Input dikali konstanta
pengali
    kecepatan_anguler = data_wz;
    input_kecepatan[0] =
kecepatan_linier+(((kecepatan_anguler*c)/(2*R))*100);
    input_kecepatan[1] =
kecepatan_linier+(((kecepatan_anguler*c)/(2*R))*100);
    input_kecepatan[2] = kecepatan_linier-
(((kecepatan_anguler*c)/(2*R))*100);
    input_kecepatan[3] = kecepatan_linier-
(((kecepatan_anguler*c)/(2*R))*100);
//    Serial.print(kecepatan_anguler);
//    Serial.print("\t");
    if(kecepatan_linier == 0 && kecepatan_anguler == 0 ||
kecepatan_linier != save_kecepatan_linier || kecepatan_anguler !=
save_kecepatan_anguler)
    {
     for(int x = 0; x<4; x++)
     {
     pwm_kecepatan[x] = 0;
     sumerror[x] = 0;
     lasterror[x] = 0;
     }
    }
    save_kecepatan_linier = kecepatan_linier;
    save_kecepatan_anguler = kecepatan_anguler;
}
void inverse_kinematic()
{
   kecepatan_x_robot        = (vlinier[0]+vlinier[1])/2;
   kecepatan_sudut_robot_inv = (-(vlinier[0])+(vlinier[1]))/c;
```

```
    kecepatan_x_robot         = kecepatan_x_robot/20;
//Merubah Kecepatan perdetik ke per50ms
    kecepatan_sudut_robot_inv = kecepatan_sudut_robot_inv/20;
}
void hitung_posisi_robot()
{
    inverse_kinematic();
//    yaw_rad = map(ypr[0], 0, 360, -180, 180)/rad;
    yaw_rad +=  kecepatan_sudut_robot_inv;
    if(yaw_rad > 3.14) yaw_rad=-3.14;
    else if(yaw_rad < -3.14) yaw_rad = 3.14;
//    yaw_rad = ypr[0]/rad;
    pos_x +=  cos(yaw_rad)*kecepatan_x_robot;
    pos_y +=  sin(yaw_rad)*kecepatan_x_robot;
    pos_tot = sqrt(pos_x*pos_x+pos_y*pos_y);

//      Serial.print(rps[0]);
//      Serial.print("\t");
//      Serial.print(rps[1]);
//      Serial.print("\t");
      Serial.print(input_kecepatan[0]);
      Serial.print("\t");
      Serial.print(input_kecepatan[2]);
      Serial.print("\t");
      Serial.print(yaw_rad);
      Serial.println("\t");
    for(int i = 0; i<4; i++)
    {
      vlinier[i] = 0;
    }
}
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
void move_mode()
{
  switch(mode_jalan)
  {
    case 0:
    //mode jalan remote
```

```
    forward_kinematic();
    input_ori = 270;
    break;
    case 1:
    //mode siap ditempat
    kecepatan_motor_kanan = 0;
    kecepatan_motor_kiri = 0;
    enable_compass = false;
    break;
    case 4:
    //mode luruskan orientasi
    input_ori = saveyaw;
    kecepatan_motor_kanan = 0;
    kecepatan_motor_kiri = 0;
    enable_compass = true;
    break;
    case 2:
    //mode bug Explore
    enable_compass = false;
    break;
    case 3:
    //mode return home
    enable_compass = false;
    break;
  }
}
////////////////////////////////////////////////////////
////////////////////////////////////////////////////////

void motor_setup()
{
  pinMode(motor1, OUTPUT);
  pinMode(motor2, OUTPUT);
  pinMode(motor3, OUTPUT);
  pinMode(motor4, OUTPUT);
  pinMode(arah1, OUTPUT);
  pinMode(arah2, OUTPUT);
  pinMode(arah3, OUTPUT);
  pinMode(arah4, OUTPUT);
```

```
}
void set_kecepatan(int motor, int pwm_val, int dir, int arah)
{
  digitalWrite(dir, arah);
  analogWrite(motor, pwm_val);
}
void set_pid()
{
    for(int x=0; x<4; x++)
    {
//        kp[x]=4, ki[x]=50; kd[x]=0.15;   //Kecepatan Motor paling
bener
        kp[x]=4, ki[x]=50; kd[x]=0.15;   //Kecepatan Motor
//          kp[x]=8, ki[x]=100; kd[x]=0.15;   //Kecepatan Motor
    }
     kp[4]=30, ki[4]=0; kd[4]=300;     //Orientasi
//     kp[0]=180, ki[0]=40; kd[0]=10;   //Kecepatan Motor
}
void pid_kecepatan()
{
  float dt = 0.05;
  forward_kinematic();
  if(enable_compass == 1) pid_orientasi();
  else pwm_kecepatan[4] = 0;
  for(int x = 0; x<4; x++)
  {
    if(input_kecepatan[x]<0)
    {
      setpoint[x] = -1*input_kecepatan[x]/2;
      logic_arah[x] = HIGH;
    }
    else if(input_kecepatan[x]>=0)
    {
      setpoint[x] = input_kecepatan[x]/2;
      logic_arah[x] = LOW;
    }
    error[x] = setpoint[x]-kecepatan[x];
    sumerror[x] += error[x]*dt; //karena dalam micros
//    if(sumerror[x]>1024) sumerror[x] = 1024;
```

```
//    else if(sumerror[x]<-1024) sumerror[x] = -1024;
    Derr[x] = (error[x]-lasterror[x])/dt;
    lasterror[x] = error[x];
    pwm_kecepatan[x] = kp[x]*error[x] + ki[x]*sumerror[x] +
kd[x]*Derr[x];
    input_kecepatan[x] = 0;
//    Serial.print(kecepatan[x]);
//    Serial.print("\t");
  }
  set_kecepatan_motor();
  hitung_kecepatan();
  tampil_rps();
  hitung_posisi_robot();
}
void pid_orientasi()
{
    double sudut_sekarang;
    sudut_sekarang = ypr[0];
    if(sudut_sekarang>179) sudut_sekarang = map(ypr[0], 180, 360, -
180, 0);
    setpoint[4] = input_ori;
    if(input_ori>179)  setpoint[4] = map(input_ori, 180, 360, -180, 0);
    error[4] = setpoint[4]-sudut_sekarang;
    sumerror[4] += error[4]; //karena dalam micros
    if(sumerror[4]>20) sumerror[4] = 20;
    else if(sumerror[4]<-20) sumerror[4] = -20;
    Derr[4] = (error[4]-lasterror[4]);
    pwm_kecepatan[4] = kp[4]*error[4] + ki[4]*sumerror[4] +
kd[4]*Derr[4];
    if(pwm_kecepatan[4]>=200) pwm_kecepatan[4] = 200;
    else if(pwm_kecepatan[4]<=-200) pwm_kecepatan[4] = -200;
    lasterror[4] = error[4];
//   Serial.print(setpoint[4]);
//   Serial.print("\t");
//   Serial.print(sudut_sekarang);
//   Serial.print("\t");
//   Serial.print(error[4]);
//   Serial.print("\t");
//   Serial.println(sumerror[4]);
```

```
}
void set_kecepatan_motor()
{
   set_kecepatan(motor1, pwm_kecepatan[0], arah1, logic_arah[0]);
   set_kecepatan(motor2, pwm_kecepatan[1], arah2, logic_arah[1]);
   set_kecepatan(motor3, pwm_kecepatan[2], arah3, logic_arah[2]);
   set_kecepatan(motor4, pwm_kecepatan[3], arah4, logic_arah[3]);
}
void tampil_rps()
{
  for(int f=0; f<5; f++)
    {
      rps[f] += display_kecepatan[f];
    }
  vlinier[0] += (rps[2]+rps[3])/2;
  vlinier[1] += (rps[0]+rps[1])/2;
}
////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
void send_data_ros()
{
  kirim_compass.data = yaw_rad;
  pos_x_alamat.data = pos_x;
  pos_y_alamat.data = pos_y;
  ori_w_alamat.data = yaw_rad;

  pos_x_arduino.publish(&pos_x_alamat);
  pos_y_arduino.publish(&pos_y_alamat);
  ori_w_arduino.publish(&ori_w_alamat);
  compass.publish(&kirim_compass);
//  imubno.publish(&imu_msg);
  nh.spinOnce();
}
```

Program Hector Mapping

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

<launch>
        <arg name="map_file"
value="/home/pi/catkin_ws/depan2.yaml"/>

   <!--node name="map_server" pkg="map_server"
type="map_server" args="$(arg map_file)">
                    <param name="frame_id" value="map"/>
          </node-->

        <arg name="port" default="/dev/ttyAMA1" />
        <arg name="baud" default="9600" />
        <arg name="frame_id" default="gps" />
        <arg name="use_GNSS_time" default="False" />
        <arg name="time_ref_source" default="gps" />
        <arg name="useRMC" default="False" />
        <node name="serial_node"  pkg="rosserial_python"
type="serial_node.py" output="screen">
        </node>

        <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="base_footprint"/>
                    <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="0.1" />
                    <param name="range_max"    type="double"
value="16.0" />
                    <param name="ignore_array" type="string"
value="" />
                    <param name="samp_rate"    type="int"
value="9"/>
                    <param name="frequency"    type="double"
value="7"/>
          </node>

          <node name="teleop"  pkg="teleop_twist_keyboard"
type="teleop_twist_keyboard.py" output="screen">
          </node>

          <!--node name="odom_arduino" pkg="ydlidar"
type="slam_out_pose_to_odom.py" output="screen">
          </node-->

          <!--node name="odomtransformer" pkg="ydlidar"
type="odomtransformer.py" output="screen">
                    <param name="odom_input" value="/nav" />
                    <param name="tf_output" value="/base_footprint"
/>
          </node-->

          <arg name="geotiff_map_file_path" default="$(find
hector_geotiff)/maps"/>

          <param name="/use_sim_time" value="false"/>

          <!--node pkg ="tf" type="static_transform_publisher"
name="map_to_odom" args="0.0 0.0 0.0 0.0 0.0 0.0 /map
/scanmatcher_odom 40"/-->
```

```
	<node pkg ="tf" type="static_transform_publisher"
name="odom_to_base_link" args="0.0 0.0 0.0 0.0 0.0 0.0
/scanmatcher_odom /base_footprint 100"/>

	<node pkg ="tf" type="static_transform_publisher"
name="base_link_to_laser" args="0.0 0.0 0.0 0.0 0.0 0.0
/base_footprint /laser_frame 40" />

	<node pkg="rviz" type="rviz" name="rviz"
		args="-d $(find
hector_slam_launch)/rviz_cfg/mapping_demo.rviz"/>

	<include file="$(find
hector_mapping)/launch/mapping_default.launch"/>

	<include file="$(find
hector_geotiff)/launch/geotiff_mapper.launch">
		<arg name="trajectory_source_frame_name"
value="scanmatcher_frame"/>
		<arg name="map_file_path" value="$(arg
geotiff_map_file_path)"/>
	</include>
	<!--node name="telusur_kanan" pkg="laser"
type="telusur_dinding.py" output="screen">
	</node-->
	<node name="reset_odom" pkg="ydlidar"
type="sendding_arduino" output="screen">
	</node>
</launch>
```

Program AMCL

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

<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="base_footprint"/>
                <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="0.1" />
                <param name="range_max"    type="double"
value="16.0" />
                <param name="ignore_array" type="string"
value="" />
                <param name="samp_rate"    type="int"
value="9"/>
                <param name="frequency"    type="double"
value="7"/>
   </node>
```

```
   <node name="serial_node"  pkg="rosserial_python"
type="serial_node.py" output="screen">
   </node>

   <arg name="map_file"
value="/home/pi/ekf_Ws/devel_isolated/mapping00603startruangka
mar.yaml"/>

   <node name="map_server" pkg="map_server" type="map_server"
args="$(arg map_file)">
                        <param name="frame_id" value="map"/>
             </node>

   <node name="teleop"  pkg="teleop_twist_keyboard"
type="teleop_twist_keyboard.py" output="screen">
   </node>


   <!--node name="odomtransformer" pkg="ydlidar"
type="odomtransformer.py" output="screen">
                        <param name="odom_input"
value="/pose_stamped" />
                        <param name="tf_output" value="/base_footprint"
/>
            </node-->

   <node name="odom_arduino" pkg="ydlidar"
type="slam_out_pose_to_odom.py" output="screen">
            </node>

   <param name="/use_sim_time" value="false"/>

   <!--node pkg ="tf" type="static_transform_publisher"
name="map_to_odom" args="0.0 0.0 0.0 0.0 0.0 0.0 /map
/scanmatch_odom 40"/-->

   <!--node pkg ="tf" type="static_transform_publisher"
name="odom_to_base_link" args="0.0 0.0 0.0 0.0 0.0 0.0 /nav
/base_footprint 40"/-->
```

```
   <node pkg ="tf" type="static_transform_publisher"
name="laser_to_base_footprint" args="0.0 0.0 0.0 0.0 0.0 0.0
/base_footprint /laser 40" />

   <arg name="init_x" default="0" />
   <arg name="init_y" default="0" />
   <arg name="init_a" default="0" />
   <remap from="map" to="map"/>

   <node pkg="amcl" type="amcl" name="amcl" output="screen">
      <!-- Publish scans from best pose at a max of 10 Hz -->
      <param name="transform_tolerance" value="0.1" />
      <param name="gui_publish_rate" value="5.0"/>
      <param name="min_particles" value="25"/>
      <param name="max_particles" value="1000"/>
      <param name="kld_err" value="0.01"/>
      <param name="kld_z" value="0.99"/>
      <!-- translation std dev, m -->
      <param name="odom_alpha1" value="0.01"/>
      <param name="odom_alpha2" value="0.01"/>
      <param name="odom_alpha3" value="0.01"/>
      <param name="odom_alpha4" value="0.01"/>
      <param name="laser_z_hit" value="0.9"/>
      <param name="laser_z_short" value="0.05"/>
      <param name="laser_z_max" value="0.05"/>
      <param name="laser_z_rand" value="0.05"/>
      <param name="laser_sigma_hit" value="0.1"/>
      <param name="laser_lambda_short" value="0.1"/>
      <param name="laser_min_range" value="0.15"/>
      <param name="laser_max_range" value="5.0"/>

      <param name="laser_max_beams" value="50"/>
      <param name="laser_model_type" value="likelihood_field"/>
      <param name="laser_likelihood_max_dist" value="4.0"/>
      <param name="update_min_d" value="0.01"/>
      <param name="update_min_a" value="0.01"/>
      <param name="resample_interval" value="1"/>
      <param name="recovery_alpha_slow" value="0.001"/>
```

```
        <param name="recovery_alpha_fast" value="0.1"/>

        <param name="use_map_topic" value="true"/>
        <param name="first_map_only" value="false"/>
        <param name="tf_broadcast" value="true"/>

        <param name="odom_frame_id" value="scanmatch_odom"/>
        <param name="global_frame_id" value="map"/>
        <param name="base_frame_id" value="base_footprint"/>
        <param name="odom_model_type" value="diff-corrected"/>

        <param name="initial_pose_x" value="$(arg init_x)"/>
        <param name="initial_pose_y" value="$(arg init_y)"/>
        <param name="initial_pose_a" value="$(arg init_a)"/>
        <param name="initial_cov_xx" value="0.00" />
        <param name="initial_cov_yy" value="0.00" />
        <param name="initial_cov_aa" value="0.00" />
        </node>

        <arg name="use_rviz" default="false" />
        <arg name="publish_covariance" default="false"/>

        <param name="/use_sim_time" value="false"/>
        <param name="/stamped_vel" value="true"/>

        <group if="$(arg publish_covariance)">
        <param
name="laser_scan_matcher_node/do_compute_covariance"
value="1"/>
        <param
name="laser_scan_matcher_node/publish_pose_with_covariance"
value="true"/>
        <param
name="laser_scan_matcher_node/publish_pose_with_covariance_sta
mped" value="true"/>
        </group>
        <node pkg="laser_scan_matcher"
type="laser_scan_matcher_node"
        name="laser_scan_matcher_node" output="screen">
```

```
        <param name="max_iterations" value="10"/>
        <param name="fixed_frame" value="scanmatch_odom"/>
        <param name="base_frame" value="base_footprint"/>
        <param name="publish_tf" value="true"/>
        <param name="publish_pose_stamped" value="true"/>
        </node>

        <node pkg="rviz" type="rviz" name="rviz"
        args="-d $(find
hector_slam_launch)/rviz_cfg/mapping_demo.rviz"/>

        <node name="reset_odom" pkg="ydlidar"
type="sendding_arduino" output="screen">
        </node>
</launch>
```

Program Navigasi ROS

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

  <arg name="no_static_map" default="false"/>

  <arg name="base_global_planner" default="navfn/NavfnROS"/>
  <arg name="base_local_planner"
default="dwa_local_planner/DWAPlannerROS"/>
  <!-- <arg name="base_local_planner"
default="base_local_planner/TrajectoryPlannerROS"/> -->

  <node pkg="move_base" type="move_base" respawn="false"
name="move_base" output="screen">

    <param name="base_global_planner" value="$(arg
base_global_planner)"/>
    <param name="base_local_planner" value="$(arg
base_local_planner)"/>
```

```
   <rosparam file="$(find costmap_2d)/launch/planner.yaml"
command="load"/>

   <!-- observation sources located in costmap_common.yaml -->
   <rosparam file="$(find
costmap_2d)/launch/costmap_common.yaml" command="load"
ns="global_costmap" />
   <rosparam file="$(find
costmap_2d)/launch/costmap_common.yaml" command="load"
ns="local_costmap" />

   <!-- local costmap, needs size -->
   <rosparam file="$(find costmap_2d)/launch/costmap_local.yaml"
command="load" ns="local_costmap" />
   <param name="local_costmap/width" value="3.0"/>
   <param name="local_costmap/height" value="3.0"/>

   <!-- static global costmap, static map provides size -->
   <rosparam file="$(find
costmap_2d)/launch/costmap_global_static.yaml" command="load"
ns="global_costmap" unless="$(arg no_static_map)"/>

   <!-- global costmap with laser, for odom_navigation_demo -->
   <rosparam file="$(find
costmap_2d)/launch/costmap_global_laser.yaml" command="load"
ns="global_costmap" if="$(arg no_static_map)"/>
   <param name="global_costmap/width" value="25.0" if="$(arg
no_static_map)"/>
   <param name="global_costmap/height" value="25.0" if="$(arg
no_static_map)"/>
  </node>

</launch>
```

Program Reset Odometry

```
#include "ros/ros.h"
#include "std_msgs/Int8.h"
#include "std_msgs/Float64.h"
#include "nav_msgs/Odometry.h"
```

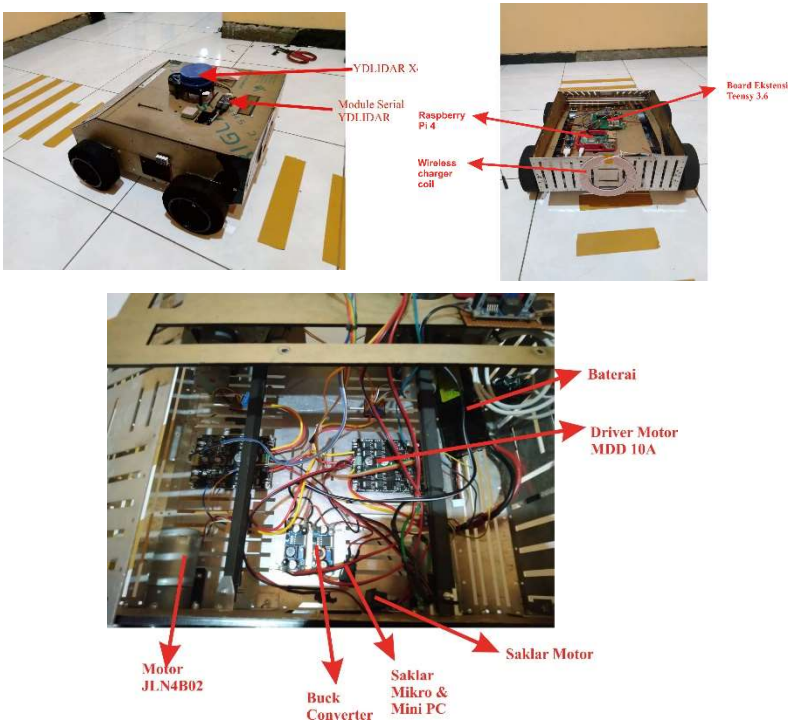
```
#include <signal.h>

void mySigintHandler(int sig)
{
	for(int x = 0; x<50; x++)
	{
	ros::NodeHandle n;
	ros::Publisher reset = n.advertise<std_msgs::Int8>("/reset",
1000);
	std_msgs::Int8 myMsg;
	myMsg.data = 1;
	reset.publish(myMsg);
	}
	ros::shutdown();
}
int main(int argc, char **argv)
{
   ros::init(argc, argv, "send_to_arduino");
	ros::NodeHandle n;
	signal(SIGINT, mySigintHandler);
	ros::Publisher reset = n.advertise<std_msgs::Int8>("/reset",
1000);
   ros::Rate loop_rate(10);
	int state_reset = 0;

   while(ros::ok()){
		std_msgs::Int8 myMsg;
		myMsg.data = state_reset;
		if(state_reset == 0)
		{
			for(int x=0;x<50;x++)
			{
				myMsg.data = 2;
				reset.publish(myMsg);
				//ROS_INFO("state_reset =  %d",
state_reset);
				ros::spinOnce();
				loop_rate.sleep();
			}
```

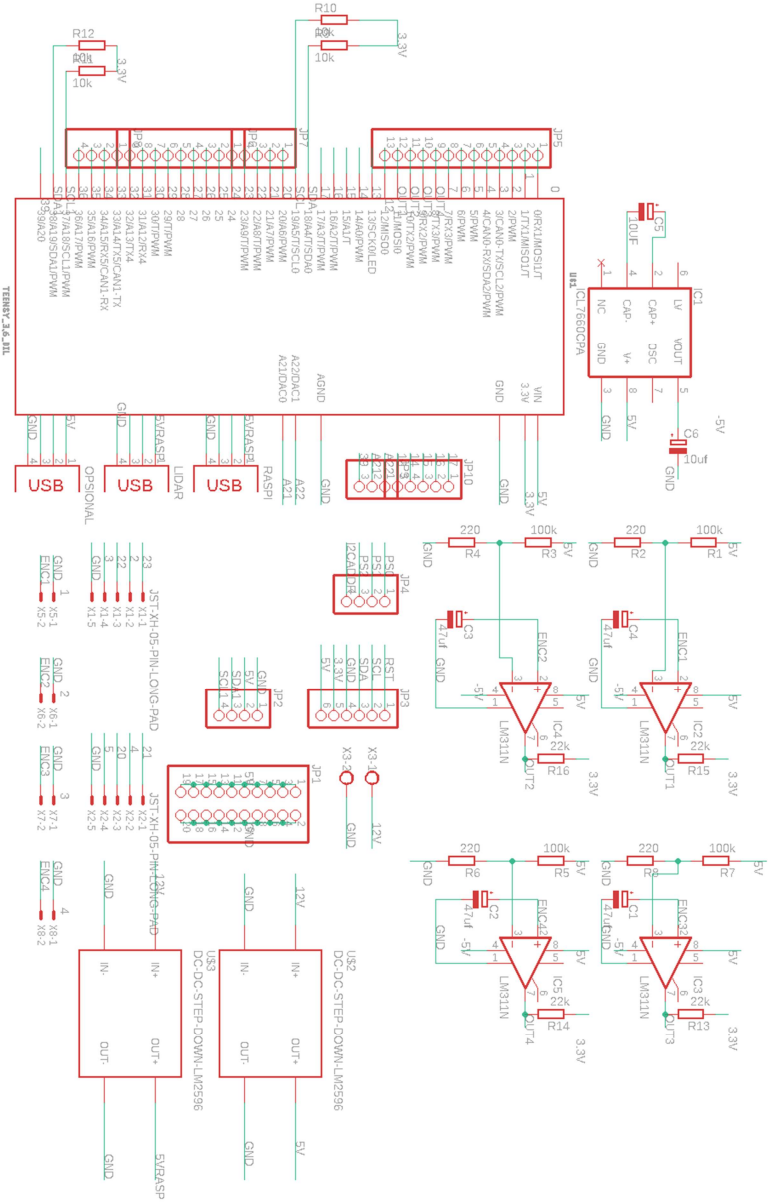
```
                                state_reset = 1;
                        }
                        else
                        {
                    //ROS_INFO("state_reset =  %d", state_reset);
                        ros::spinOnce();
                        loop_rate.sleep();
                        }
                }
                ros::spin();
    return 0;
}
```

LAMPIRAN B

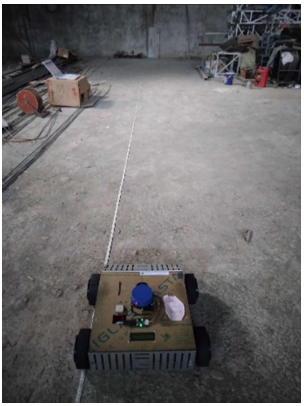
A. Dokumentasi Kegiatan

1. Realisasi Sistem

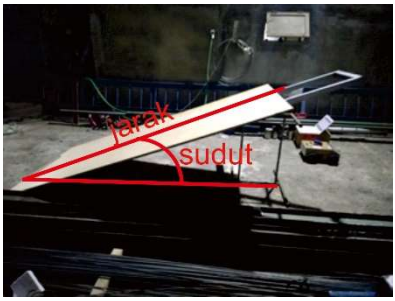


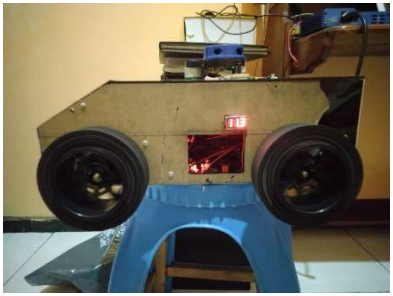

R10
10k
3.3V
R12
10k
3.3V
JP5
JP7
JP8
JP9
TEENSY 3.6_STL
0(RX1/MOSI1)T
1(TX1/MISO1)T
2PVM
3(CAN0-TX/SCL2)PVM
4(CAN0-RX/SDA2)PVM
5PVM
6PVM
7(RX3)PVM
8(TX3)PVM
9(OC1B/PWM2)PVM
10(CS0/PWM)PVM
11(MOSI0)
12(MISO0)
13(SCK0/LED)
14/A0PVM
15/A1T
16/A2/TPVM
17/A3/TPVM
18/A4/TSDA0
19/A5/TSCL0
20/A6PVM
21/A7PVM
22/A8/TPVM
23/A9/TPVM
24
25
26
27
28
29/TPVM
30/TPVM
31/A12/RX4
32/A13/TX4
33/A14/TSCA/AN1.TX
34/A15/RX5/CAN1.RX
35/A16PVM
36/A17PVM
37/A18/SCL1PVM
38/A19/SDA1PVM
39/A20
SDA1
SCL1
GND
5V
3.3V
A21/DAC1
A22/DAC0
AGND
VIN
USB
OPSIONAL
USB
LIDAR
USB
RASPI
5VRASP
GND
JP10
A21
A22
A23
A24
A25
A26
A27
A28
A29
A30
A31
A32
A33
A34
A35
A36
A37
A38
A39
IC1
ISL76600PA
C5
10UF
C6
10uf
-5V
JP4
I2CAN
PS1
PS2
PS3
PS4
JP2
JP3
SDA1
SCL1
SDA
SCL
RST
JP1
X3
X4
12V
JST-XH-05-PIN-LONG-PAD
ENC1
ENC2
ENC3
ENC4
X5:1
X5:2
X6:1
X6:2
X7:1
X7:2
X8:1
X8:2
X1:1
X1:2
X1:3
X1:4
X1:5
X2:1
X2:2
X2:3
X2:4
X2:5
220
100k
R4
R3
R2
R1
R6
R5
R8
R7
C3
C4
C2
C1
4.7uf
IC4
IC2
IC5
IC3
LM311N
R16
R15
R14
R13
2k
22k
OUT1
OUT2
OUT3
OUT4
US3
US2
DC-DC-STEP-DOWN-LM2596
IN+
IN-
OUT+
OUT-
5VRASP

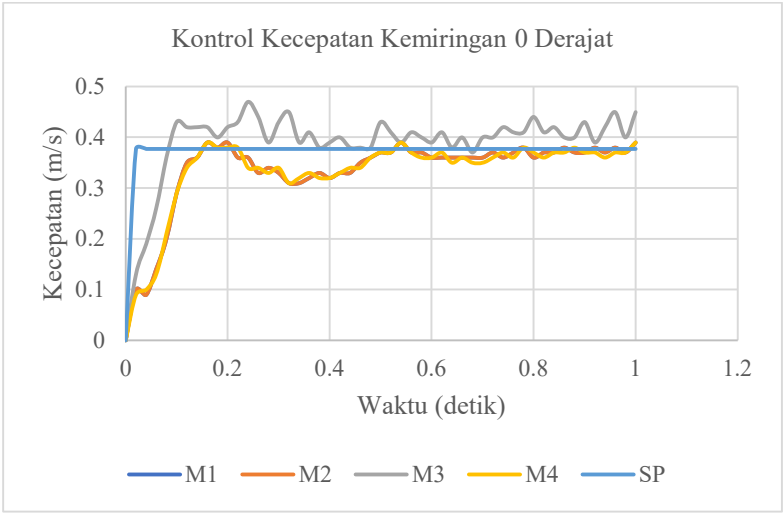
2. Pengujian LIDAR

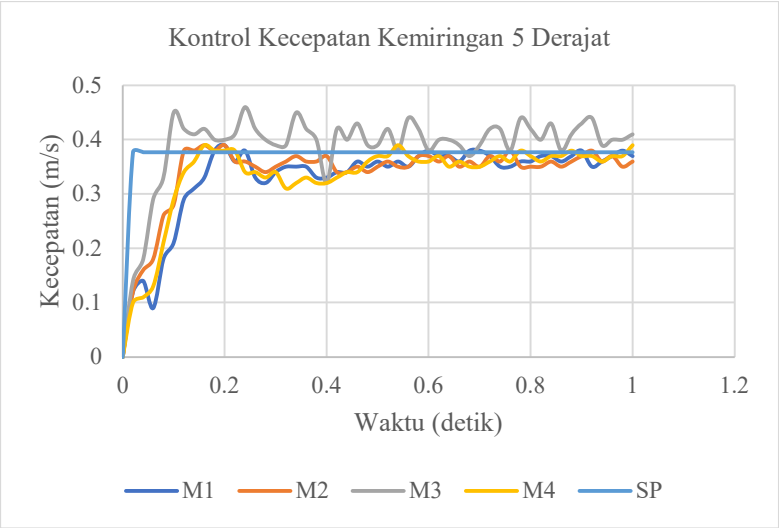


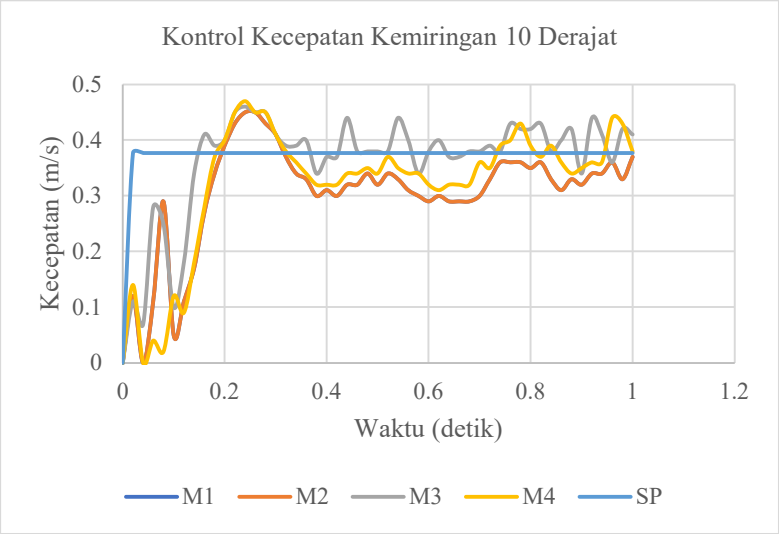
3. Pengujian Kontrol Motor

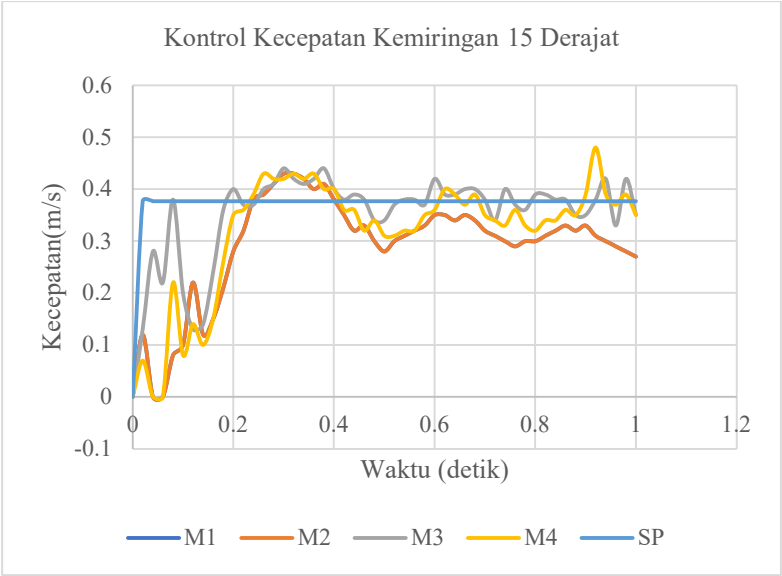


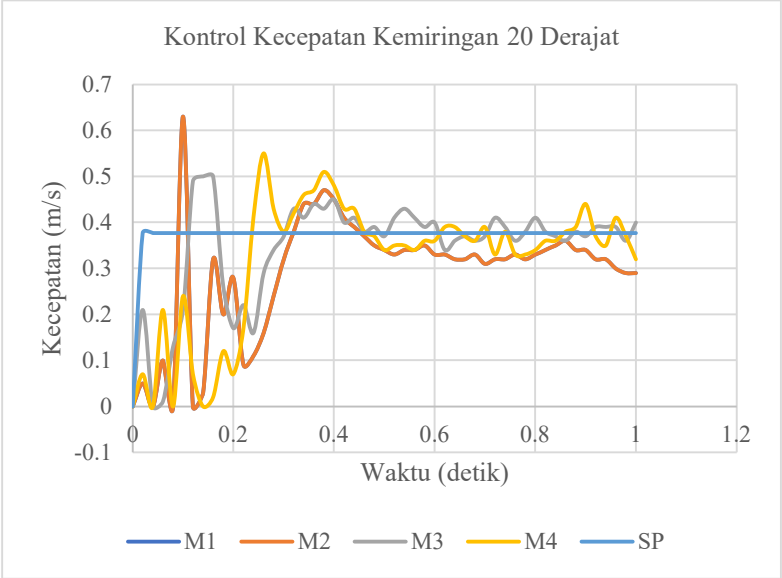





Kontrol Kecepatan Kemiringan 5 Derajat
Kecepatan (m/s)
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1.2
Waktu (detik)
M1
M2
M3
M4
SP

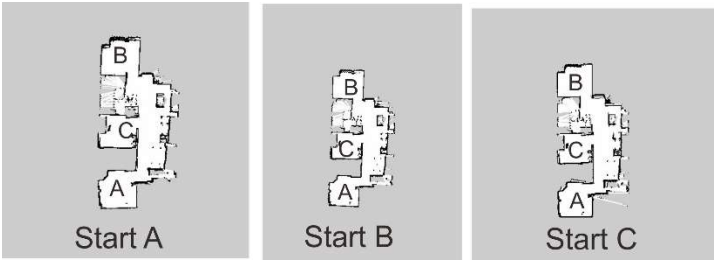

Kontrol Kecepatan Kemiringan 10 Derajat
Kecepatan (m/s)
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1
1.2
Waktu (detik)
M1
M2
M3
M4
SP

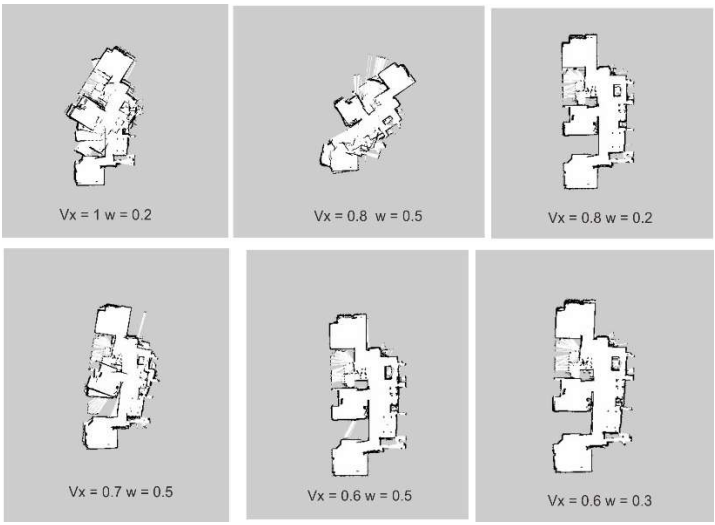

Kontrol Kecepatan Kemiringan 15 Derajat
Kecepatan(m/s)
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
0
0.2
0.4
0.6
0.8
1
1.2
Waktu (detik)
M1
M2
M3
M4
SP


Kontrol Kecepatan Kemiringan 20 Derajat
Kecepatan (m/s)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
0
0.2
0.4
0.6
0.8
1
1.2
Waktu (detik)
M1
M2
M3
M4
SP

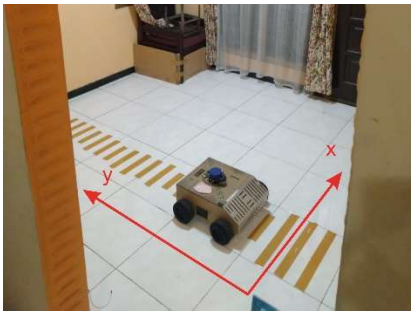
4. Realisasi Stasiun Pengisian

5. Pengujian Pemetaan

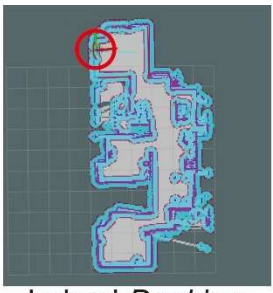




6. Pengujian Lokalisasi



7. Pengujian Navigasi Docking

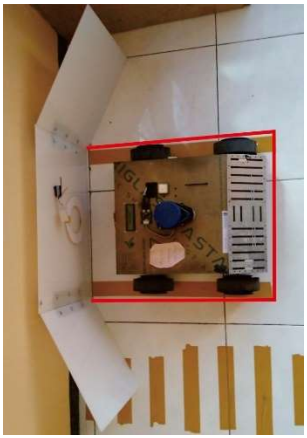


Lokasi *Docking*

Mulai (A)

Mulai (B)

Mulai (C)




1


2

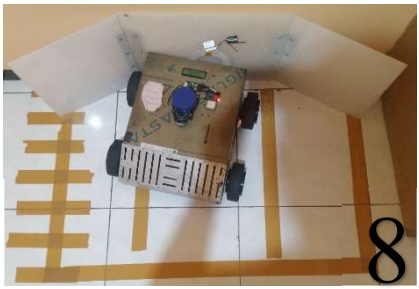

3


4


5


6


7


8

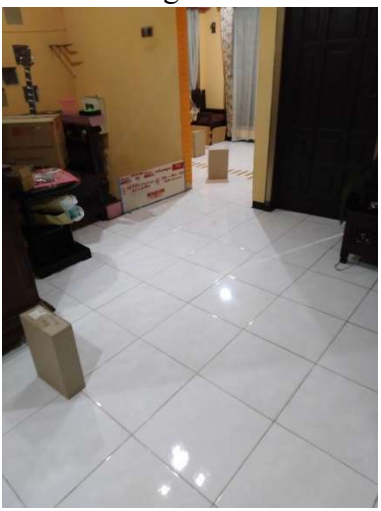

9

8. Pengujian Navigasi Docking Halangan Baru

Konfigurasi 1

Konfigurasi 2

Konfigurasi 3



LAMPIRAN C

1. Datasheet YDLIDAR X4

Measurement Performance

FIG. 1 YDLIDAR X4 MEASUREMENT PERFORMANCE

Subject	Min.	Typical	Max.	Unit	Remark
Sample Frequency	-	5000	-	Hz	5000 samples/sec
Scanning Frequency	6	-	12	Hz	PWM or voltage control
Range	0.12	-	>10	m	Indoor environment
Angular Range	-	0~360	-	Deg	-
Distance Resolution	-	<0.5	-	mm	Range<2m
		<1% of the distance			Range>2m
Angular Resolution	0.48	0.50	0.52	Deg	Frequency at 7Hz
Life Span	-	1500	-	h	

FIG. YDLIDAR X4 POWER SUPPLY SPECIFICATIONS

Subject	Min.	Typical	Max.	Unit	Remark
Supply Voltage	4.8	5	5.2	V	If the voltage exceeds the max value, it may damage the core. Too low Voltage may affect performance and even stop ranging
Voltage Ripple	0	50	100	mV	High ripple may cause working failure
Start Current	400	450	480	mA	Relatively higher current is required at startup of the device
Sleep mode Current	280	300	340	mA	System dormancy with motor rotating
Work mode Current	330	350	380	mA	

FIG. 4 YDLIDAR X4 SERIAL PORT SPECIFICATIONS

Subject	Min.	Typical	Max.	Unit	Remark
Baud rate	-	128000	-	bps	8bits, 1stop bit, no check bit
Output high voltage	1.8	3.3	3.5	V	>1.8V
Output low voltage	0	0	0.5	V	<0.5V

External Interface Definition

X4provides PH2.0-8P base interface. It contains system voltage interface, data communication interface and motor control interface.

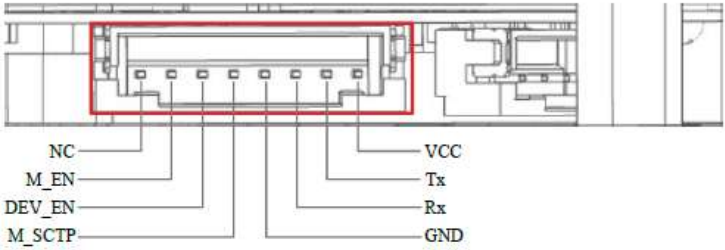


FIG. 3 YDLIDAR X4 INTERFACE DEFINITION

Pin	Type	Description	Default value	Range	Range
VCC	Power Supply	V+	5V	4.8V~5.2V	-
Tx	Output	serial port output	-	-	Data flow: lidar→peripheral
Rx	Input	serial port input	-	-	Data flow : peripheral→ lidar
GND	Power Supply	V-	0V	0V	-
M_EN	Input	Motor control	3.3V	0V~3.3V	
DEV_EN	Input	Scanning control	3.3V	0V~3.3V	
M_SCTP	Input	Motor Speed	1.8V	0V~3.3V	PWM or voltage control
NC	-	Reservation pin	-	-	-

FIG. 5 YDLIDAR X4 PWM CHART

Subject	Min.	Typical	Max.	Unit	Remark
PWM Frequency	-	10	-	KHz	PWM Square wave
Duty cycle range	50%	85%	100%		The smaller the duty cycle, the faster the speed

FIG. 6 YDLIDAR X4 LASER POWER SPECIFICATION

Subject	Min.	Typical	Max.	Unit	Remark
Laser wavelength	775	785	795	nm	Infrared Band Light
Laser power	-	3	5	mW	Peak power

FIG. 7 YDLIDAR X4 OTHER SPECIFICATIONS

Subject	Min.	Typical	Max.	Unit	Remark
Working temperature	0	20	40	℃	Working in a high temperature environment for a long time will reduce the life span.
Light environment	0	550	2000	Lux	For reference only
Weight	-	180	-	g	Net Weight

2. Datasheet Teensy 3.6

Kinetis K66 Sub-Family

180 MHz ARM® Cortex®-M4F Microcontroller.

MK66FN2M0VMD18
MK66FX1M0VMD18
MK66FN2M0VLQ18
MK66FX1M0VLQ18

The K66 sub-family members provide greater performance, memory options up to 2 MB total flash and 256 KB of SRAM, as well as higher peripheral integration with features such as Dual USB and a 10/100 Mbit/s Ethernet MAC. These devices maintain hardware and software compatibility with the existing Kinetis family. This product also offers:

- Integration of a High Speed USB Physical Transceiver
- Greater performance flexibility with a High Speed Run mode
- Smarter peripherals with operation in Stop modes

144 MAPBGA (MD) 13 mm x 13 mm Pitch 1 mm

144 LQFP (LQ) 20 mm x 20 mm Pitch 0.5 mm

Performance
- Up to 180 MHz ARM Cortex-M4 based core with DSP instructions and Single Precision Floating Point unit

System and Clocks
- Multiple low-power modes to provide power optimization based on application requirements
- Memory protection unit with multi-master protection
- 3 to 32 MHz main crystal oscillator
- 32 kHz low power crystal oscillator
- 48 MHz internal reference

Security
- Hardware random-number generator
- Supports DES, AES, SHA accelerator (CAU)
- Multiple levels of embedded flash security

Timers
- Four Periodic interrupt timers
- 16-bit low-power timer
- Two 16-bit low-power timer PWM modules
- Two 8-channel motor control/general purpose/PWM timers
- Two 2-ch quad decoder/general purpose timers
- Real-time clock

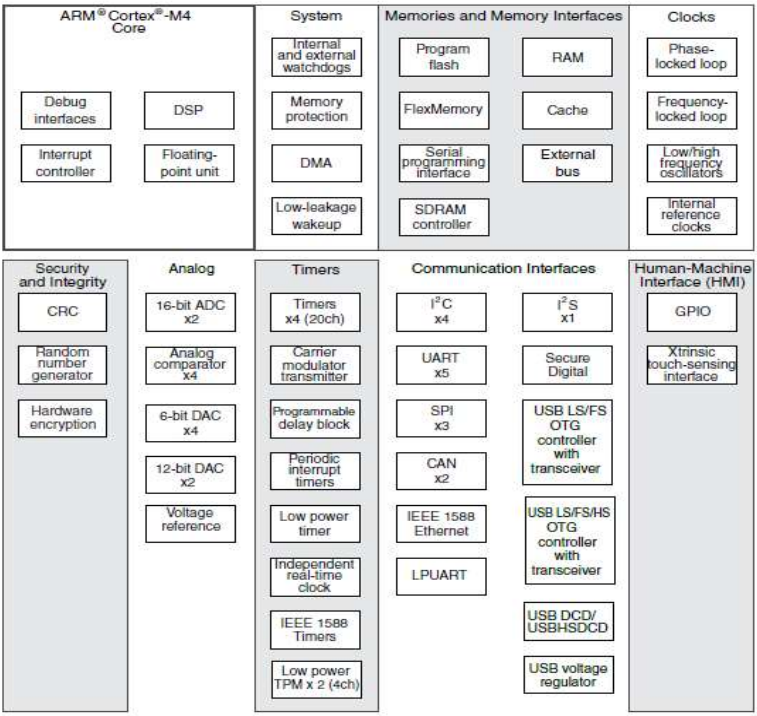
Human-machine interface
- Low-power hardware touch sensor interface (TSI)
- General-purpose input/output

Memories and memory expansion
- Up to 2 MB program flash memory on non-FlexMemory devices with 256 KB RAM
- Up to 1 MB program flash memory and 256 KB of FlexNVM on FlexMemory devices
- 4 KB FlexRAM on FlexMemory devices
- FlexBus external bus interface and SDRAM controller

Analog modules
- Two 16-bit SAR ADCs and two 12-bit DAC
- Four analog comparators (CMP) containing a 6-bit DAC and programmable reference input
- Voltage reference 1.2V

Communication interfaces
- Ethernet controller with MII and RMII interface to external PHY and hardware IEEE 1588 capability
- USB high-/full-/low-speed On-the-Go with on-chip high speed transceiver
- USB full-/low-speed OTG with on-chip transceiver
- Two CAN, three SPI and four I2C modules
- Low Power Universal Asynchronous Receiver/ Transmitter 0 (LPUART0) and five standard UARTs
- Secure Digital Host Controller (SDHC)
- I2S module

Operating Characteristics
- Voltage/Flash write voltage range:1.71 to 3.6 V
- Temperature range (ambient): -40 to 105°C



1.1 Thermal handling ratings

Symbol	Description	Min.	Max.	Unit	Notes
T_{STG}	Storage temperature	–55	150	°C	1
T_{SDR}	Solder temperature, lead-free	—	260	°C	2

1. Determined according to JEDEC Standard JESD22-A103, *High Temperature Storage Life*.
2. Determined according to IPC/JEDEC Standard J-STD-020, *Moisture/Reflow Sensitivity Classification for Nonhermetic Solid State Surface Mount Devices*.

1.2 Moisture handling ratings

Symbol	Description	Min.	Max.	Unit	Notes
MSL	Moisture sensitivity level	—	3	—	1

1. Determined according to IPC/JEDEC Standard J-STD-020, *Moisture/Reflow Sensitivity Classification for Nonhermetic Solid State Surface Mount Devices*.

1.3 ESD handling ratings

Symbol	Description	Min.	Max.	Unit	Notes
V_{HBM}	Electrostatic discharge voltage, human body model	-2000	+2000	V	1
V_{CDM}	Electrostatic discharge voltage, charged-device model	-500	+500	V	2
I_{LAT}	Latch-up current at ambient temperature of 105°C	-100	+100	mA	3

1. Determined according to JEDEC Standard JESD22-A114, *Electrostatic Discharge (ESD) Sensitivity Testing Human Body Model (HBM)*.
2. Determined according to JEDEC Standard JESD22-C101, *Field-Induced Charged-Device Model Test Method for Electrostatic-Discharge-Withstand Thresholds of Microelectronic Components*.
3. Determined according to JEDEC Standard JESD78, *IC Latch-Up Test*.

Symbol	Description	Min.	Max.	Unit
V_{DD}	Digital supply voltage	–0.3	3.8	V
I_{DD}	Digital supply current	—	300	mA
V_{DIO}	Digital[1] input voltage,including RESET_b	–0.3	V_{DD} + 0.3	V
V_{AIO}	Analog[1] input voltage, including EXTAL32 and XTAL32	–0.3	V_{DD} + 0.3	V
I_D	Maximum current single pin limit (digital output pins)	–25	25	mA
V_{DDA}	Analog supply voltage	V_{DD} – 0.3	V_{DD} + 0.3	V
V_{USB0_DP}	USB0_DP input voltage	–0.3	3.63	V
V_{USB1_DP}	USB1_DP input voltage	–0.3	3.63	V
V_{USB0_DM}	USB0_DM input voltage	–0.3	3.63	V
V_{USB1_DM}	USB1_DM input voltage	–0.3	3.63	V
V_{USB1_VBUS}	USB1_VBUS detect voltage	–0.3	6.0	V
VREG_IN0, VREG_IN1	USB regulator input	–0.3	6.0	V
V_{BAT}	RTC battery supply voltage	–0.3	3.8	V

1. Digital pins have a general purpose I/O port assigned (e.g. PTA0). Analog pins do not have an associated general purpose I/O port.

3. Datasheet Raspberry Pi 4

Symbol	Parameter	Minimum	Maximum	Unit
VIN	5V Input Voltage	-0.5	6.0	V

Table 2: Absolute Maximum Ratings

Please note that VDD_IO is the GPIO bank voltage which is tied to the on-board 3.3V supply rail.

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage[a]	VDD_IO = 3.3V	-	-	TBD	V
V_{IH}	Input high voltage[a]	VDD_IO = 3.3V	TBD	-	-	V
I_{IL}	Input leakage current	TA = +85°C	-	-	TBD	µA
C_{IN}	Input capacitance	-	-	TBD	-	pF
V_{OL}	Output low voltage[b]	VDD_IO = 3.3V, IOL = -2mA	-	-	TBD	V
V_{OH}	Output high voltage[b]	VDD_IO = 3.3V, IOH = 2mA	TBD	-	-	V
I_{OL}	Output low current[c]	VDD_IO = 3.3V, VO = 0.4V	TBD	-	-	mA
I_{OH}	Output high current[c]	VDD_IO = 3.3V, VO = 2.3V	TBD	-	-	mA
R_{PU}	Pullup resistor	-	TBD	-	TBD	kΩ
R_{PD}	Pulldown resistor	-	TBD	-	TBD	kΩ

[a] Hysteresis enabled
[b] Default drive strength (8mA)
[c] Maximum drive strength (16mA)

Table 3: DC Characteristics

Pin Name	Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital outputs	t_{rise}	10-90% rise time[a]	-	TBD	-	ns
Digital outputs	t_{fall}	90-10% fall time[a]	-	TBD	-	ns

[a] Default drive strength, CL = 5pF, VDD_IO = 3.3V

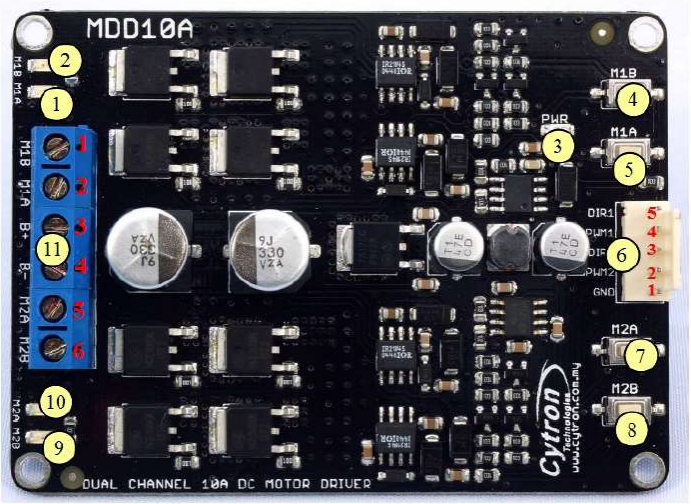
Table 4: Digital I/O Pin AC Characteristics

5.1.2 GPIO Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	SPI3_CE0_N	TXD2	SDA6
1	High	SCL0	SA4	DE	SPI3_MISO	RXD2	SCL6
2	High	SDA1	SA3	LCD_VSYNC	SPI3_MOSI	CTS2	SDA3
3	High	SCL1	SA2	LCD_HSYNC	SPI3_SCLK	RTS2	SCL3
4	High	GPCLK0	SA1	DPI_D0	SPI4_CE0_N	TXD3	SDA3
5	High	GPCLK1	SA0	DPI_D1	SPI4_MISO	RXD3	SCL3
6	High	GPCLK2	SOE_N	DPI_D2	SPI4_MOSI	CTS3	SDA4
7	High	SPI0_CE1_N	SWE_N	DPI_D3	SPI4_SCLK	RTS3	SCL4
8	High	SPI0_CE0_N	SD0	DPI_D4	-	TXD4	SDA4
9	Low	SPI0_MISO	SD1	DPI_D5	-	RXD4	SCL4
10	Low	SPI0_MOSI	SD2	DPI_D6	-	CTS4	SDA5
11	Low	SPI0_SCLK	SD3	DPI_D7	-	RTS4	SCL5
12	Low	PWM0	SD4	DPI_D8	SPI5_CE0_N	TXD5	SDA5
13	Low	PWM1	SD5	DPI_D9	SPI5_MISO	RXD5	SCL5
14	Low	TXD0	SD6	DPI_D10	SPI5_MOSI	CTS5	TXD1
15	Low	RXD0	SD7	DPI_D11	SPI5_SCLK	RTS5	RXD1
16	Low	FL0	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPI_D14	SPI6_CE0_N	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPI_D15	SPI6_MISO	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPI_D16	SPI6_MOSI	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPI_D17	SPI6_SCLK	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	SDA6
23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	SCL6
24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT0	ARM_TDO	SPI3_CE1_N
25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	SPI4_CE1_N
26	Low	SD0_DAT2	TE0	DPI_D22	SD1_DAT2	ARM_TDI	SPI5_CE1_N
27	Low	SD0_DAT3	TE1	DPI_D23	SD1_DAT3	ARM_TMS	SPI6_CE1_N

Table 5: Raspberry Pi 4 GPIO Alternate Functions

4. Datasheet Driver MDD 10A



Pin No.	Pin Name	Description
1	GND	Ground
2	PWM2	PWM input for speed control (Motor 2)
3	DIR2	Direction input (Motor 2)
4	PWM1	PWM input for speed control (Motor 1)
5	DIR1	Direction input (Motor 1)

PWM	DIR	Output A	Output B
Low	X(Don't care)	Low	Low
High	Low	High	Low
High	High	Low	High

Pin No	Pin Name	Description
1	Motor 1 Output B	Connect to motor 1 terminal B
2	Motor 1 Output A	Connect to motor 1 terminal A
3	POWER +	Positive Supply
4	POWER -	Negative Supply
5	Motor 2 Output A	Connect to motor 2 terminal A
6	Motor 2 Output B	Connect to motor 2 terminal B

5. Datasheet LM311

6.7 Switching Characteristics

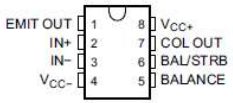
$V_{CC±}$ = ±15 V, T_A = 25°C

PARAMETER	TEST CONDITIONS	LM111, LM211, LM211Q, LM311	UNIT
		TYP	
Response time, low-to-high-level outputSee (1)	R_C = 500 Ω to 5 V, C_L = 5 pF, see (2)	115	ns
Response time, high-to-low-level outputSee (1)		165	ns

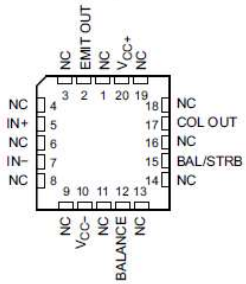
(1) The response time specified is for a 100-mV input step with 5-mV overdrive and is the interval between the input step function and the instant when the output crosses 1.4 V.
(2) The package thermal impedance is calculated in accordance with MIL-STD-883.

5 Pin Configuration and Functions

LMx11 D, JG, P, PS, or PW Package
8-Pin SOIC, CDIP, PDIP, SO or TSSOP
Top View



LM111 FK Package
20-Pin LCCC(1)
Top View



(1) NC = No internal connection

6.6 Electrical Characteristics

at specified free-air temperature, $V_{CC\pm}$ = ±15 V (unless otherwise noted)

PARAMETER		TEST CONDITIONS		T_A[1]	LM111, LM211, LM211Q			LM311			UNIT
					MIN	TYP[2]	MAX	MIN	TYP[2]	MAX	
V_{IO}	Input offset voltage	See[3]		25°C		0.7	3		2	7.5	mV
				Full range			4			10	
I_{IO}	Input offset current	See[3]		25°C		4	10		6	50	nA
				Full range			20			70	
I_{IB}	Input bias current	1 V ≤ V_O ≤ 14 V		25°C		75	100		100	250	nA
				Full range			150			300	
$I_{L(S)}$	Low-level strobe current[4]	$V_{(strobe)}$ = 0.3 V, V_{ID} ≤ −10 mV		25°C		−3			−3		mA
V_{IOR}	Common-mode input-voltage range[3]	Lower range		Full range		−14.7	−14.5		−14.7	−14.5	V
		Upper range			13	13.8		13	13.8		
A_{VD}	Large-signal differential-voltage amplification	5 V ≤ V_O ≤ 35 V, R_L = 1 kΩ		25°C	40	200		40	200		V/mV
I_{OH}	High-level (collector) output leakage current	$I_{(strobe)}$ = −3 mA, V_{ID} = 5 mV	V_{OH} = 35 V	25°C		0.2	10				nA
				Full range			0.5				µA
		V_{ID} = 5 mV, V_{OH} = 35 V		25°C					0.2	50	nA
V_{OL}	Low-level (collector-to-emitter) output voltage	I_{OL} = 50 mA	V_{ID} = −5 mV	25°C		0.75	1.5				V
			V_{ID} = −10 mV	25°C					0.75	1.5	
		V_{CC+} = 4.5 V, V_{CC-} = 0 V, I_{OL} = 8 mA	V_{ID} = −6 mV	Full range		0.23	0.4				
			V_{ID} = −10 mV	Full range					0.23	0.4	
I_{CC+}	Supply current from V_{CC+} output low	V_{ID} = −10 mV,	No load	25°C		5.1	6		5.1	7.5	mA
I_{CC-}	Supply current from V_{CC-} output high	V_{ID} = 10 mV,	No load	25°C		−4.1	−5		−4.1	−5	mA

(1) Unless otherwise noted, all characteristics are measured with BALANCE and BAL/STRB open and EMIT OUT grounded. Full range for LM111 is –55°C to 125°C, for LM211 is –40°C to 85°C, for LM211Q is –40°C to 125°C, and for LM311 is 0°C to 70°C.

(2) All typical values are at T_A = 25°C.

(3) The offset voltages and offset currents given are the maximum values required to drive the collector output up to 14 V or down to 1 V with a pullup resistor of 7.5 kΩ to V_{CC+}. These parameters actually define an error band and take into account the worst-case effects of voltage gain and input impedance.

(4) The strobe must not be shorted to ground; it must be current driven at –3 mA to –5 mA (see Figure 18 and Figure 31).

6. Datasheet ICL 7660

The ICL7660 is a monolithic CMOS power supply circuit that offers unique performance advantages over previously available devices. The ICL7660 performs supply voltage conversions from positive to negative for an input range of +1.5V to +10.0V resulting in complementary output voltages of -1.5V to -10.0V. Only two noncritical external capacitors are needed for the charge pump and charge reservoir functions. The ICL7660 can also be connected to function as voltage doublers and can generate output voltages up to +18.6V with a +10V input.

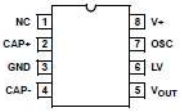
Contained on the chip are a series DC supply regulator, RC oscillator, voltage level translator, and four output power MOS switches. A unique logic element senses the most negative voltage in the device and ensures that the output N-Channel switch source-substrate junctions are not forward biased. This assures latchup free operation.

The oscillator, when unloaded, oscillates at a nominal frequency of 10kHz for an input supply voltage of 5.0V. This frequency can be lowered by the addition of an external capacitor to the OSC terminal, or the oscillator may be overdriven by an external clock.

The LV terminal may be tied to GROUND to bypass the internal series regulator and improve low voltage (LV) operation. At medium to high voltages of +3.5V to +10.0V, the LV pin is left floating to prevent device latchup.

Pinouts

ICL7660
(8 LD PDIP, SOIC)
TOP VIEW



Features

- Simple Conversion of +5V Logic Supply to ±5V Supplies
- Simple Voltage Multiplication (V_{OUT} = (-) nV_{IN})
- Typical Open Circuit Voltage Conversion Efficiency 99.9%
- Typical Power Efficiency 98%
- Wide Operating Voltage Range of 1.5V to 10.0V
- Easy to Use - Requires Only Two External Non-Critical Passive Components
- No External Diode Over Full Temperature and Voltage Range
- Pb-Free Plus Anneal Available (RoHS Compliant)

Applications

- On Board Negative Supply for Dynamic RAMs
- Localized µProcessor (8080 Type) Negative Supplies
- Inexpensive Negative Supplies
- Data Acquisition Systems

Related Literature

For a full list of related documents, visit our website:

- ICL7660 device page

Theoretical Power Efficiency Considerations

In theory a voltage converter can approach 100% efficiency if certain conditions are met.

1. The driver circuitry consumes minimal power.
2. The output switches have extremely low ON resistance and virtually no offset.
3. The impedances of the pump and reservoir capacitors are negligible at the pump frequency.

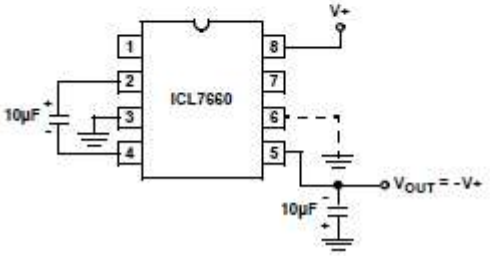


FIGURE 13A. CONFIGURATION

The ICL7660 approaches these conditions for negative voltage conversion if large values of C_1 and C_2 are used. **ENERGY IS LOST ONLY IN THE TRANSFER OF CHARGE BETWEEN CAPACITORS IF A CHANGE IN VOLTAGE OCCURS.** The energy lost is defined by:

$$E = {}^1/_2\, C_1 (V_1^2 - V_2^2)$$

where V_1 and V_2 are the voltages on C_1 during the pump and transfer cycles. If the impedances of C_1 and C_2 are relatively high at the pump frequency (refer to Figure 12) compared to the value of R_L, there will be a substantial difference in the voltages V_1 and V_2. Therefore it is not only desirable to make C_2 as large as possible to eliminate output voltage ripple, but also to employ a correspondingly large value for C_1 in order to achieve maximum efficiency of operation.

Do's and Don'ts

1. Do not exceed maximum supply voltages.
2. Do not connect LV terminal to GROUND for supply voltages greater than 3.5V.
3. Do not short circuit the output to V+ supply for supply voltages above 5.5V for extended periods, however, transient conditions including start-up are okay.
4. When using polarized capacitors, the + terminal of C_1 must be connected to pin 2 of the ICL7660, and the + terminal of C_2 must be connected to GROUND.
5. If the voltage supply driving the ICL7660 has a large source impedance (25Ω - 30Ω), then a 2.2µF capacitor from pin 8 to ground may be required to limit rate of rise of input voltage to less than 2V/µs.
6. User should insure that the output (pin 5) does not go more positive than GND (pin 3). Device latch up will occur under these conditions. A 1N914 or similar diode placed in parallel with C_2 will prevent the device from latching up under these conditions. (Anode pin 5, Cathode pin 3).

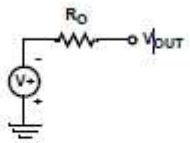


FIGURE 13B. THEVENIN EQUIVALENT

Absolute Maximum Ratings

Supply Voltage . . . . . +10.5V
LV and OSC Input Voltage . . . . . -0.3V to (V+ +0.3V) for V+ < 5.5V
(Note 6) . . . . . (V+ -5.5V) to (V+ +0.3V) for V+ > 5.5V
Current into LV (Note 6) . . . . . 20µA for V+ > 3.5V
Output Short Duration ($V_{SUPPLY} \leq 5.5V$) . . . . . Continuous

Thermal Information

Thermal Resistance (Typical, Note 4)	θ_{JA} (°C/W)	θ_{JC} (°C/W)
PDIP Package (Note 5)	110	N/A
SOIC Package	160	N/A

Maximum Storage Temperature Range . . . . . -65°C to 150°C
Pb-Free Reflow Profile (Note 5) TB493

Operating Conditions

Temperature Range . . . . . 0°C to 70°C

CAUTION: Do not operate at or near the maximum ratings listed for extended periods of time. Exposure to such conditions can adversely impact product reliability and result in failures not covered by warranty.

NOTES:

4. θ_{JA} is measured with the component mounted on an evaluation PC board in free air.
5. Pb-free PDIPs can be used for through hole wave solder processing only. They are not intended for use in Reflow solder processing applications.

Electrical Specifications

$V+ = 5V$, $T_A = 25°C$, $C_{OSC} = 0$, Test Circuit Note 7, Figure 11 on page 6 unless otherwise specified

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNITS
Supply Current	I+	$R_L = \infty$	-	170	500	µA
Supply Voltage Range - Lo	V_L+	$MIN \leq T_A \leq MAX$, $R_L = 10k\Omega$, LV to GND	1.5	-	3.5	V
Supply Voltage Range - Hi	V_H+	$MIN \leq T_A \leq MAX$, $R_L = 10k\Omega$, LV to Open	3.0	-	10.0	V
Output Source Resistance	R_{OUT}	$I_{OUT} = 20mA$, $T_A = 25°C$	-	55	100	Ω
		$I_{OUT} = 20mA$, $0°C \leq T_A \leq 70°C$	-	-	120	Ω
		$I_{OUT} = 20mA$, $-55°C \leq T_A \leq 125°C$	-	-	150	Ω
		$I_{OUT} = 20mA$, $-40°C \leq T_A \leq 85°C$	-	-	-	Ω
		$V^+ = 2V$, $I_{OUT} = 3mA$, LV to GND $0°C \leq T_A \leq 70°C$	-	-	300	Ω
		$V+ = 2V$, $I_{OUT} = 3mA$, LV to GND, $-55°C \leq T_A \leq 125°C$	-	-	400	Ω
Oscillator Frequency	f_{OSC}		-	10	-	kHz
Power Efficiency	P_{EF}	$R_L = 5k\Omega$	95	98	-	%
Voltage Conversion Efficiency	$V_{OUT\ EF}$	$R_L = \infty$	97	99.9	-	%
Oscillator Impedance	Z_{OSC}	V+ = 2V	-	1.0	-	MΩ
		V = 5V	-	100	-	kΩ

NOTES:

6. Connecting any input terminal to voltages greater than V+ or less than GND may cause destructive latchup. It is recommended that no inputs from sources operating from external supplies be applied prior to "power up" of the ICL7660.
7. In the test circuit, there is no external capacitor applied to pin 7. However, when the device is plugged into a test socket, there is usually a very small but finite stray capacitance present, of the order of 5pF.

BIODATA PENULIS

Rizky Reza Pahlevi, Lahir 25 Februari 1998 bertempat tinggal di desa Sumber Pucung, Malang Jawa Timur. Kesibukan di rumah adalah membantu orang tua berjualan komponen elektro. Teknik Elektro sudah menjadi tujuan sejak awal masuk SMA. Cita-cita yang ingin dicapai memiliki swalayan elektronika dengan cabang seluruh Indonesia. Pendidikan formal yang telah ditempuh mulai dari SDN 06 Sumber Pucung tahun 2004 - 2010, SMPN 4 Kepanjen 2010 - 2013, SMAN 1 Kepanjen 2013 – 2016. Kemudian berlanjut hingga sekarang jenjang kuliah di Teknik Elektro ITS. Prodi yang didalami adalah prodi teknik elektronika. Pada saat kuliah penulis aktif dalam unit kegiatan mahasiswa robotika dan asisten laboratorium elektronika.