Purwarupa *Autonomous Mobile Robot* Dengan *Hoverboard* dan Sensor RPLIDAR Menggunakan Algoritme Hector SLAM Dan Navfn

Bambang Gunawan Tanjung[1], Rizal Maulana[2], Rakhmadhany Primananda[3]

Program Studi Teknik Komputer, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]bambanggunawan@student.ub.ac.id, [2]rizalmaulana@ub.ac.id, [3]rakhmadhany@ub.ac.id

Abstrak

Setiap tahun, kurir ekspedisi mengalami peningkatan jumlah permintaan pengiriman barang. Peningkatan ini disebabkan oleh meningkatnya frekuensi belanja *online* pada *e-commerce*. Alhasil, banyak paket yang harus didistribusikan di gudang ekspedisi. Berdasarkan hal tersebut, diperlukan suatu sistem untuk mendistribusikan paket secara otonom di gudang sehingga proses pengangkutan barang menjadi lebih terotomatisasi. Sistem ini menggunakan sensor *encoder*, IMU GY-521, RPLIDAR, aktuator motor BLDC, dan *Robot Operating System* (ROS). *Autonomous mobile robot* (AMR) merupakan jenis robot yang paling cocok untuk sistem dengan ciri-ciri robot seperti ini. Penelitian ini berfokus pada penerapan navigasi AMR di dalam ruangan. Keunikan penelitian ini terletak pada penggunaan *hoverboard* (skuter *self-balancing*) sebagai sistem penggerak robot. Algoritme Hector SLAM digunakan untuk membuat peta lokal. Algoritme Navfn dan *Dynamic Window Approach* (DWA) untuk perencanaan global dan lokal. Untuk mendapatkan lokasi robot digunakan algoritme *Extended Kalman Filter* (EKF) dan *Adaptive Monte Carlo Localization* (AMCL). Hasil pengujian menemukan bahwa *hoverboard* bisa digunakan sebagai penggerak dengan daya angkut maksimal 40Kg. Pembacaan *odometry*, IMU, dan RPLIDAR menunjukkan hasil yang memuaskan, masing-masing sebesar 99,31%, 96,22%, dan 99,84%. Untuk pemetaan, algoritme Hector SLAM menunjukkan hasil dengan akurasi keberhasilan 80%. Pada Hector SLAM ditemukan kesimpulan menarik bahwa pembuatan peta di area tanpa dinding cenderung mengalami kegagalan. Untuk navigasi tanpa ataupun dengan halangan, perencanaan jalur global dari algoritme Navfn ternyata dapat bermanuver dengan baik, dibuktikan dengan hasil akurasi sebesar 80%.

Kata kunci: *ekspedisi, gudang, autonomous mobile robot, ROS, hoverboard, pemetaan, navigasi.*

Abstract

Every year, expedition couriers experience an increase in the number of requests for delivery of goods. This increase was due to the increasing frequency of online shopping on e-commerce. As a result, many packages must be distributed in the expedition warehouse. Based on this, a system is needed to distribute packages autonomously in the warehouse so that the process of transporting goods becomes automated. This system uses an encoder sensor, IMU GY-521, RPLIDAR, BLDC motor actuator, and Robot Operating System (ROS). An autonomous mobile robot (AMR) is the most suitable type for systems with these characteristics. This research focuses on the application of indoor AMR navigation. This research's uniqueness lies in using a hoverboard (self-balancing scooter) as a robot propulsion system. The Hector SLAM algorithm is used to create local maps. Navfn algorithm and Dynamic Window Approach (DWA) for global and local planning. For obtain the robot's location, the Extended Kalman Filter (EKF) and Adaptive Monte Carlo Localization (AMCL) algorithms. The test results found that the hoverboard is suitable for use as a prime mover with a maximum carrying capacity of 40 kilograms. The readings of odometry, IMU, and RPLIDAR showed satisfactory results, which were 99.31%, 96.22%, and 99.84%, respectively. For mapping, the Hector SLAM algorithm shows results with a success accuracy of 80%. Hector SLAM found an exciting conclusion that map-making in areas without walls tends to fail. For navigation without or with obstruction, the global path planning of the Navfn algorithm turns out to be able to maneuver well, as evidenced by the results of a success accuracy of 80%.

Keywords: *expedition, warehouse, autonomous mobile robot, ROS, hoverboard, mapping, navigation*

1. PENDAHULUAN

Setiap tahunnya, kurir ekspedisi di Indonesia mengalami peningkatan jumlah permintaan pengiriman barang (BPS 2017). Salah satu penyebabnya adalah meningkatnya frekuensi belanja *online* di platform *e-commerce* sejak tahun 2017 (Databooks 2019) dan akan terus bertumbuh hingga tahun-tahun berikutnya (Statista 2019). Jika peningkatan ini tidak diiringi dengan sistem distribusi yang lebih modern, maka akan menjadi masalah di tahun-tahun selanjutnya. Indonesia memiliki rata-rata waktu pengiriman berkisar 3.8 hari. Data ini didapat dari survei yang dilakukan oleh (Parcel Perform 2019). Survei ini juga mengemukakan bahwa 36% pelanggan mengeluhkan jasa kurir ekspedisi. Dari 36% pelanggan yang menyatakan ketidakpuasan, 90% di antaranya mengeluhkan tentang keterlambatan pengiriman, waktu transit, dan kurangnya komunikasi status pengiriman.

Dari kumpulan fakta ini, dapat disimpulkan bahwa setiap tahunnya jumlah permintaan pengiriman barang selalu meningkat, tetapi sistem distribusi di gudang penyimpanan mengalami kendala pada waktu. Proses suplai yang cepat ini mempengaruhi waktu, tenaga dan biaya yang digunakan. Dari sekian banyak masalah layanan ekspedisi yang ada, penulis mengambil sedikit porsi dari permasalahan distribusi di dalam gudang. Solusi sistem robot terotomatisasi dirasa dapat menyelesaikan permasalahan distribusi barang di dalam gudang. Bertujuan untuk mempercepat proses sortir dan perpindahan barang di dalam gudang. Sekaligus mengurangi kebutuhan tenaga angkut manusia, sehingga biaya operasional dapat berkurang secara signifikan.

Berdasarkan hal tersebut, diperlukan sistem untuk mendistribusikan paket secara otonom sehingga proses menjadi terotomatisasi. Sistem menggunakan sensor *encoder*, GY-521, RPLIDAR dan aktuator motor BLDC serta *Robot Operating System* (ROS) sebagai *middleware*. Sistem memiliki kemampuan untuk bernavigasi secara otonom dan mampu mengerti kondisi lingkungan sekitarnya, berdasarkan ciri tersebut *Autonomous mobile robot* (AMR) merupakan tipe robot yang paling cocok dengan sistem ini. AMR memiliki tugas dalam proses distribusi, yaitu untuk membawa barang dari proses penyortiran dan transit di dalam gudang.

Pada implementasinya (Utomo 2015) menyarankan untuk menggunakan sensor laser jarak jauh yang memiliki jangkauan dengan sudut 360 derajat seperti sensor RPLIDAR. Navigasi robot menggunakan perencanaan jalur global Navfn untuk mengoptimalkan waktu dan *resource*. Sebagai tambahan, digunakan perencanaan jalur lokal dengan algoritme DWA. Sebelum melakukan navigasi, robot membutuhkan sebuah peta lokal, maka dari itu Hector SLAM berperan sebagai algoritme untuk proses pemetaan. Selama proses pemetaan dan navigasi berlangsung, robot perlu mengetahui lokasi terkini secara simultan melalui proses lokalisasi. Lokalisasi yang dipakai menggunakan kombinasi algoritme AMCL dan EKF. Hasil kombinasi dari semua algoritme ini menghasilkan sistem yang dapat melakukan kemudi secara mandiri dan mampu menghindari halangan bergerak serta tetap melakukan pelacakan posisi terkini.

Hasil akhir dari penelitian ini berupa purwarupa robot otonom yang digunakan di dalam gudang penyimpanan ekspedisi dengan implementasi proses navigasi menggunakan peta lokal. Harapan dari penelitian ini, agar perusahaan jasa ekspedisi di Indonesia dapat meningkatkan efisiensi waktu, tenaga kerja dan mengurangi biaya produksi pada proses distribusi barang dengan memanfaatkan pengembangan *autonomous mobile robot*.

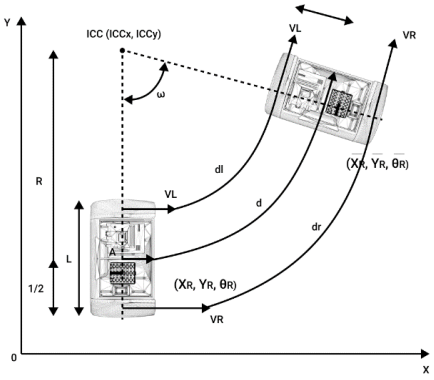
2. DASAR TEORI

2.1 Hoverboard

Hoverboard menggunakan 2 buah motor BLDC yang mampu berputar dengan kencang dan memiliki torsi yang besar. *Hoverboard* dapat digunakan sebagai *driver* motor untuk robot beroda karena memiliki *encoder* bawaan sehingga didapatkan data *odometry*. Pada penelitian ini *hoverboard* digunakan sebagai sistem penggerak utama dengan memanfaatkan motor BLDC, *motherboard*, dan baterainya.

2.2 Differential Drive

Differential drive adalah sistem penggerak dua roda dengan aktuator independen untuk setiap roda (Dudek dan Jenkin 2001). Agar robot dapat melakukan gerakan bergulir, robot harus berputar di sekitar titik yang terletak di sepanjang sumbu roda kiri dan kanan.



Gambar 1. Kinematics Differential Drive

Pada *Gambar 1* terdapat diagram kinematika dari *differential drive* dapat dituliskan persamaan berikut:

$$R = \left(\frac{L}{2}\frac{vr + vl}{vr - vl}\right) \quad (1)$$

$$\omega = \left(\frac{vr - vl}{l}\right) \quad (2)$$

Dari persamaan-persamaan di atas dapat ditarik beberapa hal, antara lain:

1. Jika $vl = vr$, maka akan diperoleh gerak lurus linear ke depan.
2. Jika $vl = -vr$, maka akan diperoleh rotasi pada titik $l/2$ atau berputar di tempat.
3. Jika $vl = 0$, maka akan diperoleh gerak poros ke kiri karena $R = l/2$, kemudian sebaliknya untuk roda kanan jika vr = 0.

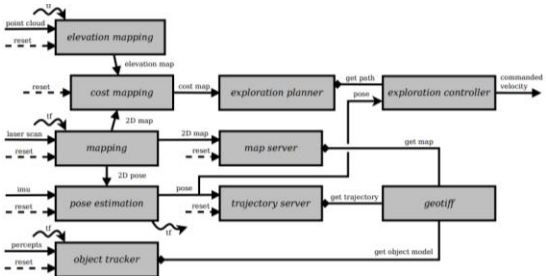
vl dan vr merupakan nilai kecepatan pada roda yang dapat dikontrol untuk mendapatkan gerak berbelok yang dibutuhkan. l merupakan jarak titik tengah di antara roda. Sedangkan R merupakan nilai $l/2$ yang merupakan jarak antara titik tengah.

2.2.1 *Simultaneous Localization and Mapping* (SLAM)

SLAM (*Simultaneous Localization And Mapping*) merupakan cara modern untuk memetakan lingkungan menjadi sebuah data menjadi sebuah peta. SLAM melakukan penjelajahan dan pemetaan lingkungan yang tidak diketahui sambil memperkirakan pose robot itu sendiri dengan menggunakan sensor yang terpasang pada robot (Pyo, dkk. 2017). Untuk melakukan pemetaan dibutuhkan pembaharuan lokasi robot terkini secara simultan. Sedangkan untuk melakukan pembaharuan lokasi dibutuhkan data peta dari hasil pemetaan. Istilah SLAM digunakan untuk permasalahan lokalisasi dalam membuat peta.

2.2.1.1 Hector SLAM

Hector SLAM berisi sekumpulan paket ROS yang digunakan untuk melakukan komputasi SLAM dengan mengkorelasikan perkiraan posisi robot dan peta di lingkungan yang tidak terstruktur (Kohlbrecher, dkk. 2014). Hector SLAM memiliki proses utama untuk pembuatan peta, yaitu Hector *Mapping*. Hector *Mapping* memanfaatkan laser Lidar dengan frekuensi tinggi tanpa menggunakan informasi *Odometry*. Hector *Mapping* menggunakan algoritma *scan matching* yang dikembangkan dengan cara menyelaraskan data dari sensor untuk membangun peta. Pada Gambar 2 terdapat skema Hector SLAM berupa proses *hector mapping*, *hector geotiff*, dan lain-lain beserta alur proses yang terjadi di dalamnya.



Gambar 2. Skema Sistem Hector SLAM
Sumber: (Kohlbrecher, dkk. 2014)

2.2.1.2 Hector *Mapping*

Hector *Mapping* adalah algoritma SLAM yang menghasilkan peta *occupancy grid* tanpa perlu menggunakan *odometry* menggunakan metode *scan matching*. Metode *scan matching* pada Hector *Mapping* berbasis pada pendekatan *Gauss-Newton* (Kohlbrecher, dkk. 2014). Dikombinasikan dengan *attitude estimation system* dan opsi unit *pitch/roll* untuk menstabilkan pemindai laser, sistem ini dapat menyediakan peta lingkungan bahkan jika tanahnya tidak datar. Terdapat dua persamaan yang menjadi formula utama pada proses pemetaan dengan Hector *Mapping*. Perpindahan posisi robot dihitung dengan persamaan 3 dan nilai H dihitung dengan persamaan 4.

$$\Delta\xi = H^{-1}\sum_{i=1}^{r}\left[\nabla M(S_i(\xi))\frac{\partial S_i(\xi)}{\partial \xi}\right]^T [1 - M(S_i(\xi))] \quad (3)$$

$$H = \sum_{i=1}^{r}\left[\nabla M(S_i(\xi))\frac{\partial S_i(\xi)}{\partial \xi}\right]^T \left[\nabla M(S_i(\xi))\frac{\partial S_i(\xi)}{\partial \xi}\right] \quad (4)$$

$\Delta\xi$ merupakan nilai perpindahan posisi robot dalam koordinat *cartesian* dan $S_i(\xi)$ merupakan koordinat hasil pemindaian lidar. Nilai M merupakan probabilitas adanya halangan pada koordinat hasil pemindaian.

2.2.2 Perencanaan Jalur Global Algoritme Navfn

Navfn diciptakan untuk melakukan optimalisasi pada algoritme *Dijkstra*. Paket Navfn menyediakan implementasi fungsi navigasi interpolasi yang cepat yang digunakan untuk membuat rencana untuk basis robot bergerak (Pyo, dkk. 2017). Karena merupakan algoritme optimalisasi dari *Dijkstra* maka Navfn memiliki kesamaan pada caranya dalam mencari tetangga, namun bedanya tidak seperti *Dijkstra* yang mengunjungi semua tetangga, Navfn hanya mengunjungi tetangga yang memiliki potensi terdekat dengan titik tujuan.

2.2.3 Perencanaan Jalur Lokal Algoritme *Dynamic Windows Approach* (DWA)

Menurut (Pyo, dkk. 2017) *Dynamic Window Approach* (DWA) adalah metode populer untuk perencanaan penghindaran halangan. Perencana jalur digunakan untuk perencanaan jalur lokal, tetapi DWA diganti karena kinerjanya yang unggul. Pertama, robot tidak berada pada koordinat sumbu x dan y, tetapi pada ruang pencarian kecepatan dengan kecepatan translasi v dan kecepatan rotasi sebagai sumbunya. Di ruang ini, robot memiliki kecepatan maksimum yang diizinkan karena keterbatasan perangkat keras, dan ini disebut *Dynamic Window*.

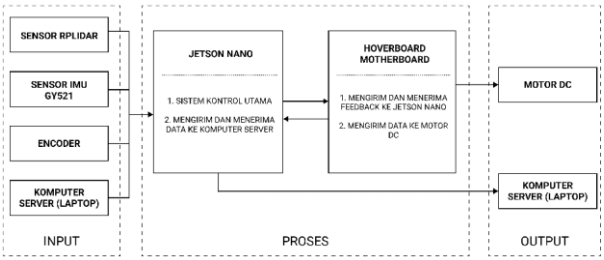
2.3 *Robot Operating System* (ROS)

ROS adalah sistem operasi meta yang menyediakan layanan seperti sistem operasi, abstraksi perangkat keras, kontrol perangkat tingkat rendah, implementasi fungsionalitas umum, pengiriman pesan antar proses, dan manajemen paket. ROS mengembangkan, mengelola, dan menyediakan aplikasi paket untuk berbagai keperluan. ROS mampu menjalankan banyak paket dan sensor sekaligus sehingga sistem tampak terlihat paralel. Keistimewaan ROS terletak pada dukungan *library*, aplikasi pihak ketiga, dan komunitas yang sangat aktif. Hal ini yang menjadi alasan utama mengapa penelitian menggunakan ROS sebagai *framework* utama sistem.

3. PERANCANGAN DAN IMPLEMENTASI

3.1 Gambaran Umum Sistem

Sistem yang dibuat pada penelitian ini merupakan sistem yang mempunyai fungsi untuk membawa barang dari titik awal menuju titik tujuan yang bergerak menggunakan roda serta mendeteksi area sekitar.



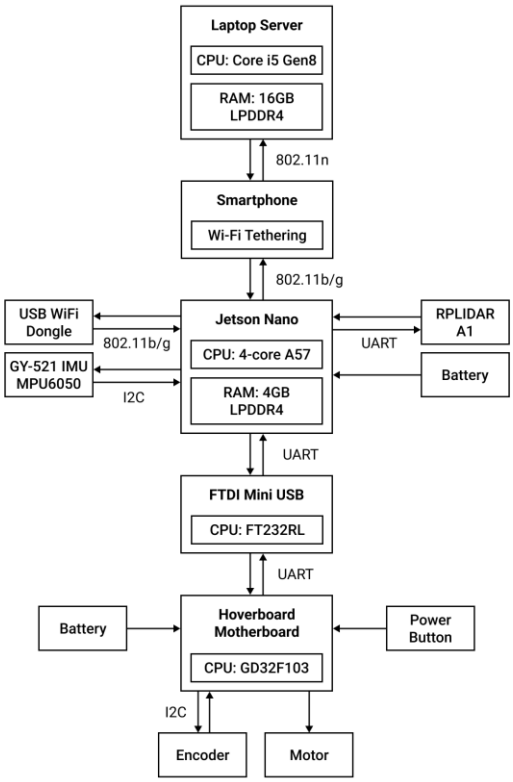
Gambar 3. Blok Diagram Sistem

Dari Gambar 3 menunjukkan blok diagram dari sistem yang memiliki tiga bagian yaitu *input*, proses, dan *output*. Bagian *input* dari sistem akan menggunakan sensor RPLIDAR, IMU dan Odometer. Ketiga sensor ini digabung menjadi satu kesatuan untuk menjalankan proses lokalisasi. Hasil dari sensor RPLIDAR akan digunakan untuk lokalisasi dengan metode AMCL di mana data laser dan map akan digabungkan sehingga menghasilkan perkiraan pemosisian. *Input* dari komputer server berupa perintah-perintah *command line* dan *input* letak koordinat tujuan melalui RViz atau terminal. Semua bagian *input* ini diteruskan ke bagian proses untuk nantinya diolah oleh Jetson Nano sebagai sistem kontrol utama. Pada bagian proses sistem menggunakan dua alat yaitu Jetson Nano dan *hoverboard*. Jetson Nano menjadi sistem kendali utama dari sistem yang bertugas untuk mencakup seluruh proses pengolahan data sensor, data odometer, dan proses SLAM serta navigasi. Sedangkan *hoverboard* sebagai penggerak utama yang dikendalikan oleh Jetson Nano.

3.2 Perancangan Perangkat Keras

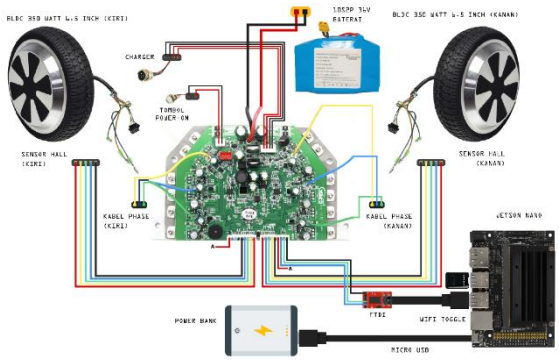
Gambar 4 merupakan diagram rancangan umum perangkat keras sistem. sistem menggunakan perangkat utama yang saling terintegrasi satu sama lain menggunakan beberapa protokol komunikasi sebagai penghubung. Jetson Nano merupakan komponen utama penyusun sistem penggerak. Sistem penggerak berperan sebagai aktuator yang sekaligus mengirimkan data berupa data

odometry. Sistem terdiri dari *motherboard hoverboard*, motor BLDC, baterai, *switch power on/off* dan Jetson Nano yang terhubung dengan FTDI melalui protokol UART. Sedangkan *motherboard hoverboard* berfungsi sebagai motor *driver* utama yang mengatur arah putar dan kecepatan putar motor BLDC.



Gambar 4. Rancangan Umum Perangkat Keras

3.2.1 Perancangan Perangkat Keras Sistem Utama



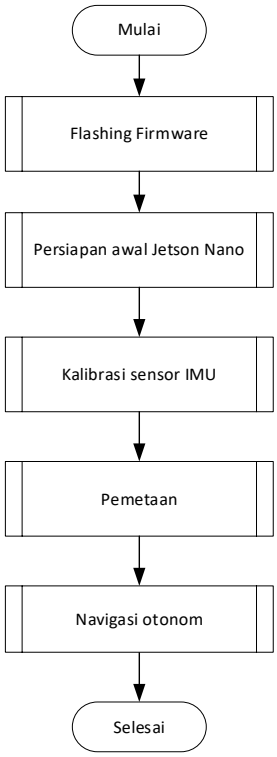
Gambar 5. Skema Sistem Penggerak Robot

Pada Gambar 5 terdapat diagram skematis dari sistem gerak robot di mana *motherboard hoverboard* yang berfungsi sebagai *driver* motor BLDC. Kemudian untuk dapat berkomunikasi dengan Jetson Nano, kita perlu menghubungkan *motherboard* dengan kabel sensor yang ada di sisi bawah *motherboard hoverboard*. Kedua sisi sensor *motherboard hoverboard* memiliki total 8 buah kabel yang masing-masing terdiri dari VCC, RX, TX, dan *ground*.

Gambar 6 merupakan gambar Rancangan 3D Keseluruhan Sistem yang terdiri dari bagian *mainboard* robot beroda sebagai penggerak, *chassis* robot sebagai kerangka dan sensor robot sebagai penglihatan dari robot.

Gambar 6. Rancangan 3D Keseluruhan Sistem

3.3 Perancangan Perangkat Lunak

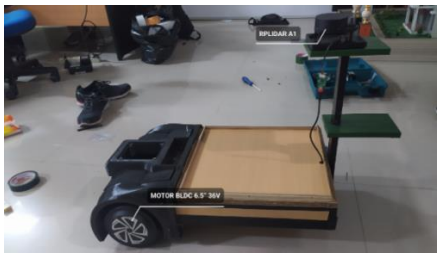


Gambar 7. Diagram Alir Program Utama

Perangkat lunak terdiri dari program perancangan utama yang menjadi keseluruhan alur dari semua proses. Perancangan program utama terbagi menjadi 4 proses besar yakni proses persiapan awal, kalibrasi sensor IMU, pemetaan dan navigasi otonom. Diagram alir proses dari program utama dapat dilihat pada Gambar 7. Semua urutan program ini sudah mencakup hal-hal atau kriteria yang dibutuhkan untuk membuat robot dengan sistem

Autonomous Mobile Robot yaitu pemetaan dan navigasi. Hal pertama yang dilakukan dalam program ini adalah melakukan kalibrasi pada sensor IMU. Proses kalibrasi yang hanya dilakukan pada sensor IMU dikarenakan sensor IMU membutuhkan nilai kalibrasi pada setiap kondisi peletakannya. Jadi jika sensor IMU dipindahkan ke tempat lain, maka perlu dilakukan kalibrasi ulang. Hal ini tidak diperlukan kedua sensor lainnya, yaitu RPLIDAR dan *encoder* pada motor BLDC dikarenakan data *raw* yang konsisten dan perangkat dilengkapi dengan *microcontroller* masing-masing.

3.4 Implementasi Keseluruhan Sistem



Gambar 8. Implementasi Keseluruhan Sistem

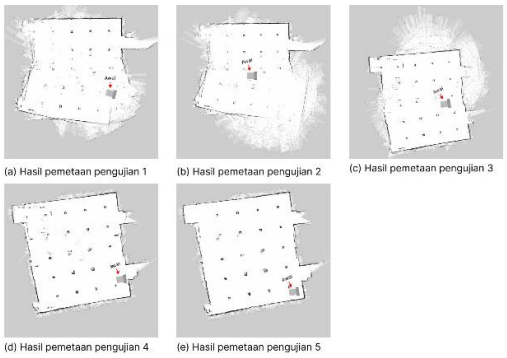
Setelah melakukan tahap perancangan maka dilakukan implementasi alat dan sistem. Untuk implementasi alat dan sistem ditunjukkan pada Gambar 8.

4. PENGUJIAN DAN ANALISIS

Pengujian pada utamanya terdiri dari pengujian pemetaan dan navigasi. Pengujian pemetaan dilakukan pada kondisi permulaan yang berbeda. Sedangkan untuk pengujian navigasi dilakukan pada kondisi posisi dan jarak yang berbeda.

4.1 Pengujian Pemetaan

Pengujian ini dilakukan dengan mengaktifkan program pemetaan dan melakukan manuver robot secara manual menggunakan *rqt robot steering*. Tujuan dari pengujian pemetaan menggunakan Hector SLAM. Hasil pengujian dari pemetaan menggunakan Hector SLAM dilakukan sebanyak 5 kali percobaan dengan posisi awal pemetaan yang berbeda-beda. Berdasarkan bentuk dari denah gedung G FILKOM yang terlihat pada dapat disimpulkan bahwa dari kelima percobaan yang dilakukan 4 dari 5 kali percobaan sesuai bentuk peta dengan denah yang ada. Kemudian dihitung akurasi dari 5 kali didapatkan hasil akurasi sebesar 80% yang terdapat pada persamaan 6.



Gambar 9 Hasil Pemetaan

$$Akurasi = 12/15 \times 100 = 80\% \quad (5)$$

Tabel 1 Hasil Pengukuran Peta Dan Denah Basemen

No	Peta Pengujian		Denah Basemen	
	Panjang	Lebar	Panjang	Lebar
1	33.519	37.9165	31.32	37.1
2	36.2777	36.9291	31.32	37.1
3	34.842	37.971	31.32	37.1
4	33.1916	38.0782	31.32	37.1
5	33.6972	37.6029	31.32	37.1

Pada pengujian pengukuran peta pada *Tabel 1*, didapatkan hasil ukuran panjang dan lebar dari peta yang dihasilkan pada proses *mapping*. Hasil pengujian pengukuran peta menghasilkan akurasi dengan nilai 94.43% yang dihitung menggunakan WAPE yang dapat dilihat pada *Tabel 2*. Dari kelima peta yang berhasil dibuat, peta ke-2 merupakan peta yang dianggap gagal pembuatannya karena hasil panjang dan lebar yang jauh dari ukuran asli denah. Peta ke-2 juga tidak mampu mendeteksi batas tembok dari gudang, maka dari itu peta ini dianggap gagal.

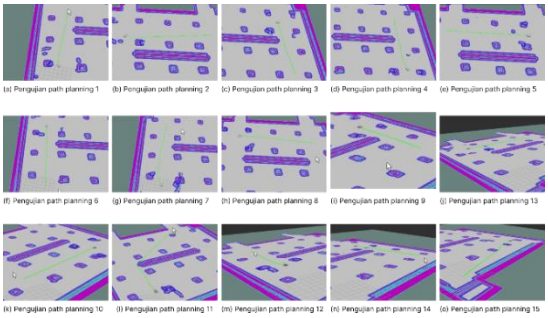
Tabel 2 Hasil Akurasi Pengujian Pengukuran Peta Dan Denah Basemen

	Total Keseluruhan			
	Panjang	Panjang	Lebar	Lebar
Total berat data	171.5275	156.6	188.4977	185.5
WAPE	0.09532247765		0.01616010782	
Rata-rata WAPE	5.57%			
Nilai akurasi	94.43%			

4.2 Pengujian Navigasi Pada Halangan Diam

Pengujian ini dilakukan sebanyak 15 kali percobaan dengan titik *goal* yang berbeda-beda. Keberhasilan dari pengujian diwakilkan dengan kecepatan waktu komputasi yang dilakukan oleh algoritme Navfn dan akurasi manuver robot saat berjalan. Tujuan dari pengujian ini yaitu untuk

mengetahui seberapa besar akurasi manuver pergerakan robot dan waktu komputasi yang dilakukan oleh sistem ketika melakukan navigasi secara *autonomous* tanpa kemudi dari *user* yang hanya memasukkan *input* berupa titik tujuan. Selain itu pengujian ini juga sebagai penentu apakah robot AMRN ini layak untuk digunakan pada area gudang kurir karena jika pergerakannya tidak stabil maka berakibat tabrakan dengan benda lain.



Gambar 10 Hasil Pengujian Perencanaan Jalur Global

Hasil pengujian direpresentasikan dengan gambar posisi awal dan akhir untuk melihat seberapa jauh posisi robot berubah dari yang seharusnya.

Tabel 3 Pengujian Waktu Komputasi Algoritme Navfn

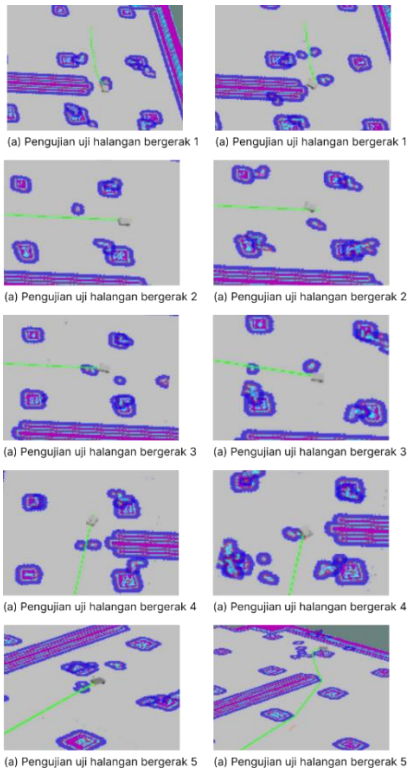
No	Waktu Komputasi Algoritme Navfn (detik)	Waktu Tempuh Manuver (detik)	Total Waktu Navigasi
1	0.37	41.53	41.9
2	0.99	56.11	57.1
3	0.84	40.29	41.13
4	0.41	59.24	59.65
5	0.88	52.08	52.96
6	1.53	52.43	53.96
7	0.63	67.06	67.69
8	1.23	46.61	47.84
9	1.32	37.53	38.85
10	0.92		
11	0.98		
12	0.67	42.35	43.02
13	1.31	54.38	55.69
14	0.45	40.72	41.17
15	1.4		
Rata-rata waktu tempuh (detik)			50.08

Pengujian akurasi dan waktu komputasi ini dilakukan sebanyak 15 kali pada masing-masing perbedaan posisi untuk mencari tahu akurasi dan waktu komputasi. Pada data Tabel 6.12 diketahui total navigasi yang berhasil adalah 12 dari 15 kali percobaan. Navigasi dikatakan berhasil apabila dapat bermanuver dari titik awal menuju titik tujuan. Terdapat 3 buah pengujian yang menunjukkan kegagalan pada lokalisasi dan perencanaan jalur, sehingga proses robot tidak pernah sampai ke titik tujuan. Didapatkan hasil akurasi yang terdapat pada persamaan 6.

$$Akurasi = 12/15 \times 100 = 80\% \quad (6)$$

Pada pengujian waktu komputasi dan tempuh dari navigasi yang terdapat pada Tabel 6.13 didapatkan hasil rata-rata waktu tempuh sebesar 50.08 detik. Analisa ini dilakukan dengan menghitung waktu titik awal dan titik tujuan robot. Dengan memanfaatkan fitur waktu yang ada pada aplikasi *Rviz*, tepatnya pada *Wall Time*, dapat dihitung selisih selama proses navigasi. Dari rata-rata dari waktu komputasi ini menunjukkan bahwa penggunaan algoritme Navfn dan DWA tidak memerlukan *resource* yang besar serta memiliki waktu komputasi yang cepat, yakni kurang dari 1 detik. Kecepatan dalam penentuan jalur ini membuat algoritme Navfn menjadi perencanaan jalur paling cocok dengan sistem di dalam gudang.

4.3 Pengujian Navigasi Pada Halangan Bergerak



Gambar 11 Hasil Pengujian Menghindari Halangan Bergerak Dengan Perencanaan Jalur Lokal

Pengujian akurasi dan waktu komputasi ini dilakukan sebanyak 5 kali pada masing-masing perbedaan posisi untuk mencari tahu

akurasi dan waktu komputasi yang dibutuhkan Navfn dalam menentukan jalur global. Dihitung akurasi dari 5 kali didapatkan hasil akurasi yang terdapat pada persamaan .

$$Akurasi = 12/15 \times 100 = 80\% \quad (7)$$

Kesimpulannya adalah sistem mampu mendeteksi halangan dengan baik dan cenderung berhenti ketika halangan di depannya melaju di depan. Namun terkadang terdapat kegagalan perencanaan jalur ulang ketika halangan bergerak terlalu dekat dan cepat. Sistem bahkan mampu menghindari halangan bergerak di depannya berkat data *point cloud* dari RPLIDAR

5. KESIMPULAN

Pada pengujian pemetaan menggunakan algoritme Hector SLAM menghasilkan nilai akurasi sebesar 80%. Ditemukan permasalahan pemilihan posisi awal pemetaan. Ketika robot berada di luar area di sekitar tembok atau area sekitar tidak dapat terbaca oleh laser RPLIDAR dikarenakan ketidakakuratan TF. Sehingga posisi robot pada sistem tidak sinkron terhadap posisi sebenarnya. Pada pengujian navigasi tanpa halangan bergerak menggunakan algoritme Navfn dan DWA menghasilkan akurasi sebesar 80%. Dari 15 kali percobaan navigasi, terdapat 3 kali percobaan yang gagal. Hal ini dikarenakan pada percobaan tersebut posisi awal navigasi, dimulai dari tengah basemen yang mana RPLIDAR tidak dapat membaca area tembok terdekat, sehingga lokalisasi mengalami *error*. Untuk waktu navigasi didapatkan rata-rata navigasi 50.08 detik pada jarak tempuh 20-30 meter.

Pada pengujian navigasi dengan halangan dinamis atau bergerak menggunakan perencanaan global Navfn dan lokal DWA. Dilakukan pengujian sebanyak 5 kali dengan nilai akurasi sebesar 80%. Nilai ini didapatkan setelah melakukan analisis manuver robot ketika halangan bergerak ke bagian depan robot. Reaksi robot menjadi penentu apakah pengujian perencanaan lokal ini berhasil atau tidak. Dari 5 percobaan terdapat 1 kegagalan, di mana robot mengalami *error* pada lokalisasinya sehingga proses navigasi tidak dapat dilanjutkan. Namun 4 di antaranya dianggap berhasil karena mampu melewati halangan dengan sangat baik.

Semua komponen dan algoritme yang digunakan terbukti bekerja dengan baik. Solusi sistem otomatisasi pada latar belakang dapat terjawab pada penelitian ini. Dari keseluruhan pengujian dan implementasi, terdapat tiga hal yang dapat diperbaiki dan menjadi masukan untuk penelitian selanjutnya. Pertama menggunakan sistem kemudi *steering Ackerman* agar kemudi mudah dikendalikan. Kedua menggunakan variasi lebih banyak pengujian pemetaan dan navigasi. Ketiga menggunakan roda *omni wheel* dengan kombinasi 4WD motor BLDC agar robot dapat bergerak ke segala arah.

6. DAFTAR PUSTAKA

BPS. 2017. *Laporan Hasil Survei Triwulanan Kegiatan Usaha Terintegrasi 2017*. Jakarta: Badan Pusat Statistik.

Databooks. 2019. “Tren Pengguna E-Commerce Terus Tumbuh.” *Dkatadata.co.id*.

Dudek, Gregory, dan Michael Jenkin. 2001. “Computational Principles of Mobile Robotics.” in *Computational Principles of Mobile Robotics*.

Kohlbrecher, Stefan, Johannes Meyer, Thorsten Graber, Karen Petersen, Uwe Klingauf, dan Oskar Von Stryk. 2014. “Hector open source modules for autonomous mapping and navigation with rescue robots.” *Lecture Notes in Computer Science*.

Parcel Perform. 2019. “Consumers are still not happy with their e-commerce delivery experience, a new survey by Parcel Perform and iPrice Group reveals | Mini Me Insights.” *Minime Insights*. Diambil 12 Maret 2022.

Pyo, YoonSeok, HanCheol Cho, RyuWoon Jung, dan TaeHoon Lim. 2017. *Robot Programming From The Basic Concept To Practical Programming and Robot Application*. First Edit. Seoul, Republic of Korea: ROBOTIS Co., Ltd.

Statista. 2019. *E-commerce in Indonesia - statistics & Facts*.

Utomo, Eko Budi. 2015. “Autonomous Mobile Robot Berbasis Landmark Menggunakan Particle Filter Dan Occupancy Grid Maps Untuk Navigasi, Lokalisasi dan Mapping.”