IMPLEMENTASI PEMETAAN PADA ROBOT BERGERAK OTONOM UNTUK NAVIGASI DALAM RUANGAN MENGGUNAKAN SENSOR RPLIDAR, ALGORITME HECTOR SLAM DAN ALGORITME DIJKSTRA

SKRIPSI

Untuk memenuhi sebagai persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
Bambang Gunawan Tanjung
NIM: 185150300111051

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2021

PERSETUJUAN

IMPLEMENTASI PEMETAAN PADA ROBOT BERGERAK OTONOM UNTUK NAVIGASI DALAM RUANGAN MENGGUNAKAN SENSOR RPLIDAR, ALGORITME HECTOR SLAM DAN ALGORITME DIJKSTRA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
Bambang Gunawan Tanjung
NIM: 185150300111051

Dosen Pembimbing I	Dosen Pembimbing 2
Rizal Maulana, S.T., M.T., M.Sc.	Rakhmadhany Primananda, S.T., M.Kom.
NIK: 2016078910091001	NIK: 2016098604061001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 20 Desember 2021

Bambang Gunawan Tanjung

NIM: 185150300111051

PRAKATA

Syukur Alhamdulillah penulis panjatkan kehadirat Allah SWT atas berkat rahmat, hidayah dan ridho-Nya sehingga penulis dapat menyelesaikan tugas akhir "skripsi" sebagai salah satu syarat yang diperlukan untuk menyelesaikan Program Sarjana (S1) Program Studi Teknik Komputer Fakultas Ilmu Komputer Universitas Brawijaya.

Dalam penyelesaian skripsi ini, penulis melakukannya dengan senang hati, penuh dengan rasa keingintahuan tinggi serta kebanggaan, karena ini merupakan kegemaran penulis atas dunia robot dan pemrograman, khususnya robot besar implementatif yang mampu menjalankan tugas secara nyata. Oleh karena itu penulis mengajukan skripsi yang berjudul "Implementasi Pemetaan Pada Robot Bergerak Otonom Untuk Navigasi Dalam Ruangan Menggunakan Sensor Rplidar, Algoritme Hector Slam Dan Algoritme Dijkstra" yang mana topik ini akan membahas permasalahan utama pada robot bergerak yaitu lokalisasi, pemetaan dan navigasi. Diharapkan proses ini akan berjalan secara otomatis pada robot bergerak dengan bantuan sensor jarak.

Dalam kepenulisan skripsi ini, penulis telah menempuh mata kuliah pendukung sistem kendali (CCE62160) dan sistem cerdas (CCE61151), guna mengimplementasikan topik pada riset ini. Selain itu dalam penyusunan skripsi ini, tidak akan terselesaikan tanpa adanya bantuan serta kemurahan hati dari berbagai pihak. Maka dari itu tidak lupa penulis panjatkan terima kasih kepada seluruh pihak yang telah memberikan inspirasi, bantuan, dan dukungan moral ataupun finansial. Ucapan terima kasih juga penulis sampaikan kepada Bapak Rizal Maulana, S.T., M.T., M.Sc. selaku dosen pembimbing pertama dan Bapak Rakhmadhany Primananda, S.T., M.Kom. selaku dosen pembimbing kedua atas bimbingan motivasi, saran dan diskusi yang diberikan kepada penulis, mulai dari persiapan usulan penelitian hingga penulis mampu menyelesaikan skripsi ini dengan baik.

Selanjutnya dengan penuh rasa hormat penulis mengucapkan terima kasih kepada kedua orang tua tercinta, Syukri Amsyah Tanjung dan Misnawati, karena semua keberhasilan ini tidak luput dari jasa, pengorbanan, limpahan cinta kasih sayang, dukungan moral dan finansial serta doa beliau yang selalu dipanjatkan setiap hari.

Dalam kesempatan ini penulis juga mengucapkan terima kasih sebesar-besarnya kepada:

1. Bapak Achmad Basuki, S.T., M.M.G., Ph.D. selaku ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
2. Bapak Eko Setiawan, S.T., M.Eng., Ph.D. selaku ketua Program Studi Teknik Komputer Fakultas Ilmu Komputer Universitas Brawijaya.

3. Bapak Gembong Edhi Setyawan, S.T., M.T., Bapak Dahnial Syauqy, S.T., M.T., M.Sc. dan Ibu Edita Rosana Widasari, S.T., M.T., M.Eng., Ph.D. selaku dosen penasehat akademik saya selama melaksanakan perkuliahan di Universitas Brawijaya.
4. Bapak dan Ibu dosen yang telah mengajarkan saya mata kuliah selama di teknik komputer maupun memberikan dukungan selama perkuliahan saya.
5. Teman-teman terdekat saya yang selalu memberikan dukungan selama perkuliahan dan mendorong saya untuk cepat menyelesaikan skripsi.
6. Teman-teman seperjuangan teknik komputer baik dari angkatan 2016, 2017, 2018, 2019, dan 2020 yang juga membantu saya pada pengerjaan skripsi ini maupun selama perkuliahan saya.
7. Nikolas Fauth dan kawan-kawan yang atas dasar idenya untuk mendaur ulang *hoverboard* bekas, sehingga penulis menemukan ide untuk membuat robot yang dibuat pada skripsi ini
8. Anggota Komunitas Hack *Hoverboard* Internasional yang senantiasa membantu menjawab pertanyaan ketika penulis sedang mengalami kesulitan.
9. Anggota Komunitas Robaka yang senantiasa melakukan diskusi mengenai perangkat lunak *hoverboard* dan menanggapi setiap error yang terjadi yang dilakukan oleh setiap orang yang sedang melakukan riset.

Akhir kata, penulis menyadari bahwasannya dalam penyusunan skripsi ini terdapat banyak kekurangan, baik berupa tata kepenulisan ataupun aspek lainnya didalam skripsi ini. Untuk itu, penulis menerima dengan lapang dada atas segala kritikan ataupun saran yang diberikan oleh pihak pembaca ataupun peneliti lainnya agar kedepannya penulis dapat memperbaiki karya tulis ini ataupun karya tulis selanjutnya menjadi lebih baik lagi.

Malang, 20 Desember 2021

Penulis

bambanggunawan@student.ub.ac.id

ABSTRAK

Bambang Gunawan Tanjung, Implementasi Robot Pengantar Barang Menggunakan Sensor RPLIDAR Dan Kombinasi Algoritme Simultaneous Localization And Mapping Pada Robot Operating System.

Pembimbing: Rizal Maulana, S.T., M.T., M.Sc. dan Rakhmadhany Primananda, S.T., M.Kom.

Bagian ini diisi dengan abstrak dalam Bahasa Indonesia. Abstrak adalah uraiansingkat (umumnya 200-300 kata) yang merupakan intisari dari sebuah skripsi. Abstrak membantu pembaca untuk mendapatkan gambaran secara cepat dan akurat tentang isi dari sebuah skripsi. Melalui abstrak, pembaca juga dapat menentukan apakah akan membaca skripsi lebih lanjut. Oleh karena itu, abstrak sebaiknya memberikan gambaran yang padat tetapi tetap jelas dan akurat tentang (1) apa dan mengapa penelitian dikerjakan: sedikit latar belakang, pertanyaan atau masalah penelitan, dan/atau tujuan penelitian; (2) bagaimana penelitian dikerjakan: rancangan penelitian dan metodologi/metode dasar yang digunakan dalam penelitian; (3) hasil penting yang diperoleh: temuan utama, karakteristik artefak, atau hasil evaluasi artefak yang dibangun; (4) hasil pembahasan dan kesimpulan: hasil dari analisis dan pembahasan temuan atau evaluasi artefak yang dibangun, yang dikaitkan dengan pertanyaan/tujuan penelitian.

Yang harus dihindari dalam sebuah abstrak diantaranya (1) penjelasan latar belakang yang terlalu panjang; (2) sitasi ke pustaka lainnya; (3) kalimat yang tidak lengkap; (3) singkatan, jargon, atau istilah yang membingungkan pembaca, kecuali telah dijelaskan dengan baik; (4) gambar atau tabel; (5) angka-angka yang terlalu banyak.

Di akhir abstrak ditampilkan beberapa kata kunci (normalnya 5-7) untuk membantu pembaca memposisikan isi skripsi dengan area studi dan masalah penelitian. Kata kunci, beserta judul, nama penulis, dan abstrak biasanya dimasukkan dalam basis data perpustakaan. Kata kunci juga dapat diindeks dalam basis data sehingga dapat digunakan untuk proses pencarian tulisan ilmiah yang relevan. Oleh karena itu pemilihan kata kunci yang sesuai dengan area penelitian dan masalah penelitian cukup penting. Pemilihan kata kunci juga bisa didapatkan dari referensi yang dirujuk.

Kata kunci: abstrak, skripsi, intisari, kata kunci, artefak

ABSTRACT

Bambang Gunawan Tanjung, Implementation of Goods Delivery Robot Using RPLIDAR Sensor and Combination of Simultaneous Localization And Mapping Algorithm on Robot Operating System.

Supervisor: Rizal Maulana, S.T., M.T., M.Sc. and Rakhmadhany Primananda, S.T., M.Kom.

This section is filled with abstracts in Indonesian. Abstract is a short description (usually 200-300 words) which is the essence of a thesis. Abstract helps readers to get a quick and accurate picture of the contents of a thesis. Through the abstract, the reader can also determine whether to read the thesis further. Therefore, the abstract should provide a concise but clear and accurate description of (1) what and why the research was carried out: a little background, research questions or problems, and/or research objectives; (2) how the research is carried out: the research design and the basic methodology/methods used in the research; (3) important results obtained: main findings, characteristics of artifacts, or evaluation results of built artifacts; (4) the results of the discussion and conclusions: the results of the analysis and discussion of the findings or evaluation of the built artifacts, which are related to the research questions/objectives.

Things to avoid in an abstract include (1) a background explanation that is too long; (2) citation to other libraries; (3) incomplete sentences; (3) abbreviations, jargon, or terms that confuse the reader, unless they are well explained; (4) pictures or tables; (5) the numbers are too many.

At the end of the abstract, several keywords are displayed (normally 5-7) to help the reader position the content of the thesis with the area of study and research problem. Keywords, along with title, author name, and abstract are usually entered in the library database. Keywords can also be indexed in the database so that they can be used for the process of searching for relevant scientific papers. Therefore, the selection of keywords that are appropriate to the research area and research problem is quite important. Selection of keywords can also be obtained from referenced references.

***Keywords**: abstract, thesis, essence, keywords, artifact*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

Dalam bab pendahuluan ini, penulis akan menjabarkan mengenai latar belakang penelitian, rumusan masalah, manfaat penelitian, serta sistematika penulisan. Penjelasan dari bab ini akan menjadi penggambaran hal-hal yang ingin disampaikan dalam penelitian ini pada bab-bab selanjutnya.

1.1 Latar Belakang

Pertumbuhan bisnis logistik di Indonesia mengalami peningkatan yang pesat. Tercatat dalam laporan Asosiasi Logistik Indonesia (ALI) pada Juli 2021 (Rabbi, 2021), bahwa arus pengiriman barang mengalami pertumbuhan hingga 40%. Kenaikan yang tinggi ini banyak disumbang oleh industri farmasi, alat kesehatan dan barang-barang konsumsi. Kenaikan ini dikarenakan oleh pandemi Covid-19 yang membuat masyarakat tidak bisa berpergian atau berbelanja dengan bebas ke supermarket atau mall besar. Sejak awal tahun 2020 dimana pandemi Covid-19 mulai menyebar luas di Indonesia, masyarakat secara masif mulai memanfaatkan internet untuk berbelanja secara online melalui e-*commerce*. Kesimpulan ini didapat dari data yang disajikan oleh katadata.com berupa Frekuensi Penggunaan Jasa Kurir selama Pandemi Covid-19. Hasil survei (MarkPlus, 2020) yang dilakukan MarkPlus, Inc. mencatat penggunaan jasa kurir untuk melakukan pengiriman barang meningkat selama masa pandemi virus corona Covid-19. Secara rinci, ada 39% responden yang mengaku bahwa frekuensi penggunaan jasa kurir meningkat signifikan saat masa pandemi dan 39% responden mengaku bahwa frekuensi penggunaan jasa kurir sedikit meningkat di masa pandemi. Data ini diambil dari total 122 responden (Bayu, 2020). Peningkatan frekuensi jasa kurir yang naik secara signifikan akan berdampak pada cepatnya proses masuk dan keluarnya barang yang terjadi di gudang logistik. Proses suplai yang cepat ini akan mempengaruhi waktu, tenaga dan biaya yang digunakan di dalam gudang penyimpanan.

Pada hasil riset (KIC & Sorclo, 2021) yang dilakukan oleh Katadata Insight Center (KIC) bersama Sirclo menunjukkan, mayoritas atau 61,8% responden memilih jasa pengiriman reguler dengan pertimbangan tepat waktu dan sebanyak 48,5% responden memilih jasa pengiriman reguler karena harga yang lebih murah (Dihni, 2021). Dari data ini dapat disimpulkan, untuk terus meningkatkan transaksi dan kepuasan pelanggan diperlukan ekspedisi pengiriman barang yang memiliki efisiensi waktu, tenaga dan biaya. Akan tetapi jika permasalahan ini diselesaikan dengan menambah jumlah pekerja manusia maka akan meningkatkan biaya produksi dan pengurangan pendapatan ekspedisi, sehingga mempengaruhi biaya pengiriman dan gaji para pekerja. Operator gudang berada di bawah tekanan lebih dari sebelumnya untuk mengikuti pengambilan, pengepakan, pengiriman, dan kegiatan logistik penting lainnya. Dengan transaksi e-commerce dan pengiriman langsung yang akan terus meningkat, kita perlu menemukan cara untuk meningkatkan produktivitas pemenuhan pesanan menjadi prioritas.

Maka dari itu untuk mengatasi permasalahan ini diperlukan robot otonom dalam pengiriman barang dibagian gudang ekspedisi. Dengan menggunakan robot otonom, ekspedisi dapat meningkatkan efisiensi kerja dan mengurangi biaya produksi dalam proses distribusi paket kiriman dari dari titik penyimpan satu ke tempat lain atau bahkan dari gudang satu ke gudang lainnya. Sehingga dengan begini maka peningkatan transaksi dan kepuasan pelanggan di *e-commerce* tidak terhambat dalam proses suplai dari ekspedisi ke pelanggan. Robot bergerak otonom adalah salah satu teknologi terbaru yang memungkinkan operasi gudang yang sibuk memproses lebih banyak pesanan, meningkatkan kapasitas, dan mempertahankan kepuasan pelanggan. Robot otonom bertugas sebagai pekerja pembawa barang atau paket dari gudang penyimpanan menuju gudang pengiriman. Pemanfaatan robot bergerak otonom dapat dilihat secara masif di pabrik-pabrik atau gudang penyimpanan untuk dipekerjakan mengantar atau memindahkan barang dari satu tempat ketempat lainnya. Hal ini dikarenakan robot mampu melakukan pekerjaan *repetitive* dan berat lebih efisien dibandingkan manusia. Operasi gudang regional yang lebih kecil biasanya tidak dapat mengganggu operasi dan menerapkan teknologi baru. Gudang ini juga tidak memiliki kapasitas untuk berinvestasi dalam otomatisasi penuh seperti sistem konveyor. Robot bergerak otonom memungkinkan pendekatan yang fleksibel dan terukur tanpa dibatasi oleh tata letak fasilitas yang ada. Robot ini mengoptimalkan kapasitas dan produktivitas tenaga kerja yang ada, terutama selama peningkatan frekuensi jumlah paket yang ada. Sebelum membahas lebih lanjut mengenai robot bergerak otonom, penulis akan menjelaskan terlebih dahulu pengertian dari robot ini.

Autonomous mobile robot (AMR) atau robot bergerak otonom adalah jenis robot yang dapat memahami dan bergerak di sekitar lingkungannya tanpa diawasi langsung oleh operator atau dibatasi pada jalur yang tetap dan telah ditentukan sebelumnya. Jenis robot ini umumnya berbentuk seperti mobil dengan permukaan datar dibagian belakang atau atap yang digunakan sebagai pijakan barang dan memiliki 2 hingga 4 buah roda atau bahkan lebih. Robot bergerak otonom memiliki serangkaian sensor canggih yang memungkinkannya memahami dan menafsirkan lingkungan yang membantunya melakukan tugas dengan cara dan jalur yang paling efisien (E. Romaine, 2020). Sebelum menentukan jalur yang paling efisien dan memulai navigasi, robot harus mampu melakukan pemetaan area kerja terlebih dahulu sejauh batas area cakupan di dalam ruangan gudang. Untuk melakukan pemetaan diperlukan lokasi robot secara akurat selama proses pemetaan yang dilakukan secara simultan. Ketika peta sudah jadi maka robot baru dapat melakukan penentuan jalur dan akhirnya melakukan navigasi. Proses berurut ini merupakan metode yang dilakukan untuk permasalahan sistem robot didalam ruangan (*Indoor positioning system)* yang memerlukan proses simultan dalam waktu yang singkat, metode komputasi ini disebut dengan *Simultaneous Localization and Mapping* (SLAM) (Galov and Moschevikin, 2014). Robot bergerak otonom ini nantinya akan melakukan pengiriman barang dari satu tempat ke tempat lainnya secara mandiri dan mampu menjaga diri walaupun banyak halangan bergerak seperti manusia disekitarnya. Dengan robot bergerak otonom

ini diharapkan dapat meningkatkan distribusi barang atau paket didalam gudang penyimpanan sehingga bisa mempercepat proses pengiriman hingga sampai ke pelanggan.

Ada beberapa penelitian yang dilakukan oleh peneliti sebelumnya dalam membangun sistem robot bergerak otonom menggunakan metode SLAM, penelitian yang pertama dilakukan oleh Eko Budi Utomo (Utomo, 2015)menggunakan gabungan beberapa sensor (ultrasonik, PSD, encoder) dan 4 buah roda yang dirancang menjadi robot beroda dengan jenis HBE Robocar. Algoritma yang digunakan adalah Hector SLAM dan A star dinamis. Penggunaan Algoritma Hector SLAM digunakan untuk lokalisasi sekaligus pembuatan peta area sekitar di dalam ruangan menggunakan data yang diberikan dari sensor dan memprosesnya menggunakan *robot operating system* (ROS) dan hasilnya akan di visualisasikan kedalam aplikasi RViz menjadi sebuah *grid-grid* penyusun peta. Sensor yang berperan dalam proses pemindaian adalah sensor ultrasonik dan PSD, sedangkan sensor akselerometer berfungsi sebagai pembaca nilai akselerasi robot. Kemudian penggunaan algoritma A star dinamis digunakan untuk *path planning* atau perencanaan jalur navigasi. *Motion* atau gerakan robot akan dilakukan jika perencanaan jalur sudah selesai, robot akan diarahkan pada jalur yang telah dibuat dengan mengirimkan sinyal PWM ke motor driver untuk menggerakkan motor DC. Kekurangan dari penelitian ini ada pada hasil pemetaan yang tidak begitu baik, dikarenakan dari hasil *mapping* menunjukkan *cell occupied* (sel berwarna putih) tidak berada di tepi peta dan waktu komputasi yang dibutuhkan sangat lama. Hal ini disebabkan karena sensor ultrasonik hanya berada di satu sudut yaitu bagian depan robot sedangkan sisi lainnya tidak ada. Kemudian jarak dari sensor ultrasonik yang menyebar dan pendek membuat hasil peta kurang baik, sehingga dibutuhkan sensor yang dapat melakukan pemindaian berputar seluas 360 derajat dengan jarak pembacaan sejauh 1 meter hingga 6 meter dengan output yang sudah terfilter dengan baik.

Penelitian kedua dilakukan oleh David Portugal, Andre Araujo dan Micael S. Couceiro (Portugal, Araújo and Couceiro, 2020) mendeskripsikan mengenai penggunaan sensor seperti *RPLIDAR A2,* Kinect V2 dan IMU untuk menciptakan pemetaan 3D menggunakan *robot operating system* (ROS) menggunakan metode Hector SLAM dan RTAB Map. Di dalam penelitian ini peneliti juga menjelaskan dengan detail proses perancangan perangkat lunak dan di aplikasikan menggunakan ROS beserta semua *library* yang digunakan untuk membangun peta dan navigasi. Kesimpulan dari penelitian ini adalah bahwasannya penggunaan RPLIDAR dan sensor Kinect mampu menciptakan peta 3 dimensi dengan hasil yang sangat baik. Hal ini dikarenakan penggunaan algoritma Hector SLAM dalam melakukan pemetaan secara simultan. Kekurangan dari penelitian ini adalah tidak adanya proses navigasi lanjutan setelah peta 3D sudah terbentuk. Penelitian ini tidak melakukan perencanaan jalur atau *path planning* berdasarkan peta 3D yang sudah dibuat. Peneliti hanya berfokus pada pembuatan peta 3D map, sehingga tidak memanfaatkan lebih lanjut hasil pemetaan tersebut menjadi navigasi robot.

Untuk itu dari kedua permasalahan peneliti tersebut, penulis mengusulkan mengembangkan penelitian menggunakan metode dan algoritma yang sama, dengan penggunaan jenis robot dan sensor yang berbeda. Robot yang dirancang pada penelitian ini merupakan robot dengan jenis transporter berukuran besar yang mampu memenuhi lingkup dan pengaplikasian secara riil pada gudang penyimpanan. Berbeda dengan penelitian sebelumnya yang menggunakan platform robot sudah jadi, pada penelitian ini penulis merakit sendiri robot menggunakan komponen *hoverboard*. Sedangkan untuk sensornya menggunakan sensor RPLIDAR, IMU, dan odometri sebagai input datanya. Sensor RPLIDAR dipilih karena kemampuannya yang mampu melakukan pemindaian sejauh 0.5 meter hingga 6 meter dengan area pemindaian seluas 360 derajat. Kemudian sensor IMU dan odometri dipilih karena kemampuannya dalam pembacaan akselerasi gerak robot. Penelitian bertujuan untuk mengembangkan purwarupa menjadi robot yang layak untuk digunakan secara langsung sesuai kondisi gudang penyimpanan. Dalam penelitian ini algoritma Hector SLAM dan Dijkstra sangat cocok digunakan untuk proses pemetaan dan navigasi pada jenis robot yang digunakan, sehingga keduanya digunakan kembali pada penelitian ini. Metode dan algoritma ini akan diimplementasikan pada unit pemrosesan robot yaitu Nvidia Jetson Nano dan *motherboard hoverboard*. Hasil pembacaan sensor akan dikirimkan ke Jetson Nano untuk diproses menjadi sebuah peta menggunakan algoritma Hector SLAM. Untuk menutupi kekurangan pada penelitian kedua, dipastikan pada penelitian ini peneliti melengkapkan fitur yang seharusnya ada pada robot bergerak otonom yaitu proses navigasi robot. Untuk itu dibutuhkan sebuah algoritma yang dapat melakukan *path planning* atau perencanaan jalur secara global, yaitu dengan algoritma Dijkstra. Algoritma Dijkstra akan bertanggung jawab pada perencanaan jalur dan hasilnya berupa *command* atau perintah yang nantinya dikirimkan ke motherboard *hoverboard* yang sudah dilakukan proses *flash, sehingga* nantinya akan memutar motor BLDC.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan sebelumnya, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana hasil akurasi dari pembacaan sensor RPLIDAR, IMU dan odometri?
2. Bagaimana hasil akurasi dan waktu komputasi algoritma Hector SLAM dalam membuat peta?
3. Bagaimana hasil perhitungan algoritma Dijkstra dalam melakukan perencanaan jalur robot untuk mencapai titik koordinat tertentu?
4. Bagaimana hasil akurasi dan waktu komputasi algoritma Dijkstra terhadap gerak navigasi robot pada perencanaan jalur robot?
5. Bagaimana hasil akurasi navigasi keseluruhan sistem robot pada saat ada atau tidak adanya halangan baik bergerak ataupun diam dalam mencapai koordinat tertentu ketika sedang membawa barang?

1.3 Tujuan

Tujuan dari adanya penelitian ini adalah sebagai berikut:

1. Menguji hasil akurasi dari pembacaan sensor RPLIDAR, IMU dan odometri.
2. Menguji hasil akurasi dan mengetahui waktu komputasi dari algoritma Hector SLAM dalam membuat peta.
3. Menguji hasil perhitungan algoritma Dijkstra dalam melakukan perencanaan jalur robot untuk mencapai titik koordinat tertentu.
4. Menguji hasil akurasi dan mengetahui waktu komputasi yang dibutuhkan algoritma Dijkstra pada perencanaan jalur robot.
5. Menguji hasil akurasi navigasi keseluruhan sistem robot pada saat ada atau tidak adanya halangan baik bergerak ataupun diam dalam mencapai koordinat tertentu ketika sedang membawa barang.

1.4 Manfaat

Pada penelitian ini manfaat yang diharapkan bisa menjadi solusi efisien dalam proses pendistribusian barang didalam gudang penyimpanan ekspedisi, sehingga proses logistik pada sistem ekspedisi pengiriman paket menjadi cepat dan terotomatisasi. Purwarupa robot bergerak otonom ini juga bisa menjadi referensi pengembang robot selanjutnya dalam pembuatan robot secara masif karena biaya pembuatan robot yang cukup terjangkau. Selain itu penelitian ini juga diharapkan bermanfaat bagi akademika di lingkungan Fakultas Ilmu Komputer Universitas Brawijaya sebagai referensi dalam melakukan penelitian mengembangkan robot berukuran besar dalam tugas akhirnya, agar penggunaan seperti perangkat lunak robot operating system (ROS) menjadi masif di lingkungan Fakultas Ilmu Komputer, sehingga mampu meningkatkan pengembangan robot mendekati pengembangan perangkat lunak yang dilakukan di industrial sehingga mampu bersaing dan menjadi nilai portofolio lebih bagi peneliti.

1.5 Batasan Masalah

Untuk menjawab rumusan masalah dan agar penelitian ini dapat terselesaikan dengan tenggat waktu yang ada, maka peneliti akan menerapkan batasan masalah sebagaimana berikut:

1. Penelitian robot hanya berfokus pada satu area yang sudah ditentukan.
2. Robot tidak dapat menaiki tangga ataupun lift secara mandiri.
3. Robot tidak dapat membuka atau menutup pintu.
4. Robot hanya mampu melakukan navigasi berdasarkan koordinat tujuan yang di input dari perangkat lunak.
5. Robot hanya mampu membawa barang tidak lebih dari 70 kg.

6. Robot tidak dapat mendeteksi objek berupa kaca, plastik transparan, air dan benda-benda lainnya yang mampu membelokkan cahaya infrared.

1.6 Sistematika Pembahasan

Tugas akhir skripsi ini terbagi atas tujuh bagian utama dengan penjabaran sebagaimana berikut:

1. BAB 1: Pendahuluan

   Bab ini berisi penjelasan mengenai beberapa hal yang memberitahukan hal-hal mendasar dari sebuah penelitian, seperti latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, waktu dan tempat pelaksanaan, dan sistematika pembahasan dari masing-masing bab.

2. BAB 2: Landasan Kepustakaan

   Bab ini membahas landasan teori yang akan digunakan dalam penelitian ini yang memiliki keterkaitan dengan pemetaan, navigasi, algoritme Hector SLAM, dan algoritme Dijkstra serta penjelasan penggunaan middle ware Robot Operating System (ROS) dan library/framework yang digunakan sebagai penunjang dalam proses penelitian, tentunya tidak keluar dari batasan masalah yang sudah ditetapkan.

3. BAB 3: Metodologi

   Bab ini menjelaskan tahapan apa saja yang perlu dilakukan oleh peneliti untuk menyelesaikan permasalahan yang dibahas dalam penelitian. Bab ini juga memuat informasi tipe penelitian apa yang diambil oleh penulis dan juga pembahasan mengenai strategi dari penelitian.

4. BAB 4: Rekayasa Kebutuhan Sistem

   Bab ini membahas penguraian kebutuhan pokok baik fungsional maupun non fungsional mengenai mekanisme dari perancangan perangkat keras dan perangkat lunak yang akan digunakan dengan penelitian ini.

5. BAB 5: Perancangan dan Impelementasi

   Bab ini berisi penjelasan mengenai proses perancangan pada sistem dan cara pengoperasiannya sebagai jawaban dari permasalahan dan rekayasa kebutuhan yang sudah didefinisikan sebelumnya.

6. Pengujian dan Analisis

   Bab ini membahas percobaan dari hasil perancangan dan implementasi mengenai data dari keluaran sistem dengan bentuk pengujian yang berbeda-beda dan melakukan analisa data dengan parameter yang berbeda untuk bisa membandingkan tingkat akurasi dan keberhasilan sistem.

7. Penutup

Bab ini berisi tentang kesimpulan yang ditarik dari hasil dari pengujian dan analisis yang dilakukan sebelumnya dari sisi perancangan, implementasi, dan pengujian sehingga mampu dijadikan acuan dalam pengembangan penelitian sejenis yang terangkum dalam saran.

BAB 2 LANDASAN KEPUSTAKAAN

Dalam bab landasan kepustakaan ini, penulis akan menjabarkan mengenai tinjauan pustaka yang telah penulis pelajari sebelumnya dan dasar teori yang berisikan teori pendukung penelitian ini. Adapun tinjauan pustaka disini adalah sekumpulan penelitian sebelumnya yang dilakukan oleh peneliti di dunia dalam lingkup serupa dan bagaimana hasil penelitian tersebut. Lalu mengenai dasar teori, penulis akan menjabarkan dasar fundamental agar tercapainya tujuan akhir penelitian ini.

2.1 Tinjauan Pustaka

Penelitian ini merupakan bentuk dari pengembangan dan gabungan dari penelitian-penelitian sebelumnya yang berhubungan dengan pemetaan, lokalisasi dan navigasi. Kajian pustaka dilakukan untuk memperdalam teori dasar yang akan dibahas dan sebagai pembelajaran dari hasil pengujian pada penelitian-penelitian sebelumnya. Pada bagian ini akan menjelaskan tentang penelitian sebelumnya yang akan menjadi landasan kepustakaan dan teori dasar, konsep, cara kerja dan sistem dari kepustakaan yang ada pada tabel 2.1.

Tabel 2.1 Tinjauan Penelitian Sebelumnya

No	Nama Penulis (Tahun), Judul Penelitian	Persamaan	Perbedaan	
			Penelitian Terdahulu	Rencana Penelitian
1	Utomo, Eko Budi. (2015). Autonomous Mobile Robot Berbasis Landmark Menggunakan Particle Filter Dan Occupancy Grid Maps Untuk Navigasi, Penentuan Posisi, Dan Pemetaan	Topik yang dibawa terdapat algoritme yang digunakan pada pemetaan sama, yakni particle filter dan Occupancy grid	Penelitian menggunakan sensor ultrasonik, PSD, akselerometer, sensor kamera dari platform robot HBE Robocar	Penelitian menggunakan sensor RPLIDAR dan IMU dari robot otonom
			Pengujian dilakukan dengan menggunakan gabungan dari sensor kamera, ultrasonik dan odometry	Pengujian dilakukan dengan menggunakan deteksi dari sensor RPLIDAR, posisi sensor akselerometer dan odometry
			Arsitektur akhir penelitian	Arsitektur akhir penelitian

			menggunakan pemetaan Occupancy grid dan navigasi menggunakan Dynamic A*	menggunakan pemetaan Hector SLAM dan navigasi menggunakan Timed Elastic Band
2	Zhang, et al. (2020). 2D LIDAR-Based SLAM and Path Planning for Indoor Rescue Using Mobile Robots	Salah satu algoritme yang digunakan sebagai perbandingan pada penelitian sama, yakni Hector SLAM	Pengujian berfokus pada perbandingan algoritme mapping menggunakan GMapping, Hector SLAM, dan Cartographer untuk mencari hasil pemetaan yang optimal	Pengujian berfokus pada algoritme mapping Hector SLAM saja
		Menggunakan platform perangkat lunak yang sama, yakni ROS	Menggunakan perangkat personal komputer notebook intel i5	Menggunakan perangkat Jetson Nano
3	Debeunne and Vivet, 2020. *A Review of Visual-LIDAR Fusion based Simultaneous Localization and Mapping*	Penelitian menggunakan konsep dan metode yang sama yakni, Pemetaan, lokalisasi dan navigasi	Menggunakan sensor kamera fusion	Menggunakan sensor RPLIDAR untuk mendapatkan kedalaman atau ukuran dari benda sekitar
4	Putra, dkk. 2020. Navigasi Indoor Berbasis Peta Pada Robot Beroda Dengan Platform Robot Operating System	Penelitian menggunakan metode pemetaan yang sama, yakni Hector SLAM	Pengontrol utama robot adalah ODROID-XU4	Menggunakan Jetson nano dan Motor Driver *Hoverboard*

5	Portugal, dkk. 2020. A Guide for 3D Mapping with Low-Cost Sensors Using ROS	Penelitian ini menggunakan sensor dan metode yang sama, yakni RPLIDAR dan Hector SLAM	Hanya melakukan proses pemetaan, tidak melakukan *path planning* dan tidak memanfaatkan peta yang sudah dibuat sehingga tidak adanya proses navigasi	Melakukan proses *path planning* dan memanfaatkan peta yang sudah dibuat sehingga terjadi proses navigasi
			Peta divisualisasikan dalam bentuk 3 dimensi	Peta divisualisasikan dalam bentuk 2 dimensi

2.1.1 Autonomous Mobile Robot Berbasis Landmark Menggunakan Particle Filter Dan Occupancy Grid Maps Untuk Navigasi, Penentuan Posisi, Dan Pemetaan

Penelitian yang dilakukan oleh Eko Budi Utomo, penulis membahas dan mengimplementasikan mengenai 3 proses utama dalam robot bergerak otonom yaitu navigasi, posisi dan pemetaan. Kesimpulan yang didapatkan dari penelitian ini (Utomo, 2015) adalah bahwa sensor yang terbatas seperti kamera dan ultrasonik bisa digunakan sebagai pemetaan dan menerapkan metode SLAM. Hal ini didapat setelah melihat hasil dari pengujian, didapati hasil berupa hasil pemetaan. Namun hasil dari pemetaan tidaklah begitu baik dikarenakan waktu komputasi yang cukup lama, selain itu dari hasil mapping menunjukkan cell occupied (warna putih) tidak berada di tepi peta. Hal ini disebabkan karena sensor ultrasonik pada robot hanya berada di bagian depan robot. Sedangkan di sisi kanan dan kiri pada badan robot tidak terdapat sensor ultrasonik. Kemudian Semakin kecil jumlah partikel, diperlukan iterasi yang lebih banyak dalam mencapai konvergen partikel (estimasi pose (x, y, θ)). Lalu Semakin banyak jumlah partikel, waktu komputasi yang diperlukan semakin lama. Hasil dari algoritmenya Penggunaan algoritme A* dapat membantu robot menemukan jalur terpendek dengan eror pose robot untuk data arah sebesar 4.34 % dan error jarak tempuh robot sebesar 4.8 %.

2.1.2 2D LIDAR-Based SLAM and Path Planning for Indoor Rescue Using Mobile Robots

Penelitian yang dilakukan oleh Xuexi Zhang, dkk., penulis mengimplementasikan 4 algoritme untuk pemetaan yakni GMapping, Hector-SLAM, Cartografer dan RGB-D. Penelitian ini membahas mengenai pemetaan dan navigasi akan tetapi yang lebih terlihat dengan detail adalah proses penggunaan

algoritma untuk proses *Mapping*. Dari pengujian yang dilakukan didapatkan kesimpulan bahwa algoritma GMapping, Hector-SLAM dan Cartographer memiliki performa yang baik di dalam ruangan. Sedangkan RGB-D sangat buruk karena tidak bisa mendeteksi pada kondisi pencahayaan yang redup. Kemudian pada kompetisi RoboCup yang di ikuti peneliti di China didapatkan data bahwa GMapping dan Cartographer (Zhang et al., 2020).

2.1.3 A Review of Visual-LIDAR Fusion based Simultaneous Localization and Mapping

Penelitian yang dilakukan oleh César Debeunne dan Damien Vivet (2021), penelitian ini (Debeunne and Vivet, 2020) berupa review yang membahas mengenai perpaduan Visual kamera dan LIDAR yang berbasis pada SLAM. Peneliti membahas tentang navigasi otonom yang membutuhkan presisi dan kecepatan pada pemetaan dan lokalisasi, peneliti juga menambahkan penggunaan algoritme SLAM banyak digunakan untuk robot bergerak, kendaraan tanpa awak, drone tanpa awak dan robot kapal selam (Debeunne and Vivet, 2020). Beberapa percobaan telah dilakukan untuk memasangkan LIDAR dan sensor visual, tetapi semuanya tetap pada tingkat fusi yang sangat longgar. Fusi terutama dilakukan dengan menggunakan hasil dari kedua langkah odometri, yang berarti bahwa deteksi LIDAR atau deteksi visual tidak dapat saling membantu, dan keputusan dibuat pada langkah yang sangat terlambat saat menggabungkan estimasi perpindahan relatif. Pendekatan lain hanya menggunakan pengukuran kedalaman LIDAR untuk menginisialisasi fitur visual secara langsung. Kesimpulan yang didapat adalah kemampuan LIDAR menggunakan sensor visual seperti kamera kurang termanfaatkan dengan baik.

2.1.4 Navigasi Indoor Berbasis Peta Pada Robot Beroda Dengan Platform Robot Operating System

Penelitian yang dilakukan oleh Tara Anggada Putra, dkk. (Putra, Muliady and Setiadikarunia, 2020) yang mengimplementasikan sistem navigasi indoor berbasis peta pada robot beroda dengan platform Robot Operating System (ROS) menggunakan algoritme Hector Mapping untuk pemetaan. Hasilnya adalah robot berhasil melakukan pemetaan lingkungan indoor dengan eror rata-rata 0,174 meter dan berhasil melakukan lokalisasi dengan eror rata-rata 0,05 meter pada koordinat x, 0,028 meter pada koordinat y, dan 1,506° pada sudut orientasi. *Path planner* memiliki tingkat keberhasilan sebesar 62,5% dalam menghasilkan jalur yang dapat dilewati robot dan tingkat keberhasilan 75% dalam mengikut jalurnya. Robot pada penelitian ini memiliki tingkat eror rata-rata 0,046 meter dalam bergerak menuju koordinat x target 0,072 meter dalam bergerak menuju koordinat y target, dan 5,163° dalam berputar menuju sudut orientasi target. Kesimpulan yang didapatkan pada jurnal ini adalah bahwa pemetaan menggunakan Hector SLAM, lokalisasi data fusion menggunakan Extended Kalman Filter (EKF), path planning dengan Field Dynamic A-Star dan Trajectory tracking dengan pengontrol on-off berhasil di implementasikan dengan ROS. Hasil yang didapat berupa kesimpulan bahwa kecepatan robot berpengaruh terhadap hasil

pemetaan. Lokalisasi Wheel Odometry berhasil menentukan posisi robot dengan tingkat kesalahan kurang dari 0.05 meter. Untuk navigasi peneliti sebelumnya menyarankan untuk menggunakan local Path Planner agar robot dapat beroperasi pada lingkungan dinamis serta menggunakan PID untuk meningkatkan performa robot

2.1.5 Sistem Navigasi Mobile Robot Dalam Ruangan Berbasis Autonomous Navigation

Penelitian yang dilakukan oleh Dwiki Erlangga (Erlangga et al., 2019) yang mengimplementasikan Navigasi Mobile Robot dalam ruangan yang menerapkan sistem kemudi otonom. Pembuatan peta menggunakan algoritme Simultaneous Localization and Mapping (SLAM) (Takleh et al., 2018) yang mengolah data dari sensor kamera RGB-D dan bumper yang di konversi ke laser scan dan point cloud digunakan untuk memperoleh perception. Sedangkan wheel encoder dan gyroscope digunakan untuk mendapatkan data odometry yang digunakan untuk membangun peta perjalanan dengan algoritme SLAM, G mapping dan melakukan autonomous-navigation. Sistem terdiri dari tiga sub-sistem yaitu: sensor sebagai input, single board computer untuk proses, dan aktuator sebagai penggerak. Hasil dari pengujian menunjukkan sistem dapat memberikan data depth yang dikonversi ke laser scan, data bumper, dan data odometry kepada single board computer berbasis ROS sehingga mobile robot yang dikendalikan secara Wireless dari Workstation dapat membangun peta grid 2 dimensi dengan akurasi total error rate sebesar 0.987%. Dengan menggunakan peta, data dari sensor, dan odometry tersebut mobile robot dapat melakukan autonomous-navigation secara konsisten dan mampu melakukan path replanning, menghindari rintangan-rintangan statis dan terus menerus melakukan localization untuk mencapai titik tujuan.

2.1.6 A Guide for 3D Mapping with Low-Cost Sensors Using ROS

Penelitian ini dilakukan oleh David Portugal, Andre Araujo dan Micael S. Couceiro (Portugal, Araújo and Couceiro, 2020) mendeskripsikan mengenai penggunaan sensor seperti *RPLIDAR A2,* Kinect V2 dan IMU untuk menciptakan pemetaan 3D menggunakan *robot operating system* (ROS) menggunakan metode Hector SLAM dan RTAB Map. Di dalam penelitian ini peneliti juga menjelaskan dengan detail proses perancangan perangkat lunak dan di aplikasikan menggunakan ROS beserta semua *library* yang digunakan untuk membangun peta dan navigasi. Kesimpulan dari penelitian ini adalah bahwasannya penggunaan RPLIDAR dan sensor Kinect mampu menciptakan peta 3 dimensi dengan hasil yang sangat baik. Hal ini dikarenakan penggunaan algoritma Hector SLAM dalam melakukan pemetaan secara simultan (Saat et al., 2019).

Berdasarkan kelima tinjauan pustaka di atas, dapat disimpulkan bahwa permasalahan komputasi pada robot bergerak otonom terdiri dari 3 proses utama, yaitu lokalisasi, pemetaan dan navigasi (Utomo, 2015). Proses pemetaan menjadi satu dengan lokalisasi dan algoritma yang paling tepat adalah algoritma Hector SLAM (Kohlbrecher et al., 2014). Kemudian dalam pengimplementasiannya robot yang digunakan pada setiap referensi penelitian tidak dirancang khusus untuk

menjalani tugas tertentu. Pada penelitian tersebut hanya berfokus pada metode komputasi yang digunakan sehingga tidak memfokuskan untuk membuat robot yang bekerja secara riil. Maka dari itu dengan penelitian ini, penulis akan melengkapi kekurangan dari penelitian sebelumnya. Data-data dari penelitian sebelumnya akan dijadikan sebagai metode pembanding dari apa yang penulis temukan dalam penelitian. Penulis akan menggunakan nilai parameter terbaik yang bisa digunakan dan tetap memperhatikan akurasi dan waktu komputasi serta penggunaan daya CPU robot. Dengan demikian, penelitian ini bertujuan untuk membuat purwarupa robot riil sesuai dengan kebutuhan yang perlukan pada distribusi gudang dibidang ekspedisi.

2.2 Dasar Teori

Pada subbab dasar teori menjabarkan terkait teori yang digunakan dalam melakukan penelitian, tujuannya adalah memperjelas sistematika dan kerangka pemikiran dari fakta yang ada dalam rumusan suatu hubungan konsep dari masing-masing ilmu pengetahuan.

2.2.1 Autonomous Mobile Robot (AMR)

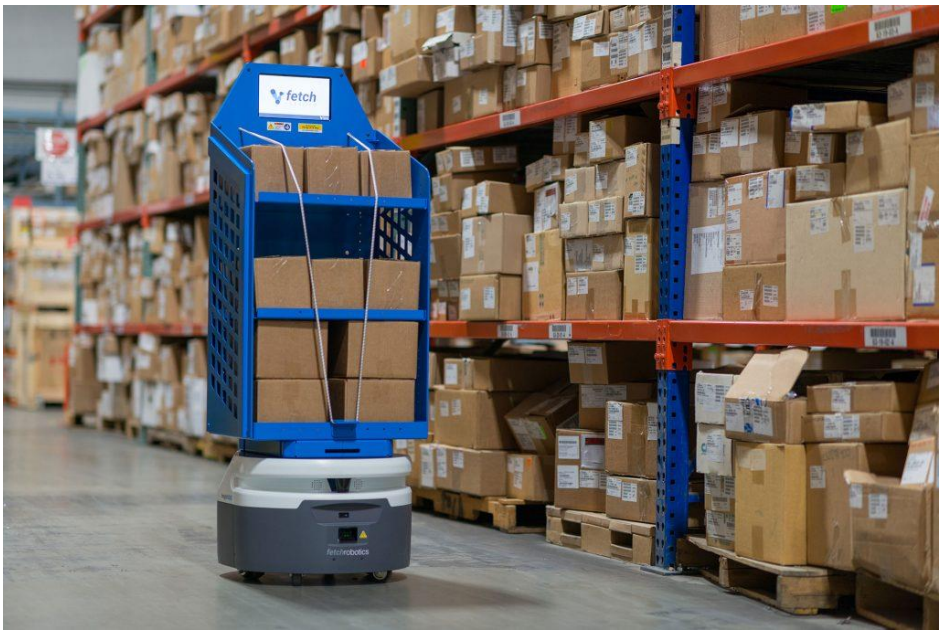
Gambar 2.1 Ilustrasi *Autonomous Mobile Robot*

Sumber : (www.panasonic.com, 2020)

Autonomous robot atau robot otonom merupakan robot yang mampu melakukan perilaku atau tugas dengan sistem otonomi tingkat tinggi (tanpa dipengaruhi input eksternal) dan sistem kontrol diri yang tinggi (Heiserman, 2021). Sedangkan m*obile robot* atau robot bergerak merupakan robot yang mampu bergerak di sekitarnya (Hu, Bhowmick and Lanzon, 2021). Jadi secara garis besar (Intel, 2020) a*utonomous mobile robot* (AMR) atau robot bergerak otonom adalah jenis robot yang dapat memahami dan bergerak di sekitar lingkungannya tanpa diawasi langsung oleh operator atau dibatasi pada jalur yang tetap dan telah ditentukan sebelumnya. Dikatakan robot bergerak karena mempunyai aktuator berupa roda untuk menggerakkan keseluruhan badan robot tersebut, sehingga robot tersebut dapat melakukan perpindahan posisi dari satu titik ke titik yang lain. Robot bergerak biasanya dianggap sebagai sub bidang robotika dan rekayasa informasi.

Robot bergerak memiliki kemampuan untuk bergerak di lingkungan mereka dan tidak terpaku pada satu lokasi fisik. Jenis robot ini umumnya berbentuk seperti mobil dengan permukaan datar dibagian belakang atau atap yang digunakan sebagai pijakan barang dan memiliki 2 hingga 4 buah roda atau bahkan lebih. Robot bergerak otonom memiliki serangkaian sensor canggih yang memungkinkannya memahami dan menafsirkan lingkungan yang membantunya melakukan tugas dengan cara dan jalur yang paling efisien (E. Romaine, 2020). Sebagai perbandingan dengan robot industri, robot bergerak dapat mengandalkan perangkat pemandu yang memungkinkan untuk melakukan perjalanan dengan rute navigasi yang telah ditentukan sebelumnya dalam ruang yang relatif terkendali. Robot bergerak dapat digunakan pada berbagai sektor, akan tetapi secara masif sektor yang lebih sering menggunakan tenaga robot ini adalah sektor yang membutuhkan pengangkutan barang dalam jumlah banyak, misalnya robot bergerak otonom pada bidang bergudangan ekspedisi yang berhubungan dengan logistik, distribusi ataupun penyortiran paket kiriman atau dibidang industri untuk proses pengangkutan bahan mentah atau muatan lainnya di dalam gudang (Erlangga et al., 2019).

2.2.1.1 *Autonomous mobile robot* dibidang gudang



Gambar 2.2 Ilustrasi robot sedang membawa barang didalam gudang

Sumber : (www.robotics247.com, 2020)

Dilansir dari website zetes.com dijelaskan bahwa, Robot bergerak otonom bekerja secara kolaboratif dengan operator gudang, baik itu memindahkan stok selama pemetikan, mengambil pengisian ulang, atau mentransfer stok massal (Zetes, 2021). Robot ini melakukan tugas dengan mengangkut barang ke tahap pemrosesan berikutnya, dengan ini memungkinkan pekerja untuk melanjutkan ke tugas berikutnya sehingga proses distribusi barang menjadi lebih cepat.

Autonomous mobile robot (AMR) memiliki karakteristik dan tugas utama dalam pekerjaannya, meliputi:

a. Beroperasi secara mandiri, AMR tidak memerlukan infrastruktur khusus untuk bekerja. Robot beroperasi dengan mulus dalam tata letak gudang yang ada tanpa gangguan besar pada operasi yang ada.

b. Sensor, peta, dan sistem pemrosesan terpasang memungkinkan robot untuk merencanakan rute dan beradaptasi secara dinamis dengan perubahan di sekitarnya.

c. AMR dapat memindahkan stok di dalam gudang atau antar fasilitas. Robot ini juga dapat melakukan tugas rutin seperti penghitungan inventaris dan pengecekan stok barang.

Operator gudang berada di bawah tekanan lebih dari sebelumnya untuk mengikuti pengambilan, pengepakan, pengiriman, dan kegiatan logistik penting lainnya. Dengan frekuensi transaksi e-commerce dan pengiriman langsung yang akan terus meningkat, menemukan cara untuk meningkatkan produktivitas pemenuhan pesanan menjadi prioritas. Robot bergerak otonom adalah salah satu teknologi terbaru yang memungkinkan operasi gudang yang sibuk memproses lebih banyak pesanan, meningkatkan kapasitas, dan mempertahankan kepuasan pelanggan.

2.2.1.2 *Autonomous mobile robot* dibidang industri

Gambar 2.3 Ilustrasi robot dibidang industri

Sumber : (www. robotnik.eu, 2021)

Robot bergerak otonom pada umumnya dimanfaatkan pada sektor industri sebagai alat untuk proses *intralogistics* (Fernandes et al., 2019). Berdasarkan pengertian dari Invata.com dikatakan bahwa *Intralogistics* adalah seni mengoptimalkan, mengintegrasikan, mengotomatisasi, dan mengelola arus logistik informasi dan barang material di dalam dinding pusat pemenuhan atau distribusi. Untuk melakukan hal ini diperlukan tenaga kerja yang mampu

mendistribusikan semua logistik secara otomatis, maka dari itu biasanya pelaku industri dengan proses alur logistik yang padat seperti pabrik penyimpanan atau *warehouse* akan memanfaatkan robot otonom sebagai pembawa barang seperti yang terlihat pada gambar 2.3.

Aplikasi robotika bergerak di bidang industri apa pun dirancang untuk mengoptimalkan proses produksi apa pun yang terbukti terlalu berbahaya atau berulang bagi pekerja. Oleh karena itu, robot industri bergerak dirancang untuk menjadi alat yang memberikan solusi dan menawarkan fungsionalitas yang menyederhanakan pekerjaan. Robot ini dilengkapi dengan fungsi robot manipulator seluler yang memiliki semua informasi dan ruang untuk gerakan untuk membantunya mengidentifikasi suatu bahan, menanganinya sesuai dengan berat, ukuran, dan kerapuhannya. Robot industri telah menjadi teknologi utama dalam industri ini, terutama untuk tugas-tugas seperti transportasi dan tugas pengambilan material.

2.2.2 Positioning System

Positioning system atau sistem penentuan posisi adalah mekanisme untuk menentukan posisi suatu objek dalam ruang (Hasan et al., 2018). Teknologi untuk tugas ini ada mulai dari cakupan dunia dengan akurasi meter hingga cakupan ruang kerja dengan akurasi sub-milimeter. Cakupan sistem penentuan posisi yang sering digunakan dalam robotika adalah dan *global positioning system (*GPS*), local positioning system (LPS)* dan *indoor positioning system* (IPS).

2.2.2.1 Global positioning system (GPS)

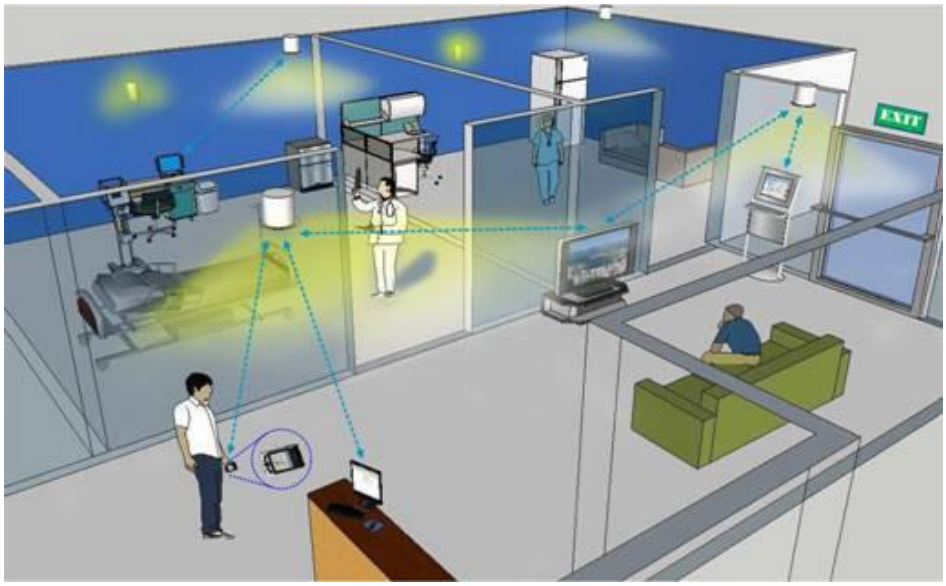
Global Positioning System (GPS) merupakan satu-satunya Sistem Satelit Navigasi Global (GNSS) yang berfungsi penuh. Dua puluh empat satelit GPS saat ini mengorbit Bumi dan mengirimkan sinyal ke penerima GPS, yang menentukan lokasi, arah, dan kecepatan penerima. Sejak satelit eksperimental pertama diluncurkan pada tahun 1978, GPS telah menjadi instrumen penting untuk navigasi, dan alat penting untuk survei tanah dan kartografi. GPS juga menyediakan referensi waktu yang tepat, yang digunakan dalam banyak aplikasi termasuk studi ilmiah tentang gempa bumi dan sinkronisasi jaringan telekomunikasi (Maddison and Ni Mhurchu, 2009).

2.2.2.2 Local positioning system (LPS)

Local Positioning System (LPS) adalah teknik yang digunakan untuk mendapatkan posisi orang atau benda di dalam suatu bangunan yang tercakup oleh jaringan area lokal. Penggunaan LPS telah menjadi suatu kebutuhan yang nyata untuk itu berbagai teknologi telah dipertimbangkan untuk memperkirakan posisi pengguna dengan akurasi yang baik. Ada banyak jenis teknologi di LPS seperti inframerah (IR), identifikasi frekuensi radio (RFID), Bluetooth, Wi-Fi, sistem ultrasound dan sistem berbasis visi (Hasan et al., 2018).

2.2.2.3 Indoor positioning system

Salah satu keterbatasan dalam global positioning system saat ini yaitu perlunya koneksi satelit, sehingga pada kondisi tertentu positioning tidak dapat dilakukan, misalnya saat berada di dalam gedung bertingkat di mana sinyal dari satelit tidak mungkin dicapai sampai ke GPS receiver. Karenanya pada kondisi tertentu, wireless positioning akan sangat dibutuhkan, terutama untuk indoor positioning. *Indoor positioning system* (IPS) adalah jaringan perangkat yang digunakan untuk menemukan orang atau objek di dalam gedung atau ruangan.Sistem penentuan posisi ini cukup rumit, bahkan teknologi GPS dan teknologi satelit lainnya kurang presisi atau bahkan gagal. Sistem ini digunakan pada ruangan tertutup seperti di dalam gedung bertingkat, bandara, gang, garasi parkir, dan lokasi bawah tanah (Chelly and Samama, 2009). Ilustrasi dari *indoor positioning* dapat dilihat pada gambar 2.6



Gambar 2.4 Ilustrasi Local Positioning System

Sumber : (www.comp-eng.binus.ac.id, 2012)

Robot akan bekerja didalam gudang maka dari itu sistem yang akan digunakan pada penelitian ini adalah *local positioning system*. Permasalahan yang ada pada sistem IPS ini adalah lokalisasi untuk deteksi posisi dan proses perencanaan gerak yang membuat ini menjadi permasalahan komputasi. Untuk menyelesaikan permasalahan ini diperlukan metode komputasi yang mampu menyelesaikan lokalisasi dan permasalahan gerak yaitu dengan SLAM.

2.2.3 Simultaneous Localization And Mapping (SLAM)

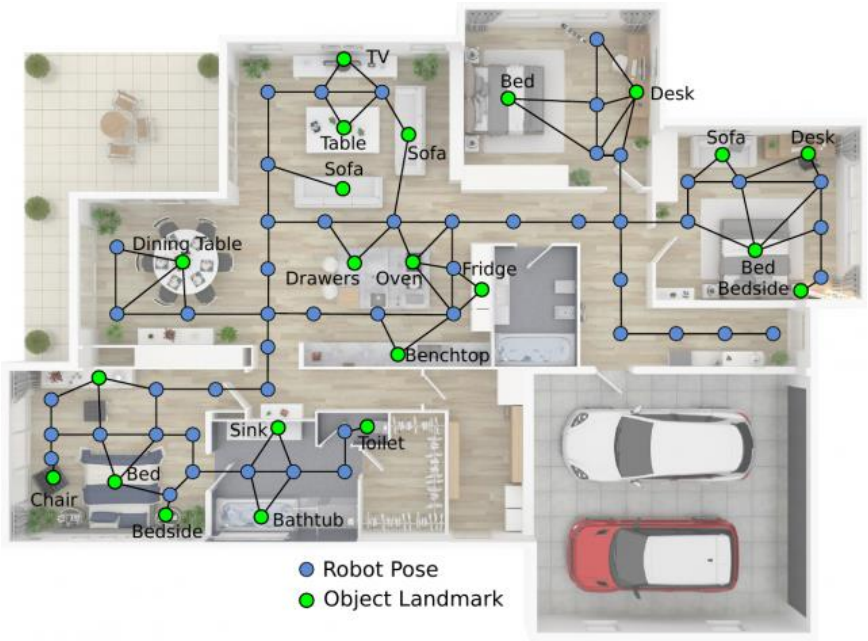
Simultaneous Localization and Mapping atau SLAM adalah sebuah metode komputasi yang dipakai untuk pemetaan menggunakan beberapa perangkat pengindraan atau sensor secara sekaligus. SLAM bukanlah algoritma khusus yang spesifik pada software tertentu, akan tetapi SLAM adalah sebuah konsep komputasi yang menggabungkan beberapa perangkat keras dan perangkat lunak

secara sekaligus untuk pemetaan lingkungan dan navigasi. Hasil akhir pemetaan sekitar dapat dibuat menggunakan pengolahan beberapa algoritma dan menggunakan sensor sebagai alat indra, proses ini dilakukan pada *middleware Robot Operating System* (ROS).

Pada pengembangan robot beroda untuk kasus pembacaan lingkungan dengan kondisi ideal, biasanya SLAM menjadi metode yang paling banyak digunakan dengan pembacaan yang dilakukan pada bidang yang rata. Pada umumnya robot dengan metode SLAM dilengkapi dengan berbagai sensor sebagai indra pembaca, seperti LIDAR atau SONAR, PID, *Deep Sensing Camera* dan sebagainya. Namun, jika melihat tren penelitian, beberapa penelitian menggunakan kamera sebagai alat pembaca keadaan sekitar. Pada teknologi SLAM yang modern, pengolahan menggunakan penggabungan beberapa titik yang terbaca, dan menghitung titik tersebut dengan teknik triangulasi untuk menciptakan data 3D secara simultan memperhitungkan data dari sudut yang lain, pengolahan dari data tersebut akan menghasilkan citra keadaan sekitar dalam bentuk 2 dimensi atau 3 dimensi. SLAM terdiri dari 2 proses utama yaitu *localization* dan *mapping*, berikut ini penjelasannya.

2.2.3.1 Localization

Localization atau lokalisasi adalah menentukan letak dan orientasi (pose) dari robot di sebuah lingkungan. Lokalisasi merupan bagian dari metode SLAM ini dikarenakan untuk melakukan pemetaan dibutuhkan data lokasi robot dengan tepat agar hasil peta sesuai dengan bentuk lokasi aslinya. Untuk melakukan lokalisasi diperlukan sensor yang mampu mendeteksi akselerasi dari robot, yaitu sensor IMU dan encoder. Sensor IMU adalah gabungan dari gyroscope dan akselerometer di dalam komponen elektronik. Sensor encoder memberikan data *odometry* yang nantinya bisa digunakan sebagai variabel untuk lokalisasi. Odometri digunakan untuk menangkap ulang lokasi dan orientasi robot menggunakan kalkulasi dari sensor *rotary encoder* pada roda robot yang bekerja secara *real time* untuk memandu lokasi robot. Namun kita juga bisa menggunakan gabungan dari keduanya, dengan tetap menyesuaikan dari metode algoritma yang ingin digunakan untuk mendapatkan hasil yang paling optimal.



Gambar 2.5 Lokalisasi didalam rumah

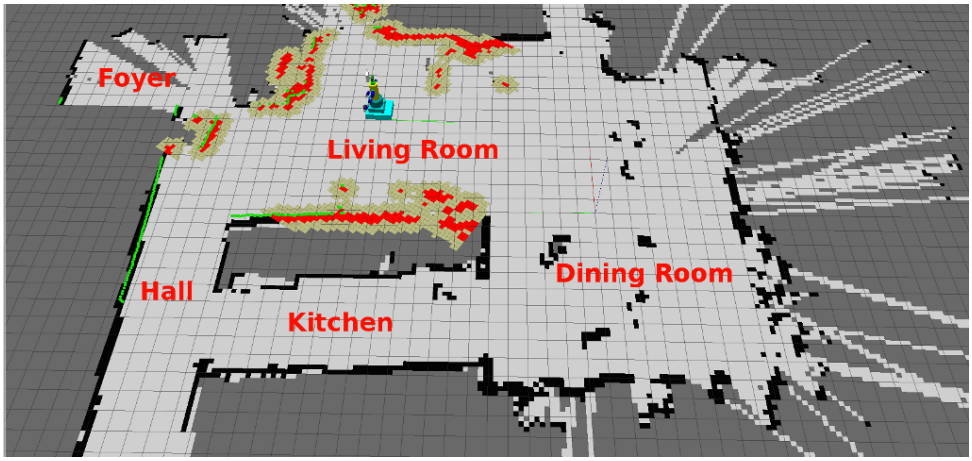
Sumber : (research.qut.edu.au, 2019)

Seperti yang terlihat pada gambar 2.5 setiap posisi dari robot diikuti dengan *object landmark* sehingga orientasi robot akan terus terpantau. Penentuan kondisi robot dalam sekala dalam *localization* terbagi menjadi tiga jenis permasalahan yaitu lokal, global, dan terlokalisasi. Lokalisasi lokal diterapkan pada kondisi ketika robot mengetahui posisi dan orientasi awal. Metode ini dapat mengestimasi posisi robot dengan menggunakan hasil pembacaan sensor robot secara terus menerus meskipun terdapat kesalahan odometri dan kesalahan pada observasi sensor. Sementara itu lokalisasi global diterapkan ketika robot tidak mengetahui posisi dan orientasi awalnya sehingga robot harus melakukan observasi pada lingkungannya agar dapat menentukan posisi. Kondisi relokalisasi merupakan salah satu permasalahan yang lebih kompleks, dimana kondisi ini terjadi ketika robot menyadari bahwa hasil observasi lingkungan robot berbeda dengan hasil pengukuran yang dilakukan pada sampel robot dalam peta tidak sesuai, sehingga menyebabkan kesalahan perhitungan. Kesalahan perhitungan ini akan dihindari menggunakan estimator yang tergabung kedalam paket *robot localization*.

2.2.3.2 Mapping

Mapping atau pemetaan adalah proses pengumpulan data dari dunia fisik dengan banyak sensor untuk membangun sebuah peta, atau proses pencocokan di mana titik-titik suatu himpunan dicocokkan dengan titik-titik himpunan lainnya sehingga membentuk sebuah peta. Pada masalah komputasi SLAM, *mapping* digunakan sebagai salah satu metode yang menjadi bagian dari algoritma. Metode ini merepresentasikan lingkungan yang ditangkap kemudian membuat peta geometris yang akurat. Bentuk dari pemrosesan peta ini bervariasi, salah satunya adalah dengan cara *grid map menggunakan array* (biasanya persegi atau heksagonal) dari yang didiskritkan untuk mewakili peta skala global dan membuat

kesimpulan tentang sel mana yang bisa ditempati dan mana yang tidak bisa (terhalang). Biasanya sel diasumsikan independen secara statistik untuk menyederhanakan perhitungan. Di bawah asumsi seperti ini pada $\mathrm{P}(m_t|x_t, m_{t-1}, o_t)$ disetel nilainya menjadi 1 jika sel peta baru konsisten dengan observasi $\mathcal{O}_t$ pada lokasi x_t dan nilai 0 jika data tidak konsisten. Data 1 dan 0 bisa dimengerti oleh komputer sehingga bisa dimanfaatkan untuk konversi ke dalam bentuk peta visualisasi. Proses ini terus berjalan secara berulang-ulang dan didapatkan bentuk peta dengan 2 warna dasar yaitu hitam yang berarti halangan terdeteksi dan putih yang berarti area tidak terdeteksi adanya halangan apa pun.



Gambar 2.6 2D Mapping di dalam ruangan

Sumber : (www.sifrobot.com, 2018)

Dari gambar 2.6 dapat dilihat bahwa proses mapping menggunakan grid sebagai koordinat alasnya. Saat ini pengembangan permasalahan komputasi pemetaan sudah mampu mencapai hasil yang luar biasa. Kita dapat memvisualisasikan area sekitar secara 3D dengan menggunakan sensor 3D LIDAR. Hal ini sudah bisa banyak ditemukan pada pengembangan *self-driving car* yang terus dikembangkan secara masif di dunia. Namun hingga pada tahun 2021, harga dari sensor 3D LIDAR sangat mahal berkisar antara 30-60 juta di pasaran. Pengembangan sensor ini terus dilakukan demi memperoleh bahan produksi yang lebih murah agar sensor dapat digunakan dan dikembangkan oleh masyarakat umum. Pada penelitian ini, penggunaan sensor 2D LIDAR dirasa sudah sangat cukup untuk jenis permasalahan robot dalam ruangan.

2.2.4 *Library Package Robot Localization*

Robot localization adalah kumpulan estimator keadaan non-linear untuk robot yang bergerak dalam ruang 3D (atau 2D). Setiap estimasi keadaan dapat menggabungkan sejumlah sensor yang berubah-ubah (IMU, odometer, sistem lokalisasi dalam ruangan, penerima GPS…) untuk melacak 15 dimensi (x, y, z, roll, pitch, yaw, $x^{\cdot}$, $y^{\cdot}$, z , $roll^{\cdot}$, $pitch^{\cdot}$, $yaw^{\cdot}$, $x^{\cdot\cdot}$, y, $z^{\cdot\cdot}$) keadaan robot. Dokumentasi paket robot_localization cukup jelas setelah mengetahui cara kerjanya. Namun, tidak memiliki tutorial langsung untuk membantu dengan langkah pertama. Ada beberapa contoh bagus tentang cara menyiapkan paket robot_localization, tetapi

mereka membutuhkan perangkat keras yang berfungsi dengan baik. Pada penelitian ini digunakan algoritma EKF untuk melakukan filtrasi pada hasil lokalisasi.

2.2.4.1 Algoritma Extended Kalman Filter pada Robot Localization

Extended Kalman Filter (EKF) digunakan sebagai salah satu algoritma filtrasi posisi dalam proses lokalisasi pada *package robot_localization* melalui paket driver ROS. Algoritma lokalisasi ini bertujuan untuk mencapai tingkat akurasi yang tinggi dan cakupan yang lebih luas. Node *e*kf_localization_node adalah implementasi dari filter Kalman yang diperluas. Ini menggunakan model gerakan omnidirectional untuk memproyeksikan keadaan ke depan dalam waktu dan mengoreksi perkiraan yang diproyeksikan menggunakan data sensor yang diperoleh. Jumlah parameter relatif besar dan tersedia untuk node estimasi keadaan dalam membuat *launcer* dan konfigurasi. Paket ini berisi *template launcher* dan file konfigurasi untuk membantu pengguna dalam melakukan lokalisasi.

Formulasi dan algoritma EKF sudah terkenal [3, 4, 5]. Kami merincinya di sini untuk menyampaikan detail implementasi yang penting. Tujuan kami adalah memperkirakan pose 3D (6DOF) penuh dan kecepatan robot bergerak dari waktu ke waktu. Proses tersebut dapat digambarkan sebagai sistem dinamis nonlinier, dengan

$$\boldsymbol{x_k} = \boldsymbol{f}(\boldsymbol{x_{k-1}}) + \boldsymbol{w_{k-1}} \tag{2.1}$$

di mana adalah status sistem robot (yaitu, pose 3D) pada waktu , f adalah fungsi transisi status nonlinier, dan 1 adalah derau proses, yang diasumsikan terdistribusi normal. Vektor keadaan 12 dimensi kami, , terdiri dari pose 3D kendaraan, orientasi 3D, dan kecepatannya masing-masing. Nilai rotasi dinyatakan sebagai sudut Euler. Selain itu, kami menerima pengukuran bentuk

$$\boldsymbol{z_k} = \boldsymbol{f}(\boldsymbol{x_k}) + \boldsymbol{v_k} \tag{2.2}$$

di mana adalah pengukuran pada waktu , adalah model sensor nonlinier yang memetakan keadaan ke dalam ruang pengukuran, dan adalah derau pengukuran yang terdistribusi normal. Tahap pertama dalam algoritma, ditunjukkan sebagai persamaan (3) dan (4), adalah melakukan langkah prediksi yang memproyeksikan estimasi keadaan saat ini dan kovarians kesalahan ke depan dalam waktu.

$$\widehat{\boldsymbol{x}}_{\boldsymbol{k}} = \boldsymbol{f}(\boldsymbol{x_{k-1}}) \tag{2.3}$$

$$\widehat{\boldsymbol{P}}_{\boldsymbol{k}} = \boldsymbol{F}\boldsymbol{P_{k-1}}\boldsymbol{F^T} + \boldsymbol{Q} \tag{2.4}$$

Untuk aplikasi kita, adalah model kinematik 3D standar yang diturunkan dari mekanika Newton. Kovarians error estimasi, , diproyeksikan melalui , Jacobian dari , dan kemudian terganggu oleh , kovarians noise proses. Kami kemudian melakukan langkah koreksi dalam persamaan (5) sampai (7):

$$\boldsymbol{K = \widehat{P}_k F^T \left(H \widehat{P}_k H^T + R \right)^{-1}} \quad (2.5)$$

$$\boldsymbol{x_k = \widehat{x}_k + K(z - H\widehat{x}_k)} \quad (2.6)$$

$$\boldsymbol{x_k = \widehat{x}_k + K(z - H\widehat{x}_k)} \quad (2.7)$$

Kami menghitung keuntungan Kalman menggunakan matriks pengamatan kami, , kovarians pengukuran kami, , dan . Kami menggunakan gain untuk memperbarui vektor keadaan dan matriks kovarians. Kami menggunakan persamaan pembaruan kovarians bentuk Joseph [6] untuk meningkatkan stabilitas filter dengan memastikan bahwa tetap positif semi-pasti. Formulasi EKF standar menetapkan bahwa harus berupa matriks Jacobian dari fungsi model observasi . Untuk mendukung array sensor yang luas, kami membuat asumsi bahwa setiap sensor menghasilkan pengukuran variabel status yang kami perkirakan. Dengan demikian, hanyalah matriks identitas. Fitur inti dari ekf_localization_node adalah memungkinkan pembaruan sebagian dari vektor status, yang juga merupakan persyaratan node estimasi status masa depan yang ditambahkan ke robot_localization. Ini penting untuk mengambil data sensor yang tidak mengukur setiap variabel dalam vektor keadaan, yang hampir selalu terjadi. Dalam praktiknya, ini dapat dicapai melalui . Khususnya, ketika hanya mengukur variabel, menjadi matriks dengan 12 peringkat , dengan satu-satunya nilai bukan nol (dalam hal ini, satu) yang ada di kolom variabel yang diukur. Karena kovarians noise proses, , bisa sulit untuk disetel untuk aplikasi tertentu [7], ekf_localization_node memaparkan matriks ini sebagai parameter kepada pengguna, memungkinkan tingkat penyesuaian tambahan.

2.2.5 *Algoritme Hector Simultaneous Localization and Mapping* (Hector SLAM)

Algoritma Hector SLAM digunakan untuk mengkorelasikan perkiraan posisi robot dan peta. *Hector Mapping* merupakan salah satu algoritme pada metode komputasi *Simultaneous Localization and Mapping* (SLAM) yang menghasilkan peta *Occupancy grid.* Pada penelitian ini algoritma memanfaatkan sistem *Light Detection and Ranging* (LIDAR) dengan frekuensi tinggi dengan menggunakan informasi *Odometry sebagai validasi posisinya*. *Hector Mapping* menggunakan algoritme *scan matching* yang dikembangkan dengan cara menyelaraskan data dari sensor untuk membangun peta. *Scan matching* adalah proses menyelaraskan hasil pemindaian (pengukuran jarak) pada waktu t dengan LIDAR dengan hasil pemindaian pada waktu t - 1 yang bertujuan untuk memperoleh posisi robot untuk kemudian memperbarui peta. Metode *scan matching* pada *Hector Mapping* berbasis pada pendekatan Gauss-Newton.

Estimasi optimal dilakukan dengan mencocokkan data laser dan peta secara optimal dalam arti bahwa optimal di bawah ini diselesaikan:

$$\boldsymbol{\xi *= \ arg\ min_{\xi} \sum_{i=1}^{N} \left[1 - \ M\left(S_i(\xi)\right)\right]^2}$$
(2.1)

Di sini $M\left(S_i(\xi)\right)$ adalah nilai dari peta dan pada $S_i(\xi)$ adalah titik koordinat akhir dari semesta $s_i \ = \ \left(s_{i,x}, s_{i,y}\right)^T$ yang memenuhi fungsi berikut:

$$S_i\xi *= \begin{bmatrix} \cos\Psi & -\sin\Psi \\ \sin\Psi & \cos\Psi \end{bmatrix} \begin{bmatrix} S_{i,x} \\ S_{i,y} \end{bmatrix} + \begin{bmatrix} \rho_x \\ \rho_y \end{bmatrix} \quad (2.2)$$

Ketika perkiraan awal dari pose ξ telah diberikan, maka akan memperbarui estimasi dari $\xi + \Delta\xi$ menggunakan pesanan pertama dari rumus perluasan Taylor dan hasilnya meliputi:

$$\Delta\xi = H^{-1} \sum_{i=1}^{N} \left[\nabla M(S_i\xi *) \frac{\partial S_i(\xi)}{\partial \xi}\right]^T \left[1 - M\left(S_i(\xi)\right)\right] \quad (2.3)$$

Ketika H itu adalah matriks *Hessian* atau perkiraan sesuatu dari itu, maka diberikan hasilnya:

$$H = \left[\nabla M(S_i\xi *) \frac{\partial S_i(\xi)}{\partial \xi}\right]^T + \left[\nabla M(S_i\xi *) \frac{\partial S_i(\xi)}{\partial \xi}\right] \quad (2.4)$$

Gambar 2.7 Hasil pemetaan menggunakan Hector SLAM

Sumber : (www.joshvillbrandt.com, 2016)

2.2.6 Path Planning

Path planning atau perencanaan jalur adalah metode komputasi untuk menemukan urutan konfigurasi yang valid yang bisa memindahkan objek dari titik sumber ke titik tujuan. Metode ini berbasis pada *path problem* dan juga digunakan pada masalah komputasi geometri. Contoh algoritme yang berbasis metode ini adalah A*, D*, Djikstra, *Probability Roadmap* dan *Rapidly-exploring random tree*. Pengaplikasian metode komputasi ini cocok digunakan pada navigasi robot, *automation, self-driving car,* robot bedah, animasi karakter digital, dan lain

sebagainya. Metode ini memiliki keunggulan utama terletak pada pemindaian jalur secara dinamis dan kontinu. Terdapat 2 perangkat yang dapat digunakan dalam proses navigasi, yakni o*dometry* dan *inertial measurement unit* (IMU). Odometri memperhitungkan rotasi roda robot untuk membantu mengukur seberapa jauh jaraknya menggunakan pembacaan sensor rotary encoder. IMU juga digunakan untuk mengukur kecepatan dan akselerasi sebagai cara untuk melacak posisi robot menggunakan pembacaan *gyroscope* dan akselerometer.

Konsep dasarnya *path planning* adalah dengan menghitung jalur kontinu yang menghubungkan konfigurasi awal dan konfigurasi tujuan, sambil menghindari tabrakan dengan rintangan yang diketahui. Geometri robot dan rintangan dideskripsikan dalam ruang kerja 2D atau 3D, sedangkan gerakan direpresentasikan sebagai jalur dalam. Di dalam motion planning terdapat 2 hal yang akan selalu dibahas dalam penelitian ini, yaitu *global planner* dan *local planner. Global planner* bekerja dengan informasi yang didapat pada keseluruhan peta, jadi semua data yang ter visualisasi pada peta disebut sebagai *global planner*. Sedangkan *local planner* hanya bekerja dengan informasi yang didapatnya saat ini dari sensor dan merencanakan jalur yang panjangnya hanya sekitar 1 meter.

Terdapat 4 konsep pada proses memahami ruang dan pengambilan jalur pada permasalahan *motion planning* antara lain:

1. *Configuration space,* yaitu melakukan konversi dari dimensi 3 ke dimensi 2 (datar) dan merepresentasikan warna abu-abu sebagai halangan yang telah di konfirmasi atau di atur sedari awal atau sudah dilakukan pemetaan.
2. *Free space*, yaitu proses menghindari segala jenis tabrakan terhadap himpunan konfigurasi dari rintangan yang ada. *Forward kinematic* menentukan posisi geometri robot, dan pengujian deteksi tabrakan apakah geometri robot bertabrakan dengan geometri lingkungan.
3. *Target space*, bagian ruang kosong yang menunjukkan ke mana robot ingin bergerak. Pada *global planning* target dapat diamati oleh sensor robot, namun di dalam *local planning* robot tidak bisa melakukan pengamatan secara langsung karena keterbatasan data jarak demi proses optimal, untuk menyelesaikan permasalahan ini, robot dicoba untuk melewati beberapa ruang target virtual yang masing-masing terletak di dalam area yang dapat diamati di sekitar robot, ruang pada target disebut *sub-goal.*
4. *Obstacle space,* bagian ruang yang tidak bisa dimasuki robot. Ruang ini berlawanan dengan ruang bebas

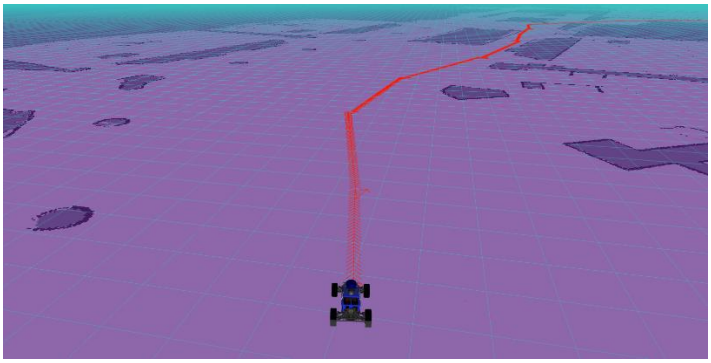
Konsep dasar ini mendasari algoritma-algoritma seperti *Grid-based search, Interval-based search, Geometric algorithm, Artificial potential fields* dan *Sampling-based algorithm.*

Gambar 2.8 Gerak robot menghindari halangan dengan *path planning*

Sumber : (www.venturebeat.com, 2018)

2.2.6.1 Global Planner

Perencana global merencanakan jalur global di sekitar rintangan dan rintangan baru apa pun berdasarkan frekuensi yang ditentukan oleh parameter *frequency planner*. Penghindaran rintangan (pemeriksaan tabrakan) terjadi di perencana lokal tempat cmd_vel diproduksi dan didasarkan pada parameter *controller frequency*. *Local planner* mencoba mengikuti g*lobal planner* sedekat mungkin yang berarti mempertimbangkan bagian dari perencana global pada suatu waktu.

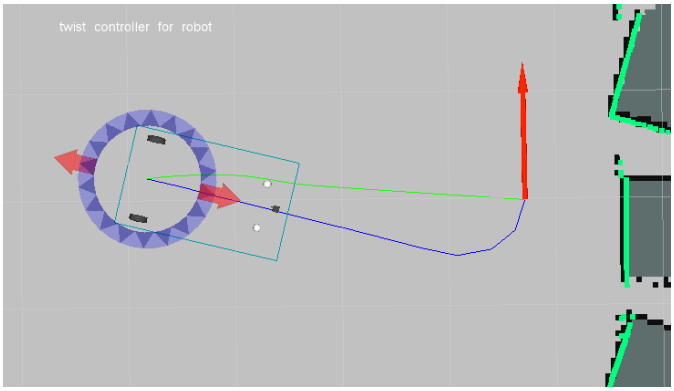


Gambar 2.9 Ilustrasi Global Planner

Sumber : (www. mushr.io, 2021)

2.2.6.2 Local Planner

Local planner atau perencana lokal merupakan metode untuk mengubah jalur global menjadi titik jalan yang sesuai (Marin-Plaza et al., 2018). Jadi, untuk menghitung ulang jalur pada titik tertentu tingkat, peta direduksi menjadi lingkungan kendaraan dan diperbarui saat kendaraan bergerak. Hal ini tidak mungkin untuk gunakan seluruh peta karena sensor tidak dapat memperbarui peta di semua wilayah dan sejumlah besar sel akan meningkat biaya komputasi. Oleh karena itu, dengan lokal yang diperbarui peta dan titik arah global, perencanaan lokal menghasilkan strategi penghindaran untuk rintangan dinamis dan mencoba untuk mencocokkan lintasan sebanyak mungkin ke titik jalan yang disediakan dari perencana global.



Gambar 2.10 Ilustrasi Local Planner

Sumber : (www. nswers.ros, 2021)

2.2.7 Algoritme Dijkstra

Algoritma Dijkstra adalah algoritma untuk menemukan jalur terpendek antara node dalam grafik, yang dapat mewakili suatu jaringan. Dengan Algoritma Dijkstra, kita dapat menemukan jalur terpendek antara node dalam grafik. Khususnya dapat menemukan jalur terpendek dari sebuah simpul (disebut "simpul sumber") ke semua simpul lain dalam grafik, menghasilkan pohon jalur terpendek. Algoritma ini digunakan dalam perangkat GPS untuk menemukan jalur terpendek antara lokasi saat ini dan tujuan. Ini memiliki aplikasi luas di industri, khususnya di domain yang membutuhkan jaringan pemodelan.

Gambar 2.11 Perencanaan Jalur Terpendek Menggunakan Algoritma Djikstra

Sumber : (www.programiz.com, 2020)

Pada gambar 2.14 dapat dilihat bahwa algoritma dijkstra biasanya digunakan pada perencanaan jalur menggunakan GPS atau *Global Positioning System.* Pada penelitian ini, Dijkstra dijadikan algoritma perencanaan jalur menggunakan sistem global, akan tetapi batas global yang digunakan adalah hasil dari pemetaan yang dilakukan sebelumnya.

2.2.7.1 Dasar algoritma Dijkstra

Algoritma Dijkstra terdiri dari beberapa tahapan berupa langkah-langkah secara sistematis, antara lain sebagai berikut:

1. Algoritma Dijkstra pada dasarnya dimulai pada simpul yang dipilih (simpul sumber) dan menganalisis grafik untuk menemukan jalur terpendek antara simpul itu dan semua simpul lain dalam grafik.
2. Algoritme melacak jarak terpendek yang diketahui saat ini dari setiap node ke node sumber dan memperbarui nilai-nilai ini jika menemukan jalur yang lebih pendek.
3. Setelah algoritme menemukan jalur terpendek antara node sumber dan node lain, node tersebut ditandai sebagai "dikunjungi" dan ditambahkan ke jalur tersebut.
4. Proses berlanjut sampai semua node dalam grafik telah ditambahkan ke jalan. Dengan cara ini, kita memiliki jalur yang menghubungkan node sumber ke semua node lain mengikuti jalur terpendek yang mungkin untuk mencapai setiap node.

Algoritma Dijkstra hanya dapat bekerja dengan graf yang memiliki bobot positif. Ini karena, selama proses, bobot sisi harus ditambahkan untuk menemukan jalur terpendek. Dari tahapan tersebut di rumuskan pseudocode sebagai berikut.

Tabel 2.2 Pseudocode Algoritma Dijkstra

Psudocode algoritme Dijkstra

function dijkstra(G, S)
    for each vertex V in G
        distance[V] <- infinite
        previous[V] <- NULL
        If V != S, add V to Priority Queue Q
    distance[S] <- 0
 	
    while Q IS NOT EMPTY
        U <- Extract MIN from Q
        for each unvisited neighbour V of U
            tempDistance <- distance[U] + edge_weight(U, V)
            if tempDistance < distance[V]
                distance[V] <- tempDistance
                previous[V] <- U
    return distance[], previous[]

Setelah melakukan perencaan jalur maka robot sudah dapat melakukan navigasi. Dengan memanfaatkan data dari peta dan tangkapan sensor maka robot bisa bernavigasi dengan lancar dan mampu menghindari halangan bergerak seperti manusia.

2.2.8 Navigasi Robot

Secara terminologi umum *navigation* atau navigasi adalah penentuan kedudukan (*position*) dan arah perjalanan baik di medan sebenarnya atau di peta. Navigasi biasanya digunakan pada istilah pelayaran kapal, penerbangan atau perjalanan. Menurut Montello, D. R. (2005) dalam bukunya yang berjudul "*Navigation"*, dijelaskan bahwa navigasi dikoordinasikan dan gerakan yang

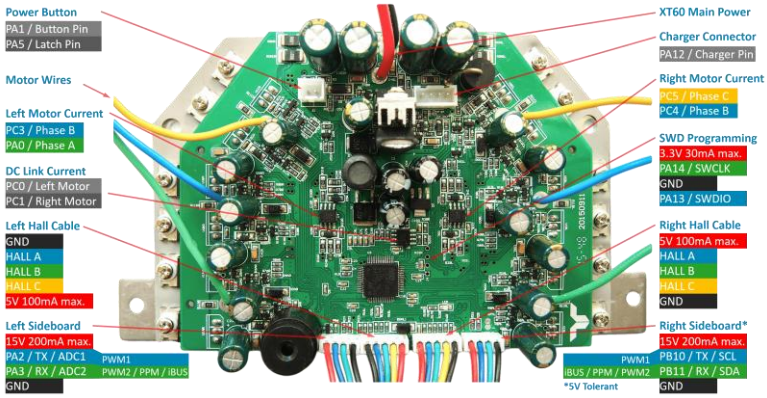
diarahkan pada tujuan melalui lingkungan oleh organisme atau mesin cerdas. Beliau juga menjelaskan bahwa Ilmu saraf telah menunjukkan peran struktur otak tertentu dalam pemeliharaan orientasi dan telah menemukan bukti untuk neuron yang menyala secara istimewa sebagai respons terhadap lokasi atau arah hewan. Peneliti kecerdasan buatan mengembangkan model komputer yang menguji teori kognisi navigasi atau hanya membuat robot yang kompeten (Montello, 2009). Sedangkan untuk pengertian navigasi pada SLAM dikutip dari penelitian Eko Budi Utomo dalam tesisnya mengatakan bahwa navigasi diartikan sebagai proses atau aktivitas untuk merencanakan atau menuju jalur secara langsung dalam sebuah misi yang diberikan pada sebuah autonomous mobile robot dari satu tempat ke tempat yang lain tanpa kehilangan arah atau mengalami tabrakan dengan objek yang lain (Utomo, 2015).

2.2.9 Komponen Utama Robot Pengantar Barang

Komponen utama robot pengantar barang berisikan alat-alat yang digunakan guna membangun sistem robot ini. Pada sub bab ini akan dijelaskan spesifikasi dan pengertian dari setiap komponen yang digunakan. Berikut ini adalah komponen utama yang dibutuhkan dalam membuat robot pengantar barang.

2.2.9.1 Motherboard Hoverboard

Motherboard adalah papan logika utamanya yang berisi unit pemrosesan pusat dan *chip* memori. Pada komputer yang menggunakan arsitektur terbuka, *motherboard* telah memasang selot di mana kartu logika tambahan khusus dapat dimasukkan untuk menambah fungsi primitif yang didukung oleh konfigurasi minimum sistem komputer (Reilly, 2003). Pada penelitian ini, *motherboard hoverboard* digunakan sebagai *mainboard* atau papan pemrosesan utama dari robot sekaligus sebagai driver motor dc. Di dalam *motherboard* terdapat *chip* mikroprosesor dengan seri GD32F103RCT6 yang digunakan sebagai otak dari kontroler sekaligus sebagai driver untuk mengontrol kecepatan dan arah putaran roda. *Motherboard* ini terdiri dari berbagai regulator, input/output, *converter* tegangan, dan berbagai komponen elektronik lainnya seperti transistor, kapasitor serta mosfet. gambar 2.12 merupakan gambar *motherboard hoverboard*.



Gambar 2.12 *Motherboard Hoverboard*

Sumber : (github.com/EFeru/*hoverboard*-firmware-hack-FOC, 2021)

2.2.9.2 Nvidia Jetson Nano

Mikro komputer adalah komputer kecil yang relatif murah dengan mikroprosesor sebagai unit pemrosesan pusat (CPU) (dictionary.com, 2021). Ini termasuk mikroprosesor, memori dan sirkuit input/output (I/O) yang dipasang pada satu papan sirkuit tercetak (PCB) dan Jetson Nano termasuk salah satu mikrokomputer. Pada penelitian ini menggunakan mikrokontroler Jetson Nano sebagai pemrosesan utama kedua setelah mikrokontroler *hoverboard*. Jetson Nano Developer Kit adalah adalah komputer kecil dan kuat yang memungkinkan Anda menjalankan beberapa jaringan saraf secara paralel untuk aplikasi seperti klasifikasi gambar, deteksi objek, segmentasi, dan pemrosesan ucapan (Williman and Jelinek, 1976). Bentuk fisik dari Nvidia jetson nano dapat dilihat pada gambar 2.13.

Gambar 2.13 Nvidia Jetson Nano

Sumber: (www.nvidia.com, 2021)

2.2.9.3 Motor BLDC Hoverboard 6.5Inch 350 Watt

Brushless DC Motor atau motor BLDC adalah motor sinkron yang menggunakan catu daya listrik arus searah (DC). Ini menggunakan pengontrol loop tertutup elektronik untuk mengalihkan arus DC ke belitan motor yang menghasilkan medan magnet yang secara efektif berputar di ruang angkasa dan yang diikuti oleh rotor magnet permanen. Kontroler menyesuaikan fase dan amplitudo pulsa arus DC untuk mengontrol kecepatan dan torsi motor. Sistem kontrol ini merupakan alternatif dari komutator mekanis (sikat) yang digunakan di banyak motor listrik konvensional (Wang, 2012).

Gambar 2.14 Motor BLDC 6.5” 36V 350W dan *build in* Encoder

Sumber : (www.*hoverboard*forlife.blogspot.com, 2016)

Pada gambar 2.14 dapat dilihat bahwa didalam motor BLDC terdapat sensor encoder yang berfungsi untuk mengetahui posisi motor berdasarkan data yang dikirimkan dari HALL sensor. Pembahasan lebih lanjut mengenai Encoder akan dibahas pada bagian selanjutnya.

2.2.9.4 Baterai 18650 Lithium Ion 42V 10S2P 4400mAh

Baterai lithium-ion atau baterai Li-ion adalah jenis baterai isi ulang di mana ion lithium bergerak dari elektroda negatif melalui elektrolit ke elektroda positif selama pelepasan, dan kembali saat pengisian. Baterai Li-ion menggunakan senyawa lithium interkalasi sebagai bahan pada elektroda positif dan biasanya grafit pada elektroda negatif. Baterai Li-ion memiliki kepadatan energi yang tinggi dan self-discharge rendah (Lain, Brandon and Kendrick, 2019).

Gambar 2.15 Baterai Lithium Ion 42V 10S2P 4400mAh

Sumber : (www.streetsaw.com, 2021)

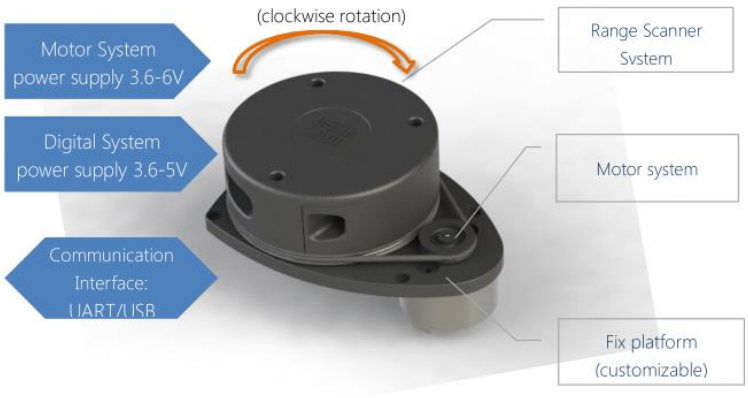
2.2.9.5 STLink V2 Programmer

STLink bukanlah bagian dari komponen utama robot, alat ini digunakan dalam proses *flashing motherboard hoverboard*. Tepatnya STlink berfungsi sebagai perantara USB Flashing antara laptop dengan motherboard. Flashing dilakukan untuk mereset semua kodingan yang ada di motherboard sekaligus menuliskan ulang motherboad dengan kodingan baru yang sudah di modifikasi.

Gambar 2.16 STLink V2

Sumber: (indonesia.alibaba.com, 2021)

2.2.9.6 Sensor RPLIDAR A1



Gambar 2.17 Sensor RPLIDAR

Sumber: (www.RoboPeak.com, 2013)

RPLIDAR (RoboPeak *Light Distance and Ranging*) adalah solusi pemindai laser 2D (LIDAR) 360 derajat berbiaya rendah yang dikembangkan oleh RoboPeak. Sistem dapat melakukan pemindaian 360 derajat dalam jarak 6 meter. Metode yang digunakan adalah LIDAR, LIDAR (*Light detection and ranging*) merupakan teknologi sensor jarak jauh yang menggunakan laser untuk memindai halangan atau dinding dan menghitung jaraknya. Data titik 2D yang dihasilkan dapat

digunakan dalam pemetaan, pelokalan, dan permodelan objek/lingkungan. Pada gambar 2.17 dapat dilihat bentuk fisik dari RPLidar dengan keterangan berupa fitur utama dan nilai tegangannya.

Frekuensi pemindaian RPLIDAR mencapai 5,5 hz saat pengambilan sampel 360 titik setiap putaran. Dan dapat dikonfigurasi hingga maksimum 10 hz. RPLIDAR pada dasarnya adalah sistem pengukuran triangulasi laser. Sistem ini dapat bekerja dengan sangat baik di semua jenis lingkungan dalam dan luar ruangan tanpa sinar matahari. Pada LIDAR 360 derajat, proses ini berlangsung secara berkesinambungan. Hasil *scan* atau pemindaian LIDAR ini berupa titik-titik awan (*point clouds*) dalam bidang 2 dimensi, karena sensor ini hanya memindai bidang horizontal. Jika LIDAR memindai bidang vertikal dan horizontal maka point clouds yang dihasilkan berupa ruang 3 dimensi. Point clouds 3 dimensi memungkinkan pemetaan lebih detail tentang lingkungan robot tapi membutuhkan komputasi yang lebih *powerful*.

RPLIDAR berisi sistem pemindai jangkauan dan sistem motor. Setelah menyalakan setiap sub-sistem, RPLIDAR mulai berputar dan memindai searah jarum jam. Pengguna bisa mendapatkan data pemindaian jarak jauh melalui antarmuka komunikasi (Port serial/USB). RPLIDAR didasarkan pada prinsip rentang triangulasi laser dan menggunakan akuisisi penglihatan berkecepatan tinggi dan perangkat keras pemrosesan yang dikembangkan oleh RoboPeak. Sistem mengukur data jarak lebih dari 2000 kali per detik dan dengan output jarak resolusi tinggi (<1% dari jarak). RPLIDAR memancarkan sinyal laser inframerah termodulasi dan sinyal laser tersebut kemudian dipantulkan oleh objek yang akan dideteksi. Sinyal kembali di sampel oleh sistem akuisisi visi di RPLIDAR dan DSP yang tertanam di RPLIDAR mulai memproses data sampel dan nilai jarak keluaran dan nilai sudut antara objek dan RPLIDAR melalui antarmuka komunikasi. Sistem pemindai rentang kecepatan tinggi dipasang pada rotator yang berputar dengan sistem pengkodean sudut bawaan. Selama rotasi, pemindaian 360 derajat dari lingkungan saat ini akan dilakukan. Hasil dari pemindaian ini akan dikirimkan melalui komunikasi UART melalui adapter UART dan dihubungkan ke Jetson Nano menggunakan port USB. Data yang didapat berupa koordinat dari setiap titik laser yang terbaca. Teknologi Light Detection and Ranging (LIDAR) dengan memanfaatkan teknik *time-of-flight* (TOF) akan dibahas lebih detail pada bagian selanjutnya.

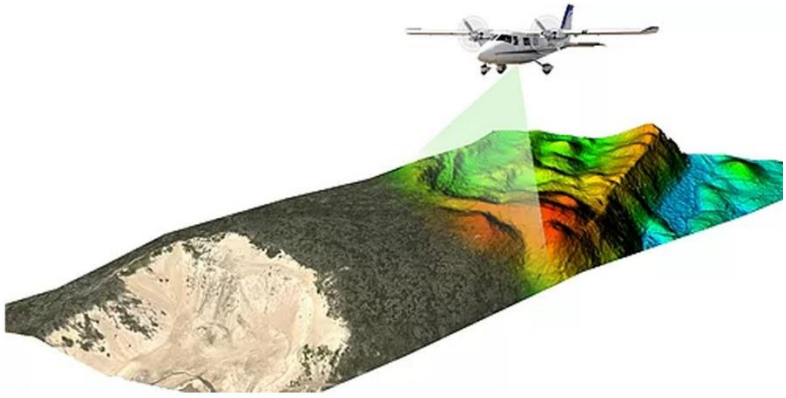
2.2.9.7 Sensor IMU GY-521 MPU-6050 6DOF

Gambar 2.18 Sensor IMU GY-521

Sumber: (www.digiwarestore.com, 2013)

GY-521 MPU-6050 merupakan modul sensor yang berfungsi sebagai akselerometer dan giroskop yang dikemas dalam bentuk satu modul yang kokoh. Modul ini mampu membaca data pada sudut x, y, dan z dalam satu waktu bersamaan. Modul ini sudah dilengkapi dengan fitur ADC atau Analog to Digital Converter dengan resolusi 16-bit sehingga data yang dihasilkan akan lebih teliti dan lebih bagus. Dengan menggunakan sensor ini, maka perubahan maupun pergerakan benda pada ruang dapat diamati perubahan nilai dalam sudut x, y, dan z. GY-521 MPU-6050 juga sudah dilengkapi dengan antarmuka I2C sehingga bisa dihubungkan dengan Arduino Uno, maupun perangkat mikrokontroler lainnya yang memiliki fitur I2C. Selain itu, modul ini juga dilengkapi dengan pin Interrupt (INT) sehingga bisa dimanfaatkan sebagai program interupsi, jika terdapat banyak baris program.

2.2.10 *Light Detection and Ranging* (LIDAR)



Gambar 2.19 Pendeteksian Permukaan Tanah Menggunakan LIDAR

Sumber : (www. digiwarestore.com, 2019)

Light Detection and Ranging atau LIDAR merupakan metode pendeteksian objek yang menggunakan prinsip pantulan sinar laser untuk mengukur jarak objek

yang ada di permukaan. Lidar juga dapat digunakan untuk membuat representasi digital 3-D dari area di permukaan bumi dan dasar laut, karena perbedaan waktu kembali laser, dan dengan memvariasikan panjang gelombang laser. Ini memiliki aplikasi terestrial, udara, dan seluler . Adaptasi teknologi ini pertama kali digunakan untuk keperluan penerbangan yang terjadi pada tahun 1960-an dan baru digunakan sebagai sistem pemetaan pada tahun 1980-an hingga sekarang. Prinsip kerja dari LIDAR ini sendiri sebetulnya cukup sederhana, LIDAR melakukan perhitungan jarak dengan cara mengeluarkan sinar laser transmitter ke suatu permukaan, kemudian dihitung berapa lama waktu yang dibutuhkan oleh sinar laser untuk kembali ke reseptor (Donager, Sánchez Meador and Blackburn, 2021).

Jika dianalogikan, ini sama seperti ketika kita mengarahkan cahaya senter ke suatu permukaan. Sebenarnya yang terjadi adalah kita dapat melihat pantulan cahaya senter dari suatu permukaan ke retina Anda, namun karena prosesnya sangat cepat, maka hal tersebut terjadi seolah-olah secara instan. Kecepatan cahaya sangat tinggi, yaitu sekitar 300.000 KM/detik jadi kita tidak bisa pergerakannya secara langsung. Untuk menghitung jarak, LIDAR menggunakan rumus sebagai berikut:

$$d = \frac{c * t}{2} \tag{2.8}$$

d = jarak antara sensor dan objek yang diukur (m)

c = Kecepatan cahaya (3 x 10^8 m/s)

t = Waktu tempuh cahaya/sinyal

Perangkat LIDAR menembakkan sinar laser secara cepat ke suatu permukaan, bahkan beberapa perangkat LIDAR menembakkan sekitar 150.000 pulsa laser per detik. Kemudian komponen sensor pada LIDAR menghitung waktu yang dibutuhkan dari setiap pulsa laser untuk memantul dari suatu permukaan ke sensor sehingga didapat hasil penghitungan jarak dengan akurasi tinggi

2.2.11 Odometri

Odometri adalah penggunaan data dari pergerakan aktuator untuk memperkirakan perubahan posisi dari waktu ke waktu. Odometri merupakan teknik lokalisasi yang banyak digunakan pada robot beroda, namun teknik ini biasanya memiliki kesalahan yang cukup besar. Akan tetapi algoritme *Extended Kalman Filter* (EKF) dapat digunakan untuk mengurangi kesalahan tersebut dengan menggabungkan data *Wheel Odometry* dengan data sensor lainnya seperti inertial measurement unit (IMU) (Maruf and Hasan, 2017). *Wheel* Odometri merupakan teknik lokalisasi yang menghitung posisi robot berdasarkan perputaran roda (kecepatan putaran roda atau jumlah perputaran roda). Kecepatan putaran roda atau jumlah perputaran roda dapat diukur menggunakan *rotary encoder.*

Odometri digunakan untuk memperkirakan posisi relatif terhadap posisi awal. Untuk memperkirakan posisi relatif robot, digunakan perhitungan jumlah pulsa yang dihasilkan oleh sensor rotary encoder setiap satuan ukuran yang kemudian

dikonversi menjadi satuan millimeter. Untuk mendapatkan jumlah pulsa setiap satu kali putaran roda digunakan rumus sebagai berikut (Utomo, 2015):

$$keliling\ roda = 2\,\pi\,r$$

$$pulsa\ per\ mm = \frac{resolusi\ encoder}{keliling\ roda} \quad (2.9)$$

Pada sistem gerak diferensial terdapat dua roda, yaitu roda kanan dan roda kiri dan dimisalkan jumlah pulsa_per_mm untuk roda kanan adalah *right_encoder* dan roda kiri adalah *left_encoder* dan jarak antara dua roda adalah *wheel_base* maka didapatkan jarak tempuh (*distance*) dan sudut orientasi (θ). Rumusnya adalah sebagai berikut:

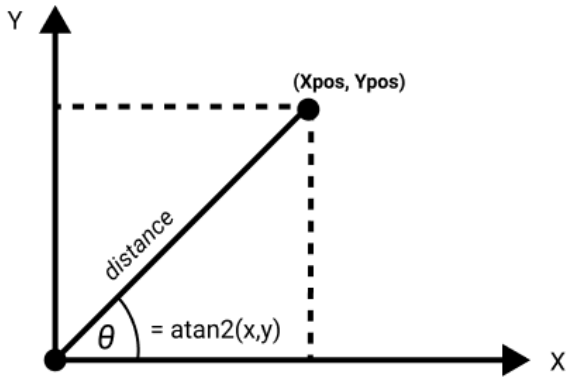
$$distance = \frac{left\ enc + right\ enc}{2}$$

$$\theta = \frac{(left\ enc - right\ enc)}{wheel\_base} \quad (2.10)$$

Karena θ adalah sudut dalam radian maka untuk mengetahui sudut dalam derajat (*heading*) digunakan rumus sebagai berikut :

$$heading = \theta \times \frac{100}{\pi} \quad (2.11)$$

Dari ketentuan diatas didapatkan bahwa nilai *heading* akan bernilai negatif (-) ketika robot berputar melawan arah jarum jam dan akan bernilai positif (+) ketika robot berputar searah dengan jarum jam. Dengan mengetahui jarak dan sudut (*distance* dan θ) maka kita dapat mengetahui koordinat X dan koordinat Y dengan persamaan trigonometri.



Gambar 2.20 Sistem Koordinat Pose

Dari ilustrasi diatas maka koordinat dari robot dapat kita ketahui dengan rumus :

$$X_{pos} = distance\ x \sin\ (\theta)$$

$$Y_{pos} = distance\ x \cos\ (\theta) \quad (2.12)$$

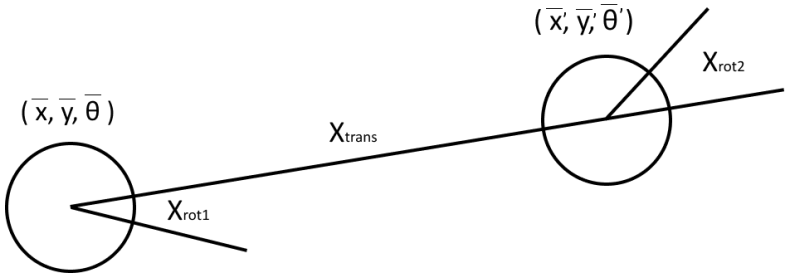
Robot yang bergerak dari $\bar{x},\ \bar{y}, \bar{\theta}$ ke $\bar{x}'$, $\bar{y}'$, $\bar{\theta}'$ maka berlaku translasi dan rotasi dari x_{t-1} ke x_t antara dua titik keadaan robot adalah.

$$X_{trans} = \sqrt{(\vec{x} - \bar{x})^2 + (\dot{\bar{y}} - \bar{y})^2}$$

$$X_{rot\,1} = \text{a}\tan^2(\dot{\bar{y}} - \bar{y},\ \vec{x} - \bar{x}) - \bar{\theta}$$

$$X_{rot\,2} = \overrightarrow{\theta} - \bar{\theta} - X_{rot1} \qquad (2.13)$$

Persama tadi berdasarkan model odometri yang digunaka. Model ini dapat dilihat pada gambar 2.20.



Gambar 2.21 Sistem Odometri Robot

Adapun syarat penggunaan dari $atan^2$

$$atan^2\,(y,x) = \begin{cases} \text{atan}\left(\frac{y}{x}\right), & jika\ x < 0 \\ sign(y)\left(\pi - \text{atan}\left(\left|\frac{y}{n}\right|\right)\right), & jika\ x \geq 0 \\ 0 & jika\ x = y = 0 \\ sign(y)\frac{\pi}{2} & \end{cases} \qquad (2.14)$$

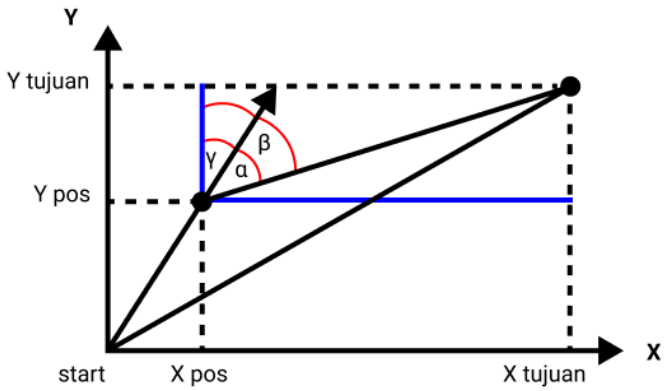
Untuk menentukan error arah hadap dari robot terhadap titik tujuan maka digunakan teorema pythagoras yang akan menghasilkan posisi saat ini dan jarak terhadap titik tujuan, berikut perhitungannya:

$$x = X\ tujuan - X_{pos}$$

$$y = Y\ tujuan - X_{pos}$$

$$target\ distance = \sqrt{x^2 + y^2} \qquad (2.15)$$

Arah hadap dari robot yang telah diketahui sehingga kita dapat menghitung error arah hadap (*heading error*) robot terhadap titik tujuan.



Gambar 2.22 Sistem Koordinat Tujuan

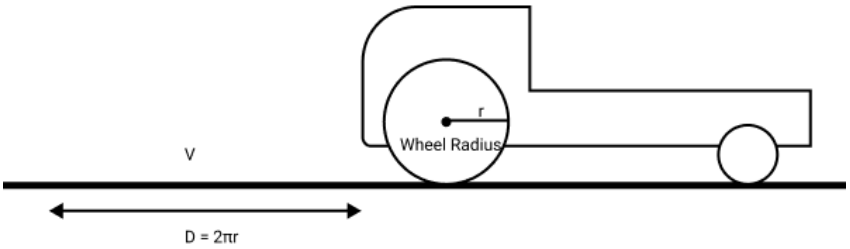
Pada gambar 3.11. menunjukkan ilustrasi untuk mencari *heading error* (α) dimana β adalah target bearing yaitu sudut antara posisi robot saat ini terhadap titik tujuan. Sedangkan garis berwarna biru adalah garis bantu yang masing-masing sejajar dengan sumbu ?? dan sumbu ??. Untuk mendapat nilai dari β, maka digunakan rumus sebagai berikut:

$$\beta = \arctan \frac{(Y\ tujuan - Y\ pos)}{X\ tujuan - Xpos} \quad (2.16)$$

dan

$$\alpha = \beta - \gamma \quad (2.17)$$

Berikut ini adalah motion model pada robot bergerak otonom



Gambar 2.23 Rotasi Roda Untuk Perhitungan Jarak

Diameter roda robot r berputar dengan kecepatan sudut ω maka berlaku:

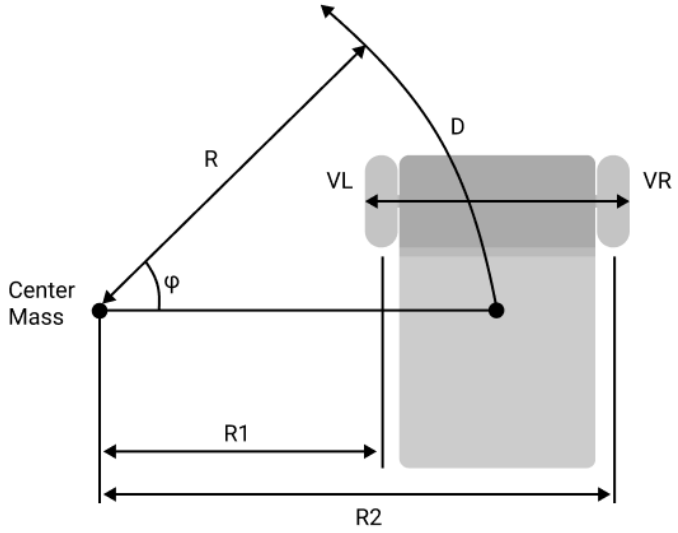
$$V = r.\omega \quad (2.18)$$

Dan jika robot bergerak selama t dengan kecepatan V maka jarak D didapatkan dengan

$$D = V.t \quad (2.19)$$

Jarak D adalah sama dengan keliling roda robot, maka berlaku

$$D = 2\pi r \quad (2.20)$$



Gambar 2.24 Perputaran Roda Robot Pada Sumber Center of Mass

Kemudian untuk radius titik tengah robot R yaitu jarak yang dibentuk oleh R1 dan R2. Dengan kecepatan kiri disimbolkan dengan VL dan kecepatan kanan disimbolkan dengan VR. Berlaku perbandingan:

$$V_L : V_R = V_R : V_2 \tag{2.21}$$

Dengan persamaan diatas maka

$$\frac{V_L}{R\frac{L}{2}} = \frac{V_R}{R\frac{L}{2}} \tag{2.22}$$

Kemudian untuk mendapatkan R radius putaran maka:

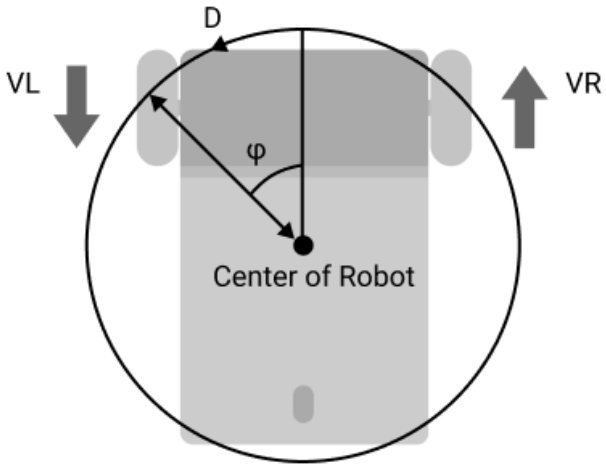
$$R = \frac{L}{2} \times \frac{V_R + V_L}{V_R + V_L} \tag{2.23}$$

Saat R = ∞ dan $V_R = V_L$ robot bergerak maju dan ketika R = 0 dan $V_R = -V_L$ maka robot perputar ditempat. Nilai D bisa didapatkan dengan asumsi kecepatan tidak berubah selama robot bergerak, maka berlaku:

$$D = \frac{V_L + V_R}{2} \times \mathrm{t} \tag{2.24}$$

Dan kita dapatkan sudut putar sebesar

$$\varphi = \frac{D}{R} \tag{2.25}$$



Gambar 2.25 Robot Berputar Ditempat

Untuk gerakan robot perputar ditempat maka berlaku:

$$\theta = \frac{D}{R}\ (radian) \tag{2.26}$$

2.2.12 Inertial Measurement Unit (IMU) 6 Derajat Kebebasan

Inertial Measurement Unit (IMU) merupakan alat ukur yang memanfaatkan sistem pengukuran seperti giroskop dan akselerometer untuk memperkirakan posisi relatif, kecepatan, dan akselerasi dari suatu benda. Sensor ini terdiri atas akselerometer dengan 3 derajat kebebasan dan gyroskop dengan 3 derajat kebebasan.

2.2.12.1 Akselerometer

Akselerometer adalah perangkat elektromekanis yang dapat merasakan gaya statisataupun dinamis dari percepatan linear, yaitu laju perubahan kecepatan suatu benda.Percepatan diukur dalam meter per detik kuadrat (m/s). Kekuatan statis termasuk gravitasi,sementara kekuatan dinamis dapat mencakup getaran dan gerakan. Akselerometer berisi piringkapasitif internal. Beberapa di antaranya bersifat tetap, sementara yang lain bergerak secarainternal. Selama lempeng ini bergerak dalam hubungan satu sama lain, kapasitansi antaramereka berubah. Dari perubahan-perubahan dalam kapasitansi, percepatan dapat ditentukan.

2.2.12.2 Giroskop

Giroskop adalah perangkat elektromekanikal yang berfungsi mengukur perubahan sudutseputar sumbu tetap terhadap ruang inersia. Satuan kecepatan sudut yang diukur dalam derajatper detik (° / s) atau rotasi per detik (RPS). Sebuah giroskop, dapat mengukur rotasi sekitar tigasumbu yaitu, x, y, dan z.Ketika giroskop diputar, sensor akan beresonansi kecil yang kemudiandiubah menjadi perubahan kecepatan sudut. Getaran inilah yang akan dikonversikan menjadisinyal listrik.

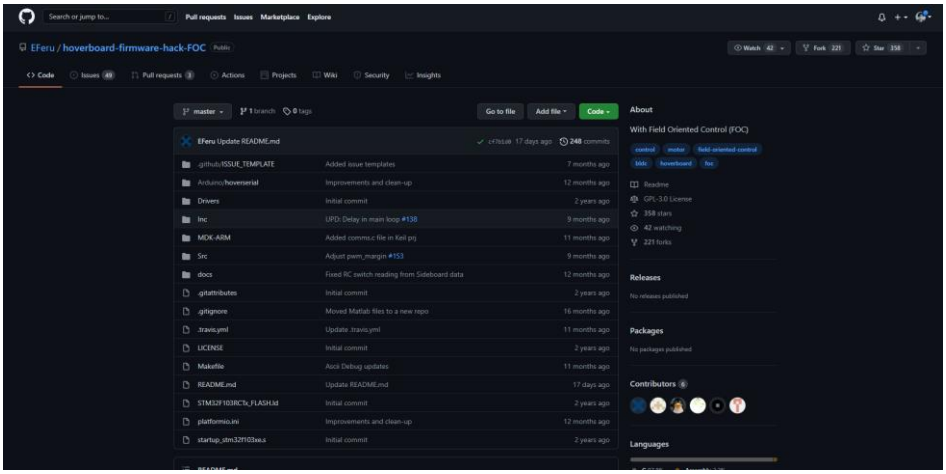
2.2.13 *Firmware*

Dalam komputasi, firmware adalah kelas perangkat lunak komputer tertentu yang menyediakan kontrol tingkat rendah untuk perangkat keras khusus. Firmware, seperti BIOS komputer pribadi, dapat berisi fungsi dasar perangkat dan dapat menyediakan layanan abstraksi perangkat keras ke perangkat lunak tingkat tinggi seperti sistem operasi. Untuk perangkat yang tidak terlalu rumit, firmware dapat bertindak sebagai sistem operasi lengkap perangkat, melakukan semua fungsi kontrol, pemantauan, dan manipulasi data. Contoh umum perangkat yang berisi firmware adalah sistem tertanam (menjalankan perangkat lunak tertanam), peralatan rumah tangga dan penggunaan pribadi, komputer, dan periferal komputer.

2.2.13.1 Hoverboard Firmware

Hoverboard firmware merupakan perangkat lunak khsus yang ditanamkan pada *hoverboard controller* atau *motherboard hoverboard* untuk mengontrol perangkat keras yang ada pada *hoverboard* seperti, motor BLDC, baterai, driver motor, sensor giroskop, sensor akselerometer dan charger. Firmware sangat dibutuhkan sebagai perantara kontrol perangkat keras pada hoverboard. Firmware ditanamkan menggunakan teknik *flashing* kedalam *motherboard hoverboard* sehingga pengembang dapat mengubah isi dari pemrograman motherboard sesuai dengan apa yang diinginkan. Untuk melakukan ini dibutuhkan adapter berupa STLink dan software STMCube Programmer untuk melakukan flashing pada *motherboard hoverboard.* Data yang disimpan kedalam motherboad merupakan data dari file biner hasil compile menggunakan C++. Jadi firmware ditulis menggunakan bahasa pemrograman C dan nantinya dicompile menggunakan compiler. Output dari hasil compile tersebut berupa file biner yang dapat di upload langsung menggunakan aplikasi STMCube Programmer melalui adapter STLink.

Pada penelitian ini, peneliti menggunakan firmware khusus untuk motherboard hoverboard dengan nama “hoverboard firmware hack FOC” yang sudah tersedia di github yang ditulis pertama kali oleh Emanuel Feru pada tahun 2018. Firmware ini bersifat open source dan sudah digunakan oleh banyak pengembang robot yang menggunakan komponen hoverboard. Hingga tahun 2021, firmware ini masih terus dikembangkan untuk hasil yang lebih baik. gambar 2.26 merupakan *repository* github dari *firmware hoverboard.*



Gambar 2.26 *Repository Hoverboard Firmware Hack FOC*

2.2.14 *Robot Operating System* (ROS)

Robot Operating System (ROS) dikembangkan pada tahun 2007 oleh *Stanford Artificial Intelligence Laboratory* (SAIL) dengan dukungan Stanford AI Robot project. Pada tahun 2008, pengembangan ROS dilanjutkan oleh Willow Garage lembaga penelitian robotik dengan dukungan lebih dari 20 lembaga penelitian. Banyak lembaga penelitian yang mulai menggunakan ROS dengan menambahkan hardware dan berbagi contoh source code-nya. Banyak perusahaan juga mengadopsi untuk menggunakan ROS (Saputro, 2016).

*Robot Operating System (*ROS) sebenarnya adalah *meta-operating system* atau *framework* yang bersifat *open source* yang dapat digunakan untuk robot. ROS mempunyai sebuah service seperti halnya sebuah sistem operasi pada umumnya, termasuk abstraksi perangkat keras, kendali perangkat tingkat bawah, implementasi dari fungsi-fungsi yang biasa digunakan, penyampaian pesan/data diatara proses serta management package. ROS juga menyediakan alat dan library yang memungkinan kita untuk mendapatkan, membangun, memprogram hingga menjalankan program melalui banyak komputer. Dengan ROS, kita bisa memulai untuk memprogram dan mengendalikan sebuah robot dengan mudah, dikarenakan ada banyak contoh dan source code yang disediakan, termasuk sensor dan perangkat yang akan menambahkan fungsionalitas baru pada robot, seperti navigasi otonom dan persepsi visual. Kita patut berterimakasih kepada Komunitas opensource yang mengembangkan ROS , sehingga menghindarkan kita dalam membuang -buang waktu dan effort untuk melakukan riset yang sama berulang – ulang

ROS menyediakan layanan yang dirancang untuk kluster komputer yang heterogen seperti abstraksi perangkat keras, kontrol perangkat tingkat rendah, implementasi fungsionalitas yang umum digunakan, penyampaian pesan antar proses, dan manajemen paket. Menjalankan set proses berbasis ROS direpresentasikan dalam arsitektur grafik di mana pemrosesan berlangsung di node yang dapat menerima, menampilkan dan *multipleks* data sensor, kontrol, status, perencanaan, aktuator, dan pesan lainnya. Terlepas dari pentingnya

reaktivitas dan *latensi* rendah dalam kontrol robot, ROS sendiri bukanlah OS waktu nyata (RTOS). Namun, dimungkinkan untuk mengintegrasikan ROS dengan kode waktu nyata. Kurangnya dukungan untuk sistem waktu nyata telah diatasi dalam pembuatan ROS 2.0 revisi utama dari ROS API yang akan memanfaatkan perpustakaan dan teknologi modern untuk fungsionalitas inti ROS dan menambahkan dukungan untuk kode waktu nyata dan perangkat keras yang disematkan.

ROS memiliki konsep dengan kerumitan yang sangat tinggi. Untuk menjabarkan dengan detail setiap bagian dari ROS sangatlah sulit, maka dari itu peneliti hanya akan menjabarkan beberapa istilah yang akan sering dipakai dalam penelitian ini. ROS memiliki tiga level konsep: Filesystem Level, Computation Graph Level, dan Community level. Dari ketiga tingkat dan konsep ini, peneliti hanya akan membahas konsep *Computational Graph Level*. Hal ini dikarenakan konsep ini yang akan lebih penting dan lebihsering digunakan secara langsung dibandingkan yang lainnya. Bagian selanjutnya akan membahas mengenai konsep *computational graph* pada ROS.

2.2.14.1 ROS Computational Graph Level

Grafik Komputasi adalah jaringan peer-to-peer dari proses ROS yang memproses data bersama-sama. Konsep Grafik Komputasi dasar ROS adalah node, Master, Server Parameter, pesan, layanan, topik, dan tas, yang semuanya menyediakan data ke Grafik dengan cara yang berbeda. Konsep-konsep ini diimplementasikan dalam repositori ros_comm.

1. ***Node***, adalah proses yang melakukan komputasi. ROS dirancang untuk menjadi modular pada skala halus; sistem kontrol robot biasanya terdiri dari banyak node.
2. ***Master***, menyediakan registrasi nama dan pencarian ke Grafik Komputasi lainnya. Tanpa Master, node tidak akan dapat menemukan satu sama lain, bertukar pesan, atau memanggil layanan.
3. ***Parameter Server,*** memungkinkan data disimpan dengan kunci di lokasi pusat. Saat ini adalah bagian dari Guru.
4. ***Message***, node berkomunikasi satu sama lain dengan mengirimkan pesan. Pesan hanyalah struktur data, terdiri dari bidang yang diketik. Tipe primitif standar (integer, floating point, boolean, dll.) didukung, seperti juga array tipe primitif. Pesan dapat menyertakan struktur dan array bersarang (seperti struktur C).
5. ***Topic***, pesan dirutekan melalui sistem transportasi dengan semantik terbitkan/berlangganan. Sebuah node mengirimkan pesan dengan memublikasikannya ke topik tertentu. Topik adalah nama yang digunakan untuk mengidentifikasi isi pesan. Sebuah node yang tertarik pada jenis data tertentu akan berlangganan topik yang sesuai. Mungkin ada beberapa penerbit dan pelanggan secara bersamaan untuk satu topik, dan satu node dapat menerbitkan dan/atau berlangganan beberapa topik. Secara umum,

penerbit dan pelanggan tidak menyadari keberadaan satu sama lain. Idenya adalah untuk memisahkan produksi informasi dari konsumsinya. Logikanya, seseorang dapat menganggap topik sebagai bus pesan yang diketik dengan kuat. Setiap bus memiliki nama, dan siapa pun dapat terhubung ke bus untuk mengirim atau menerima pesan selama itu adalah jenis yang tepat.

6. ***Service***, model publish/subscribe adalah paradigma komunikasi yang sangat fleksibel, tetapi transportasi satu arah banyak-ke-banyaknya tidak sesuai untuk interaksi permintaan/balasan, yang sering diperlukan dalam sistem terdistribusi. Permintaan / balasan dilakukan melalui layanan, yang ditentukan oleh sepasang struktur pesan: satu untuk permintaan dan satu untuk balasan. Sebuah node penyedia menawarkan layanan dengan nama dan klien menggunakan layanan dengan mengirimkan pesan permintaan dan menunggu balasan. Pustaka klien ROS umumnya menyajikan interaksi ini kepada pemrogram seolah-olah itu adalah panggilan prosedur jarak jauh.
7. ***Bags***, adalah format untuk menyimpan dan memutar ulang data pesan ROS. Kantong adalah mekanisme penting untuk menyimpan data, seperti data sensor, yang mungkin sulit dikumpulkan tetapi diperlukan untuk mengembangkan dan menguji algoritme.

2.2.14.2 Basic Operation pada ROS

ROS memiliki beberapa perintah operasi dasar yang sering digunakan pada imbuhan awal setiap perintah yang dilakukan, antara lain sebagai berikut:

1. **roslaunch**, merupakan alat untuk dengan mudah meluncurkan beberapa node ROS secara lokal dan jarak jauh melalui SSH, serta mengatur parameter pada Server Parameter. Ini mencakup opsi untuk secara otomatis menghidupkan kembali proses yang telah mati. roslaunch mengambil satu atau lebih file konfigurasi XML (dengan ekstensi .launch) yang menentukan parameter yang akan disetel dan node yang akan diluncurkan, serta mesin tempat mereka harus dijalankan.
2. **roscore,** merupakan spesialisasi alat roslaunch untuk memunculkan sistem ROS "inti". Roslaunch akan secara otomatis memulai roscore jika mendeteksi bahwa itu belum berjalan (kecuali argumen --wait diberikan). Untuk informasi lebih lanjut, silakan lihat dokumentasi roscore. Catatan: karena kondisi balapan, roslaunch tidak boleh digunakan untuk menjamin instance roscore tunggal**.**
3. **rosrun,** merupakan bagian dari rosbash. rosrun memungkinkan kita untuk menjalankan executable dalam paket arbitrer dari mana saja tanpa harus memberikan path lengkapnya atau cd/roscd di sana terlebih dahulu.
4. **rostopic,** perintah untuk mengakses topik yang ditentukan dalam grafik komputasi.

5. **rosparam,** perintah untuk mendapatkan, mengatur, menghapus parameter dari ROS Parameter Server.
6. **rosservice,** perintah untuk mengakses layanan yang didefinisikan dalam grafik komputasi.

2.2.15 Catkin workspace pada ROS

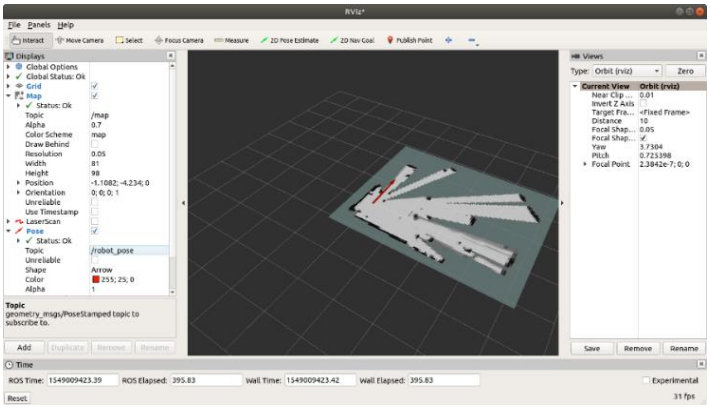
Catkin workspace adalah folder tempat memodifikasi, membangun, dan menginstal paket catkin. Ruang kerja catkin dapat berisi hingga empat ruang berbeda yang masing-masing memiliki peran berbeda dalam proses pengembangan perangkat lunak. Paket catkin dapat dibangun sebagai proyek mandiri, dengan cara yang sama seperti proyek cmake biasa dapat dibangun, tetapi catkin juga menyediakan konsep ruang kerja, di mana Anda dapat membangun beberapa paket yang saling bergantung sekaligus sekaligus.

2.2.16 Alat atau Aplikasi Pendukung ROS dari Pihak Ketiga

Tampilan antar muka pada *robot operating system* (ROS) hanya berupa terminal command, tidak memiliki GUI ataupun perangkat lunak interaksi lainnya. Untuk menjalankan, memonitoring dan melakukan *debugging* pengembang harus menggunakannya didalam terminal. Beruntungnya pada saat ini sudah banyak aplikasi yang mendukung pengembangan ROS dan memudahkan pengembang dalam memantau hasil kerja. Contohnya untuk aplikasi 3D simulasi dan visualisasi menggunakan Rviz dan Gazebo, kemudian untuk *plotting,* kontrol manual, *managing* dan visualisasi sistem menggunakan aplikasi rqt beserta pluginnya.

2.2.16.1 Rviz

Rviz adalah antarmuka grafis ROS yang memungkinkan Anda memvisualisasikan banyak informasi, menggunakan *plugin* untuk berbagai jenis topik yang tersedia. Rviz sebuah aplikasi yang digunakan untuk 3D visualisasi dalam *middle ware* ROS. Untuk dapat memvisualisasikan informasi yang dipublikasikan oleh kamera ZED, perlu untuk mengonfigurasi pengaturan global dengan benar. Rviz memungkinkan tampilan real-time dari posisi dan sikap lokal kendaraan, data LIDAR, dan peta 3D *Cartografer*. Dimungkinkan juga untuk menetapkan target posisi lokal dan memiliki perpustakaan navigasi ROS dengan memindahkan kendaraan ke target. Terdapat 2 sistem utama pada penelitian perencanaan gerakan robot ini. Sistem ini meliputi pengiriman node dengan pengaturan masing-masing sudut tiap *joint* pada RViz dan Pengiriman perintah pada aktuator *servo dynamixel*. Kedua sistem ini dapat divisualisasikan melalui RViz. gambar 2.28 merupakan ilustrasi tentang hubungan kedua sistem tersebut untuk membentuk gerakan pada visualisasi dan robot sesungguhnya secara sama seperti yang ditunjukkan pada gambar 2.27.

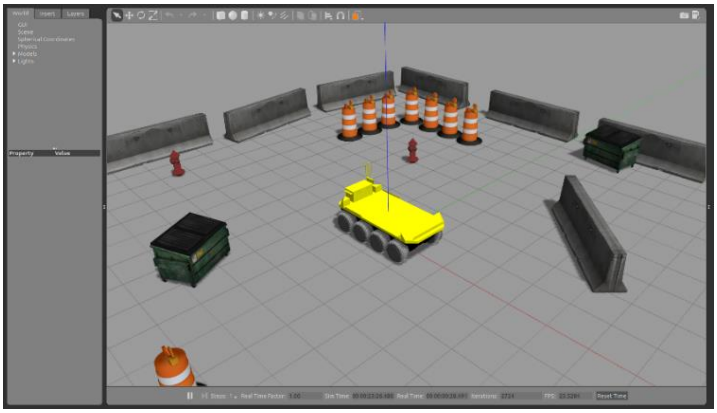


Gambar 2.27 Visualisasi simulasi *mapping* menggunakan Rviz

Sumber : (ardupilot.org, 2021)

2.2.16.2 Gazebo 3D Simulator

Simulator robotika adalah simulator yang digunakan untuk membuat aplikasi robot fisik tanpa bergantung pada mesin yang sebenarnya, sehingga menghemat biaya dan waktu. Dalam beberapa kasus, aplikasi ini dapat ditransfer ke robot fisik tanpa modifikasi. Gazebo menawarkan kemampuan untuk secara akurat dan efisien menyimulasikan populasi robot di lingkungan indoor dan outdoor yang kompleks. Di ujung jari Anda adalah mesin fisika yang kuat, grafis berkualitas tinggi, dan antarmuka program dan grafis yang nyaman. Yang terbaik dari semuanya, Gazebo gratis dengan komunitas yang dinamis.



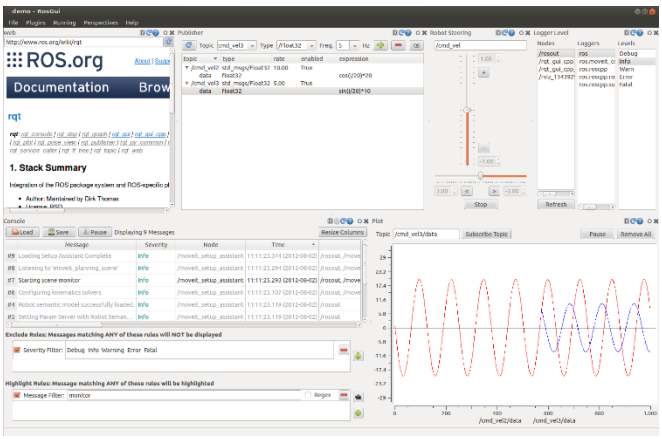
Gambar 2.28 Visualisasi simulasi robot menggunakan Gazebo

Sumber : (www.clearpathrobotics.com, 2015)

2.2.16.3 Rqt

rqt adalah kerangka kerja perangkat lunak ROS yang mengimplementasikan berbagai alat GUI dalam bentuk plugin. Seseorang dapat menjalankan semua alat GUI yang ada sebagai jendela yang dapat dipasang di dalam rqt! Alat masih dapat berjalan dalam metode mandiri tradisional, tetapi rqt memudahkan untuk mengelola semua berbagai jendela didalam satu layar. Hanya dengan mengetikkan *command* $ rqt pada terminal, maka akan langsung menunjukkan GUI di mana kita dapat memilih plugin yang tersedia di sistem. Pengembang dapat

membuat plugin mereka sendiri untuk rqt dengan Python atau C++. Lebih dari 20 plugin (per Feb 2013) telah dibuat dan lebih banyak lagi dijadwalkan untuk pengembangan. Tampilan antar muka aplikasi rqt dapat dilihat pada gambar 2.29



Gambar 2.29 Semua plugin didalam aplikasi rqt

Sumber : (www. ros.guru, 2021)

2.2.17 *Protokol Komunikasi*

Robot Operating System (ROS) atau Sistem Operasi Robot adalah rangkaian *middleware* robotika *open source*. *Middleware* robot dirancang untuk mengelola kompleksitas dan heterogenitas perangkat keras dan aplikasi, mempromosikan integrasi teknologi baru, menyederhanakan desain perangkat lunak, menyembunyikan kompleksitas komunikasi tingkat rendah dan heterogenitas sensor dari sensor, meningkatkan kualitas perangkat lunak, menggunakan kembali perangkat lunak robotik infrastruktur di berbagai upaya penelitian, dan untuk mengurangi biaya produksi.

2.2.17.1 Protokol Universal Synchronous and Asynchronous Receiver-Transmitter (USART)

Robot Operating System (ROS) atau Sistem Operasi Robot adalah rangkaian *middleware* robotika *open source*. *Middleware* robot dirancang untuk mengelola kompleksitas dan heterogenitas perangkat keras dan aplikasi, mempromosikan integrasi teknologi baru, menyederhanakan desain perangkat lunak, menyembunyikan kompleksitas komunikasi tingkat rendah dan heterogenitas sensor dari sensor, meningkatkan kualitas perangkat lunak, menggunakan kembali perangkat lunak robotik infrastruktur di berbagai upaya penelitian, dan untuk mengurangi biaya produksi.

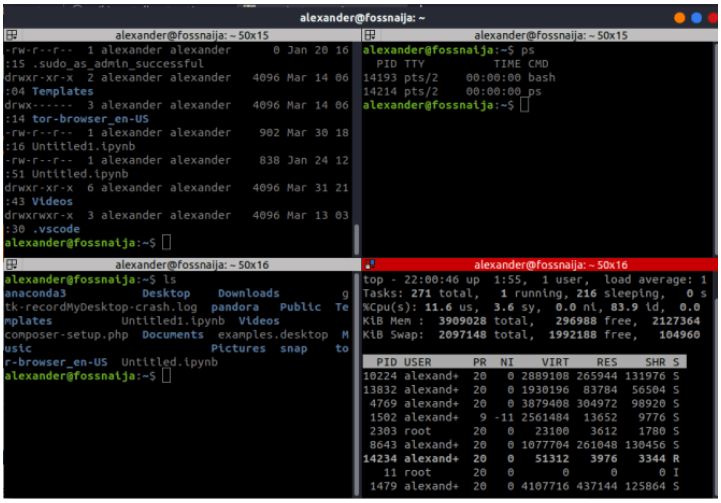
2.2.17.2 Protokol Inter-Integrated Circuit (I2C)

Robot Operating System (ROS) atau Sistem Operasi Robot adalah rangkaian *middleware* robotika *open source*. *Middleware* robot dirancang untuk mengelola kompleksitas dan heterogenitas perangkat keras dan aplikasi, mempromosikan integrasi teknologi baru, menyederhanakan desain perangkat lunak, menyembunyikan kompleksitas komunikasi tingkat rendah dan heterogenitas

sensor dari sensor, meningkatkan kualitas perangkat lunak, menggunakan kembali perangkat lunak robotik infrastruktur di berbagai upaya penelitian, dan untuk mengurangi biaya produksi.

2.2.18 *Terminator (Terminal Command Application)*

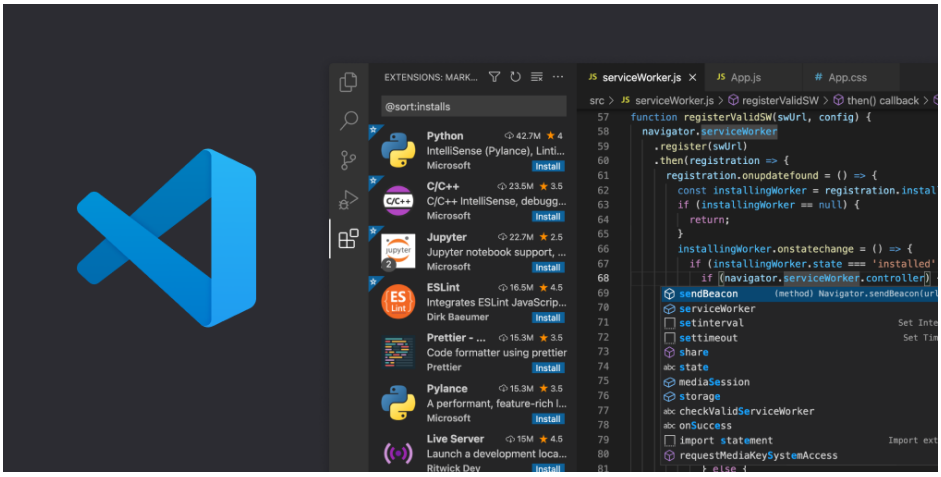
Terminator adalah terminal alternatif untuk Linux yang hadir dengan sedikit fitur dan fungsionalitas tambahan yang tidak akan akan temukan di aplikasi terminal default. Misalnya di terminator kita dapat membagi layar terminal secara horizontal dan vertikal sesuai keinginan. Pengguna juga dapat memiliki beberapa terminal dalam satu *windows*. Dengan terminator, pengguna juga dapat secara efisien mengisi area layar yang luas dengan terminal. Dengan begitu menjalankan robot menjadi efisien karena tab yang baru dibuka akan membelah tampilan terminal, tidak menutupnya secara keseluruhan. Tampilan dari terminator dapat dilihat pada gambar 2.



Gambar 2.30 Tampilan aplikasi terminal Terminator

2.2.19 Visual Studio Code (IDE)

Visual Studio Code adalah Integrated Development Environment (IDE) yang dibuat oleh Microsoft untuk Windows, Linux, dan macOS. Fitur termasuk dukungan untuk debugging, penyorotan sintak, penyelesaian kode cerdas, cuplikan, pemfaktoran ulang kode, dan Git yang disematkan. Pengguna dapat mengubah tema, pintasan keyboard, preferensi, dan pemasangan ekstensi yang menambahkan fungsionalitas tambahan.

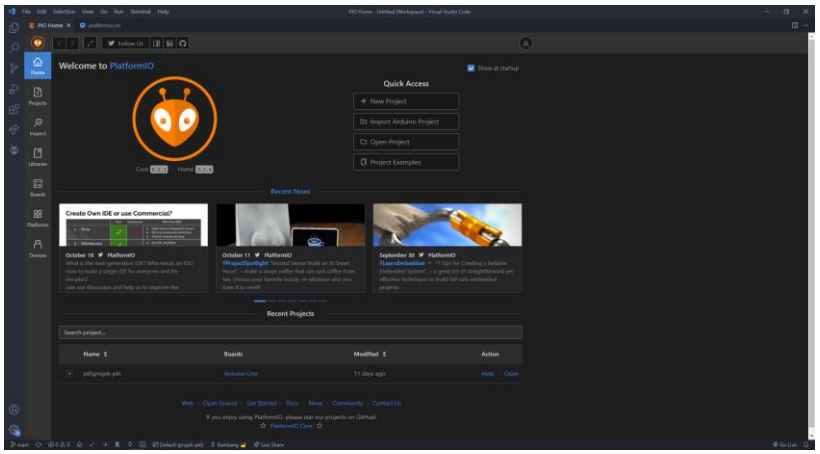


Gambar 2.31 Visual Studio Code

Sumber : (www.code.visuastudio.com, 2021)

2.2.20 PlatformIO (*Visual Studio Code Extension*)

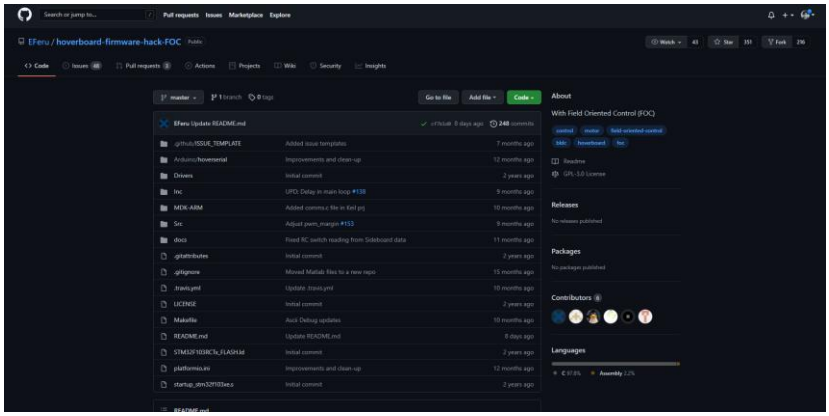
PlatformIO adalah alat IDE profesional lintas-platform, lintas-arsitektur, multi-kerangka untuk sistem tertanam dan insinyur perangkat lunak yang menulis aplikasi tertanam. Dengan menyediakan antarmuka IDE universal menggunakan PlatformIO, kita dapat memprogram perangkat keras dengan cara yang lebih ramah pengembangan. Dengan menggunakan PlatformIO pengembang dapat melakukan compile, build, upload, debugging dan management project dengan mudah.



Gambar 2.32 Extensi Platform IO yang ada di Visual Studio Code

2.2.21 *Hoverboard* Firmware

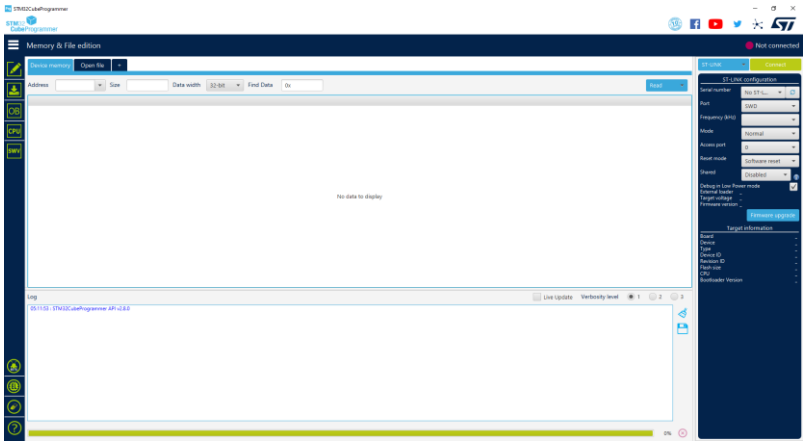
Frmware adalah kelas perangkat lunak komputer tertentu yang menyediakan kontrol tingkat rendah untuk perangkat keras khusus perangkat. Firmware, seperti BIOS komputer pribadi, mungkin berisi fungsi dasar perangkat, dan dapat menyediakan layanan abstraksi perangkat keras ke perangkat lunak tingkat tinggi seperti sistem operasi



Gambar 2.33 Firmware *Hoverboard* yang dibuat oleh Emanuel Feru di github

2.2.22 STM32Cube Programmer

STM32CubeProgrammer adalah alat perangkat lunak multi-OS all-in-one untuk memprogram produk STM32. Ini menyediakan lingkungan yang mudah digunakan dan efisien untuk membaca, menulis, dan memverifikasi memori perangkat melalui antarmuka debug (JTAG dan SWD) dan antarmuka bootloader (UART, USB DFU, I2C, SPI, dan CAN). STM32CubeProgrammer menawarkan berbagai fitur untuk memprogram memori internal STM32 (seperti Flash, RAM, dan OTP) serta memori eksternal. STM32CubeProgrammer juga memungkinkan pemrograman dan unggah opsi, verifikasi konten pemrograman, dan otomatisasi pemrograman melalui skrip. STM32CubeProgrammer dikirimkan dalam versi GUI (antarmuka pengguna grafis) dan CLI (antarmuka baris perintah).



Gambar 2.34 Tampilan muka aplikasi STM32Cube Programmer

BAB 3 METODOLOGI PENELITIAN

Dalam bab metodologi penelitian, penulis akan menjelaskan bagaimana tahapan peneliti meneliti, merancang, mengimplementasi, dan menilai sistem purwarupa Robot Pengantar Barang Menggunakan Sensor RPLIDAR Dan Kombinasi Algoritme *Simultaneous Localization And Mapping* Pada *Robot Operating System* dapat bekerja secara real time dan efisien. Harapannya pada tahapan ini, penulis juga dapat merancang kembali penelitian sebelumnya yang telah diterapkan menjadi lebih efisien lagi.

3.1 Tipe Penelitian

Tipe penelitian yang digunakan pada penelitian ini adalah tipe implementatif pengembangan yang mana, implementatif menunjukkan bahwa penelitian ini memerlukan pengujian melalui perangkat lunak dan perangkat keras, tidak hanya analisis data saja. Hasil dari penelitian ini adalah sebuah purwarupa robot bergerak otonom dari *hoverboard* yang mampu memetakan area sekitar dan mampu menghindari halangan.

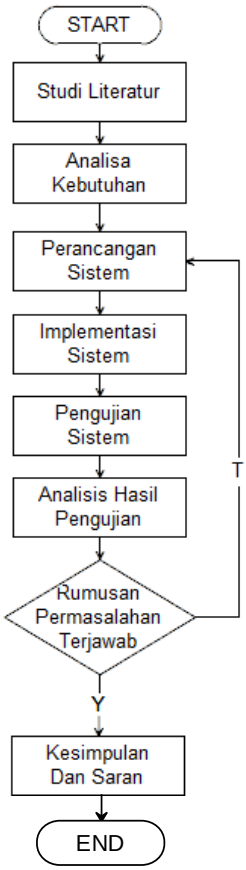
3.2 Strategi dan Rancangan Penelitian

Pada penelitian ini, dibutuhkan strategi dan rencana yang jelas agar memenuhi tujuan yang hendak dicapai. Untuk memenuhi hal tersebut penulis akan menjelaskan secara umum setiap perangkat keras dan perangkat lunak yang digunakan sebagai sistem dari robot bergerak. Untuk memenuhi tujuan yang telah dirumuskan pada bab sebelumnya maka disusun perancangan sistem yang menjelaskan peran dari perangkat keras sebagai penyedia sumber pemrosesan untuk melakukan komputasi dan perangkat lunak yang berperan sebagai pengolahan data dan pemrosesan *mapping, localization and navigation*.

Pemecahan masalah pada penelitian ini dilakukan dengan menjalankan 3 tahap, yaitu pengambilan data untuk *mapping*, pengolahan data untuk *localization,* dan hasil pemrosesan peta-lokasi untuk proses *navigation*. Pada tahap awal, pengambilan data berdasarkan perhitungan jarak oleh hasil pembacaan 2 sensor, yakni RPLIDAR dalam bentuk sinar inframerah yang ditembakkan dan *Inertial Measurement Unit* atau IMU. Perhitungan dari kedua sensor ini nantinya akan menghasilnya lokasi robot berupa posisi x, y, z dan benda di sekitarnya seluas 6-6 meter yang menghasilkan bentuk visual pada perangkat lunak yang ditampilkan dalam monitor laptop/Jetson nano berupa partikel titik merah yang sangat banyak, sehingga membentuk pola berdasarkan area sekitar seperti air yang di wadah maka air akan berbentuk seperti wadah. Kemudian setelah peta sudah dibangun, selanjutnya adalah tahap akhir yaitu navigasi, dengan memanfaatkan sensor RPLIDAR, IMU, peta dan gabungan algoritma navigasi, kita dapat melakukan proses navigasi dengan memasukkan titik tujuan pada visualisasi peta. Dengan begitu robot akan berjalan ke dari titik awal (posisi robot saat ini) menuju titik tujuan (posisi input pengguna).

3.2.1 Metode Penelitian

Demi sepenuhnya mencapai tujuan, maka pada penelitian ini dilakukan secara struktur dan sistematis dalam metode penelitian. Metode yang digunakan untuk mengembangkan suatu karya harus sesuai dengan kebutuhan pembuatan karya. Kebutuhan awal yang diperlukan untuk merancang robot ini yaitu pendalaman informasi terkait permasalahan, pengumpulan, pengkajian teori, dan penelitian sebelumnya yang telah dilakukan dengan tujuan mendukung pengembangan robot ini sehingga dapat ditemukan solusi dari permasalahan dalam pengembangan karya ini. Jurnal ilmiah, artikel ilmiah, situs web, buku, dan lain sebagainya digunakan sebagai studi literatur pada penelitian ini. Setelah melakukan studi literatur, maka langkah selanjutnya adalah melakukan definisi masalah yang ingin diselesaikan, misalnya seperti perangkat lunak (*software*), perangkat keras (*hardware)*, algoritma yang digunakan, dan lain sebagainya. Setelah itu dirumuskan dasar teori yang berisi penelitian secara ilmiah sangat diperlukan sebagai referensi penunjang dalam pengembangan robot ini sehingga menghasilkan karya dari permasalahan yang ada dan bisa dipertanggungjawabkan. Hasil pengkajian teori dari studi literatur yang relevan digunakan dalam mengaplikasikan proses rekayasa kebutuhan sistem yang sesuai lalu melakukan perancangan sistem atau gambaran sistem, implementasi sistem, dan pengujian sistem. Sistem yang telah dilakukan pengujian kemudian dilakukan analisis. Dari hasil analisis pengujian akan digunakan untuk menginformasikan bahwa sistem ini belum atau telah memenuhi kebutuhan dalam permasalahan yang diajukan. Diagram alir tahapan penelitian dapat dilihat pada gambar 3.1.



Gambar 3.1 Diagram Alir Penelitian

Penggunaan diagram alir penelitian seperti gambar 3.1 akan membuat proses lebih tepat arah dan teratur. Setelah analisis hasil pengujian apabila penulis merasa bahwa rumusan permasalahan tidak terjawab, maka penulis akan memperbaiki sistem pada perancangan sistem, kemudian melakukan implementasi sistem, pengujian sistem, dan kembali melakukan analisis hasil pengujian. Setelah analisis hasil berhasil menjawab rumusan permasalahan, maka penulis akan mendapatkan kesimpulan dan saran yang dibutuhkan dalam penelitian.

3.2.2 Lokasi Penelitian

Lokasi penelitian dilakukan pada Gedung Fakultas Ilmu Komputer dan ruang lingkup sekitar penulis. Untuk lokasi pengujian dilakukan pada tempat dengan desain denah bangunan yang bisa diukur.

3.2.3 Teknik Pengumpulan Data

Pengambilan data sensor yang telah di kalibrasi, observasi parameter berdasarkan dokumentasi dan data dari terminal Linux digunakan sebagai teknik pengumpulan data dalam penelitian ini. Dengan menggunakan data yang diperoleh dari hasil proses observasi langsung yang dilakukan pada semua sensor, diharapkan penulis mengetahui betul bahwa data semua sensor telah benar berdasarkan pengukuran asli dan dokumentasi *datasheet* setiap sensor. Hal ini

dilakukan agar pada pengujian, hasil bisa lebih objektif karena sensor sudah berjalan sebagaimana mestinya dan menghindari kesalahan sehingga berbeda dari penelitian sebelumnya. Observasi yang dilakukan bertujuan untuk mengetahui apakah sistem yang dibuat berjalan sesuai dengan fungsi dan kebutuhannya. Proses observasi meliputi percobaan setiap parameter pada algoritma yang telah ditentukan dan dicari nilai yang optimal. Pencarian nilai optimal ini didasari pada hasil riset pada penelitian yang serupa. Hal ini dilakukan untuk mengetahui apakah parameter sebagai perhitungan dipengaruhi oleh lingkungan yang dilihat dari hasil *log* pada terminal Linux dan hasil visualisasi bentuk peta dan navigasi pada aplikasi Rviz yang merupakan salah satu *tool* di dalam ROS. Selanjutnya sampel data hasil observasi digunakan sebagai data untuk proses analisa lebih lanjut.

3.3 Teknik Analisis Data

Analisa data dilakukan menggunakan data yang telah dikumpulkan sebelumnya. Hal ini dilakukan untuk menguji kemampuan sistem yang akan menjawab rumusan permasalahan yang ada untuk dievaluasi agar berguna untuk penelitian ke depannya. Berikut adalah aspek yang diuji pada sistem.

1. Pengujian tingkat keakuratan posisi robot yang dihasilkan oleh ROS melalui sinkronasi IMU dengan data sensor RPLIDAR
2. Pengujian tingkat ketepatan bentuk peta berdasarkan denah bangunan asli dari hasil pemetaan yang dihasilkan oleh ROS melalui pemindaian dari data sensor RPLIDAR dan IMU
3. Pengujian tingkat keakuratan navigasi robot terhadap halangan menggunakan algoritme Timed Elastic Bands yang diperoleh dari data pemetaan yang telah dibuat.
4. Pengujian keakuratan pergerakan robot serta waktu komputasi yang dibutuhkan sistem untuk berjalan pada posisi awal menuju posisi yang telah ditentukan yang diperoleh dari masukkan titik koordinat dari perangkat lunak ROS.

3.4 Peralatan Pendukung

Peralatan pendukung diperlukan untuk menunjang penelitian ini agar bisa berjalan dengan lancar dan sistem bisa berjalan sesuai kebutuhan. Peralatan ini terdiri dari perkakas atau peralatan yang berfungsi untuk membantu penulis dalam pembuatan robot. Dibawah ini akan disebutkan peralatan pendukung terkait perangkat keras serta perangkat lunak yang digunakan pada penelitian ini, sebagai berikut.

1. Perangkat keras

- Nvidia Jetson Nano
- Arduino Uno
- Motherboard Hoverboard

- STLink V2
- FTDI to USB adapter
- Motor Brushless DC
- Baterai 10S2P 4000mAh
- Kabel power-on dan *charger port*
- Charger adapter hoverboard
- Sensor RPLidar dan adapter UART
- Sensor GY-521
- Hoverboard motor BLDC encoder (*build-in*)
- 3D Printer dan filamen
- Keyboard dan Mouse
- Kabel Jumper
- Power bank
- Kabel micro USB
- Personal Komputer

2. Perangkat lunak

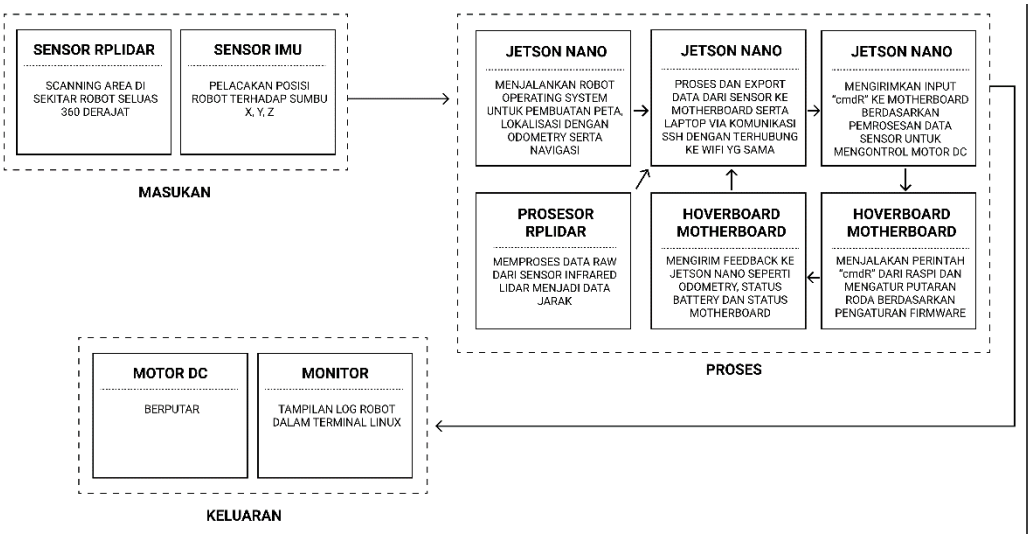
- *Hoverboard Firmware*
- *STMCube Programmer*
- Compiler *Catkin workspace*
- Compiler *makefile*
- Terminator (Terminal linux)
- Robot operating system (ROS)
- Visual Studio Code (IDE)
- Platform IO (VSCode *Extension)*
- VNC Viewer

BAB 4 REKAYASA KEBUTUHAN

Dalam bab analisis kebutuhan, penulis akan menjabarkan kebutuhan yang diperlukan guna mengimplementasikan sistem yang penulis ajukan dalam penelitian ini. Rekayasa kebutuhan mencakup gambaran umum sistem, analisis kebutuhan fungsional dan kebutuhan non-fungsional, kebutuhan perangkat keras, kebutuhan perangkat lunak dan batasan desain sistem. Oleh karena itu disusunlah suatu analisis rekayasa yang diperlukan agar sistem dapat diimplementasikan nantinya sebagai berikut.

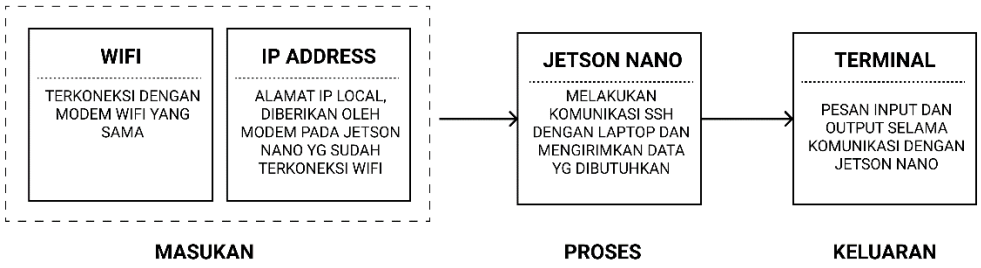
4.1 Gambaran Umum Sistem

Sistem yang dibuat pada penelitian ini merupakan sistem yang mempunyai fungsi membawa barang dari titik awal menuju titik akhir atau tujuan. Sistem ini akan dimulai difokuskan terhadap perancangan perangkat keras pada robot. Pembacaan sensor RPLIDAR nantinya akan diproses menggunakan metode PID control melalui ROS *hoverboard* driver. Pengolahan sistem akan dibagi menjadi 2 alur yaitu desain perangkat keras dan desain sistem perangkat lunak, seperti ditunjukkan pada gambar 4.1.



Gambar 4.1 Blok Diagram Sistem Mainboard dan Sensor Robot

Dari gambar 4.1 dijelaskan bahwa input adalah pembacaan data dari sensor RPLIDAR untuk mendeteksi jarak yang didapat secara real-time. Selain itu didapatkan pula data dari sensor IMU untuk mengetahui akselerasi dari sensor. Kemudian data dari input yang diterima oleh sensor RPLIDAR dan IMU diproses oleh jetson nano menggunakan metode PID control melalui *Hoverboard* ROS Driver. Pada bagian output, motor DC akan menggerakkan roda robot sehingga membuat robot dapat berjalan mengikuti objek. Hal utama yang perlu diperhatikan bahwa, objek yang dapat diikuti oleh robot hanya satu objek saja dan ruangan yang digunakan untuk percobaan harus cukup luas dan tidak terdapat obstacle dikarenakan dapat mengganggu pembacaan data dari sensor RPLIDAR.



Gambar 4.2 Blok Diagram Komunikasi Sistem Dengan Laptop

Dari gambar 4.2 dijelaskan sistem komunikasi robot dengan laptop sebagai monitor menggunakan sistem SSH dengan memanfaatkan jetson nano yang terhubung dengan modem wifi yang sama. Selain itu kelebihan dari penggunaan SSH adalah pengontrolan robot dapat dilakukan secara remote hanya dengan menggunakan koneksi internet, pada implementasinya jaringan yang digunakan adalah jaringan wifi dengan modem wifi yang sama kemudian jetson nano dan laptop saling terhubung kedalam wifi yang sama

4.2 Analisis Kebutuhan Sistem

Pada sub bab terkait analisis kebutuhan sistem, proses ini bertujuan agar peneliti tahu segala kebutuhan yang diperlukan oleh sistem pada perangkat keras dan perangkat lunak, baik secara fungsional dan non-fungsional.

4.2.1 Kebutuhan Fungsional

Kebutuhan fungsional merupakan kebutuhan sistem yang di dalamnya berisi proses apa saja yang harus disediakan oleh sistem, seperti bagaimana sistem merespons input dan bagaimana respons sistem pada kondisi tersebut. Kebutuhan fungsional disajikan kedalam tabel yang dapat diliat pada tabel 4.1.

Tabel 4.1 Kebutuhan Fungsional Sistem

No	Kebutuhan Sistem	Skenarion Pengujian
1	Sensor RPLIDAR dapat membaca jarak benda sekitar robot dengan pemindaian pada area seluas 360 derajat. Nilai yang dihasilkan berupa nilai *distance* atau jarak dari *range* 0.15 meter hingga 6 meter	Dapat mengirimkan data ke mikrokontroller dan pengujian hasil data berupa pengukuran jarak antara robot dengan halangan menggunakan alat pengukur sebenarnya seperti meter ukur
2	Sensor Inertial measurement unit (IMU) dapat membaca nilai derajat kemiringan dan gerak akselerasi pada robot. Berfungsi untuk melakukan	Dapat mengirimkan posisi robot pada mikrokontroller terhadap posisi sebenarnya robot menggunakan pemantauan

	Sensing proyeksi terhadap kemiringan dan gerak akselerasi robot	secara langsung dan proyeksi melalui 3D visual Rviz
3	Motor DC Brushless atau BLDC dapat menggerakkan robot berdasarkan nilai PWM yang diberikan dari motherboard *hoverboard*	Diujinya jumlah putaran motor DC menggunakan odometry dan perhitungan secara manual sehingga dipastikan data odometry adalah putaran sebenarnya dan robot bergerak sesuai dengan yang didesain
4	Sistem mikrokontroler didalam robot mampu mengirim dan menerima data sensor IMU untuk dilaksanakan	Diujinya pengiriman data dan penerimaan data ketika diam dan bergerak
5	Sistem mikrokontroler didalam robot mampu mengirim dan menerima data sensor RPLIDAR untuk dilaksanakan	Diujinya pengiriman data ketika robot melakukan proses pembuatan peta atau *mapping*
6	Mikrokontroler dapat memproses data dari sensor menjadi kesimpulan arah gerak dan kecepatan sistem pada aplikasi ROS	Diujinya navigasi robot menggunakan data yang diberikan oleh sensor
7	Sistem *Motherboard Hoverboard* dapat menggerakkan kedua motor serta memberi tegangan pada semua komponen elektronik.	Diujinya kecepatan berdasarkan perputaran roda melalui encoder motor BLDC menggunakan metode debuging
8	Aplikasi Rviz dapat memantau pengontrolan sistem secara otomatis ketika bergerak menggunakan sistem otonom robot operating sistem	Diujinya penggunakan Rviz untuk melihat hasil pengontrolan otonom secara real-time

4.2.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional adalah batasan dari sistem terhadap layanan yang diberikan. Batasan tersebut dapat berupa waktu, batasan proses pengembangan dan batasan terhadap standar tertentu.

4.2.2.1 Kebutuhan Perangkat Keras

1. Motherboard Hoverboard

   Motherboard Hoverboard digunakan sebagai driver dan controller pada motor BLDC. Hoverboard motherboard menjadi komponen penting yang menyimpan data firmware yang telah di *flash* sebelumnya. Motherboard merupakan komponen wajib yang ada pada penelitian, ini dikarenakan tanpa adanya motherboard hoverboard maka tidak akan bisa mengendalikan motor dan akan sangat sulit jika menggunakan motor BLDC driver eksternal karena permasalahan biaya dan kompatibilitas dengan firmware yang ada. Berikut

adalah spesifikasi yang diambil dari *datasheet* motherboard *hoverboard* yang ditunjukkan pada tabel 4.2.

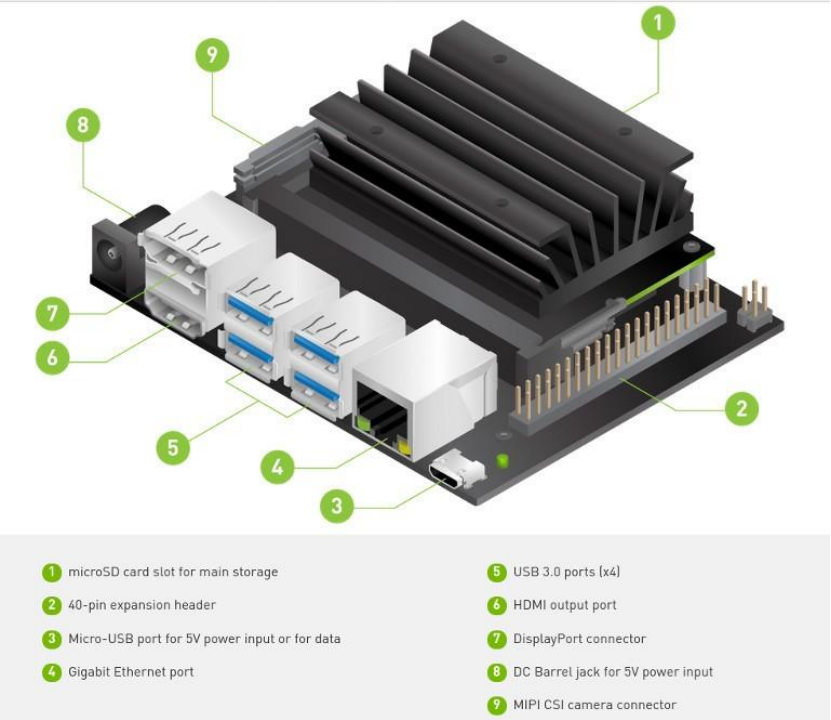
Tabel 4.2 Spesifikasi Motherboard *Hoverboard*

Sumber : (beta.ivc.no dan www.st.com, 2021)

Keterangan	Spesifikasi
Prosesor	GD32F103RCT6 ARM Cortex-M3 32-bit MCU
Flash memory	256 KB sampai 512 KB
SRAM	64 KB
Clock speed	108MHz
Flasher	ST-Link V2 *Programming* Unit
Tegangan operasi	36V
Pengisian daya	42V 2A
Port *Fast* I/O	80 Port (45P, 9ADC, 16PWM, 6RXTX, 4SC, 3.3V, 5V, GND)
Pin Digital I/O motor DC	6 (2xHALL A-C)
Pin Digital I/O sensor	4 (PA2, PA3, PB10, PB11)
Tegangan, arus dan kekuatan maksimal pada Motor DC	36V(1000 r.p.m.) 1-25A 350W
Tegangan dan arus DC Pin I/O motor	5V 100mA
Tegangan dan arus DC Pin I/O sensor	15V 200mA
A/D converters	3 x 12-bit
D/A converters	2 x 12-bit
Komunikasi	USART, SPI, I2C, I2S, USB, CAN

2. Nvidia Jetson Nano

Nvidia Jetson nano merupakan pemrosesan inti dari sistem pada penelitian ini, fungsinya sebagai tempat pemrosesan input dari ketiga sensor dan menjalankan fungsi mulai dari lokalisasi, pemetaan dan perencanaan jalur serta komunikasi SSH-VNC pada slave (laptop). Semua metode, algoritma dan komunikasi yang digunakan akan diproses didalam sini. Alasan dipilihnya Jetson nano sebagai mikrokomputer dan pemrosesan utama adalah pertimbangan kemampuan dari perangkat ini yang memiliki CPU Quad-core dengan *clock speed* 1.43 GHz dan RAM sebesar 4 GB LPDDR.



Gambar 4.3 Nvidia Jetson Nano

Sumber: (www.embeddednesia.com, 2020)

Dengan spesifikasi seperti ini diharapkan Nvidia Jetson Nano mampu melakukan pemrosesan *mapping* dan *navigating.* Proses pemetaan menggunakan sumber daya yang cukup banyak karena terdapat banyak sistem ROS yang berjalan dibelakang layar ditambah dengan sistem perencanaan gerak menggunakan Dijkstra yang banyak menguras CPU untuk menemukan jalur terpendek. Memori RAM sebesar 4 GB dirasa sudah mumpuni untuk menjalankan semua proses ini. Walaupun pemrosesan pada Rviz dilakukan di Slave atau Laptop, akan tetapi untuk melakukan *load data* hasil pemetaan juga dibutuhkan sumber daya CPU dan RAM yang besar. Oleh karena itu, kesimpulannya adalah Nvidia Jetson Nano mampu menjalankan sistem secara baik pada penelitian ini. Berikut adalah *datasheet* dari mikrokontroler Jetson nano 3B yang ditunjukkan pada tabel 4.3.

Tabel 4.3 Spesifikasi Mikrokontroler Jetson nano 3B

Sumber : (developer.nvidia.com, 2021)

Keterangan	Spesifikasi
CPU	Quad-core ARM A57 @ 1.43 GHz
GPU	NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores
Memory	4 GB 64-bit LPDDR4, 1600MHz 25.6 GB/s
Storage	microSD (not included)

Video Encode	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)
Video Decode	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)
Kamera	2x MIPI CSI-2 DPHY lanes
Konektivitas	Gigabit Ethernet, M.2 Key E
Display	HDMI 2.0 and display port
USB	4x USB 3.0, USB 2.0 Micro-B
Lainnya	GPIO, I2C, I2S, SPI, UART
Mechanical	69.6 mm x 45 mm 260-pin edge connector
Memory Eksternal	64GB Mikro SD
Power	Mikro USB 5V 2.5A dan DC Barrel Jack 5V 4A

3. Motor DC Brushless

Motor DC Brushless atau motor BLDC adalah motor sinkron yang menggunakan catu daya listrik arus searah (DC). Ini menggunakan pengontrol loop tertutup elektronik untuk mengalihkan arus DC ke belitan motor yang menghasilkan medan magnet yang secara efektif berputar di ruang angkasa dan yang diikuti oleh rotor magnet permanen. Kontroler menyesuaikan fase dan amplitudo pulsa arus DC untuk mengontrol kecepatan dan torsi motor. Sistem kontrol ini merupakan alternatif dari komutator mekanis (sikat) yang digunakan di banyak motor listrik konvensional. Berikut adalah *datasheet* motor dc brushless yang ditunjukkan pada tabel 4.4.

Tabel 4.4 Spesifikasi Motor DC Brushless

Sumber : (www.aliexpress.com, 2021)

Keterangan	Spesifikasi
Tipe	Gear Motor, Motor Skuter
Komunikasi	Brushless
Kecepatan	5 - 1500 rpm
Tegangan	36V
Arus	1A - 25A
Power Motor	250 - 350W
Kecepatan Maksimal	26km/h
Kecepatan rata-rata	800rpm
Diameter roda	6.5”
Panjang kabel	26 cm

4. Catu Daya

Catu daya merupakan hal penting dari sistem yang digunakan untuk penyedia sumber tegangan untuk mengaktifkan keseluruhan sistem. Selain itu, catu daya tergolong jenis yang portabel dengan daya tampung energi listrik yang dapat diisi ulang kembali. Untuk penggunaan catu daya harus dapat menyesuaikan tegangannya yang dibutuhkan oleh sistem (Cahyadi, et al., 2016). Ada dua catu daya yang digunakan pada robot ini, pertama yaitu baterai li-ion dengan besar tegangan 32V dan kuat arus 2A. Catu daya ini digunakan untuk memberikan power ke motherboard hoverboard dan yang nantinya akan disalurkan lagi ke motor BLDC. Kemudian catu daya yang kedua adalah power bank ke Jetson Nano dengan besar tegangan 5V dan kuat arus 1A. Berikut adalah tabel spesifikasi Baterai Li-ion 10S2P dan Power bank yang masing-masing ditunjukkan pada tabel 4.5 dan tabel 4.6.

Tabel 4.5 Spesifikasi Catu Daya untuk Suplai Motherboard Hoverboard

Sumber : (www. hovertech.co.za, 2021)

Keterangan	Spesifikasi
Teknologi	Lithium-Ion Battery
Ukuran Cell	18650 Lithium Ion
Tegangan ke motherboard	25.2V - 36V
Tegangan maksimal	42V
Arus sistem kerja	1A - 20A
Tenaga	75W hingga 158W
Maksimal pemakaian arus	15A
Arus Pengisian	< 5A
Tegangan Pengisian	43.2V
Kapasitas	4400 mAh
Struktur	10S2P & 10S1P

Tabel 4.6 Spesifikasi Catu Daya untuk Suplai Jetson Nano

Sumber : (www.shopee.co.id, 2021)

Keterangan	Spesifikasi
Teknologi	Quick Charge 3.0
Tipe Baterai	Lithiun Polymer
Output 1	USB A DC 5V/3A, 9V/2A, 12V/1,5A
Output 2	USB A DC 5V/3A, 9V/2A, 12V/1,5A
Output 3	Type C 5V/3A, 9V/2,33A, 12V/1,7A
Total Output	20W
Kapasitas	10000 mAh

5. RPLIDAR A1

RPLIDAR A1 digunakan sebagai input utama dari sistem berupa tangkapan jarak dari tembakan laser yang antinya akan diproses oleh sistem. Penggunaan RPLIDAR didasarkan dari kemampuannya mendeteksi objek seluas 360 derajat dan sejauh 6 meter. Selain itu RPLIDAR juga sangat praktis untuk digunakan karena kita tidak perlu menghitung nilai ataupun kalibrasi data karena RPLIDAR memiliki prosesornya sendiri dan mampu mengirimkan data setengah jadi berupa koordinat-koordinat data jarak. Atas semua pertimbangan dan kemampuannya yang powerfull maka dari itu RPLIDAR dipilih menjadi sensor utama pada penelitian ini. Berikut adalah spesifikasi sensor RPLIDAR yang ditunjukkan pada tabel 4.7.

Tabel 4.7 Spesifikasi RPLIDAR A1

Sumber : (www.slamtec.com, 2021)

Keterangan	Spesifikasi
Measuring Range	0.15 – 12 meter
Angular Field of View	0 - 360°
Angular Resolution:	≤1°
Sampling frequency	4000 - 8000 Hz
Rotational Speed	5.5Hz
Laser wave length	775 - 795nm
Pulse width	110 us
System Voltage	5V
System Current	100mA
Power Consumption	0.5W
Output	UART Serial 3.3 Voltage level
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）

6. Sensor IMU GY-521

Sensor GY-521 memiliki 3 DOF akselerometer dan 3 DOF giroskop yang dapat mendeteksi nilai akselerasi dan gerakan roll, pitch atau yaw pada robot. Sensor GY-521 digunakan sebagai input dari sistem berupa mendeteksian posisi robot secara lokal. Hal ini sangat berguna dalam proses lokalisasi, walaupun proses lokalisasi dapat dilakukan hanya dengan menggunakan data dari odometri, akan tetapi masih perlu dilakukan validasi lebih lanjut lagi menggunakan sensor yang mampu membaca akselerasi robot. Ini dikarenakan terkadang ketika motor BLDC

tersangkut atau berputar ditempat maka data yang dikirimkan akan dianggap sedang melakukan suatu gerakan. Tentunya hal ini tidak diinginkan karena dapat membuat robot tidak bergerak semestinya dan posisinya akan berbeda ketika di aplikasi Rviz dan keadaan didunia riil. Maka dari itu atas pertimbangan ini, sensor GY-521 dipilih karena sangat dibutuhkan untuk memvalidasi nilai dari odometri dan membuat robot untuk tetap diposisi semestinya walaupun motor BLDC berputar ditempat. Berikut adalah spesifikasi sensor IMU GY-512 yang ditunjukkan pada tabel 4.8.

Tabel 4.8 Spesifikasi Sensor IMU GY-521 MPU6050

Sumber : (www.tokopedia.com, 2021)

Keterangan	Spesifikasi
Chipset	MPU-5060
Tegangan sistem kerja	3.3 - 5V DC
Jalur Komunikasi	I2C up to 400kHz
Range Gyroscope	250, 500, 1000, 2000/s
Range Gyroscope	2, 4, 6, 8, 16g
Pin	VCC, RX, TX, GND, RST, B0, SCL, SDA

7. STLink V2 Programmer

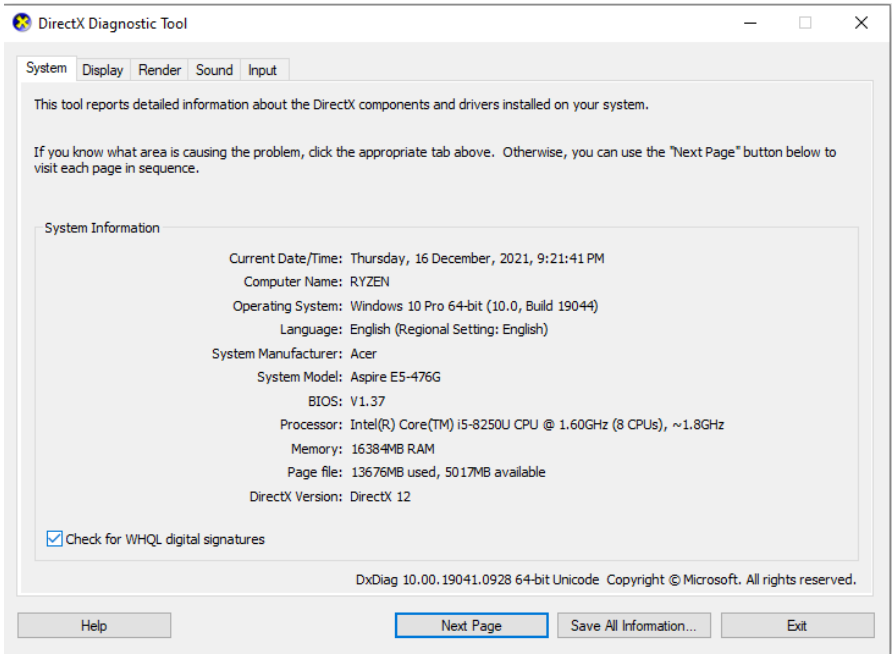
STLink bukanlah bagian dari komponen utama robot, alat ini digunakan dalam proses *flashing motherboard hoverboard*. Tepatnya STlink berfungsi sebagai perantara USB Flashing antara laptop dengan motherboard. Flashing dilakukan untuk mereset semua kodingan yang ada di motherboard sekaligus menuliskan ulang motherboad dengan kodingan baru yang sudah di modifikasi.

Gambar 4.4 Perangkat STLink V2 Programmer

Sumber: (indonesia.alibaba.com, 2021)

8. Personal Komputer (Laptop)

Laptop merupakan sebuah komputer fleksibel yang dapat dijinjing dengan mudah. Laptop yang digunakan seperti pada gambar 4.5 berjenis Acer Aspire E5-476G. Pada penelitian ini, laptop memiliki peran yang sangat penting dan banyak berkontribusi pada penelitian ini. Diantaranya adalah berguna sebagai perangkat dalam memprogram sistem pada tahap prototyping awal, baik dalam memprogram perangkat keras maupun perangkat lunak. Kemudian juga digunakan sebagai setting up Master - Slave pada robot operating system. Robot bergerak otonom merupakan masternya sedangkan laptop berperan sebagai slave. Fungsi slave (laptop) adalah menerima semua jenis data feedback yang dikirim atau diexport dari master (robot) yang kemudian data-data tersebut dapat digunakan untuk proses visualisasi menggunakan aplikasi *third party* Rviz. Dengan begini Mikrokomputer pada robot tidak perlu banyak menghabiskan resource untuk melakukan pemetaan sekaligus menampilkan visualisasi dari peta tersebut. Terkait spesifikasi personal komputer akan ditunjukkan pada gambar



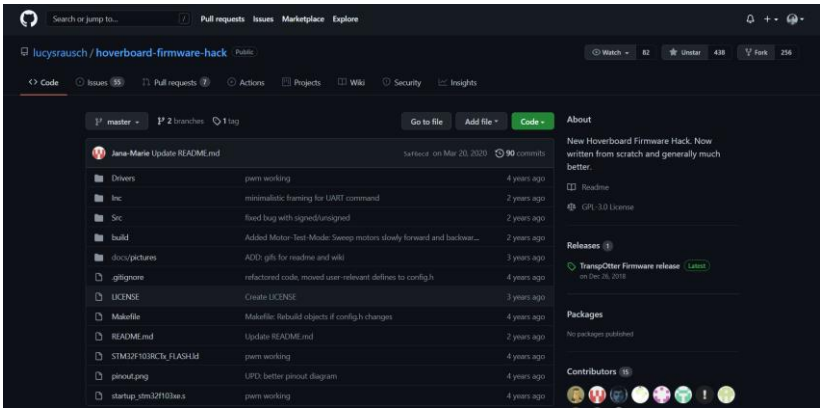
Gambar 4.5 Spesifikasi Laptop Acer e5-476G

4.2.2.2 Kebutuhan Non-Fungsional Perangkat Lunak

Kebutuhan non-fungsional adalah batasan dari sistem terhadap layanan yang diberikan. Batasan tersebut dapat berupa waktu, batasan proses pengembangan dan batasan terhadap standar tertentu.

1. Hoverboard Firmware

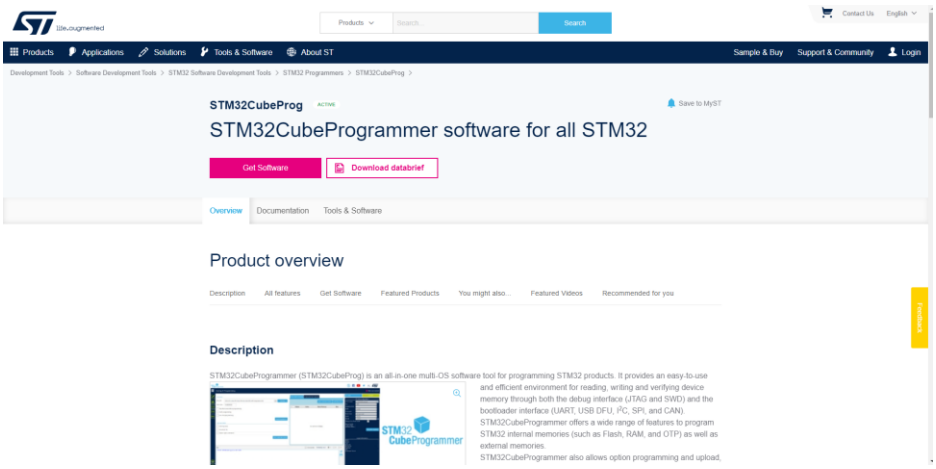
Firmware adalah kelas perangkat lunak komputer tertentu yang menyediakan kontrol tingkat rendah untuk perangkat keras khusus perangkat. Firmware, seperti BIOS komputer pribadi, mungkin berisi fungsi dasar perangkat, dan dapat menyediakan layanan abstraksi perangkat keras ke perangkat lunak tingkat tinggi seperti sistem operasi



Gambar 4.6 *Hoverboard* Firmware Hack di Github

2. STM32Cube Programmer

STM32CubeProgrammer adalah alat perangkat lunak multi-OS all-in-one untuk memprogram produk STM32. Ini menyediakan lingkungan yang mudah digunakan dan efisien untuk membaca, menulis, dan memverifikasi memori perangkat melalui antarmuka debug (JTAG dan SWD) dan antarmuka bootloader (UART, USB DFU, I2C, SPI, dan CAN). STM32CubeProgrammer menawarkan berbagai fitur untuk memprogram memori internal STM32 (seperti Flash, RAM, dan OTP) serta memori eksternal. STM32CubeProgrammer juga memungkinkan pemrograman dan unggah opsi, verifikasi konten pemrograman, dan otomatisasi pemrograman melalui skrip. STM32CubeProgrammer dikirimkan dalam versi GUI (antarmuka pengguna grafis) dan CLI (antarmuka baris perintah).

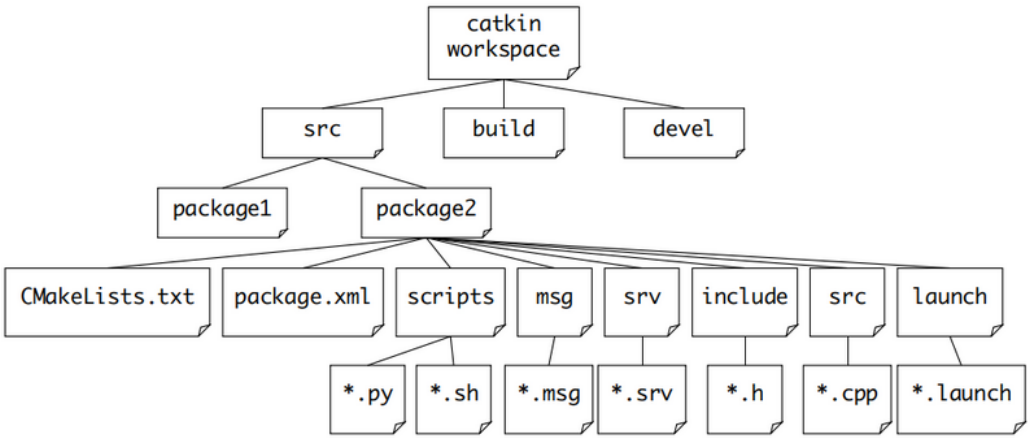


Gambar 4.7 STM32Cube Programmer

3. Catkin workspace

Ruang kerja catkin adalah direktori (folder) tempat membuat atau memodifikasi paket catkin yang ada. Struktur catkin menyederhanakan proses pembuatan dan pemasangan untuk paket ROS. Ruang kerja catkin dapat berisi hingga tiga atau lebih subdirektori yang berbeda (/build, /devel, dan /src), yang masing-masing memiliki peran berbeda dalam proses pengembangan perangkat lunak. Catkin merupakan komponen paling penting didalam ROS, hal ini

dikarenakan untuk menjalankan *middle ware* ROS kita perlu *build* terlebih dahulu kodingan ROS yang telah ditulis dan disimpan didalam folder /src. Kemudian pindahkan posisi terminal pada direktory /catkin_ws dan ketik $ catkin_make, maka catkin akan bekerja dan melakukan building file ke folder /build dan /devel. Jika sudah selesai maka perintah ROS sudah dapat digunakan dan kita dapat menjalankan sistem berdasarkan pemrograman yang dilakukan di folder /src seperti yang terdapat pada gambar 4.8.



Gambar 4.8 Diagram pohon direktory catkin workspace

Sumber : (www.medium.com/swlh, 2021)

4. Makefile

Make adalah utilitas untuk membangun dan memelihara grup program (dan jenis file lainnya) dari kode sumber. Tujuan dari utilitas make adalah untuk menentukan secara otomatis bagian mana dari program besar yang perlu dikompilasi ulang, dan mengeluarkan perintah yang diperlukan untuk mengkompilasi ulang (Computer Hope, 2021). Make digunakan untuk menginstall aplikasi dan dependensi ROS, melakukan compile firmware, melakukan flash firmware dan melakukan compiling pada file C lainnya dengan hasil output biner.

5. Terminator

Terminator adalah terminal alternatif untuk Linux yang hadir dengan sedikit fitur dan fungsionalitas tambahan yang tidak akan Anda temukan di aplikasi terminal default. Aplikasi ini digunakan sebagai terminal utama dalam melakukan penelitian ini, hal ini dikarenakan fiturnya lebih kaya dari pada terminal *default* milik linux. Kelebihan terminal ini adalah mampu membelah atau membagi tampilan tabnya menjadi beberapa bagian. Pada umumnya jika kita menggunakan terminal *default* linux maka fitur ini tidak ada, jika ingin membuka tab baru maka tab sebelumnya akan hilang. Ketidakefisienan ini membuat proses monitoring robot menjadi kurang nyaman, lagi pula kita sangat butuh melihat output yang dikirimkan pada tab terminal sebelumnya untuk memastikan dan melanjutkan proses pemanggilan perintah. Jika selalu berulang-ulang untuk membuka dan menutup tab, maka proses monitoring dan pengujian pada penelitian ini akan

menjadi lama. Untuk itu dibutuhkan terminal yang mampu membelah diri didalam satu tampilan *window.*

6. Visual Studio Code (IDE)

Visual Studio Code adalah teks editor ringan dan handal yang dibuat oleh Microsoft untuk sistem operasi multiplatform, artinya tersedia juga untuk versi Linux, Mac, dan Windows. Teks editor ini secara langsung mendukung bahasa pemrograman JavaScript, Typescript, dan Node.js, serta bahasa pemrograman lainnya dengan bantuan plugin yang dapat dipasang via marketplace Visual Studio Code (seperti C++, C#, Python, Go, Java, dst). Editor ini digunakan karena kemudahan dalam auto koreksi dan kemudahan navigasi antar file. Selain itu fitur debuggingnya dalam melakukan *tracking error* akan sangat membantu sekali.

7. *Extension* Platform IO (*VSCode Extension*)

Platformio adalah aplikasi console yang bisa diintegrasikan dengan IDE favorit atau text editor seperti Arduino IDE, Atom, Eclipse, Sublime, VIM, Visual Studio Code dan lain sebagainya. Namun beberapa waktu ini, Platformio telah merilis Platformio IDE yang dibangun diatas Atom IDE. yang membuat pemrograman berbagai jenis mikrokontroler kini lebih mudah, hanya pada satu aplikasi IDE. Penggunaan extension ini dikarenakan kemudahannya dalam proses compiling dan *flash* ke motherboard hoverboard. Dengan cukup melakukan 1 kali klik maka proses compile dan upload akan sekaligus dilakukan sehingga mempercepat proses pengembangan robot didalam penelitian ini.

8. VNC *Viewer*

VNC *viewer* merupakan software remote-control yang memungkinkan untuk mengontrol komputer lain melalui koneksi network. Pencetan keyboard dan mouse click dikirimkan dari satu komputer ke komputer lainnya sehingga seseorang dapat mengelola sebuah dekstop, server dan alat yang terhubung jaringan tanpa harus di lokasi yang sama. VNC viewer memungkinkan Jetson Nano terhubung dan dapat dikendalikan dengan laptop dari jarak jauh menggunakan jaringan wifi atau LAN.