AUTONOMOUS MOBILE ROBOT DENGAN METODE *SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM)* BERBASIS *LIDAR*

PROYEK AKHIR

Laporan akhir ini dibuat dan diajukan untuk memenuhi salah satu syarat kelulusan Sarjana Terapan Politeknik Manufaktur Negeri Bangka Belitung

Disusun Oleh :

Mashur Arbi Maulana	NIRM	1051811
Reynaldi Novian	NIRM	1051822

POLITEKNIK MANUFAKTUR NEGERI BANGKA BELITUNG

TAHUN 2022

LEMBAR PENGESAHAN

AUTONOMOUS MOBILE ROBOT DENGAN METODE *SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM)* BERBASIS *LIDAR*

Oleh:

Mashur Arbi Maulana	1051811
Reynaldi Novian	1051822

Laporan akhir ini telah disetujui dan disahkan sebagai salah satu syarat kelulusan Program Sarjana Terapan Politeknik Manufaktur Negeri Bangka Belitung

Menyetujui,

Pembimbing 1	Pembimbing 2
Muhammad Iqbal Nugraha, M.Eng	I Made Andik Setiawan, M.Eng, Ph.D

Penguji 1	Penguji 2
Ocsirendi, M.T	Aan Febrianysah, M.T

PERNYATAAN BUKAN PLAGIAT

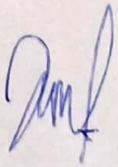
Yang bertanda tangan dibawah ini:

Nama Mahasiswa 1 : Mashur Arbi Maulana NIRM: 1051811
Nama Mahasiswa 2 : Reynaldi Novian NIRM: 1051822

Dengan Judul : *Autonomous Mobile Robot* Dengan *Metode Simultaneous Localization And Mapping (SLAM)* Berbasis *LIDAR*

Menyatakan bahwa laporan akhir ini adalah hasil kerja kami sendiri dan bukan merupakan plagiat. Pernyataan ini kami buat dengan sebenarnya dan bila ternyata dikemudian hari ternyata melanggar pernyataan ini, kami bersedia menerima sanksi yang berlaku.

Sungailiat, 13 November 2021

Nama Mahasiswa	Tanda Tangan
Mashur Arbi Maulana	
Reynaldi Novian	

ABSTRAK

Robot mobil otonom adalah sebuah robot yang memiliki kemampuan untuk mengenali lingkungan serta dapat melokalisasikan posisinya di lingkungan tersebut. Robot mobil otonom sangat diperlukan untuk bekerja di lingkungan yang berbahaya atau beresiko tinggi untuk manusia seperti bangunan yang sudah lama tidak terpakai. Untuk dapat bergerak otonom, maka robot harus dapat mengenali lingkungan sekitarnya dan juga dapat mengetahui lokasinya pada lingkungan tersebut. Pengenalan lingkungan ini berbentuk pemetaan dua dimensi (2D) dan lokalisasi atau pemosisian robot dari peta tersebut. Metode pemetaan dan lokalisasi yang digunakan adalah Simultaneous Localization and Mapping (SLAM) dengan bantuan sensor jarak yaitu Light Detection and Ranging (LIDAR) dan menggunakan Robot Operating System (ROS). Dari hasil pengujian, robot mobil mampu melakukan pemetaan dan menyampaikan informasi posisi robot pada lingkungan dengan persentase error pada saat pemetaan sebesar 0,20% dan lokalisasi sebesar 1,52%. Untuk error pembacaan jarak dan sudut pada sensor adalah 0,77%. Secara keseluruhan robot mobil ini dapat diterapkan pada area ruangan kosong dengan halangan sederhana.

Kata Kunci: LIDAR, Robot, ROS, SLAM

ABSTRACT

Autonomous mobile robot is a robot that has the ability to recognize the environtment and can localize its position in that environtment. Which is important for working in dangerous area. To move independently, the robot must be able to identify its environment and know its position in the form of two-dimensional map. This research used Simultaneous Localization And Mapping (SLAM) method combined with of range sensor Light Detection And Raning (LIDAR) and use Robot Operating System(ROS) for identification the evironment. As a results, Autonomous robot is capable of Mapping and delivering information of the robot can map and localize its position in an environment with Mapping error 0,20% and 1,52% Localization. Distance measurement in various angle tendts to stable with mean error 0,77%. Overall, the robot can be used in an empty area or an area with simple obstacle.

Keywords: LIDAR, Robot, ROS, SLAM

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh. Segala puji bagi Allah SWT., yang telah melimpahkan taufiq dan hidayah-Nya, sehingga dapat menyusun laporan proyek akhir ini dengan Judul "*Autonomous Mobile Robot* Dengan Metode *Simultaneous Localization And Mapping* (*SLAM*) Berbasis *LIDAR*" dan dapat menyelesaikan Program Studi Diploma IV Teknik Elektronika di Politeknik Manufaktur Negeri Bangka Belitung. Sholawat serta salam semoga selalu tercurahkan kepada Nabi Muhammad SAW beserta keluarga dan sahabat-sahabatnya, serta semoga semua umatnya senantiasa dapat menjalankan syari'at-syari'atnya, dan mendapatkan syafaat di hari akhir.

Dalam penyusunan laporan proyek akhir ini banyak terdapat kekurangan mengingat terbatasnya kemampuan yang dimiliki, namun berkat rahmat Allah SWT, serta pengarahan dari berbagai pihak, akhirnya laporan proyek akhir ini dapat diselesaikan. Semoga laporan proyek akhir ini dapat bermanfaat untuk kepentingan bersama. Untuk itu disampaikan ucapan terima kasih kepada:

1. Ibu dan Ayah tercinta serta seluruh keluarga yang dengan penuh keikhlasan, dukungan, dan kesungguhan hati memberikan bantuan moral dan spiritual yang tak ternilai harganya.
2. Bapak I Made Andik Setiawan, M.Eng, Ph.D Selaku Direktur di Politeknik Manufaktur Negeri Bangka Belitung yang telah banyak memberikan kemudahan dalam menyelesaikan pendidikan.
3. Bapak Muhammad Iqbal Nugraha, M.Eng dan bapak I Made Andik Setiawan, M.Eng, Ph.D Selaku dosen pembimbing dan Ibu Nofriyani, M.Tr.T yang telah membantu mengarahkan dan memberi saran-saran dalam pembuatan dan penyusunan laporan proyek akhir ini.
4. Dosen dan Staf Pengajar di Politeknik Manufaktur Negeri Bangka Belitung yang telah mendidik, membina dan mengantarkan penulis untuk menempuh kematangan dalam berfikir dan berperilaku.

5. Teman-teman seperjuangan dan semua pihak yang telah memberikan bantuannya.
6. Sahabat - sahabat yang selalu memberikan dukungan selama ini dan mitra kerja selama mengerjakan proyek akhir ini yang selalu berjuang bersama-sama.

Setelah melalui proses yang panjang dan penuh tantangan, pembuatan alat dan laporan proyek akhir ini dapat terselesaikan meskipun masih banyak kekurangan dan jauh dari kesempurnaan. Diharapkan laporan proyek akhir ini dapat bermanfaat bagi kita semua dan penulis khususnya. Semoga Allah senatiasa melimpahkan taufiq dan hidayah-Nya kepada penulis dan semua pihak yang telah membantu dalam pembuatan alat dan penulisan laporan proyek ini, Wassalamu'alaikum Warahmatullahi Wabarakatuh.

Sungailiat, 13 November 2021

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Ilmu pengetahuan dan teknologi robotika telah berkembang dengan sangat pesat. Pengaruh dari perkembangan ini, robot yang sebelumnya bergerak secara manual kini sudah bisa bergerak secara otomatis (*Autonomous*) (Taufik, 2013). Salah satu jenis pergerakan dari autonomous robot yaitu, *autonomous mobile robot*. *Autonomous mobile robot* adalah sebuah kemampuan yang dimiliki oleh sebuah robot untuk berjalan secara mandiri tanpa bantuan dari *remote control* (manual). Untuk mendapatkan pergerakan *autonomous, mobile robot* harus mampu mengenali lingkungan yang ada di sekitarnya dan juga harus mengetahui posisinya dilingkungan tersebut (pemetaan & lokalisasi) dengan bantuan sistem navigasi (Rahman, 2020). *Autonomous mobile robot* ini dapat dimanfaatkan untuk melakukan pekerjaan di lokasi - lokasi yang dianggap rawan terhadap gangguan keamanan. Sehingga manusia tidak perlu lagi untuk turun langsung dalam melakukan pemetaan di tempat yang berbahaya (Prayoga et al., 2017).

Salah satu metode navigasi yang paling sederhana adalah dengan cara mendeteksi jarak sebuah benda terhadap penghalang disekitar robot menggunakan sensor *Ultrasonic*. Namun sensor *ultrasonic* yang diterapkan pada *mobile robot* ini memiliki keterbatasan, seperti pembacaan jaraknya yang pendek dan rendahnya akurasi pembacaan terhadap sudut (Zahra et al., 2018). Oleh karena itu, untuk menutupi keterbatasan pembacaan jarak dan sudut digunakanlah sebuah sensor jarak yaitu *Light Detection and Ranging (LIDAR).*

Tujuan pembuatan *mobile robot* ini, robot mampu menggantikan posisi manusia untuk melakukan pemetaan dan lokalisasi di tempat – tempat yang berbahaya seperti sebuah bangunan yang sudah lama tidak terpakai. Pembuatan *mobile robot* ini menggunakan metode *Hector SLAM* dibantu dengan sensor *RPLidar A1M8* dan *Robot Operating System (ROS).*

Simultaneous Localizaton And Mapping (SLAM) adalah suatu cara yang dapat diterapkan pada *Autonomous mobile robot*, dimana metode ini dapat membantu robot untuk membuat suatu peta dan pada saat yang bersamaan mengindikasikan posisinya pada peta yang dibuat (Prayoga et al., 2010).

Untuk membantu menggunakan metode *SLAM* dibutuhkan sebuah sistem pengolah data sensor yang digunakan untuk mengendalikan *Autonomous mobile robot*, yaitu *Robot Operating System (ROS).*

1.2 Rumusan Masalah

Adapun rumusan masalah pada proyek akhir ini adalah sebagai berikut :

1. Bagaimana merancang *mobile robot* dengan metode *Hector SLAM* berbasis *RPLidar A1M8.*
2. Bagaimana membuat *mobile robot* yang dapat membuat sebuah peta dan sekaligus dapat menampilkan posisinya secara *real-time* pada peta yang dibuat.

1.3 Batasan Masalah

Adapun batasan masalah dalam proyek akhir ini adalah :

1. Robot ini hanya dapat membuat peta 2D menggunakan *Robot Operating System (ROS).*
2. Robot ini hanya bisa beroperasi di dalam ruangan (indoor).
3. Robot tidak bisa melewati rintangan yang tingginya kurang dari posisi sensor *RPLidar A1M8* yaitu dibawah 18 cm.
4. Robot ini tidak bisa mendeteksi permukaan berbahan kaca.

1.4 Tujuan Proyek Akhir

Tujuan dari pembuatan *Autonomous Mobile Robot* dengan Metode *Simultaneous Localization And Mapping (SLAM)* Berbasis *LIDAR* ini adalah :

1. Dapat membuat *mobile robot* otomatis yang dapat membuat peta dari suatu lokasi tertentu yang belum diketahuinya menggunakan metode *Hector SLAM* berbasis *RPLidar A1M8*.
2. Dapat membuat *mobile robot* yang dapat berjalan secara otomatis serta dapat menghindari halangan berdasarkan data jarak pada sudut 0° hingga 360° dari sensor *RPLidar A1M8*.

BAB II

DASAR TEORI

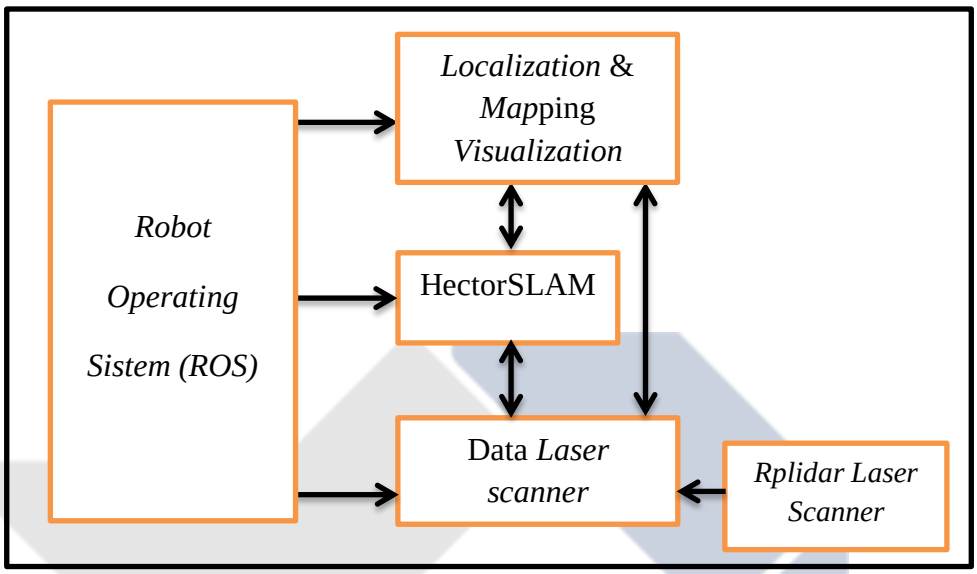
2.1. *Simultaneous Localization And Mapping (SLAM)*

SLAM (Simultaneous Localization And Mapping) adalah suatu metode yang digunakan sebuah robot untuk membuat suatu peta sekaligus bisa melokalisasikan dirinya di peta yang dibuatnya. Metode ini memungkinkan robot untuk mengerjakan dua tugas secara bersamaan. yaitu memetakan lokasi serta mengetahui posisinya di dalam peta yang dibuatnya (Prayoga et al., 2010). Metode *SLAM* ini banyak sekali digunakan pada pengembangan *mobile robot* dengan kasus pembacaan suatu tempat yang memiliki bidang yang rata. Agar dapat membaca dengan baik, robot harus dilengkapi dengan bermacam sensor pendeteksi sebagai indra pembacanya. Bisa memakai *LIDAR* maupun *SONAR* (Sidharta, 2019). Terdapat berbagai cara yang bisa digunakan untuk mengaplikasikan *algoritma SLAM* secara 2D, yaitu *Hector SLAM, GMapping, Karto SLAM, Core SLAM,* dan *Lago SLAM*. *Algoritma SLAM* dapat dikelompokan menjadi 3 bagian, yang pertama berdasarkan sensor yang digunakan untuk mengukur jarak, bisa menggunakan laser, *ultrasound* atau *odometry*, kedua berdasarkan metode perhitungan, bisa menggunakan *algoritma kalman filter* atau *partikel filter*, dan yang ketiga berdasarkan strukturnya (Utomo, 2015). *Algoritma SLAM* ini semuanya terdapat pada *package Robot Operating System* (Rahman, 2020).

2.1.1 *Hector SLAM*

Hector SLAM adalah sebuah *algoritma SLAM* yang proses pembuatan peta dan lokalisasi tanpa harus adanya data dari *odometry*. *Hector SLAM* menggunakan data dari sensor *LIDAR* untuk menjalankan *algoritma scan matching*. *Scan matching* ini adalah suatu proses menyamakan hasil pembacaaan pengukuran jarak dari hasil pembacaan waktu t – 1 pada sensor *LIDAR* yang hasilnya digunakan untuk mendapatkan posisi robot unuk memperbarui peta yang akan

dibuat. Pada saat proses *scan matching* dilakukan, sistem akan memperbarui peta sesuai dengan hasil pembacaan sensor *LIDAR*.Dibawah ini merupakan gambar dari blok diagram sistem *Hector SLAM* (Saat et al., 2020)



Gambar 2. 1 Blok Diagram Sistem *SLAM*

2.1.2 *GMapping*

Algoritma SLAM GMapping adalah sebuah *algoritma* yang berbasis pada *partikel filter* dari Rao-Blackwelized, yang mengusulkan probabilitas untuk mempertimbangkan tindakan terakhir yang diambil oleh perangkat laser dan *odometry*. Ini dilakukan dengan mencari jalan yang lebih dekat ke lokasi yang diperkirakan, dengan cara mendefinisikan probabilitas setiap *landmark* dengan cara mengukur dan menambahkan informasi *odometry* (Lemus et al., 2014). Metode *SLAM GMapping* mirip dengan metode *Hector SLAM* hanya saja metode *Hector SLAM* tidak menggunakan data *odometry*. *SLAM GMapping* membutuhkan daya yang relatif lebih ringan dibandingkan dengan metode *SLAM* lainnya. Metode *SLAM GMapping* ini membutuhkan data dari sensor *LIDAR* dan data *odometry* untu melakukan pemetaan dan juga lokalisasi (Rahman, 2020).

2.1.3 *CEKF-SLAM*

Algoritma CEKF-SLAM sebuah *algoritma* yang berdasar pada *Kalman filter* untuk proses pembuatan *map* nya. *Map* yang dihasilkan berupa *feaures based maps*. *Algoritma CEKF-SLAM* juga mengoptimalkan sebuah filter terkompresi untuk menunda pembaruan kovarians, ini bertujuan untuk meningkatkan kepresisian tanpa mengurangi akurasi dari proses pembuatan *map*. *Algoritma CEKF-SLAM* ini menggunakkan sensor ultrasound dan juga *odometry* (Lemus et al., 2014).

2.1.4 *DP-SLAM*

Algoritma DP-SLAM adalah sebuah *algoritma* yang berdasar pada *partikel filter*. *Algoritma* ini menghasilkan peta *grid*. Tujuannya adalah untuk mengurangi penggunaan sumber daya komputer agar menghindari salinan peta berturut – turut pada tahap *resampling*. *Algoritma* ini memungkinkan mengetahui perubahan setiap partikel yang dibuat sebelumnya, hal ini dapat mengurangi waktu untuk menyalin data ketika partikel baru dibuat. *Algoritma DP-SLAM* ini menggunakan sensor *odometry laser* untuk proses pembuatan *map* (Lemus et al., 2014).

2.1.5 *EKFM-SLAM*

Algoritma EKFM-SLAM hampir sama dengan *algoritma CEKF-SLAM*, kedua metode ini berdasar pada *Kalman filter*, namun yang membedakan *algoritma* ini adalah pada *EKFM-SLAM* menambahkan sensor monocular camera pada proses pemuatan *map*. Peta yang dihasilkan pun berbeda, pada *EKFM-SLAM* menghasilkan peta grids (Lemus et al., 2014).

2.1.6 *Tiny SLAM*

Algoritma Tiny SLAM adalah sebuah *algoritma* yang berdasar pada *partikel filter* dengan pemindaian resolusi tinggi pada *grid map*. Pemindaian laser memperbarui lebih dari 1 *landmark* pada permukaan. *Algoritma Tiny SLAM* ini menggunakan sensor laser dan *odometry* untuk proses pembuatan peta (Lemus et al., 2014).

2.2. *Robot Operating System (ROS)*

Robot Operating System (ROS) merupakan sebuah *middleware* yang digunakan untuk menghubungkan perangkat keras robot dengan sistem operasi komputer secara fleksibel. Di dalam *ROS* juga memuat *tools* dan *library* yang bisa digunakan untuk mengembangkan berbagai macam program untuk sistem robot (Rahman, 2020). Tujuan penggunaan *ROS* juga untuk memudahkan para pengembang robot dalam membuat sistem robot yang di inginkan tanpa harus membuat pengkodean dari awal serta dapat mengembangkan kode sumber bersama – sama (Jalil, 2018). *ROS* dapat dibagi menjadi 3 bagian yaitu *level file systems*, level grafik komputasi dan level komunitas. Di dalam *level file systems ROS* memuat *packages, metapackages, packages manifests, metapackage manifests, message dan service*. Di dalam level grafik komputasi memuat level *nodes, master, parameter server, message, topic, services, dan bags*. Dan di dalam level komunitas ini juga bisa memisahan *resources ROS* agar dapat saling bertukar pengetahuan dan perangkat lunak. *Resources* ini terdiri dari distribusi *ROS, repository, ROS wiki, bug ticket system, maling list, ROS answer and blog* (Jalil, 2019). Di dalam *ROS* juga terdapat *package RViz* yang dapat menampilkan hasil peta 2D. *RViz (ROS Visualization)* merupakan suatu *package* yang terdapat di dalam *ROS* yang dapat digunakan sebagai media visualisasi 2D atau 3D untuk memvisualisasikan robot dan data sensor yang telah di dapatkan dari sensor *LIDAR.*

2.3. *Light Detection And Ranging (LIDAR)*

Light detection and ranging (LIDAR) adalah suatu sensor yang memanfaatkan pantulan sinar laser untuk mengukur objek yang ada disekitarnya. Cara kerja sensor *LIDAR* ini cukup mudah dipahami yaitu dengan melakukan perhitungan jarak terhadap bidang datar yang berada di sekitarnya dengan cara mengeluarkan sinar laser dari transmitter ke sebuah bidang datar kemudian menghitung berapa lama waktu yang dibutuhkan untuk sinar itu diterima oleh *receptor* (Prayoga et al., 2010). Hasil dari sensor *LIDAR* berupa jarak pada sudut. Untuk mengukur berapa lama waktu yang dibutuhkan sensor untuk mengukur

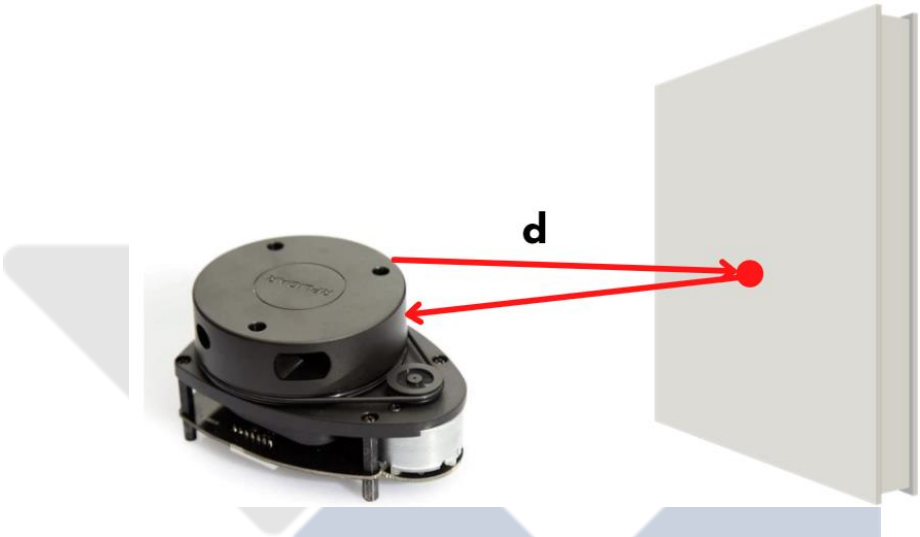
jarak antara sensor dengan bidang datar dapat dicari menggunakan rumus (Putra, 2016).:

$$d = c\frac{t}{2} \qquad (2.1)$$

Ket : d : Jarak antara sensor dan objek yang diukur (m)

c : Kecepatan cahaya ($3x10^8\ m/s$)

t : Waktu tempuh sinyal (s)



Gambar 2. 2 Sistem *RPLidar A1M8*(Prayoga et al., 2017)

Spesifikasi dari *RPLidar A1M8* ditunjukan pada tabel dibawah.

Tabel 2. 1 Spesifikasi *RPLidar A1M8*(Slamtec.com, n.d.)

Parameter	*Specification*
Measuring Range	*0.15m - 12m*
Sampling Frequency	*8K*
Rotational Speed	*5.5Hz*
Angular Resolution	*≤1°*
Dimensions	*96.8 x 70.3 x 55mm*
System Voltage	*5V*
System Current	*100mA*
Power Consumption	*0.5W*
Output	*UART Serial (3.3 voltage level)*

Temperature Range	*0℃-40℃*
Angular Range	*360°*
Range Resolution	*≤1% of the range （≤12m)*
	≤2% of the range （12m～16m)
Accuracy	*1% of the range （≤3 m）*
	2% of the range （3-5 m）
	2.5% of the range （5-25m）

2.4. Penelitian Sebelumnya

Penelitian mengenai pemanfaatan *Hector SLAM* untuk pembuatan *Mapping* serta *Localization* sudah banyak di lakukan sebelumnya, seperti penelitian yang dilakukan oleh (Sartika et al., 2015), mengenai implementasi *Hector SLAM* pada robot pencari korban gempa. Dimana pada penelitiannya ini dia menggunakan *algoritma Hector SLAM* dan sensor *Hokuyo URG04-LX 2D* laser scanner untuk proses *Mapping* serta *Localization.* Penggunaan *Hector SLAM* ini karena robot yang dibuat hanya mengandalkan pembacaan dari sensor *Hokuyo URG04-LX* tanpa data *odometry*. Dimana hasil dari penelitian ini robot mampu melakukan *Mapping* dan *Localization* sesuai dengan area yang sebenarnya,

Selanjutnya penelitian yang dilakukan oleh (Rahman, 2020), yang membahas tentang Penerapan *SLAM GMapping* dengan *Robot Operating System* Menggunakan Laser Scanner Pada *Turtlebot*. Dimana pada penelitiannya menggunakan *algoritma SLAM GMapping* dan menggunakan sensor *LIDAR* dibantu dengan data *odometry*. Hasil dari penelitian ini robot mampu membuat peta pada *Occupancy grid* 2D menggunakan *algoritma SLAM GMapping* menggunakan laser scanner.

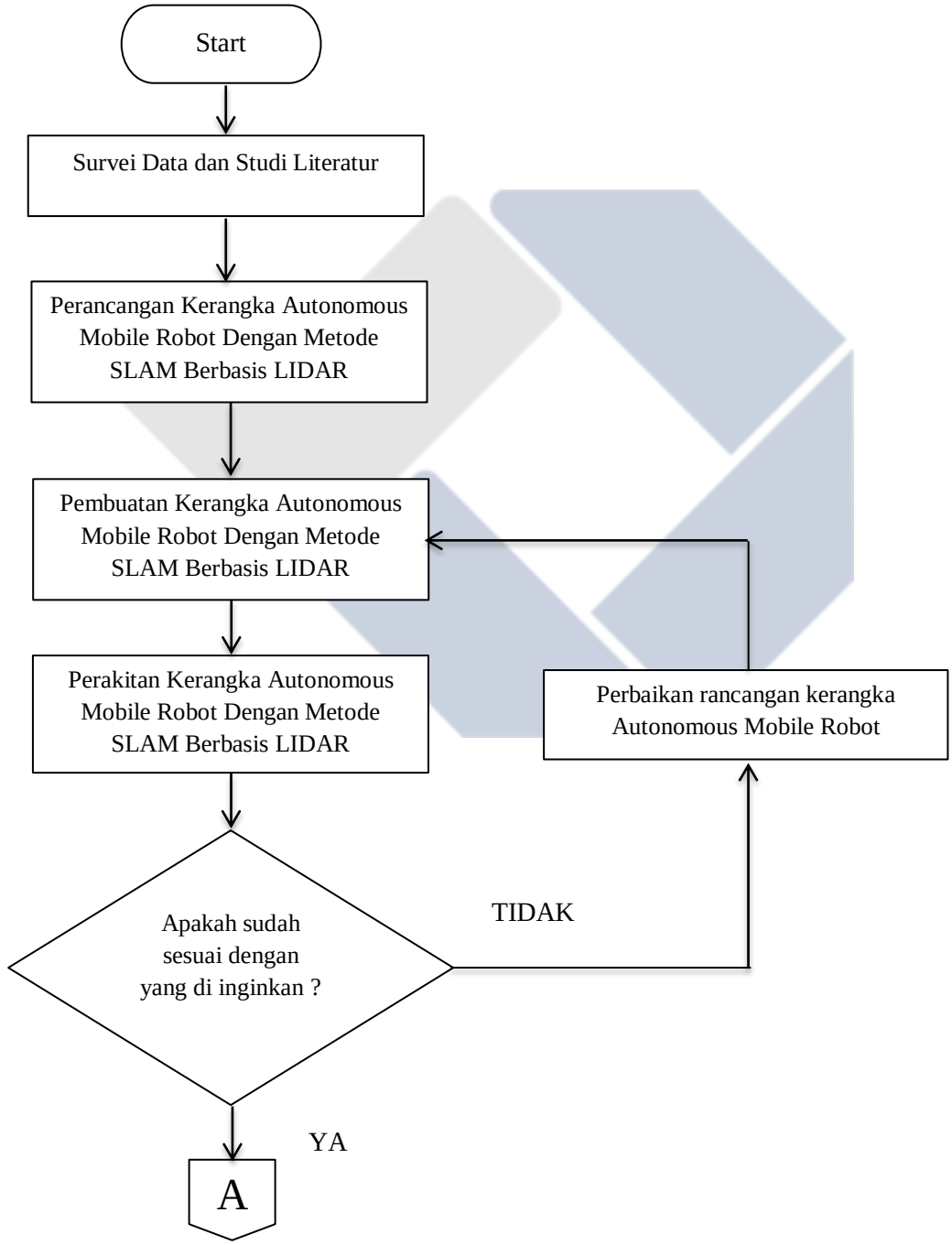
Berdasarkan pembahasan diatas metode *Hector SLAM* lebih mudah di implementasikan dibandingkan dengan metode *SLAM GMapping* karena metode *Hector SLAM* dapat melakukan pemetaan dan lokalisasi hanya dengan

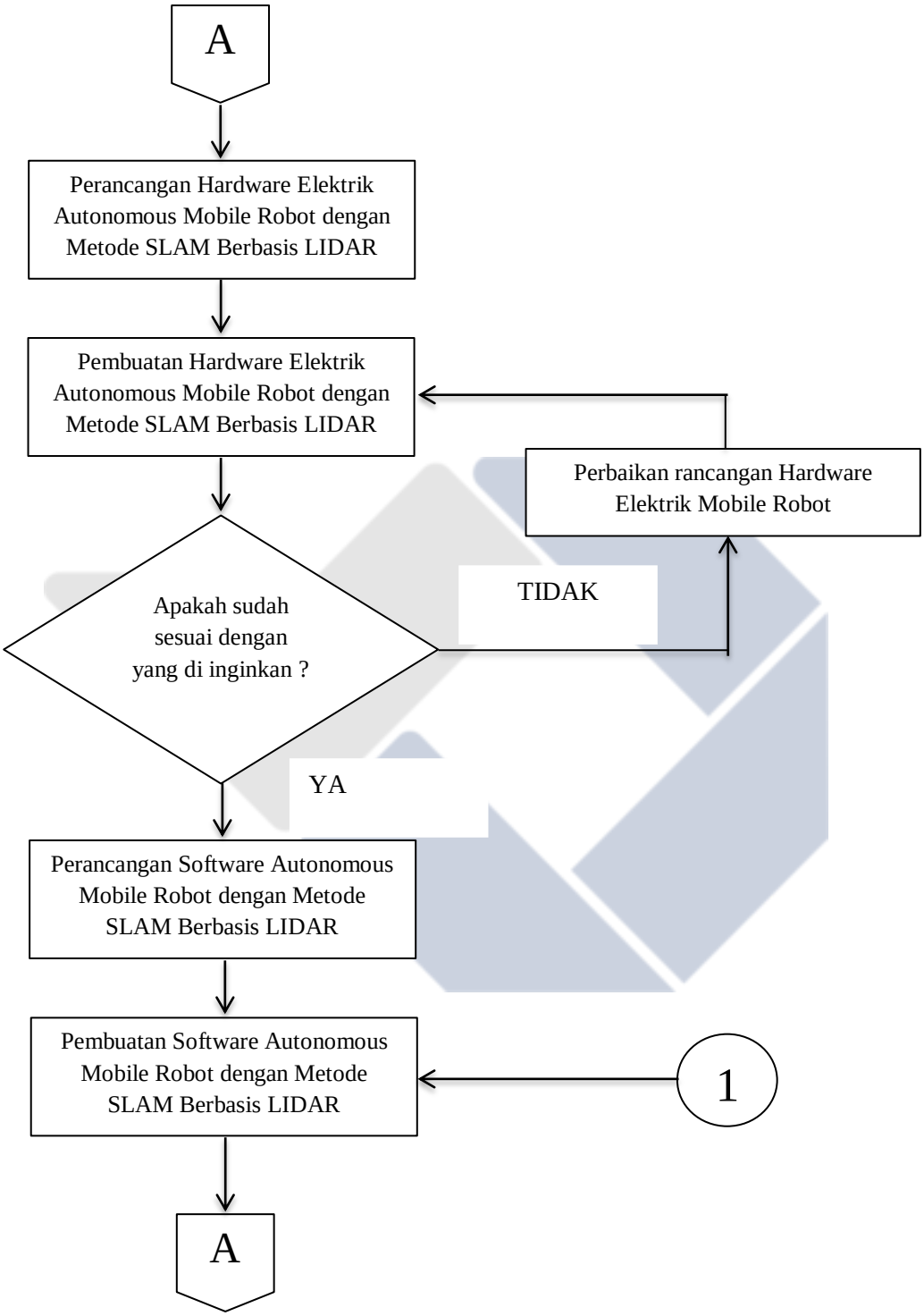
menggunakan data dari sensor *LIDAR* tanpa harus menggunakan data *odometry* seperti metode *SLAM GMapping*. Untuk penggunaan sensor pada *mobile robot*, sensor *Hokuyo URG04-LX* sangat baik diterapkan untuk proses pemetaan dan lokalisasi serta autonomous movement pada *mobile robot* dibandingkan dengan sensor *LIDAR* karena jarak minimal pengukuran pada sensor *Hokuyo URG04-LX* mencapai rentang 0,02 m sedangkan pada *LIDAR* jarak minimal yang yang dicapai 0,15 m . Namun sensor *Hokuyo URG04-LX* hanya bisa mendeteksi hingga sudut 240° dibandingkan sensor *LIDAR* yang mampu mendeteksi hingga 360°. Selain itu sensor *Hokuyo URG04-LX* memiliki harga yang relatif mahal ketimbang sensor *LIDAR*. Oleh karena itu pada penelitian ini *mobile robot* yang dibuat akan menggabungkan metode *Hector SLAM* dibantu dengan sensor *LIDAR* dengan harapan *mobile robot* dapat melakukan pemetaan dan lokalisasi serta dapat menghindari rintangan yang diberikan.

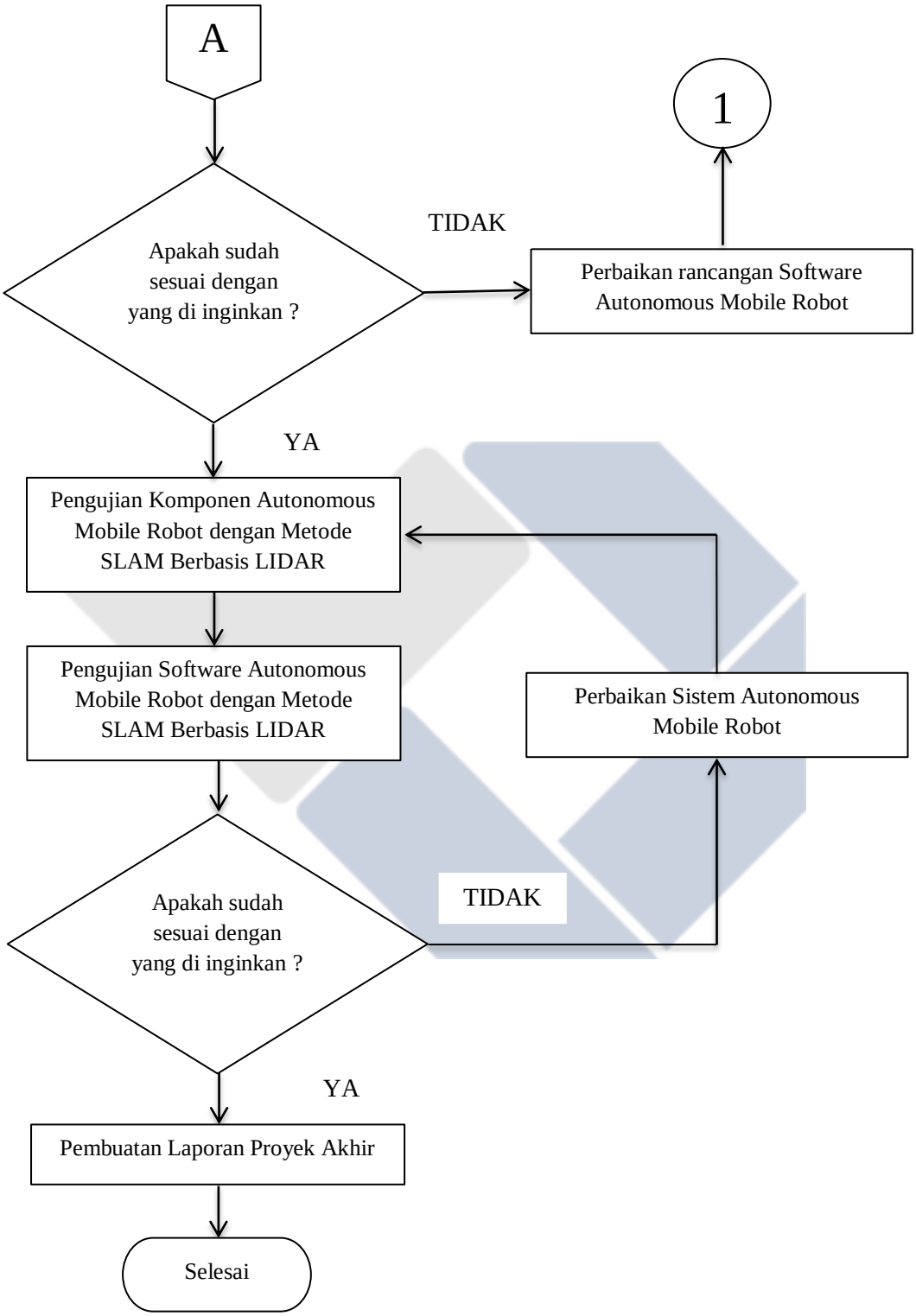
BAB III

METODE PELAKSANAAN

Dalam pelaksanaan proyek akhir ini, dilakukan tahapan proyek akhir yang bertujuan untuk mempermudah dalam proses pembuatan proyek akhir.




A
Perancangan Hardware Elektrik Autonomous Mobile Robot dengan Metode SLAM Berbasis LIDAR
Pembuatan Hardware Elektrik Autonomous Mobile Robot dengan Metode SLAM Berbasis LIDAR
Perbaikan rancangan Hardware Elektrik Mobile Robot
Apakah sudah sesuai dengan yang di inginkan ?
TIDAK
YA
Perancangan Software Autonomous Mobile Robot dengan Metode SLAM Berbasis LIDAR
Pembuatan Software Autonomous Mobile Robot dengan Metode SLAM Berbasis LIDAR
1
A



Gambar 3. 1 *Flowchart* Tahapan Pembuatan *Autonomous Mobile Robot* Dengan Metode *SLAM* Berbasis *LIDAR*

3.1 Survei Data dan Studi Literature

Survei data, merupakan proses pengumpulan data dari berbagai sumber. Yang dibutuhkan dalam proses pengerjaan proyek akhir ataupun penyusunan makalah proyek akhir. Pada tahap ini meliputi beberapa aspek dalam pengumpulan data secara langsung yaitu (data primer) dan pengumpulan data secara tidak langsung (data sekunder). Pengumpulan data secara langsung (data primer), data yang diperoleh dari hasil konsultasi dengan dosen pembimbing. Sedangkan pengumpulan data secara tidak langsung (data sekunder). Yaitu data yang didapatkan dari referensi-referensi buku dan jurnal yang masih berkaitan dengan proyek akhir

Studi Literature dilakukan dengan mencari referensi yang bersumber baik dari buku dan jurnal yang diambil dari internet. Referensi dibutuhkan untuk mengetahui perkembangan *autonomous mobile robot* dengan metode *SLAM* berbasis *LIDAR* yang sudah ada saat ini. Studi literature dilakukan untuk mendapatkan rumusan masalah, kebutuhan komponen dan peralatan yang akan digunakan di dalam perancangan *hardware* dan *software.*

3.2 Tahap Perencanaan Alat

3.2.1 Perancangan *Autonomous Mobile* Robot dengan Metode SLAM Berbasis *LIDAR*

Perancangan *autonomous mobile* robot pada proyek akhir ini dilakukan untuk membuat bagian bagian pada *mobile* robot, yang meliputi kerangka *base* robot, penempatan posisi sensor dan penentuan dimensi/ukuran material yang akan digunakan pada *mobile* robot.

3.2.2 Pembuatan Kerangka *Mobile Robot*

Pembuatan kerangka *mobile* robot ini dilakukan diluar area Polman Babel dengan membuat konstruksi *base mobile* robot. Dimulai dengan pembuatan *base mobile* robot sebagai media penghubung Motor DC dengan roda *mobile* robot,

pembuatan *base mobile* robot sebagai dudukan sensor, dan *hardware* yang akan digunakan pada *mobile* robot.

3.2.3 Perakitan Kerangka *Mobile* Robot

Perakitan *Autonomous Mobile* Robot dilakukan dengan cara merakit setiap bagian dari *mobile* robot yang telah dirancang. Mulai dari pemasangan *base* robot, pemasangan Motor DC pada *base* robot, pemasangan roda pada *base* robot yang akan digunakan sebagai pergerakan robot.

3.2.4 Perancangan *Hardware* Elektrik *Mobile* Robot

Proses perancangan *hardware autonomous mobile robot* menggunakan metode *SLAM* berbasis *LIDAR*. Menggunakan beberapa komponen elektrik seperti *Power Bank*, sensor *LIDAR*, Motor DC, *Raspberry pi 4*, dan *Motor Driver*, perancangan *hardware* elektrik ini menggunakan *Software FRITZING.*

3.2.5 Perakitan *Hardware* Elektrik *Mobile* Robot

Proses perakitan *hardware elektrik autonomous mobile robot* menggunakan metode *SLAM* berbasis *LIDAR* dilakukan di luar area Politeknik Manufaktur Negeri Bangka Belitung. Dengan merakit komponen-komponen sesuai dengan rancangan *hardware* elektrik yang sudah dirancang.

3.2.6 Pengujian Komponen *Mobile* Robot

Pengujian komponen elektrik yang dilakukan untuk mengetahui komponen-komponen yang akan digunakan bekerja sesuai dengan fungsinya. Berikut uji coba komponen-komponen elektrik:

1. Uji coba *Raspberry Pi 4*
2. Uji coba *RPLidar A1M8*
3. Uji coba Motor DC

3.2.7 Pembuatan *Package Autonomous Mobile Robot* dengan *Metode SLAM* Berbasis *LIDAR*

Pembuatan *package* yang akan dilakukan meliputi :

1. Pembuatan sistem kerja robot pada *ROS* untuk pergerakan *Autonomous* berdasarkan data dari sensor *RPLidar A1M8*
2. Pembuatan *Mapping* dan *Localization* pada *ROS* menggunakan *Hector SLAM*

3.2.8 Pengujian *Package* Mobile robot

Pengujian ini dilakukan untuk mengetahui hasil akhir dari *Mobile* robot yang telah dibuat, apakah telah bekerja sesuai dengan yang diinginkan atau belum.

1. Pengujian *Autonomous Movement* pada *mobile* robot.
2. Pengujian *Hector SLAM* untuk *Mapping* dan *Localization* pada *mobile* robot.

3.2.9 Pembuatan Laporan Proyek Akhir

Tahap ini merupakan tahap terakhir dalam pembuatan proyek akhir yang bertujuan untuk merangkum keseluruhan detail mengenai alat yang dibuat, yang meliputi latar belakang, tujuan, rumusan masalah, batasan masalah, landasan teori, metode pelaksanaan, pembahasan, serta kesimpulan dan saran.

BAB IV

PEMBAHASAN

Bab 4 ini menguraikan proses pengerjaan proyek akhir berdasarkan metode yang telah disampaikan pada bab sebelumnya. Secara umum bab ini menguraikan tentang:

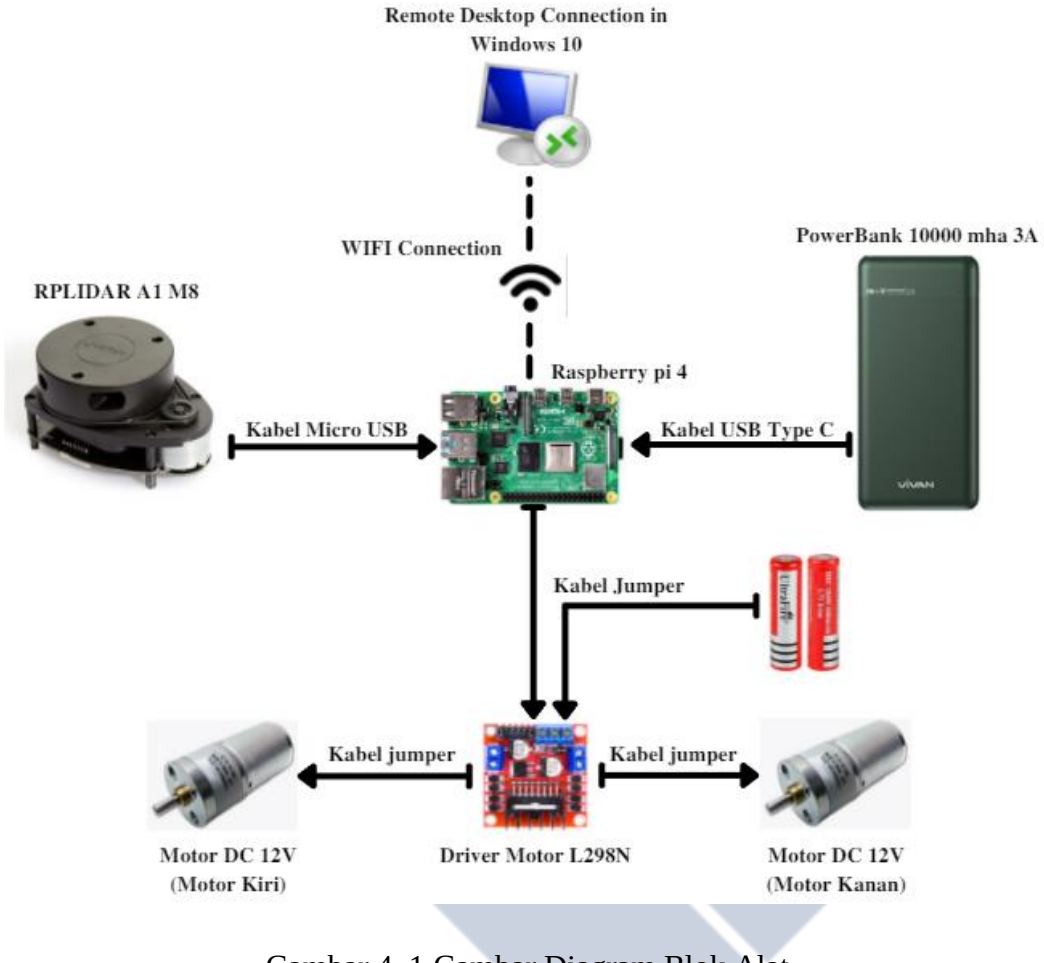
1. Deskripsi Alat
2. Diagram Blok Alat
3. Perancangan Kerangka *Mobile robot*
4. Pembuatan Kerangka *Mobile robot*
5. Perakitan Kerangka *Mobile robot*
6. Perancangan *Hardware Elektrik Mobile robot*
7. Perakitan *Hardware Elektrik Mobile robot*
8. Uji Coba *Hardware Elektrik Mobile robot*
9. Pembuatan *Package Mobile robot*
10. Uji Coba *Package Mobile robot*
11. Uji Coba Keseluruhan

4.1 Deskripsi Alat

Autonomous Mobile Robot Dengan Metode *Simultaneous Localization and Mapping berbasis LIDAR* adalah sebuah robot yang digunakan untuk membuat sebuah peta di sebuah tempat yang belum diketahuinya dan sekaligus dapat melokalisasikan posisinya pada peta yang telah dibuat. Proses pemetaan dan lokalisasi ini menggunakan metode *Hector SLAM* dibantu dengan sensor *RPLidar A1M8* dan *Robot Operating Sistem (ROS).* Agar dapat melakukan pemetaan diseluruh arena robot ini menggunakan *Autonomous Movement* dengan metode navigasi sederhana yaitu memanfatkan data berupa jarak pada sudut 0° hingga 360° yang dikirimkan oleh sensor *RPLidar A1M8*

4.2 Diagram Blok Alat

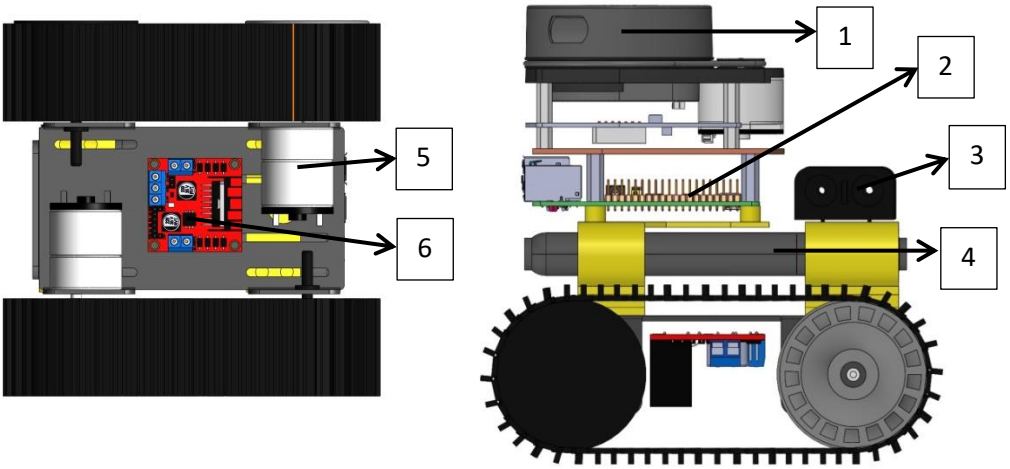
Sambungan komponen *mobile robot* ditunjukan oleh Gambar 4.1



Gambar 4. 1 Gambar Diagram Blok Alat

4.3 Perancangan Kerangka *Mobile robot*

Pada tahap perancangan kerangka *mobile robot* ini, *mobile robot* dirancang sesuai dengan konsep yang telah dibuat. Untuk penggerak mobile robot ini menggunakan *wheel tank*, ini dimaksudkan agar robot dapat di fungsikan di medan dengan permukaan rata maupun tidak rata sehingga bisa digunakan di segala jenis medan. Rancangan kerangka *mobile robot* ini dibuat menggunakan software *autodesk inventor* . Rancangan kerangka *mobile robot* ini ditunjukan pada Gambar 4.2 dibawah ini.



(a) (b)

Keterangan :

1. *RPLidar A1M8*
2. *Raspberry Pi 4*
3. *Socket Baterai*
5. *Motor DC*
6. *Motor Driver L298N*

Gambar 4. 2 Gambar Rancangan Kerangka *Mobile robot* (a) Tampak Samping; (b) Tampak Bawah

4.4 Pembuatan Kerangka *Mobile robot*

Pada tahap pembuatan kerangka *mobile robot* ini dilakukan proses pembuatan *base mobile robot* serta *part - part* penunjang untuk dudukan dari komponen komponen yang dipakai. Untuk *base* dari *mobile robot* dengan membeli di salah satu toko online. *Mobile robot* ini sudah siap dirangkai. Untuk *part - part* penunjang lainnya seperti dudukan dari *powerbank, Raspberry* pi 4 serta *RPLidar A1M8* dibuat menggunakan 3D printing dan menggunakan pcb bolong. Untuk foto *base* dari *mobile robot* serta *part - part* penunjangnya di tunjukan pada gambar 4.3 dibawah ini.

Gambar 4. 3 Gambar kerangka *Mobile robot* dan *Part* Penunjang

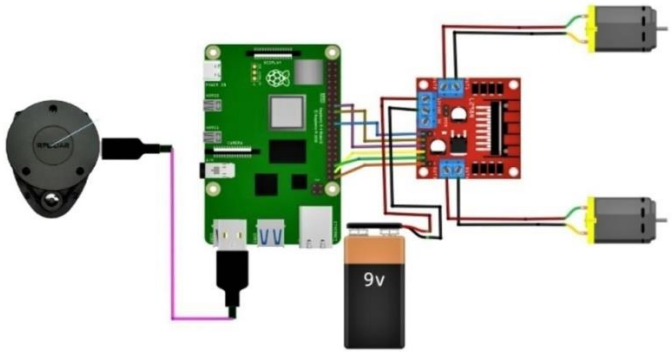
4.5 Perakitan Kerangka *Mobile robot*

Pada tahap perakitan kerangka *mobile robot* ini dilakukan proses *assembly* atau menghubungkan komponen komponen ke *base* dari *mobile robot* sesuai dengan rancangan yang telah dibuat sebelumnya. Untuk hasil perakitan kerangka *mobile robot* bisa dilihat pada gambar 4.4 dibawah ini.

Gambar 4. 4 Gambar *Mobile robot* Setelah di Rakit

4.6 Perancangan *Hardware Elektrik Mobile robot*

Pada tahap perancangan *hardware elektrik mobile robot* ini dirancang dengan konsep yang telah dibuat. Rancangan *hardware elektrik mobile robot* ini dilakukan dengan *software fritzing*. Rancangan *hardware elektrik mobile robot* ini ditunjukan pada gambar 4.5 dibawah ini.



Gambar 4. 5 Gambar *Rancangan Hardware Elektrik*

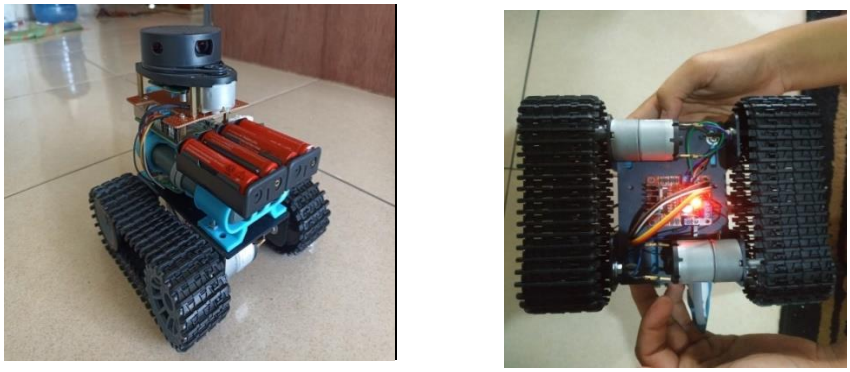
Rancangan elektrik mobile robot ini menggunakan sebuah sensor *RPLidar A1M8*, satu buah *Raspberry Pi 4*, satu buat *Motor Driver (L298N)*, dua buah Motor DC dan 1 buah Baterai. *Wiring* pada gambar rancangan elektrik bisa dilihat pada keterangan dibawah ini.

Keterangan :

Pin GPIO 24 (pin 18 *Raspberry*)	Dihubungkan ke pin IN1 pada L298N
Pin GPIO 23 (pin 16 *Raspberry*)	Dihubungkan ke pin IN2 pada L298N
Pin GPIO 25 (pin 22 *Raspberry*)	Dihubungkan ke pin ENA pada L298N
Pin GPIO 20 (pin 38 *Raspberry*)	Dihubungkan ke pin IN3 pada L298N
Pin GPIO 16 (pin 36 *Raspberry*)	Dihubungkan ke pin IN4 pada L298N
Pin GPIO 21 (pin 40 *Raspberry*)	Dihubungkan ke pin ENB pada L298N
Pin GND (pin 3 *Raspberry*)	Dihubungkan ke pin GND pada L298N
Port USB	Dihubungkan ke *RPLidar A1M8*

4.7 Perakitan *Hardware Elektrik Mobile robot*

Pada tahap perakitan hardware elektrik *mobile robot* ini dilakukan proses menghubungkan komponen - komponen ke *Raspberry* menggunakan kabel jumper sesuai dengan desain yang telah dibuat sebelumnya. Untuk hasil perakitan *hardware elektrik mobile robot* bisa dilihat pada gambar 4.6 dibawah ini.



(a) (b)

Gambar 4. 6 Gambar *kelistrikan Mobile robot* Setelah Dirakit (a) Tampak Atas; (b) Tampak Bawah

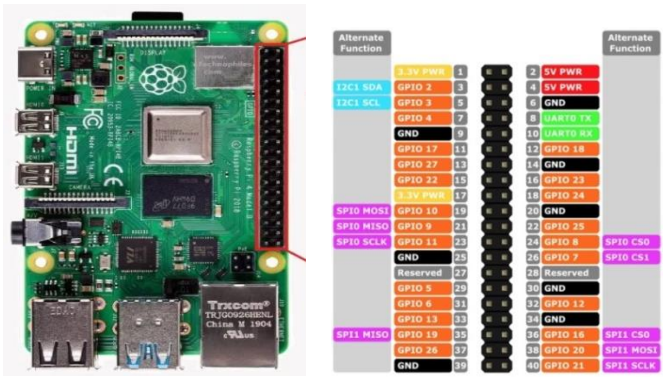
4.8 Uji Coba Komponen *Elektrik Mobile robot*

Uji coba komponen komponen dari *mobile robot* ini dilakukan untuk mengecek, apakah komponen komponen yang digunakan bisa berfungsi dengan baik atau tidak. Komponen komponen yang akan dicek adalah

1. *Raspberry pi 4*
2. *RPLidar A1M8*
3. Motor DC

4.8.1 Uji Coba *Raspberry pi 4*

Uji coba *Raspberry pi* ini dilakukan untuk mengetahui bagaimana cara kerja dari *Raspberry pi*. Selain itu untuk mengetahui apakah *Raspberry pi* ini berfungsi dengan baik atau tidak. Berikut adalah gambar dari *Raspberry pi 4*



Gambar 4. 7 Gambar *Raspberry Pi 4*

Spesifikasi dari *Raspberry pi 4* yang digunakan ditunjukan pada tabel 4.1 dibawah.

Tabel 4. 1 Spesifikasi *Raspberry Pi 4*

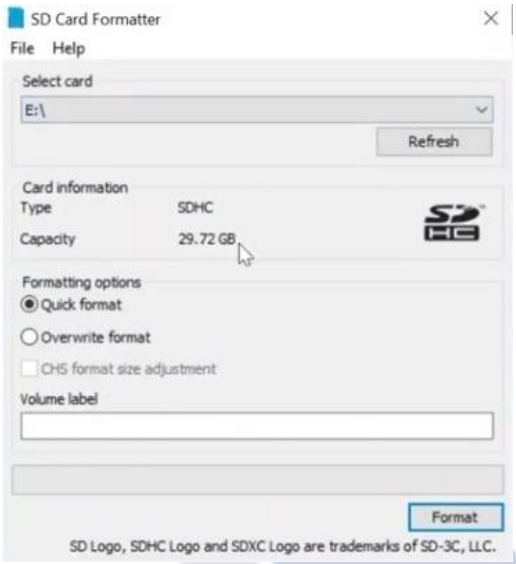
Nama Komponen	Spesifikasi komponen
SoC	*Broadcom BCM2711quad-core Cortex-A72 (ARMv8) @ 1.5GHz with VideoCore VI GPU supporting OpenGL ES 3.0 graphics*
System Memori	*8GB LPDDR4*
Storage	*MicroSD card slot*
Video Output & Display I/F	• *2x micro HDMI ports up to 4Kp60* • *3.5mm AV port with composite video (and stereo audio)* • *2-lane MIPI DSI display port*
Video	• *Decode – H.265 up to 4Kp60, H.264 up to 1080p60* • *Encode – H.264 up to 1080p30*
Camera	*2-lane MIPI CSI camera port*
Audio	*Stereo audio via AV port, digital audio via HDMI ports*
Connectivity	• *Gigabit Ethernet (RJ45)* • *Dual-band (2.4 GHz/5.0 GHz) 802.11b/g/n/ac WiFi 5* • *Bluetooth 5.0 BLE*
USB	*2x USB 3.0 ports, 2x USB 2.0 ports.*
Expansion	*Standard 40-pin GPIO header fully backward-compatible with previous Raspberry Pi boards*
Power Supply	• *5V DC via USB-C connector (minimum 3A)* • *5V DC via GPIO header (minimum 3A)* • *Power over Ethernet (PoE) via optional PoE HAT*

Dimensions	*85 x 56 mm (same as other model B boards)*
Temperature Range	*0° – 50°C*

Raspberry pi 4 ini nantinya akan digunakan sebagai mini pc yang di dalamnya akan memuat *ROS*, *Package* sensor *RPLidar A1M8*, *Package Autonomous Movement*, dan *Package* dari *Hector SLAM*. Ada beberapa tahapan yang harus dilakukan sebelum menguji coba kinerja dari *Raspberry pi 4*. Yang pertama memasukan Operation System (OS) pada *Raspberry*, Yang kedua membuat tampilan *Raspberry terdisplay* di monitor laptop menggunakan koneksi wifi, dan Yang Ketiga *menginstall Robot Operating System* (*ROS*).

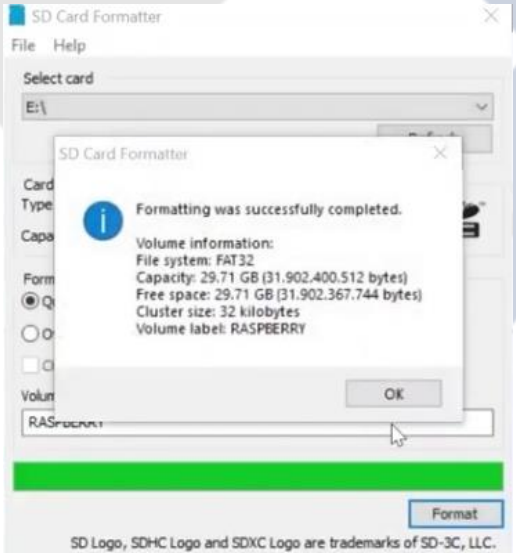
Tahapan pertama yang harus dilakukan adalah, memasukan Sistem Operasi (OS) ke dalam *Raspberry*. Ada banyak system operasi yang kompatibel dengan *Raspberry pi 4* seperti : *Raspbian, Ubuntu, Ubuntu Mate, Twister, Kali Linux* dan masih banyak lagi. Namun OS yang *support* terhadap *ROS* Cuma *Raspbian, Ubuntu*, dan *Ubuntu Mate*. Jadi untuk OS yang dipakai kali ini adalah Ubuntu Mate karena ukuran nya yang relative kecil dibandingkan dengan Ubuntu biasa dan *Raspbian*. Untuk langkah – langkah memasukan OS ke *Raspberry pi 4* sebagai berikut.

1. *Download* OS *Ubuntu Mate* version 20.04. (disesuaikan tergantung jenis *Raspberry* dan kapasitas ram dari *Raspberry*).
2. *Download SD Card Formater*. (digunakan untuk memformat *SD Card* sebelum di masukan *Operation System* nya).
3. *Download Balena Etcher*. (digunakan untuk memasukan file *Operation System* kedalam *SD Card*).
4. Setelah *download* semuanya, selanjutnya masukan *SD Card* ke laptop.
5. Kemudian buka aplikasi dari *SD Card Formatter* untuk melakukan pemformatan.



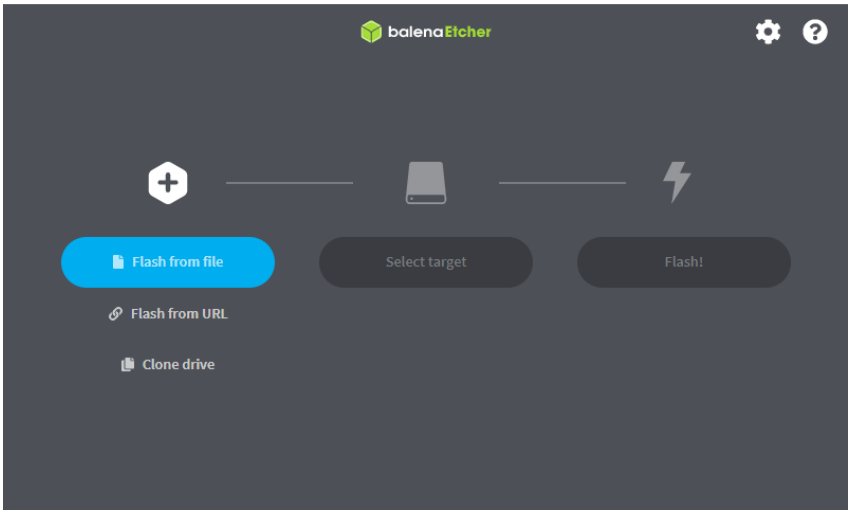
Gambar 4. 8 Tampilan Awal *SD Card Formatter*

6. Kemudian pilih *SD Card* yang akan di format, dan jika sudah klik Format. Jika sudah tampilannya akan seperti ini.



Gambar 4. 9 Tampilan *SD Card* Setelah Di Format

7. Kemudian buka aplikasi *Balena Etcher* untuk memasukan *OS* ke dalam *SD Card* yang telah diformat.

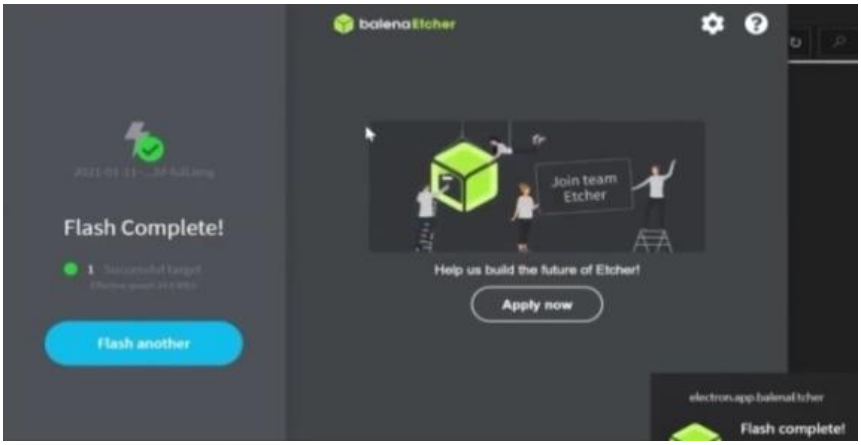


Gambar 4. 10 Tampilan Awal *Software Balena Etcher*

8. Pilih file yang akan digunakan, kemudian pilih *SD Card* yang sudah diformat sebelumnya.

Gambar 4. 11 Memilih File Yang Akan Digunakan

9. Klik *Flash*, kemudian jika sudah selesai maka tampilannya akan seperti ini.



Gambar 4. 12 Tampilan Proses *Flash*

10. Kemudian cabut *SD Card* dari laptop dan masukan ke dalam *Raspberry*.
11. Kemudian hubungkan *Raspberry* ke monitor menggunakan kabel *micro hdmi to hdmi*.
12. Jika sudah sambungkan *Raspberry* dengan *power* untuk melihat apakah proses yang dilakukan sebelumnya itu berhasil atau tidak.
13. Jika berhasil maka akan diminta memasukan zona waktu dan diminta untuk memasukan nama serta *password.*
14. Setelah itu klik *continue*, akan muncul tampilan seperti ini.

Gambar 4. 13 Tampilan Awal *Ubuntu Mate*

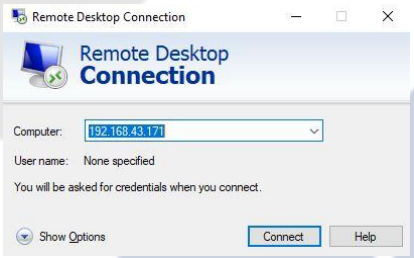
15. *Ubuntu Mate* siap digunakan.

Setelah *Ubuntu Mate* selesai di *Install* maka di lanjutkan pada Tahapan kedua. Tahapan kedua ini adalah membuat tampilan *Raspberry* bisa *terdisplay* di monitor PC/Latop menggunakan koneksi *wifi*. Karena *mobile robot* yang dibuat bergerak secara fleksibel maka tahapan ini sangat penting dilakukan untuk membuat *mobile robot* bergerak dengan leluasa. Banyak sekali cara untuk membuat tampilan *Raspberry* bisa *terdisplay* ke monitor PC/Laptop. Bisa menggunakan *VNC Viewer* dan *Xrdp*. Kali ini digunakan cara *Xrdp* karena dirasa cukup mudah untuk di implementasikan. Untuk langkah – langkahnya sebagai berikut :

1. Pada *Ubuntu Mate*, klik kanan lalu pilih open terminal
2. Pada saat terminal sudah terbuka, masukan perintah '*sudo apt install xrdp*'. Perintah ini digunakan untuk menginstall xrdp.
3. Kemudian jika sudah masukan perintah '*sudo systemctl enable-now xrdp*'. Perintah ini digunakan untuk mengaktifkan *xrdp* yang telah di *install.*
4. Setelah itu masukan perintah '*sudo ufw allow from any to any port 3389 proto tcp*'. Perintah ini digunakan untuk mengktifkan fitur *xrdp firewall* untuk menerima koneksi *xrdp* dari mana saja melalui *port 3389* dengan *protocol tcp*.
5. Kemudian masukan perintah '*ip a*'. perintah ini digunakan untuk melihat ip *address* dari *Raspberry*.
6. Setelah itu lepaskan kabel yang menyambungkan antara *Raspberry* dengan monitor.
7. Siapkan *wifi* yang bisa diakses oleh PC/Laptop dan juga *Raspberry*. Contohnya *Tathering Portabel* dari *Handphone.*

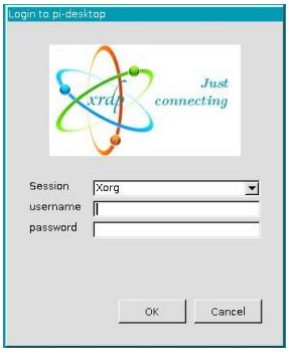
Gambar 4. 14 Tampilan *Tathering Hotspot*

8. Jika PC/Laptop dan *Raspberry* sudah terhubung ke koneksi wifi yang sama, lanjutkan dengan membuka *Remote Dekstop Protocol (RDP)* pada laptop. PC/Laptop dan *Raspberry* harus terhubung ke satu buah koneksi *wifi*. Jika tidak terhubung ke dalam satu koneksi *wifi* maka akan terjadi *error* atau tidak berhasil.
9. Kemudian masukan *Ip Address* dari *Raspberry*.



Gambar 4. 15 Tampilan *Remote Dekstop*

10. Masukan *Username* serta *password* dari *Raspberry pi*.



Gambar 4. 16 Tampilan Awal *Xrdp*

11. Tampilan *Ubuntu Mate* akan seperti ini.

Gambar 4. 17 Tampilan *Ubuntu Mate*

12. *Raspberry* sudah terdisplay ke PC/Laptop menggunakan koneksi *wifi*.

Setelah Tahap kedua selesai, selanjutnya dilakukan penginstalan *ROS* pada *Raspberry*. *ROS* ini yang nantinya yang akan mengatur semua kerja dari robot. Mulai dari *Autonomous Movement* serta *Mapping* dan *Localization* nya. *ROS* juga memiliki banyak jenis. Untuk penginstalannya tergantung versi dari *Ubuntu Mate* nya. Contoh *Ubuntu Mate Version 20.04* kompatibel terhadap *ROS Noetic*. Ini sudah dari website *ROS* nya. Untuk cara penginstalan *ROS Noetic* cukup mengikuti langkah - langkah yang sudah diberikan pada website *ROS*. Untuk langkah – langkahnya sebagai berikut.

1. Open terminal pada Ubuntu, kemudian ikuti langkah – langkah yang terdapat pada website *ROS*.
2. *Setup Source.list*
   - Mengizinkan komputer untuk menerima software dari *package.ros.org.*

```
sudo sh -c 'echo "deb http://packages.ROS.org/ROS/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ROS-latest.list'
```

3. *Setup keys*

```
sudo apt install curl # if you haven't already installed curl

curl -s https://raw.githubusercontent.com/ROS/ROSdistro/master/ROS.asc | sudo apt-key add -
```

4. *Installation*

- Sebelum melakukan penginstalan pastikan Ubuntu telah diperbarui ke pembaruan terbaru

```
sudo apt update
```

5. Setelah Ubuntu ter update, selanjutnya bisa dilakukan penginstalan *ROS Noetic*. Untuk penginstalan memiliki 3 opsi
   - Opsi Pertama ini sangat disarankan karena seluruh *package* 2D/3D seperti *Gazebo* dan *RViz* sudah bisa langsung digunakan

```
sudo apt install ROS-noetic-desktop-full
```

   - Opsi Kedua hanya melakukan penginstalan *ROS-base* dan sudah terdapat *tools* grafis seperti *rqt* dan *RViz.*

```
sudo apt install ROS-noetic-desktop
```

   - Opsi Ketiga hanya menginstal dasar *ROS Noetic* saja. Jadi perlu melakukan penginstallan *RViz* jika ingin menggunakan nya.

```
sudo apt install ROS-noetic-ROS-base
```

6. Untuk mengakses *ROS* yang telah terinstall harus mengkonfigurasikan terlebih dahulu pada saat membuka terminal.

```
source /opt/ROS/noetic/setup.bash
```

   - bisa juga mengakses *ROS* tanpa harus mengkofigurasi terlebih dahulu menggunakan perintah

```
echo "source /opt/ROS/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

7. *ROS Noetic* pun selesai di install

4.8.2 Uji Coba Sensor *RPLidar A1M8*

Uji coba sensor *RPLidar A1M8* ini dilakukan untuk mengetahui bagaimana cara kerja dari sensor *LIDAR*. Selain itu untuk mengetahui apakah sensor *RPLidar A1M8* dapat berfungsi dengan baik atau tidak.

Untuk uji coba ini dilakukan tiga tahap. Yang Pertama menggunakan *software Frame grabber* di windows dan yang kedua pada *ROS* menggunakan *Package RViz* untuk visualisasinya di linux, dan yang ketiga uji coba keakurasian pembacaan sensor *RPLidar A1M8* Ketika membeli sensor *RPLidar A1M8* ini di dalamnya terdapat 3 buah komponen yaitu: *RPLidar A1M8* nya sendiri, Kabel jumper yang digunakan untuk menghubungkan *RPLidar A1M8* nya ke *converter*, dan converter untuk menghubungkan sensor *RPLidar A1M8* ke PC/*Raspberry* menggunakan kabel USB. Komponen *RPLidar A1M8* ditunjukan pada gambar 4.8 dibawah.

Gambar 4. 18 *RPLidar A1M8*

RPLidar A1M8 ini mempunyai 7 kaki atau pin untuk keterangan pin nya di tunjukan tabel 4.3 dibawah.

Tabel 4. 2 *Pin RPLidar A1M8*

PIN	KETERANGAN
PIN 1 (KAKI 1)	GND
PIN 2 (KAKI 2)	RX
PIN 3 (KAKI 3)	TX
PIN 4 (KAKI 4)	V 5.0

PIN 5 (KAKI 5)	GND
PIN 6 (KAKI 6)	MOTOCTL
PIN 7 (KAKI 7)	VMOTO

Uji coba *RPLidar A1M8* yang pertama dilakukan di windows menggunakkan *software Frame Grabber*. Untuk langkah-langkah nya sebagai berikut.

1. Hubungkan *RPLidar A1M8* ke *Converter* menggunakan kabel jumper yang sudah disiapkan.

Gambar 4. 19 Sambungan *RPLidar A1M8* ke *Converter*

2. Hubungkan *RPLidar A1M8* ke PC menggunakan kabel USB yang telah dihubungkan ke *Converter.*

Gambar 4. 20 Sambungan *RPLidar A1M8* ke kabel USB

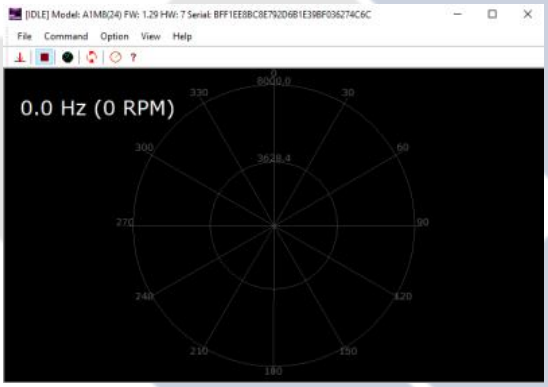
3. Ketika *RPLidar A1M8* aktif maka led pada *converter* akan menyala.

Gambar 4. 21 Indikator Led *converter* pada saat *RPLidar A1M8* aktif

4. *Download software Frame Grabber.*

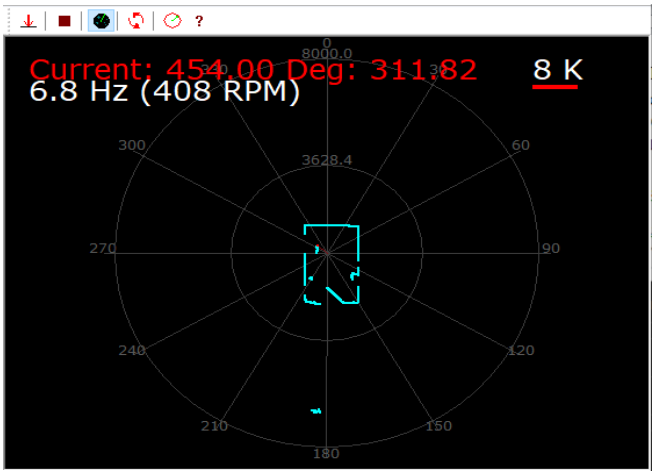
Gambar 4. 22 Tampilan *Software Frame Grabber*

5. Jika sudah di *Download*, buka *software Frame Grabber*.
6. Lalu setting COM serial nya.
7. Untuk melihat COM serial *RPLidar A1M8* cukup buka *device manager* lalu pilih *Ports* (COM&LPT)
8. Jika sudah maka akan tampil tampilan seperti ini.



Gambar 4. 23 Tampilan Awal *Software Frame Grabber*

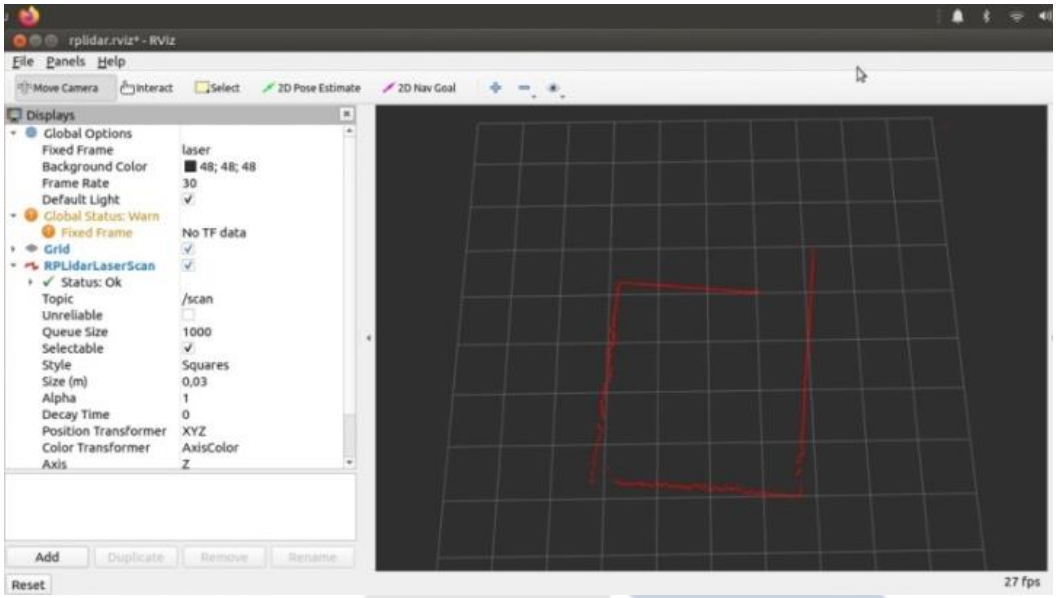
9. Kemudian tinggal klik tombol Start Scan, dan tampilan *RPLidar A1M8* seperti ini.



Gambar 4. 24 Tampilan *Software Frame Grabber* Pada Saat Berjalan

Uji coba yang kedua pada *ROS* menggunakan *package RViz* untuk memvisualisasikan hasil scan sensor *RPLidar A1M8*.

1. Buat sebuah folder untuk meletakan semua *package* - *package* yang akan digunakan dengan cara ketik perintah :
   `$ mkdir ~/catkin_ws/src`
2. Setelah folder dibuat, masuk ke folder `catkin_ws/src` dan *install package RPLidar A1M8* yang ada pada *ROS* dengan mengetikan perintah :
   `$ cd catkin_ws/src` kemudian,
   `$ git clone`
   `https://github.com/Slamtec/rplidar_ros.git`
3. Kemudian build *package RPLidar A1M8* dengan menggunakan perintah :
   `$ cd ~/catkin_ws` kemudian ketik `$ catkin_make`
4. Setelah *package* terbuild, hubungkan *RPLidar A1M8* ke *Raspberry pi 4* menggunakan kabel USB. Kemudian masukan perintah berikut ini untuk melihat port yang terhubung ke *RPLidar A1M8*:
   `$ ls -l /dev/ttyUSB0`
5. Kemudian berikan izin akses *RPLIDAR A1M8* ke *Raspberry pi 4* dengan perintah :
   `$ sudo chmod 666 /dev/ttyUSB0`
6. Kemudian open terminal baru dan jalankan *roscore*
   `$ roscore`
7. Setelah *roscore* berjalan, selanjutnya jalan kan *Package RPLidar A1M8* A1M8 dengan cara :
   - Buka terminal baru lalu masukan perintah
     `$ roslaunch rplidar_ros rplidar.launch`
   - Ubah topic fixed frame menjadi `\laser`
   - Kemudian tambahkan display *LaserScan* setelah itu ubah topiknya menjadi `/scan`
   - Maka tampilan *RPLidar A1M8* pada *ROS* akan seperti ini.



Gambar 4. 25 Tampilan *RPLidar A1M8* Pada *ROS*

Uji coba yang ketiga bertujuan untuk mengetahui ke akurasian pembacaan jarak dari sensor *RPLidar A1M8* Pengujian Sensor ini dilakukan dengan cara membandingkan data yang telah didapatkan. Data tersebut berupa nilai jarak yang ditampilkan di terminal (pada *Raspberry pi 4*) dan nilai jarak yang diambil secara manual dengan cara mengukur jarak pada suatu sudut yang ditentukan dari sensor *RPLidar A1M8* ke halangan yang berada pada arah sudut tersebut. Sensor ini mampu mendeteksi jarak pada sudut 0° hingga 360° dan dapat melakukan pembacaan jarak hingga 12 meter terhadap halangan namun memiliki kekurangan yaitu memiliki jarak pembacaan minimum dimana data akan bernilai *infinity* jika sensor *RPLidar A1M8* berada kurang dari 0,5 meter terhadap halangan. Pada pengujian ini dilakukan 5 kali percobaan pembacaan jarak pada sudut 0°,90°,180°, dan 270° untuk mengetahui ke akurasian pembacaan jarak pada sensor *RPLidar A1M8* dengan cara membandingkan jarak pembacaan pada sensor dengan jarak aslinya. Tabel 4.3 menampilkan hasil uji coba *RPLidar A1M8.*

Tabel 4. 3 Uji coba sensor

No	Pengukuran (cm)				Terukur (cm)				*Error* (%)			
	0°	90°	180°	270°	0°	90°	180°	270°	0°	90°	180°	270°
1	160	160	160	160	161	160	162	161	0,62	0	1,25	0,62
2	160	160	160	160	161	161	161	162	0,62	0,62	0,62	1.25
3	160	160	160	160	162	161	162	162	1,25	0,62	1,25	1,25
4	160	160	160	160	162	160	161	161	1,25	0	0,62	0,62
5	160	160	160	160	162	161	161	161	1,25	0,62	0,62	0,62
Rata – Rata *Error*									0,77 %			

Berdasarkan tabel diatas dapat diketahui bahwa perbandingan pembacaan jarak antar sudut yang telah ditentukan memiliki rata – rata *error* dari pembacaan sensor *RPLidar A1M8* sebesar 0,77%. Disimpulkan bahwa pengukuran sensor RPLidar A1M8 dapat dikatakan sangat stabil dan baik.

4.8.3 Uji Coba Motor DC

Uji coba ini dilakukan untuk melihat apakah Motor DC yang digunakan dapat berfungsi dengan baik atau tidak. Untuk membantu pengujian motor dc ini kami memanfaatkan pin *GPIO* yang sudah ada pada *Raspberry pi 4* serta menggunakan *Motor Driver L298N* untuk mengatur kecepatan dan rotasi dari Motor DC. Untuk sumbernya digunakan baterai *Lithium – ion* 3,7v sebanyak 4 buah yang dihubungkan secara seri agar dapat mencapai tegangan 12v. Gambar Motor DC bisa dilihat pada tabel 4.4 dibawah ini.

Gambar 4. 26 Gambar Motor DC

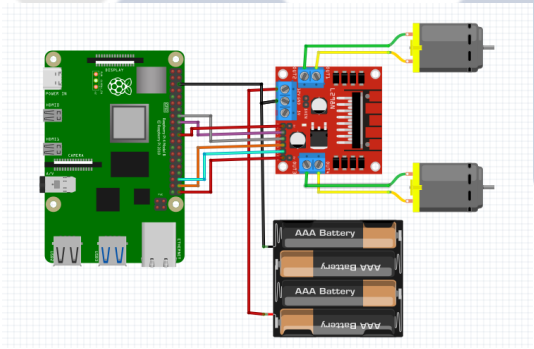
Motor DC yang digunakan adalah tipe *SZdiot* yang memiliki tegangan kerja 12v dengan berat 100 gram. Motor DC jenis ini memiliki Speed 170rpm

hingga 350rpm. Untuk melihat spesifikasi dari Motor DC jenis ini bisa melihat Tabel 4.4 dibawah ini.

Tabel 4. 4 Spesifikasi Motor DC

Parameter	***Specification***
Nama Merek	*SZdiot*
Rated Voltage	*DC12V*
No-load current	*100mA*
No-load speed	*170-350 rpm*
Weight	*100g*

Berikut adalah gambar *interface* antara *Raspberry pi 4* dengan Motor DC, *Motor Driver*, dan Baterai. Rangkaian interface ini dibuat dengan menggunakan *Software fritzing*. Interface antara Raspberry Pi 4 ini memanfaatkan pin GPIO yang ada pada Raspberry Pi 4 yang dihubungkan ke pin yang ada pada Motor Driver L298N.



Gambar 4. 27 *Wiring* Motor DC

Keterangan :

Pin GPIO 24 (Kabel Ungu)	Dihubungkan ke pin IN1 pada L298N
Pin GPIO 23 (Kabel Abu – Abu)	Dihubungkan ke pin IN2 pada L298N
Pin GPIO 25 (Kabel Merah)	Dihubungkan ke pin ENA pada L298N
Pin GPIO 20 (Kabel Orange)	Dihubungkan ke pin IN3 pada L298N
Pin GPIO 16 (Kabel Biru)	Dihubungkan ke pin IN4 pada L298N

Pin GPIO 21 (Kabel Merah)	Dihubungkan ke pin ENB pada L298N
Pin GND (Kabel Hitam)	Dihubungkan ke pin GND pada L298N
Positif Baterai	Dihubungkan ke pin 12v pada L298N
Negatif Baterai	Dihubungkan ke pin GND pada L298N
Keluaran Motor 1	Dihubungkan ke pin OUT1 dan OUT2 pada L298N
Keluaran Motor 2	Dihubungkan ke pin OUT3 dan OUT4 pada L298N

Berikut adalah program uji coba yang dibuat pada *Raspberry pi 4* :

```
Import RPi.GPIO as GPIO
in1 = 24
in2 = 23
in3 = 20
in4 = 16
en1 = 25
en2 = 21
GPIO.setmode(GPIO.BCM)
GPIO.setup(in1,GPIO.OUT)
GPIO.setup(in2,GPIO.OUT)
GPIO.setup(in3,GPIO.OUT)
GPIO.setup(in4,GPIO.OUT)
GPIO.OUTPUT(in1,GPIO.HIGH)
GPIO.OUTPUT(in2,GPIO.LOW)
GPIO.OUTPUT(in3,GPIO.LOW)
GPIO.OUTPUT(in4,GPIO.HIGH)
P1.start(100)
P2.start(100)
```

Setelah dimasukan program diatas Motor DC dapat bergerak sesuai dengan yang diharapkan. Untuk kondisi pergerakan dari motor dc ditampilkan oleh tabel dibawah.

Tabel 4. 5 Kondisi Pergerakan Motor DC

Kondisi	**In1**	**In2**	**In3**	**In4**
Maju	*HIGH*	*LOW*	*LOW*	*HIGH*
Mundur	*LOW*	*HIGH*	*HIGH*	*LOW*
Kanan	*LOW*	*HIGH*	*LOW*	*HIGH*
Kiri	*HIGH*	*LOW*	*HIGH*	*LOW*

Setelah mengetahui kondisi pergerakan dari Motor DC yang ditunjukan tabel 4.5 diatas. Selanjutnya dilakukan percobaan dengan menggunakan *base* dari *mobile robot*. Hal ini bertujuan untuk mendapatkan pergerakan *mobile robot* yang lurus (tidak menyerong). Untuk mendapatkan pergerakan lurus ini cukup menyamakan nilai PWM p1.*start* dan p2.*start* pada *mobile robot*.

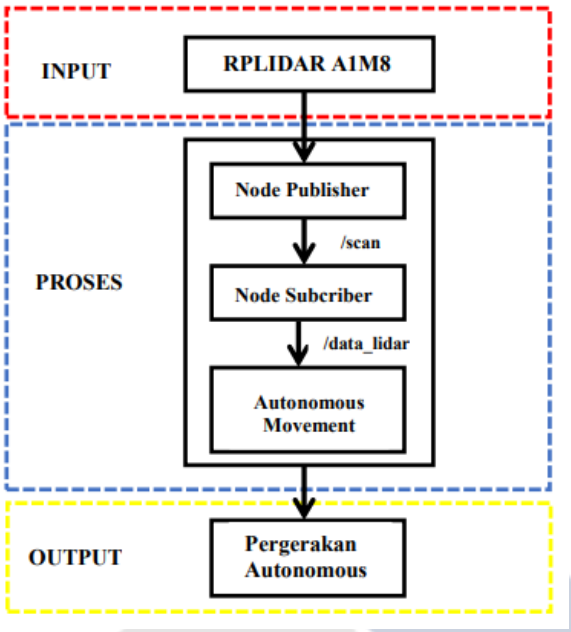
4.9 Pembuatan *Package Mobile robot*

Pembuatan *package mobile robot* ini dibagi menjadi 2 bagian :

- *Autonomous Movement*
- *Mapping & Localization menggunakan Hector SLAM*

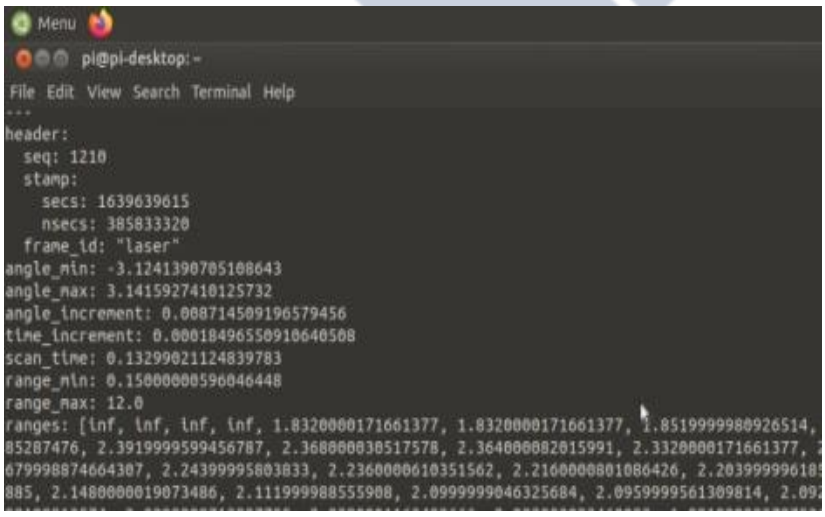
4.9.1 *Autonomous Movement*

Autonomous Movement yang dibuat pada penelitian ini dihasilkan dari pembacan sensor *RPLidar A1M8*. Data pembacaan dari sensor ini berupa nilai jarak pada sudut 0° hingga 360° yang akan di publish oleh sebuah *node publisher* dan kemudian akan di *subscribe* oleh *node subsrcriber* yaitu *node* yang mengeksekusi program *autonomous movement*. Berikut adalah gambar blok diagram dari proses dari *Autonomous Movement*.



Gambar 4. 28 Blok Diagram *Autonomous Movement*

Mula – mula sensor *RPLidar A1M8* akan *mempublish* data melalui *topic "/scan"*. Kemudian *topic "/scan"* tersebut di *Subscribe* oleh *node Subscriber (data_lidar). Node Subcriber* disini berfungsi untuk memproses data yang diterima dari *topic "/scan"* dan kemudian mengeksekusi pergerakan *autonomous* untuk menggerakan *mobile robot*. Data dari *topic "/scan"* ditunjukan pada gambar berikut.

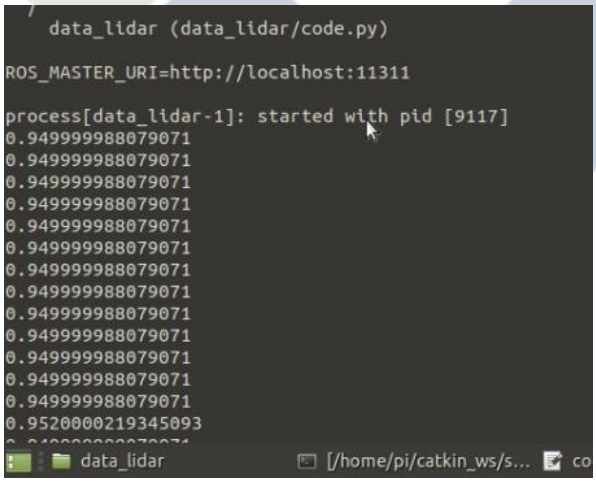


Gambar 4. 29 *Data topic "/scan"*

Karena kesulitan dalam melihat data jaraknya. Dibuatlah suatu *node* khusus untuk memudahkan melihat data jarak dari sensor *RPLidar A1M8*. Berikut program untuk menampilkan data jarak dari *topic "/scan"*.

```
#! /usr/bin/env python3
Import rospy
From sensor_msgs.msg import LaserScan
Def callback(msg):
     Print (msgs.ranges([360])
Rospy.init_node(data_lidar)
Sub = rospy.Subscriber('/scan',LaserScan, callback)
Rospy.spin()
```

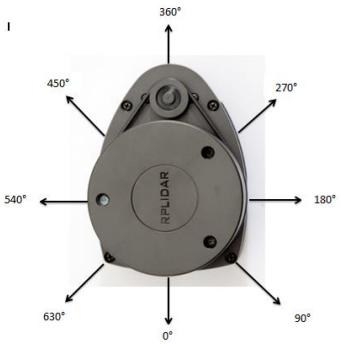
Berdasarkan Program yang dibuat diatas, data jarak yang ditampilkan berdasarkan sudut 360° terhadap *obstacle*. Berikut adalah data jarak yang dihasilkan setelah program dijalankan.



Gambar 4. 30 Data Jarak

Hasil yang ditampilkan yaitu pada sudut 360° dengan jarak terhadap halangan sejauh 0.94 meter. Kemudian untuk mengetahui posisi sudut yang ada pada sensor *RPLidar A1M8* yaitu dengan memasukan nilai sudut yang berbeda pada program yang telah dibuat. Pencarian posisi sudut ini dimaksudkan untuk

mempermudah pada saat pembuatan program *Autonomous Movement* yang akan dipakai pada mobile robot ini. Posisi sudut yang didapatkan ditampilkan pada Gambar 4.14



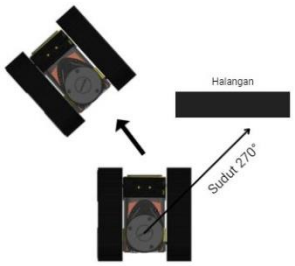
Gambar 4. 31 Posisi Sudut Pada Sensor *RPLidar A1M8*

Selanjutnya mencoba menggerakan *mobile robot* berdasarkan data jarak pada sudut yang sudah diketahui. Untuk dapat menggerakan *mobile robot* secara otomatis dilakukan penggabungan program menggerakan Motor DC dengan pembacaan sensor *RPLidar A1M8.* Program *Autonomous Movement* dibuat dengan mengkombinasikan sudut 180°, 270°, 360°, 450°, dan 540° untuk membaca jarak terhadap halangan. Program *Autonomous movement* diletakan pada Lampiran 2.. Tabel dibawah menunjukan hasil uji coba pada program *Autonomous Movement.*

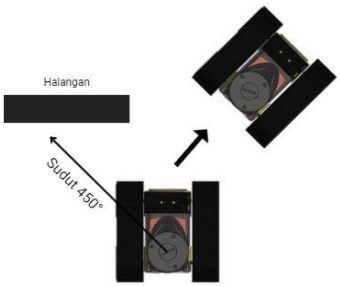
Tabel 4. 6 Pergerakan *Mobile Robot*

Logika	**Pergerakan Robot**
Jika sensor tidak mendeteksi adanya halangan, robot akan berjalan lurus	

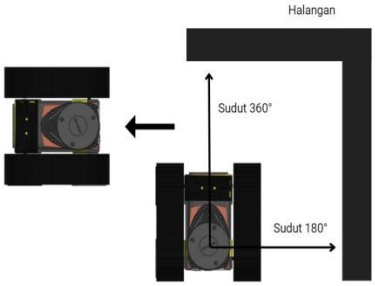
Jika jarak pada sudut 270° terbaca 0.5 meter terhadap halangan, maka pergerakan robot akan serong kiri



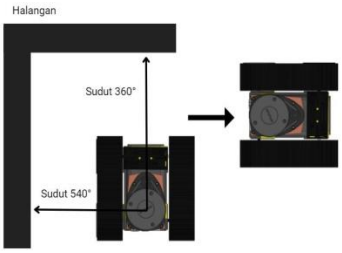
Jika jarak pada sudut 450° terbaca 0.5 meter terhadap halangan, maka pergerakan robot akan serong kanan



Jika jarak pada sudut 360° dan 180° terbaca 0.5 meter terhadap halangan, maka pergerakan robot akan belok ke kiri



Jika jarak pada sudut 360° dan 540° terbaca 0.5 meter terhadap halangan, maka pergerakan robot akan belok ke kanan



Dari percobaan *autonomous movement* sederhana ini, didapatkan hasil bahwa *mobile robot* mampu bernavigasi dan menghindari halangan menggunakan data dari sensor *RPLidar A1M.8.*

4.9.2 *Mapping dan Localization* Menggunakan *Hector SLAM*

Proses *Mapping dan Localization* pada penelitian ini menggunakan metode *Hector SLAM*. Untuk dapat menggunakan *Hector SLAM* harus menginstall *packagenya* terlebih dahulu. Berikut langkah – langkah penginstallannya :

1. Buka folder *src* untuk meletakan *package Hector SLAM* yang akan digunakan dengan cara ketik perintah :
   ```
   $ cd catkin_ws/src
   ```
2. *Install Hector SLAM* dengan mengetikan perintah pada Terminal :
   ```
   $ git clone https://github.com/tu-darmstadt-
   ros-pkg/hector_slam.git
   ```
3. Setelah terinstall setting parameter *Hector SLAM* karena parameter sebelumnya masih settingan default. Caranya buka folder *package Hector SLAM* yang telah terinstall dengan perintah :
   ```
   $ cd catkin_ws/src/hetor_slam/hector_Mapping/launch
   ```
4. Buka file *Mapping_default.launch*, kemudian *setting* parameternya dengan cara seperti berikut :

   Parameter sebelumnya :
   ```
   <param name="map_frame" value="map" />
   <arg name="base_frame" default="base_footprint"/>
   <arg name="odom_frame" default="nav"/>
   <!--<node pkg="tf"
   type="static_transform_publisher"
   name="map_nav_broadcaster" args="0 0 0 0 0 0 map
   nav 100"/>-->
   ```
   Ubah menjadi :
   ```
   <param name="map_frame" value="map" />
   <arg name="base_frame" default="base_link"/>
   ```

```
<arg name="odom_frame" default="base_link"/>
<node pkg="tf" type="static_transform_publisher"
name="base_to_laser_broadcaster" args="0 0 0 0 0 0
base_link laser 100" />
```

5. Buka folder *Hector_SLAM*_launch kemudian buka file *tutorial.launch* dan setting parameternya dengan cara seperti berikut :

   Parameter sebelumnya :

   ```
   <param name="/use_sim_time" value="true"/>
   ```

   Ubah menjadi :

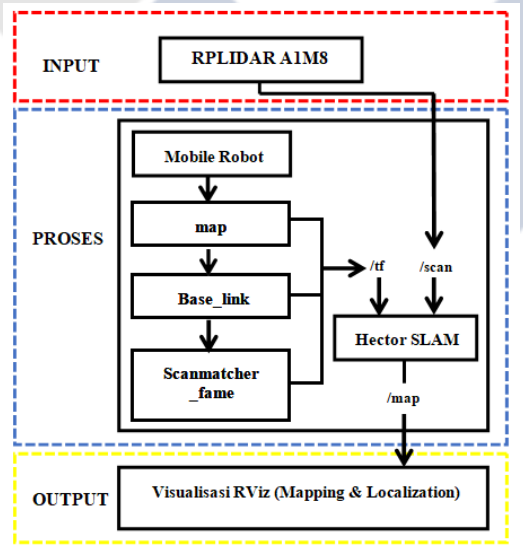
   ```
   <param name="/use_sim_time" value="false"/>
   ```

6. Setelah parameter disetting *build* kembali *package Hector SLAM* :

   `$ cd ~/catkin_ws` kemudian ketik `$ catkin_make`

7. *package Hector SLAM* telah siap digunakan.

Berikut adalah gambar blok diagram dari proses *Mapping dan Localization* menggunakan *Hector SLAM.*

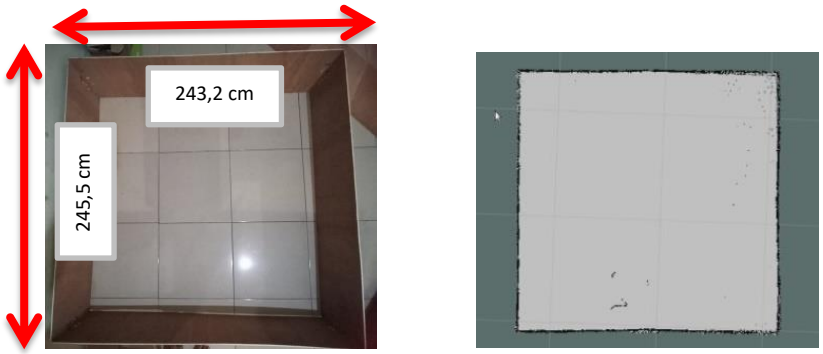


Gambar 4. 32 Blok Diagram *Mapping dan Localization*

Mula – mula sensor *RPLidar A1M8* akan mempublish sebuah data ke dalam *topic /scan*, dimana data dari *topic /scan* ini dapat dilihat pada *'sensor_msg/LaserScan'*, data yang dikirimkan oleh sensor *RPLidar A1M8* akan

digunakan untuk membuat visualisasi *map* dua dimensi (2D) menggunakan metode *scan matching* yang terdapat pada *Hector SLAM*. Karena *mobile robot* yang dibuat pada penelitian ini tidak menggunakan data *odometry*, jadi untuk mendapatkan koordinat robot secara terus menerus *ROS* mempunyai sebuah *package* yaitu "/tf" *(transform). Transform* sendiri berfungsi sebagai standart cara untuk menentukan atau melacak koordinat *frame* dari *mobile robot*. Pada penelitian ini "/tf" yang digunakan adalah *map, base_link dan scanmatcher frame. Frame map* memungkinkan peneliti untuk mengetahui posisi awal robot. *base_link* adalah sebuah koordinat *frame* yang berada pada *body* robot. *base_link* digunakan apabila *algoritma* yang digunakan tidak menggunakan data *odometry*. Sedangkan *scanmatcher frame* digunakan untuk mendapatkan koordinat *mobile robot* secara terus menerus sehingga ketika jenis "/tf" ini digabungan akan muncul *frame* serta koordinat *mobile robot* pada peta yang dibentuk. Hasil *Mapping dan Localization* akan tervisualisasi di *RViz* melalui fitur */map*. Fitur */map* disini berfungsi untuk mengimplementasikan pembuatan *map* dua dimensi (2D) berdasarkan data dari "/tf" *(package transform)* dan */scan* (sensor *RPLidar A1M8*). Parameter *Hector SLAM* ditunjukan pada Lampiran 2. Kemudian dilakukan pengujian *Mapping dan Localization* pada *Hector SLAM*.

Pengujian *Mapping* pada *Hector SLAM* ini bertujuan untuk mengetahui keakurasian ukuran hasil *Mapping* terhadap arena sebenarnya. Pengujian *Mapping* dilakukan pada arena percobaan berbahan papan dengan Panjang 243,2 cm dan Lebar 245,5 cm dengan cara membandingkan hasil ukuran arena sebenarnya dengan hasil pengukuran pada *map* yang telah dibuat. Pengukuran pada arena sebenarnya dilakukan secara manual dan pengukuran pada hasil *Mapping* menggunakan fitur *measuremen*t pada *RViz*. Pada pengujian ini dilakukan 5 kali percobaan untuk mendapatkan nilai panjang dan lebar dari arena pengujian. Arena pengujian *Mapping* ditunjukan Gambar 4.16 dan hasil uji coba *Mapping* ditampilkan oleh tabel 4.6.



(a) (b)

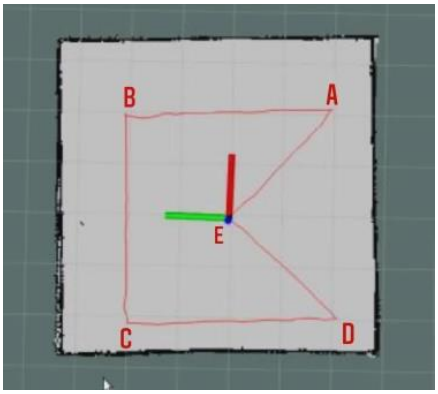
Gambar 4. 33 Arena Pengujian (a) Ruangan sebenarnya; (b) Hasil *Mapping*

Tabel 4. 7 Tabel Pengujian *Mapping* Pada *Hector SLAM*

No	**Pengukuran (cm)**		**Terukur (cm)**		***Error* (%)**	
	P	**L**	**P**	**L**	**P**	**L**
1	234,2	245,5	234,5	245,2	0,123	0,122
2	234,2	245,5	234,8	245,0	0,246	0,20
3	234,2	245,5	235,1	245,1	0,411	0,162
4	234,2	245,5	234,2	244,2	0	0,529
5	234,2	245,5	234,6	245,7	0,164	0,081
Rata – Rata *Error*					0,2038 %	

Percobaan yang dilakukan mendapatkan hasil rata – rata *error* sebesar 0,2038 %. Dari hasil yang didapatkan dapat disimpulkan bahwa penerapan *Hector SLAM* untuk melakukan *Mapping* pada arena pengujian sudah sangat baik MOBILE ROBOT dikarenakan *error* yang dihasilkan sangat kecil yaitu 0,2038 %. Hal ini menunjukan bahwa hasil *Mapping* menyerupai arena sebenarnya.

Pengujian lokalisasi bertujuan untuk mengetahui ke akuratan nilai koordinat yang ditampilkan *RViz* saat *mobile robot* berpindah dari satu titik ke titik lain yang telah di tentukan. Pengujian lokalisasi dilakukan dengan cara membandingkan nilai koordinat yang ditetapkan dengan nilai koordinat yang ditampilkan pada *RViz.* Pada pengujian ini menggunakan 5 titik perpindahan yang ditandai dengah huruf A, B, C, D, dan E. hasil uji coba lokalisasi ditampilkan oleh Gambar 4.17 dan tabel 4.7



Gambar 4. 34 Uji Coba Lokalisasi

Tabel 4. 8 Tabel Hasil Uji Coba Lokalisasi

Titik	**Posisi (cm)**		**Posisi pada *Rviz* (cm)**		***Error* (%)**	
	X	**Y**	**X**	**Y**	**X**	**Y**
A	80	-80	81,25	-80,65	1,56	0,81
B	80	80	79,56	80,17	0,55	0,175
C	-80	80	-80,74	78,83	0,32	1,46
D	-80	-80	-80,14	83,03	0,175	3,78
E	0	0	0,12	0,014	1,2	1,4
Rata – Rata *Error*					1,523 %	

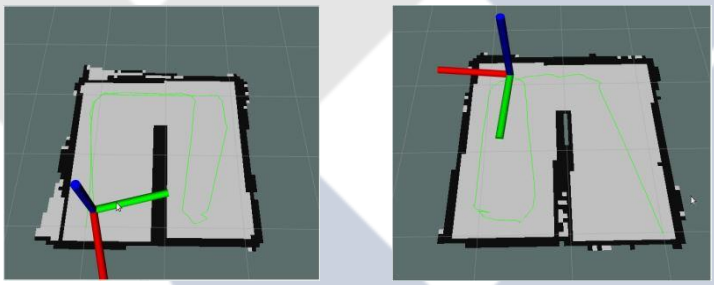
Percobaan yang dilakukan mendapatkan hasil rata – rata *error* sebesar 1,523%. *Error* ini disebabkan kesalahan penentuan titik tengah yang ada pada sensor *RPLidar A1M8* dan perpindahan dari titik ke titik lainnya dilakukan secara manual.

4.10 Uji Coba Keseluruhan

Uji coba keseluruhan ini dilakukan untuk mengetahui apakah sistem *Autonomous Movement yang* dibuat mampu membantu *mobile robot* untuk melakukan proses pemetaan dan lokalisasi di arena yang telah disiapkan. Gambar 4.18 menampilkan arena percobaan pada penelitian ini.

Gambar 4. 35 Arena Pengujian

Pengujian ini dilakukan sebanyak 2 kali dengan titik start yang berbeda. Gambar 4.18 menunjukan hasil pemetaan dan lokalisasi menggunakan *Autonomous Movement.*



(a) (b)

Gambar 4. 36 Hasil Uji Coba Keseluruhan (a) Titik Start 1; (b) Titik Start 2

Dari hasil percobaan keseluruhan ini didapatkan bentuk peta yang sudah menyerupai bentuk arena aslinya. Hasil peta yang kurang baik seperti Gambar (a) disebabkan oleh pergerakan *autonomous movement* yang terlalu cepat pada saat menyusuri arena. Untuk mendapatkan hasil peta yang baik seperti Gambar (b) *mobile robot* harus bergerak secara perlahan pada saat menyusuri arena yang disiapkan. Dapat disimpulkan pergerakan automomous pada *mobile robot* akan mempengaruhi bentuk *Mapping* dan keakuratan lokalisasi. Semakin lambat pergerakan *mobile robot* pada saat proses *Mapping* dan lokalisasi maka semakin bagus dan akurat hasil *Mapping* dan lokalisasi yang ditampilkan pada *RViz.*

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah melakukan penelitian tentang *Autonomous Mobile Robot* Dengan Menggunakan Metode *Simultaneous Localization And Mapping* dapat disimpulkan sebagai berikut :

1. Hasil pembacaan jarak pada sensor *RPLidar A1M8* memiliki rata – rata *error* sebesar 0,77% terhadap jarak sebenarnya.
2. Dari hasil pengujian *Mapping* dengan *Hector SLAM* menggunkan sensor *RPLidar A1M8* terhadap arena pengujian memiliki rata–rata *error* sebesar 0,2038%.
3. Ujicoba lokalisasi yang dilakukan dengan *Hector SLAM* (fungsi *transform*) terhadap arena percobaan memiliki rata–rata *error* hingga 1,523%.

5.2 Saran

Dari hasil proyek akhir yang telah didapatkan ada beberapa saran yang perlu disampaikan untuk mengembangkan proyek akhir ini kedepannya, yaitu:

1. Pada penelitian ini fungsi *package transform* hanya digunakan untuk menampilkan koordinat *mobile robot*. Untuk itu disarankan lebih mengembangkan fungsi *package transform* agar tidak hanya mampu menampilkan koordinat *mobile robot* melainkan dapat membantu *mobile robot* bergerak ke sebuah koordinat yang telah ditetapkan.
2. Pada penelitian ini robot tidak dapat melewati halangan yang tingginya dibawah posisi sensor LIDAR yaitu sekitar 18 cm, untuk itu disarankan menambahkan sebuah sensor jarak pada robot agar dapat mendeteksi halangan yang tingginya dibawah sensor LIDAR.

DAFTAR PUSTAKA

Jalil, A. (2018). Robot Operatyng System (Ros) Dan Gazebo Sebagai Media Pembelajaran Robot Interaktif. *ILKOM Jurnal Ilmiah*, *10*, 284–289.

Jalil, A. (2019). Pemanfaatan Middleware Robot Operating System (Ros) Dalam Menjawab Tantangan Revolusi Industri 4.0. *ILKOM Jurnal Ilmiah*, *11*(1), 45–52. https://doi.org/10.33096/ilkom.v11i1.412.45-52

Lemus, R., Díaz, S., Gutiérrez, C., Rodríguez, D., & Escobar, F. (2014). SLAM-R Algorithm Of Simultaneous *Localization* and *Mapping* Using RFID For *Obstacle* Location and Recognition. *Journal of Applied Research and Technology*, *12*(3), 551–559. https://doi.org/10.1016/S1665-6423(14)71634-7

Prayoga, S., Budianto, A., Budi, A., & Atmaja, K. (2017). Sistem Pemetaan Ruangan 2D Menggunakan Lidar. *9*(1), 73–79.

Prayoga, S., Sumantri, K. R., A, A. B. K., & Batam, P. N. (2010). *Ekf slam menggunakan lidar*. 135–140.

Putra, I. W. K. E. (2016). Sistem Kerja Sensor Laser pada LIDAR. *Jurnal Media Komunikasi Geografi*, *17*(1), 59–70.

Rahman, A. (2020). Penerapan SLAM G*Mapping* dengan Robot Operating System Menggunakan Laser Scanner pada Turtlebot. *Jurnal Rekayasa Elektrika*, *16*(2). https://doi.org/10.17529/jre.v16i2.16491

Saat, S., Abd Rashid, W. N., Tumari, M. Z. M., & Saealal, M. S. (2020). Hectorslam 2d *Mapping* For Simultaneous *Localization* And *Mapping* (Slam). *Journal of Physics: Conference Series*, *1529*(4). https://doi.org/10.1088/1742-6596/1529/4/042032

Sartika, E. M., Elektro, J. T., Maranatha, U. K., & Uno, A. (2015). *Implementasi Hector Slam Pada Robot Pencari Korban Gempa the Implementation of Hector Slam on the*. 383–391.

Sidharta, H. A. (2019). *Pengenalan Slam (Simultaneous Localization And*

Mapping) Pada Lidar.

Slamtec.com. (n.d.). *Spesifikasi RPLidar A1M8*. www.Slamtec.Com. Retrieved January 28, 2022, from https://www.slamtec.com/en/Lidar/A1Spec

Taufik, A. S. (2013). Sistem Navigasi Waypoint pada Autonomous Mobile Robot. *Jurnal Mahasiswa TEUB*, *1*(1), 1–6.

Utomo, E. B. (2015). *Autonomous Mobile Robot Berbasis Landmark Menggunakan Particle Filter Dan Occupancy Grid Maps Untuk Navigasi , Landmark Using Particle Filter and Occupancy Grid Maps for Navigation , Localization , and Mapping*. 1–93. https://repository.its.ac.id/id/eprint/41577

Zahra, L., Sani, M. I., & Siregar, S. (2018). Perancangan Dan Implementasi *Mapping* System Untuk Navigasi Roner (Robot Cleaner). *E-Proceeding of Applied Science*, *4*(3), 2092–2101.

LAMPIRAN 1

DAFTAR RIWAYAT HIDUP

1. **Data Diri**

Nama	: Mashur Arbi Maulana
Tempat,Tanggal Lahir	: MENTOK, 05 – 10 - 2000
Alamat Rumah	: GANG URIP KAMPUNG TEGALREJO, SUNGAI BARU, MENTOK
Telp	: -
Hp	: 081369270818
Email	: arbimaulana856@gmail.com
Jenis Kelamin	: LAKI –LAKI
Agama	: ISLAM

2. **Riwayat Pendidikan**

SD Negeri 3 MENTOK (2006 - 2012)

SMP Negeri 3 MENTOK (2012 - 2015)

SMK Negeri 1 MENTOK (2015 - 2018)

3. **Pendidikan Non Formal**

-

Sungailiat, 18 Januari 2022

Mashur Arbi Maulana

DAFTAR RIWAYAT HIDUP

1. **Data Diri**

Nama	: Reynaldi Novian
Tempat,Tanggal Lahir	: CUPAT, 22 -11 -2000
Alamat Rumah	: JLN. IMAM BONJOL NO.116, AIR RUAI, PEMALI
Telp	: -
Hp	: 081360501816
Email	: reynaldinovian211@gmail.com
Jenis Kelamin	: LAKI –LAKI
Agama	: ISLAM

2. **Riwayat Pendidikan**

SD Negeri 14 PEMALI (2006 - 2012)

SMP Negeri 1 PEMALI (2012 - 2015)

SMA Negeri 1 PEMALI (2015 - 2018)

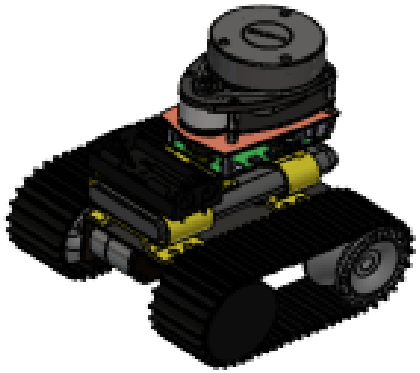
3. **Pendidikan Non Formal**

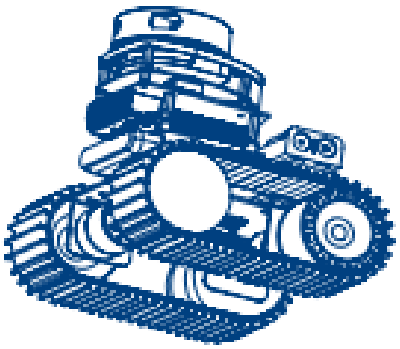
-

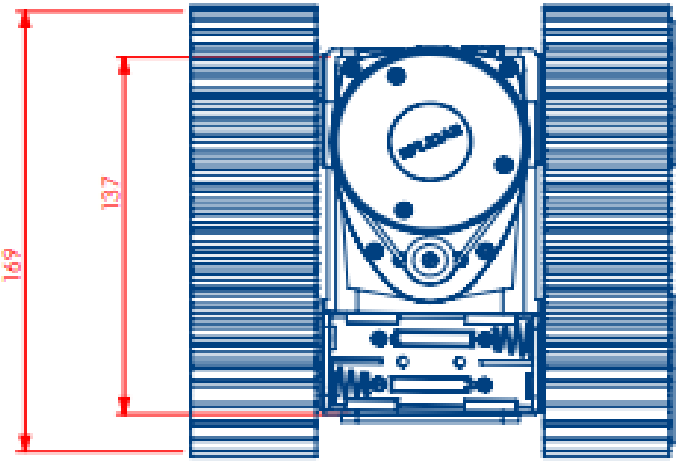
Sungailiat, 18 Januari 2022

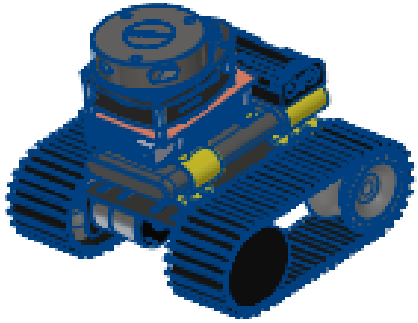
Reynaldi Novian

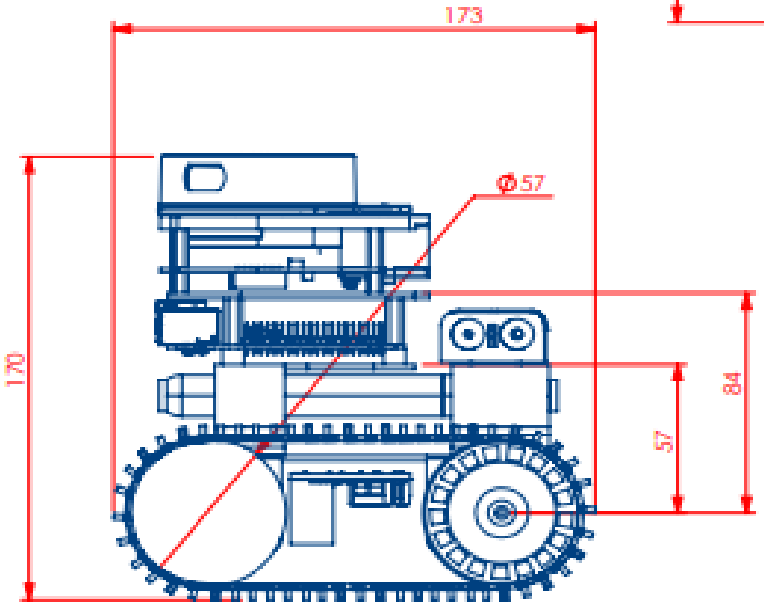
LAMPIRAN 2

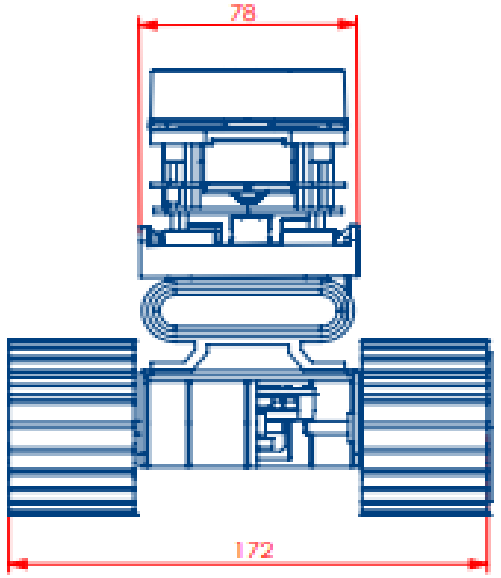












			FINISHING : MACHINE ROUGH SURFACE FINISH WITH BLUE PAINT COATING
	NAMA	PARAF	TANGGAL
DIGAMBAR	Rahmat		
ENGINEERING CHECK			
DISETUJUI			
QC CHECK			
MENGETAHUI			

DO NOT SCALE DRAWING — REVISION : 0

KETERANGAN :
1. DIBUAT 5 PCS

JUDUL :

NO GBR :

A4

Document Type : Standard ISO

MATERIAL:

TANGGAL : 23 OKT 2020 — LOKASI : SELINDUNG

SCALE : 1 : 20, 1:50 — LEMBAR 1 DARI 1

LAMPIRAN 3

Program Autonomous Movement

```
#! /usr/bin/env python3
import rospy
from sensor_msgs.msg import LaserScan
import RPi.GPIO as GPIO
from time import sleep
import time
#in1, in2 dan en1 = motor kanan
in1 = 24
in2 = 23
#in3, in4 dan en2 = motor kiri
in3 = 20
in4 = 16
en1 = 25
en2 = 21
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(in1,GPIO.OUT)
GPIO.setup(in2,GPIO.OUT)
GPIO.setup(en1,GPIO.OUT)
GPIO.setup(in3,GPIO.OUT)
GPIO.setup(in4,GPIO.OUT)
GPIO.setup(en2,GPIO.OUT)
GPIO.output(in1,GPIO.LOW)
GPIO.output(in2,GPIO.LOW)
p1=GPIO.PWM(en1,255)
GPIO.output(in3,GPIO.LOW)
GPIO.output(in4,GPIO.LOW)
p2=GPIO.PWM(en2,255)
```

```
def callback(msg):
      print ("jarak pada derajat 180 = "+str(msg.ranges[180]))
      print ("jarak pada derajat 270 = "+str(msg.ranges[270]))
      print ("jarak pada derajat 340 = "+str(msg.ranges[340]))
      print ("jarak pada derajat 352 = "+str(msg.ranges[352]))
      print ("jarak pada derajat 360 = "+str(msg.ranges[360]))
      print ("jarak pada derajat 368 = "+str(msg.ranges[368]))
      print ("jarak pada derajat 380 = "+str(msg.ranges[380]))
      print ("jarak pada derajat 450 = "+str(msg.ranges[450]))
      print ("jarak pada derajat 540 = "+str(msg.ranges[540]))
      GPIO.output(in1,GPIO.HIGH)
      GPIO.output(in2,GPIO.LOW)
      GPIO.output(in3,GPIO.LOW)
      GPIO.output(in4,GPIO.HIGH)
      p1.start(40)
      p2.start(40)
#1  MAJU (Jika tidak ada titik yang aktif)
if msg.ranges[540]> 0.5 and msg.ranges[450]> 0.5 and
msg.ranges[360]> 0.5 and msg.ranges[270]> 0.5 and msg.ranges[180]>
0.5:
          GPIO.output(in1,GPIO.HIGH)
          GPIO.output(in2,GPIO.LOW)
          GPIO.output(in3,GPIO.LOW)
          GPIO.output(in4,GPIO.HIGH)
#2  MAJU (Jika titik yang aktif adalah 540, 450, 270, dan 180)
if msg.ranges[540]< 0.5 and msg.ranges[450]< 0.5 and
msg.ranges[360]> 0.5 and msg.ranges[270]< 0.5 and msg.ranges[180]<
0.5:
          GPIO.output(in1,GPIO.HIGH)
          GPIO.output(in2,GPIO.LOW)
          GPIO.output(in3,GPIO.LOW)
          GPIO.output(in4,GPIO.HIGH)
```

```
#3  MAJU (Jika titik yang aktif adalah 540 dan 180)

if msg.ranges[540]< 0.5 and msg.ranges[450]> 0.5 and
msg.ranges[360]> 0.5 and msg.ranges[270]> 0.5 and msg.ranges[180]<
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

#.....................................................................

#4 SERONG KIRI (Jika titik yang aktif adalah 540, 270, dan 180)

if msg.ranges[540]< 0.3 and msg.ranges[450]> 0.5 and
msg.ranges[360]> 0.5 and msg.ranges[270]< 0.5 and msg.ranges[180]<
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(70)

          p2.ChangeDutyCycle(35)

#5 SERONG KIRI (Jika titik yang aktif adalah 540 dan 270)

if msg.ranges[540]< 0.3 and msg.ranges[450]> 0.5 and
msg.ranges[360]> 0.5 and msg.ranges[270]< 0.5 and msg.ranges[180]>
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(70)

          p2.ChangeDutyCycle(35)
```

```
#6 SERONG KIRI (Jika titik yang aktif adalah 270 dan 180)

if msg.ranges[540]> 0.5 and msg.ranges[450]> 0.5 and
msg.ranges[360]> 0.5 and msg.ranges[270]< 0.5 and msg.ranges[180]<
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(65)

          p2.ChangeDutyCycle(35)

#7 SERONG KIRI (Jika titik yang aktif adalah 270)

if msg.ranges[540]> 0.5 and msg.ranges[450]> 0.5 and
msg.ranges[360]> 0.5 and msg.ranges[270]< 0.5 and msg.ranges[180]>
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(65)

          p2.ChangeDutyCycle(35)

#.............................................................
#8 BELOK KIRI (Jika titik yang aktif adalah 360)

if msg.ranges[540]> 0.5 and msg.ranges[450]< 0.5 and
msg.ranges[360]< 0.5 and msg.ranges[270]< 0.5 and msg.ranges[180]<
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.HIGH)

          GPIO.output(in4,GPIO.LOW)

          p1.ChangeDutyCycle(68)

          p2.ChangeDutyCycle(0)
```

```
#9 BELOK KIRI (Jika titik yang aktif adalah 450, 360, 270, dan
180)

if msg.ranges[540]> 0.5 and msg.ranges[450]< 0.5 and
msg.ranges[360]< 0.5 and msg.ranges[270]< 0.5 and msg.ranges[180]<
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.HIGH)

          GPIO.output(in4,GPIO.LOW)

          p1.ChangeDutyCycle(68)

          p2.ChangeDutyCycle(0)

#10 BELOK KIRI (Jika titik yang aktif adalah 360, 270, dan 180)

if msg.ranges[540]> 0.5 and msg.ranges[450]> 0.5 and
msg.ranges[360]< 0.5 and msg.ranges[270]< 0.5 and msg.ranges[180]<
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.HIGH)

          GPIO.output(in4,GPIO.LOW)

          p1.ChangeDutyCycle(68)

          p2.ChangeDutyCycle(0)

#11 BELOK KIRI (Jika titik yang aktif adalah 360 dan 270)

if msg.ranges[540]> 0.5 and msg.ranges[450]> 0.5 and
msg.ranges[360]< 0.5 and msg.ranges[270]< 0.5 and msg.ranges[180]>
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.HIGH)

          GPIO.output(in4,GPIO.LOW)

          p1.ChangeDutyCycle(68)

          p2.ChangeDutyCycle(0)
```

```
#12 BELOK KIRI (Jika titik yang aktif adalah 360 dan 180)

if msg.ranges[540]> 0.5 and msg.ranges[450]> 0.5 and
msg.ranges[360]< 0.5 and msg.ranges[270]> 0.5 and msg.ranges[180]<
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.HIGH)

          GPIO.output(in4,GPIO.LOW)

          p1.ChangeDutyCycle(68)

          p2.ChangeDutyCycle(0)

#13 BELOK KIRI (Jika titik yang aktif adalah 352)

if msg.ranges[540]> 0.5 and msg.ranges[450]> 0.5 and
msg.ranges[352]< 0.5 and msg.ranges[270]> 0.5 and msg.ranges[180]>
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.HIGH)

          GPIO.output(in4,GPIO.LOW)

          p1.ChangeDutyCycle(68)

          p2.ChangeDutyCycle(0)

#14  BELOK KIRI (Jika titik yang aktif adalah 450, 340, dan 270,
dan 180)

if msg.ranges[540]> 0.5 and msg.ranges[450]< 0.5 and
msg.ranges[340]< 0.5 and msg.ranges[270]< 0.5 and msg.ranges[180]<
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.HIGH)

          GPIO.output(in4,GPIO.LOW)

          p1.ChangeDutyCycle(68)

          p2.ChangeDutyCycle(0)
```

```
#15 BELOK KIRI (Jika titik yang aktif adalah 340, 270, dan 180)

if msg.ranges[540]> 0.5 and msg.ranges[450]> 0.5 and
msg.ranges[340]< 0.5 and msg.ranges[270]< 0.5 and msg.ranges[180]<
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.HIGH)

          GPIO.output(in4,GPIO.LOW)

          p1.ChangeDutyCycle(68)

          p2.ChangeDutyCycle(0)

#16 BELOK KIRI (Jika titik yang aktif adalah 340 dan 270)

if msg.ranges[540]> 0.5 and msg.ranges[450]> 0.5 and
msg.ranges[340]< 0.5 and msg.ranges[270]< 0.5 and msg.ranges[180]>
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.HIGH)

          GPIO.output(in4,GPIO.LOW)

          p1.ChangeDutyCycle(68)

          p2.ChangeDutyCycle(0)

#17 BELOK KIRI (Jika titik yang aktif adalah 340 dan 180)

if msg.ranges[540]> 0.5 and msg.ranges[450]> 0.5 and
msg.ranges[340]< 0.5 and msg.ranges[270]> 0.5 and msg.ranges[180]<
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.HIGH)

          GPIO.output(in4,GPIO.LOW)

          p1.ChangeDutyCycle(68)

          p2.ChangeDutyCycle(0)
```

```
#18 SERONG KANAN (Jika titik yang aktif adalah 540, 450, dan 180)

if msg.ranges[540]< 0.5 and msg.ranges[450]< 0.5 and
msg.ranges[360]> 0.5 and msg.ranges[270]> 0.5 and msg.ranges[180]<
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(20)

          p2.ChangeDutyCycle(53)

#19 SERONG KANAN (Jika titik yang aktif adalah 450 dan 180)

if msg.ranges[540]> 0.5 and msg.ranges[450]< 0.5 and
msg.ranges[360]> 0.5 and msg.ranges[270]> 0.5 and msg.ranges[180]<
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(20)

          p2.ChangeDutyCycle(53)

#20 SERONG KANAN (Jika titik yang aktif adalah 450)

if msg.ranges[540]> 0.5 and msg.ranges[450]< 0.5 and
msg.ranges[360]> 0.5 and msg.ranges[270]> 0.5 and msg.ranges[180]>
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(20)

          p2.ChangeDutyCycle(53)
```

```
#21 SERONG KANAN (Jika titik yang aktif adalah 540 dan 450)

if msg.ranges[540]< 0.5 and msg.ranges[450]< 0.5 and
msg.ranges[360]> 0.5 and msg.ranges[270]> 0.5 and msg.ranges[180]>
0.5:

          GPIO.output(in1,GPIO.HIGH)

          GPIO.output(in2,GPIO.LOW)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(20)

          p2.ChangeDutyCycle(53)

#22 BELOK KANAN (Jika titik yang aktif adalah 540, 450, 360, 270,
dan 180)

if msg.ranges[540]< 0.5 and msg.ranges[450]< 0.5  and
msg.ranges[360]< 0.5 and msg.ranges[270]< 0.5 and msg.ranges[180]<
0.5:

          GPIO.output(in1,GPIO.LOW)

          GPIO.output(in2,GPIO.HIGH)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(0)

          p2.ChangeDutyCycle(50)

#23 BELOK KANAN (Jika titik yang aktif adalah 540, 450, 360, dan
270)

if msg.ranges[540]< 0.5 and msg.ranges[450]< 0.5 and
msg.ranges[360]< 0.5 and msg.ranges[270]< 0.5 and msg.ranges[180]>
0.5:

          GPIO.output(in1,GPIO.LOW)

          GPIO.output(in2,GPIO.HIGH)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(0)

          p2.ChangeDutyCycle(50)
```

```
#24 BELOK KANAN (Jika titik yang aktif adalah 540, 450, dan 360)

if msg.ranges[540]< 0.5 and msg.ranges[450]< 0.5 and
msg.ranges[360]< 0.5 and msg.ranges[270]> 0.5 and msg.ranges[180]>
0.5:

          GPIO.output(in1,GPIO.LOW)

          GPIO.output(in2,GPIO.HIGH)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(0)

          p2.ChangeDutyCycle(50)

#25 BELOK KANAN (Jika titik yang aktif adalah 540 dan 360)

if msg.ranges[540]< 0.5 and msg.ranges[450]> 0.5 and
msg.ranges[360]< 0.5 and msg.ranges[270]> 0.5 and msg.ranges[180]>
0.5:

          GPIO.output(in1,GPIO.LOW)

          GPIO.output(in2,GPIO.HIGH)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(0)

          p2.ChangeDutyCycle(50)

#26 BELOK KANAN (Jika titik yang aktif adalah 450 dan 360)

if msg.ranges[540]> 0.5  and msg.ranges[450]< 0.5 and
msg.ranges[360]< 0.5 and msg.ranges[270]> 0.5 and msg.ranges[180]>
0.5:

          GPIO.output(in1,GPIO.LOW)

          GPIO.output(in2,GPIO.HIGH)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(0)

          p2.ChangeDutyCycle(50)
```

```
#27 BELOK KANAN (Jika titik yang aktif adalah 540, 450, 380, dan
270)

if msg.ranges[540]< 0.5 and msg.ranges[450]< 0.5 and
msg.ranges[380]< 0.5 and msg.ranges[270]< 0.5 and msg.ranges[180]>
0.5:

          GPIO.output(in1,GPIO.LOW)

          GPIO.output(in2,GPIO.HIGH)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(0)

          p2.ChangeDutyCycle(50)

#28 BELOK KANAN (Jika titik yang aktif adalah 540, 450 ,dan 380)

if msg.ranges[540]< 0.5 and msg.ranges[450]< 0.5 and
msg.ranges[380]< 0.5 and msg.ranges[270]> 0.5 and msg.ranges[180]>
0.5:

          GPIO.output(in1,GPIO.LOW)

          GPIO.output(in2,GPIO.HIGH)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(0)

          p2.ChangeDutyCycle(50)

#29 BELOK KANAN (Jika titik yang aktif adalah 540 dan 380)

if msg.ranges[540]< 0.5 and msg.ranges[450]> 0.5 and
msg.ranges[380]< 0.5 and msg.ranges[270]> 0.5 and msg.ranges[180]>
0.5:

          GPIO.output(in1,GPIO.LOW)

          GPIO.output(in2,GPIO.HIGH)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(0)

          p2.ChangeDutyCycle(50)
```

```
#30 BELOK KANAN (Jika titik yang aktif adalah 450 dan 380)

if msg.ranges[540]> 0.5  and msg.ranges[450]< 0.5 and
msg.ranges[380]< 0.5 and msg.ranges[270]> 0.5 and msg.ranges[180]>
0.5:

          GPIO.output(in1,GPIO.LOW)

          GPIO.output(in2,GPIO.HIGH)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(0)

          p2.ChangeDutyCycle(50)

#31  BELOK KANAN (Jika titik yang aktif adalah 368)

if msg.ranges[540]> 0.5  and msg.ranges[450]> 0.5 and
msg.ranges[368]< 0.5 and msg.ranges[270]> 0.5 and msg.ranges[180]>
0.5:

          GPIO.output(in1,GPIO.LOW)

          GPIO.output(in2,GPIO.HIGH)

          GPIO.output(in3,GPIO.LOW)

          GPIO.output(in4,GPIO.HIGH)

          p1.ChangeDutyCycle(0)

          p2.ChangeDutyCycle(55)

rospy.init_node('data_lidar')

sub = rospy.Subscriber('/scan',LaserScan, callback)

rospy.spin()
```

Program *Mapping* dan *Localization*

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


<launch>
  <arg name="tf_map_scanmatch_transform_frame_name"
default="scanmatcher_frame"/>


  <arg name="base_frame" default="base_link"/>
  <arg name="odom_frame" default="base_link"/>
  <arg name="pub_map_odom_transform" default="true"/>
  <arg name="scan_subscriber_queue_size" default="5"/>
  <arg name="scan_topic" default="scan"/>
  <arg name="map_size" default="2048"/>


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


    <!-- Frame names -->
    <param name="map_frame" value="map" />
    <param name="base_frame" value="base_link" />
    <param name="odom_frame" value="base_link" />


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


    <!-- Map size / start point -->
    <param name="map_resolution" value="0.020"/>
    <param name="map_size" value="$(arg map_size)"/>
    <param name="map_start_x" value="0.5"/>
    <param name="map_start_y" value="0.5" />
    <param name="map_multi_res_levels" value="2" />


    <!-- Map update parameters -->
    <param name="update_factor_free" value="0.4"/>
```

```
    <param name="update_factor_occupied" value="0.9" />
    <param name="map_update_distance_thresh" value="0.01"/>
    <param name="map_update_angle_thresh" value="0.9" />
    <param name="laser_z_min_value" value = "-1.0" />
    <param name="laser_z_max_value" value = "1.0" />
    <param name="map_pub_period" value = "0.1" />

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

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

    <param name="tf_map_scanmatch_transform_frame_name"
value="$(arg tf_map_scanmatch_transform_frame_name)" />
  </node>

  <node pkg="tf" type="static_transform_publisher"
name="base_to_laser_broadcaster" args="0 0 0 0 0 0 base_link laser
100"/>
</launch>
```