

BAB I

PENDAHULUAN

1.1 Latar Belakang

Waiter Robot atau robot pelayan adalah salah satu jenis robot yang telah diperkenalkan pada awal tahun 2000 di berbagai industri makanan dan minuman di beberapa negara (Quigley et al., n.d.). Robot pelayan atau waiter-bot dirancang membantu distribusi logistik, makanan di restoran cepat saji dan lingkungan rumah sakit (Prayoga & Kurniawan, 2018). Menurut Bruno Siciliano (2016), tantangan utama dalam merealisasikan robot layanan yang handal Adalah 1) kemampuan robot untuk bernavigasi secara otonom pada lingkungan dinamis 2) Kemampuan berinteraksi yang intuitif dengan pengguna (human-robot interaction).

Dalam menghadapi tantangan navigasi dan interaksi, diperlukan robot yang memiliki pergerakan yang baik dan halus, selain itu antarmuka pengguna atau Graphical User Interface (GUI) memegang peranan sebagai jembatan antara operator dan sistem robot (Yoga Prihastomo & Winanti, 2024). Pengembangan antarmuka yang responsif sering kali menghadapi tantangan integrasi platform. Singla (2023) dan Lundh (2022) mengusulkan penggunaan kerangka kerja python seperti Kivy untuk membangun aplikasi kontrol yang fleksibel dan dapat berjalan secara multi-platform, memungkinkan operator untuk memantau status robot dan mengganti mode operasi secara real-time.

Pada penelitian sebelumnya, Salam (2024) mengembangkan sistem navigasi robot otonom waiter-bot berbasis Robot Operating System (ROS) untuk pengantaran logistic secara autonomous dalam mencapai misi dan penelitian lanjutan Faizal (2025) mengembangkan sistem pemetaan Waiter-Bot menggunakan metode sensor fusion diantaranya Rotary Encoders, Inertial Measurement Unit (IMU) dan Kinect agar menghasilkan peta yang optimal digunakan. Waiter-Bot mampu melakukan pemetaan dan pergerakan secara otonom, namun terdapat beberapa kendala yang menjadi isu kritis pada waiter-bot yaitu kendala mekanis pada sistem kendali gerak yang ditemukan tercatat kasar akibat sinyal kontrol motor yang mendadak. Borenstein & Feng (1996) dalam analisis mendalam mengenai sistem odometry errors menjelaskan bahwa percepatan atau perlambatan mendadak pada robot beroda adalah penyebab utama terjadinya selip roda. Selip ini sangat fatal bagi odometri karena menyebabkan ketidaksesuaian antara data putaran roda yang dibaca encoder dengan pergerakan aktual robot di lantai. Akumulasi kesalahan odometri ini dapat menyebabkan robot gagal mencapai titik target dengan presisi. Dari sisi mapping, pembacaan odometri buruk menyebabkan robot sering kehilangan orientasi (lost localization) saat berada di tengah ruangan besar, kualitas peta grid (occupancy grid map) sangat rentan terhadap drift odometri dimana kesalahan kecil pada estimasi sudut (angular error) dapat mengakibatkan peta yang dihasilkan terdistorsi (Siegwart et al., 2011).

Untuk mengatasi permasalahan gerak motor yang kasar diperlukan teknik kendali motor yang efisien dan untuk meningkatkan pengalaman interaksi robot dengan pengguna diperlukan GUI sebagai perangkat antarmuka pengguna.

Berdasarkan uraian di atas, Penelitian ini bertujuan untuk mengimplementasikan PWM ramp rate untuk kontrol gerak motor yang lebih halus guna meningkatkan akurasi pemetaan dan navigasi otonom, serta mengintegrasikan sistem tersebut ke dalam antarmuka pengguna (GUI) Kivy untuk pengoperasian yang lebih baik.

1.2 Rumusan Masalah

1. Bagaimana laju performa sistem Waiter-Bot dengan implementasi PWM ramp rate?
2. Bagaimana implementasi Kivy pada sistem GUI pada Waiter-Bot?

1.3 Tujuan Penelitian

1. Mengimplementasikan pendekatan PWM ramp rate dengan sistem kendali
2. Menganalisis performa pemetaan setelah pengintegrasian PWM ramp rate
3. Menguji performa sistem Waiter-Bot secara keseluruhan
4. Mengimplementasikan aplikasi antarmuka (GUI) pada Waiter-Bot.

1.4 Manfaat Penelitian

1. Meningkatkan kehalusan pergerakan Waiter-Bot
2. Memudahkan Penggunaan Waiter-Bot

1.5 Tinjauan Pustaka

1.5.1 Penelitian Terkait

RANCANG BANGUN SISTEM PEMETAAN DAN LOKALISASI BERBASIS ALGORITMA SLAM MENGGUNAKAN DEPTH SENSOR KINECT PADA MOBILE ROBOT – Fariz Achmad Faizal (2025)

Memfokuskan penelitiannya pada rancangan sistem pemetaan dan lokalisasi pada mobile robot menggunakan algoritma SLAM (Simultaneous Localization and Mapping). Sistem ini mengintegrasikan sensor visual depth camera (Kinect Xbox 360) dengan metode sensor fusion (menggabungkan data encoder dan IMU). Penelitian ini berhasil mendemonstrasikan kemampuan robot dalam memetakan ruangan skala kecil dan skala besar.

RANCANG BANGUN SISTEM NAVIGASI ROBOT OTONOM WAITER BOT BERBASIS ROBOT OPERATING SYSTEM – Abd. Salam (2024)

Melakukan rancang bangun sistem navigasi otonom pada robot waiter-bot berbasis ROS dengan Navigation Stack dan Dynamic Window Approach (DWA). Robot dilengkapi dengan kemampuan mengenali batas jalur lintasan dan melakukan navigasi menuju goal yang telah ditentukan dalam peta digital

SISTEM NAVIGASI PADA PROTOTYPE ROBOT KURSI BERODA UNTUK PENYANDANG DISABILITAS - Muhammad Takbir Machmud (2019)

Mengembangkan sebuah prototipe robot kursi roda yang dilengkapi sistem navigasi untuk penyandang disabilitas. Penelitian ini memadukan mekanisme pandu arah berbasis gambar menggunakan sensor kinect dan pandu arah berbasis suara. Fokus penelitian ini pada aspek bantuan disabilitas (assistive device).

INTRODUCTION TO AUTONOMOUS MOBILE ROBOTS SECOND EDITION - Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza (2011)

Dalam literatur standar robotika, menjelaskan secara teoritis bahwa perubahan kecepatan yang mendadak atau jerk (sentakan) dapat menghasilkan gaya inersia yang besar pada bodi robot. Gaya ini memicu terjadinya osilasi atau guncangan hebat pada struktur robot, yang tidak hanya merusak kenyamanan tetapi juga menyebabkan sensor

(seperti kamera atau LiDAR) kehilangan kestabilan pembacaan data, sehingga mempersulit lokalisasi robot.

1.5.2 Robot Pelayan (Service Robot)

Robotika adalah cabang ilmu interdisipliner yang menggabungkan teknik mesin, teknik listrik, dan ilmu komputer yang berhubungan dengan desain, konstruksi, operasi, dan aplikasi robot. Dalam perkembangannya, definisi robot telah bergeser dari sekadar mesin otomatis yang diprogram ulang (reprogrammable) menjadi sistem cerdas yang mampu mempersepsi lingkungan dan mengambil keputusan secara otonom. Struktur dasar robot dimodifikasi untuk mendukung mobilitas di lingkungan indoor maupun outdoor. Menurut Anshar (2025) ada lima komponen yang membangun robot: perwujudan, persepsi, aksi, otak dan kekuatan, dan otonomi.

Klasifikasi robot secara global dibagi menjadi dua domain utama: robot industri dan robot layanan. robot pelayan (Service Robot) didefinisikan sebagai robot yang melakukan tugas-tugas yang bermanfaat bagi manusia atau peralatan, tidak termasuk aplikasi otomasi industri manufaktur. dalam studi tinjauan komprehensifnya mengkategorikan robot pelayan menjadi dua jenis:

1. Robot Pelayan Personal (Personal Service Robot), digunakan untuk kepentingan individu non-komersial. Contohnya robot pembersih lantai (vacuum cleaner) dan robot pendamping edukasi.
2. Robot Pelayan Profesional (Professional Service Robot), dioperasikan untuk tujuan komersial dan sering dikelola oleh operator terlatih. Contohnya robot logistik gudang, robot medis, dan lainnya (Gonzalez-aguirre et al., 2021).

Waiter-Bot termasuk dalam robot kategori pelayan professional, tantangan utama bagi robot jenis ini adalah mobilitas. Robot harus mampu berpindah dari titik A ke titik B (navigasi) tanpa menabrak objek statis maupun dinamis, serta memiliki tingkat otonomi yang cukup untuk menangani ketidakpastian lingkungan tanpa intervensi manusia yang terus-menerus.



Gambar 1 Service Robot

1.5.3 Kinematik Robot Beroda (Differential Drive)

Mekanisme pergerakan (locomotion) adalah aspek paling fundamental dalam desain robot mobil (mobile robot). Untuk aplikasi dalam ruangan (indoor) yang memiliki permukaan datar, differential drive adalah pilihan yang paling efisien dari segi energi dan pengendalian. mekanisme ini terdiri dari dua roda penggerak independen yang terletak pada satu sumbu putar yang sama, dilengkapi dengan satu atau lebih roda kastor (caster wheel) pasif sebagai penyeimbang stabilitas statis (Lynch & Park, 2017). Keunggulan utama mekanisme ini adalah sifatnya yang meskipun robot tidak dapat bergerak secara lateral (menyamping) secara instan, ia memiliki kemampuan yaitu dapat berputar 360 derajat di tempat dengan memutar kedua roda pada kecepatan yang sama tetapi berlawanan arah. Kemampuan manuver ini sangat krusial untuk operasional di lingkungan yang sempit dan penuh rintangan.

$$VR = r \cdot \theta R \quad 1$$

$$VL = r \cdot \theta L \quad 2$$

r : Jari-jari roda penggerak (meter)

L : Jarak antara pusat roda kiri dan roda kanan (meter)

θR : Kecepatan sudut roda kanan (rad/s)

θL : Kecepatan sudut roda kiri (rad/s)

Dimana,

$$VL = \frac{Vr + Vl}{Vr - vl} = \frac{r}{2} (\theta R + \theta L) \quad 3$$

vr : Kecepatan roda kanan terhadap lantai

vl : Kecepatan roda kiri terhadap lantai

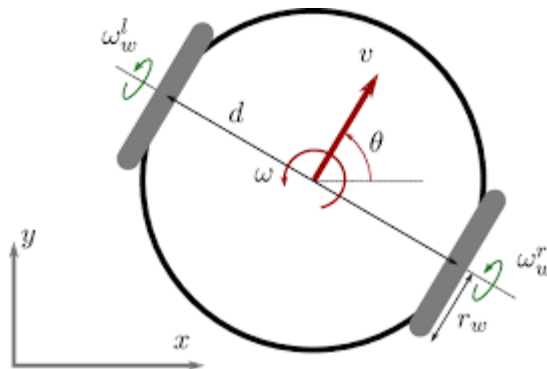
Kecepatan rotasi/angular robot (ω) disekitar pusat sumbu sesaat (Instantaneous Center of Curvature) dihitung dari selisih kecepatan roda dibagi dengan lebar wheelbase:

$$\omega = \frac{Vr - Vl}{L} = \frac{r}{L} (\theta R - \theta L) \quad 4$$

Kinematika balik diperlukan dalam sistem kendali untuk mengonversi perintah kecepatan target robot ($v_{target}, \omega_{target}$) yang dihasilkan oleh algoritma navigasi ROS menjadi kecepatan putar motor yang harus dieksekusi.

$$Vr = v_{target} + \frac{\omega_{target} \cdot L}{2} \quad 5$$

$$Vl = v_{target} - \frac{\omega_{target} \cdot L}{2} \quad 6$$



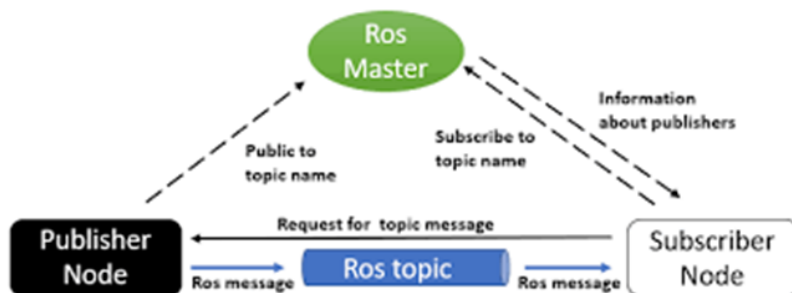
Gambar 2 Differential Drive

1.5.4 Robot Operating System (ROS)

ROS adalah Sistem Operasi Robot bersifat open source yang telah memiliki beberapa library untuk membuat software robot. ROS bertujuan untuk memudahkan pengembang robot tanpa harus membuat kode dari awal serta dapat dikembangkan bersama-sama (Jalil, 2018).

Dalam pengembangan sistem robotika modern yang kompleks, integrasi antara berbagai perangkat keras (sensor, aktuator) dan perangkat lunak (algoritma persepsi, navigasi) merupakan tantangan utama. Untuk mengatasi hal ini, penelitian ini memanfaatkan Robot Operating System (ROS). ROS bukanlah sistem operasi dalam pengertian tradisional (seperti Windows atau Linux), melainkan sebuah meta-operating system atau perangkat lunak penengah (middleware) yang berjalan di atas sistem operasi host (biasanya Ubuntu Linux). ROS menyediakan layanan standar yang diharapkan dari sebuah sistem operasi, termasuk abstraksi perangkat keras, pengendalian perangkat tingkat rendah (low-level device control), fungsi umum ROS biasanya digunakan sebagai penyampaian pesan antar-proses (inter-process communication), dan manajemen package.

Keunggulan utama ROS adalah arsitekturnya yang bersifat modular dan terdistribusi (peer-to-peer). Sistem ROS terdiri dari sejumlah proses independen yang disebut Nodes, yang saling berkomunikasi melalui kerangka kerja komputasi graf (Computation Graph) (Joseph, 2015). Arsitektur ini memungkinkan pengembangan sistem yang robust, di mana kegagalan pada satu node (misalnya node kamera mati) tidak serta merta mematikan seluruh sistem robot.



Gambar 3 Robot Operating System

1.5.4.1 Node dan ROS Master

Elemen komputasi dasar dalam ROS adalah Node. Node adalah sebuah proses eksekusi (executable) yang melakukan komputasi tertentu. Dalam konteks Waiter-Bot pada penelitian ini, sistem terdiri dari puluhan node yang berjalan secara paralel, contohnya: .

1. Node untuk mengelola driver sensor Kinect (freenect_node).
2. Node untuk algoritma pemetaan (slam_gmapping).
3. Node untuk antarmuka mikrokontroler (rosserial_python).
4. Node untuk logika navigasi (move_base).

Agar node-node tersebut dapat saling menemukan dan berkomunikasi, diperlukan sebuah entitas pusat yang disebut ROS Master (dijalankan dengan perintah `roscore`). ROS Master berfungsi sebagai Name Service. Ketika sebuah node baru aktif, ia akan mendaftarkan nama dan jenis informasi yang dimilikinya ke ROS Master. Tanpa ROS Master yang berjalan, node tidak akan bisa saling mendeteksi, dan sistem robot tidak dapat beroperasi.

1.5.4.2 Komunikasi Topics dan Messages

Komunikasi data antar-node dalam ROS dilakukan melalui mekanisme streaming data yang disebut Topics. Topic adalah sebuah nama unik (bus) yang digunakan node untuk bertukar pesan.

Message atau pesan berupa data yang dikirimkan melalui Topic dibungkus dalam struktur data yang disebut Message (.msg). Pesan ini dapat berupa tipe data primitif (integer, float, boolean) atau struktur data yang kompleks. Contoh pesan yang krusial dalam penelitian ini adalah `geometry_msgs/Twist` digunakan untuk mengirim perintah kecepatan linear dan angular ke motor. `sensor_msgs/LaserScan` yang digunakan untuk mengirim data jarak dari sensor Kinect ke algoritma SLAM. Serta `nav_msgs/Odometry` digunakan untuk mengirim estimasi posisi robot dari encoder

Topic mengimplementasikan komunikasi many-to-many. Artinya, satu Topic dapat memiliki banyak pengirim dan banyak penerima. Sifat komunikasi Topic adalah unidirectional (satu arah) dan berkelanjutan (streaming), yang sangat cocok untuk data sensor yang memiliki frekuensi tinggi, seperti data kamera atau encoder

1.5.4.3 Publisher dan Subscriber

Mekanisme pertukaran data pada ROS didasarkan pada pola desain Publisher/Subscriber. Pola ini memisahkan pengirim data (Publisher) dari penerima data (Subscriber) demi efisiensi dan fleksibilitas sistem.

Publisher adalah node yang menghasilkan data dan mengirimkannya ke Topic tertentu. Sebagai contoh, node `rosserial` pada robot ini bertindak sebagai Publisher yang menerbitkan data odometri (`/odom`) secara terus-menerus. Publisher tidak perlu mengetahui siapa yang akan membaca datanya, ia hanya bertugas menyiarkan data ke Topic

Subscriber adalah node yang membutuhkan data dan "subscribe" ke Topic tertentu. Sebagai contoh, node `gmapping` bertindak sebagai Subscriber yang mendengarkan data dari Topic `/scan` (data laser) dan `/odom` (data posisi) untuk menyusun peta. Ketika data baru tersedia di Topic tersebut, fungsi callback pada Subscriber akan dieksekusi untuk memproses data. Secara teknis, proses komunikasi ROS sebagai berikut:

1. Publisher memberi tahu ROS Master bahwa ia akan menerbitkan data ke Topic.
2. Subscriber memberi tahu ROS Master bahwa ia ingin membaca data dari Topic.
3. ROS Master mencocokkan nama Topic tersebut dan memberikan alamat dan Port milik Publisher kepada Subscriber.
4. Setelah mendapatkan alamat, Subscriber menghubungi Publisher secara langsung.
5. Data kemudian dikirimkan langsung dari Publisher ke Subscriber

Mekanisme ini sangat efisien untuk mentransmisikan data bandwidth tinggi seperti video atau point cloud pada robot, serta meminimalkan latensi (delay) yang sangat kritikal dalam sistem kendali real-time (Kane & Kane, 2018)

1.5.5 Sistem Navigasi dan Pemetaan

Kemampuan navigasi otonom pada waiter-bot bergantung pada integrasi kompleks antara persepsi sensor, estimasi posisi, dan perencanaan jalur. Dalam arsitektur Robot Operating System (ROS), sistem ini dibagi menjadi beberapa subsistem utama yaitu Sensor Fusion, Visual Odometry, SLAM, dan Path Planning.

1.5.5.1 Sensor Fusion

Untuk mendapatkan estimasi posisi robot yang akurat, tidak cukup hanya mengandalkan satu jenis sensor. Metode Sensor Fusion digunakan untuk menggabungkan data dari berbagai sumber guna meminimalkan ketidakpastian (uncertainty). Fusi dilakukan menggunakan algoritma Extended Kalman Filter (EKF) melalui paket `robot_pose_ekf` di ROS. Rotary Encoder memberikan data odometri interoseptif berdasarkan putaran roda. Keunggulannya adalah frekuensi update yang tinggi dan presisi dalam jangka pendek. Inertial Measurement Unit (IMU) mengukur percepatan linear dan kecepatan sudut (angular velocity) menggunakan akselerometer dan giroskop. IMU sangat efektif dalam mendeteksi rotasi robot yang cepat yang mungkin tidak terdeteksi oleh encoder saat roda tergelincir. Namun, IMU memiliki kelemahan berupa bias drift yang menyebabkan kesalahan posisi membesar seiring waktu jika tidak dikoreksi.

1.5.5.2 Visual Odometry (Kinect dan RTAB-Map)

Selain odometri roda dan inersia, penelitian ini menerapkan Visual Odometry (VO) sebagai sumber estimasi posisi tambahan, khususnya untuk mengatasi situasi di mana roda mengalami selip total. VO bekerja dengan menganalisis urutan gambar dari kamera untuk mengestimasi pergerakan robot.

1.5.5.3 SLAM (Simultaneous Localization and Mapping)

SLAM adalah metode komputasi yang memungkinkan robot membangun peta lingkungan yang belum diketahui sekaligus memperkirakan posisinya di dalam peta tersebut secara bersamaan, robot butuh peta untuk mengetahui posisi, tetapi butuh posisi yang akurat untuk menggambar peta.

1.5.5.4 Sistem Navigasi (AMCL dan DWA Local Planner)

Setelah peta terbentuk, robot memerlukan sistem navigasi untuk bergerak dari titik awal ke titik tujuan (goal) secara otonom. AMCL (Adaptive Monte Carlo Localization) adalah algoritma probabilistik untuk melacak posisi robot pada peta yang sudah ada. Berbeda dengan SLAM yang membangun peta, AMCL hanya mencocokkan data sensor saat ini dengan peta statis.

Perencanaan Jalur (Path Planning) Sistem navigasi ROS (Navigation Stack) menggunakan dua lapis perencanaan yaitu Global Planner dalam menghitung jalur

terpendek dari posisi robot ke tujuan menggunakan algoritma Jalur ini bersifat statis dan mengabaikan rintangan dinamis baru. Sedangkan Local Planner Dynamic Window Approach (DWA) bertugas mengendalikan motor untuk mengikuti jalur global sambil menghindari halangan dinamis.

1.5.6 Sistem Kendali Motor dan PWM Ramp Rate

Pada robot bergerak (mobile robot), sistem kendali motor bertindak sebagai jembatan antara perintah navigasi tingkat tinggi (High-Level) yang dihasilkan oleh ROS dengan eksekusi fisik tingkat rendah (Low-Level) pada aktuator. Kualitas pergerakan robot sangat bergantung pada bagaimana sinyal kendali dikelola untuk menangani dinamika beban dan inersia.

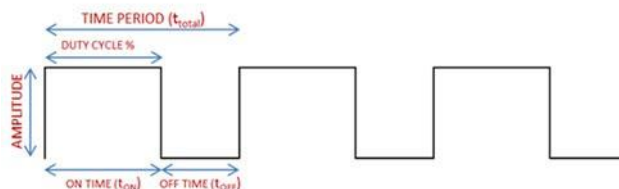


Gambar 4 Sistem Kendali

1.5.6.1 Pulse Width Modulation (PWM)

Pulse Width Modulation (PWM) atau Modulasi Lebar Pulsa adalah teknik yang digunakan untuk mengontrol daya yang disuplai ke perangkat listrik, khususnya motor DC, dengan cara memvariasikan lebar pulsa sinyal digital. Alih-alih memberikan tegangan analog yang bervariasi (yang tidak efisien dan menghasilkan panas berlebih pada transistor), PWM menswitch tegangan antara logika High (VCC) dan Low (GND) dengan frekuensi tinggi.

Parameter utama dalam PWM adalah Siklus Kerja (Duty Cycle), yang didefinisikan sebagai persentase waktu sinyal berada dalam kondisi ON terhadap total periode sinyal. Tegangan rata-rata yang diterima motor dapat dihitung dengan persamaan:



Gambar 5 Pulse Width Modulation (PWM)

Di mana:

Siklus kerja (0% - 100%).

Periode gelombang PWM ($1/f$).

Tegangan suplai input (misal: 12V).

Dengan mengatur nilai siklus kerja, mikrokontroler dapat mengatur kecepatan putar motor DC secara linear. Namun, penerapan perubahan secara instan (step change) dapat menimbulkan masalah mekanis yang serius.

1.5.6.2 PWM Ramp Rate

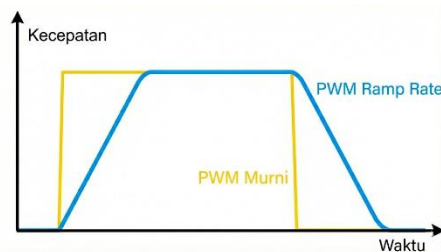
Dalam sistem kendali, Ramp Rate didefinisikan sebagai laju perubahan maksimum yang diizinkan dari suatu variabel keluaran terhadap waktu. Dalam konteks pengendalian motor robot (motor control), ramp rate secara spesifik merujuk pada batasan akselerasi (percepatan) dan deselerasi (perlambatan)

Secara sederhana, ramp rate adalah mekanisme yang mencegah motor berubah dari kecepatan 0 ke kecepatan maksimum secara instan (step change). Sebaliknya, motor dipaksa untuk menaikkan kecepatannya secara bertahap mengikuti kemiringan (slope) tertentu hingga mencapai target

Jika kita memplot grafik antara Kecepatan (v) terhadap Waktu (t), ramp rate merepresentasikan kemiringan (gradient) dari grafik tersebut. Tanpa Ramp (Step Input) grafik berbentuk kotak tegak lurus. Perubahan kecepatan terjadi dalam waktu mendekati nol, yang secara teoritis membutuhkan gaya tak hingga. Dengan Ramp (Trapezoidal Profile) grafik berbentuk trapesium. Sisi miring naik adalah akselerasi, bagian datar adalah kecepatan konstan, dan sisi miring turun adalah deselerasi. Secara matematis, ramp rate (R) dirumuskan sebagai:

Δv : Perubahan kecepatan (satuan unit kecepatan atau RPM).

Δt : Waktu yang dibutuhkan untuk perubahan tersebut



Gambar 6 PWM Ramp Rate

1.5.7 Interaksi Manusia dan Komputer (Human Machine Interface)

Dalam konteks robotika, antarmuka berfungsi untuk menyederhanakan kompleksitas sistem ROS yang berbasis teks (command line interface) menjadi bentuk visual yang intuitif (Graphical User Interface). Yoga Prihastomo dan Winanti (2024) menyatakan bahwa tren desain industri saat ini menuntut adanya visualisasi data real-time yang memungkinkan operator mengambil keputusan cepat tanpa harus memahami kode pemrograman yang rumit.



Gambar 7 Human Machine Interface

1.5.7.1 Kivy Framework

Untuk membangun antarmuka pada penelitian ini, digunakan kerangka kerja (framework) Kivy. Kivy adalah pustaka open-source berbasis Python yang dirancang untuk pengembangan aplikasi multitouch dan lintas platform (cross-platform). Kivy sangat efektif untuk menjembatani komunikasi antara pengguna awam dengan sistem robot yang kompleks karena fleksibilitasnya dalam menangani event loop secara asinkron

1.5.7.2 Front-End dan Back-End pada Sistem Robot

Dalam rekayasa perangkat lunak aplikasi robot ini, diterapkan pola arsitektur yang memisahkan antara lapisan presentasi (Front-End) dan lapisan logika bisnis (Back-End). Pemisahan ini krusial untuk menjaga agar antarmuka tidak mengalami freeze (macet) saat robot sedang melakukan komputasi berat.

Front-End adalah bagian dari perangkat lunak yang berinteraksi langsung dengan pengguna. Front-End bekerja dalam sebuah Main Loop yang terus-menerus memperbarui tampilan layar (misalnya 60 frame per detik). Front-End tidak boleh melakukan pemrosesan data yang berat, tugasnya hanya menerima perintah operator dan mengirimkannya ke Back-End.

Back-End adalah "otak" yang bekerja di balik layar untuk memproses perintah dan mengendalikan robot, Mengelola siklus hidup proses ROS (roscore, nodes), memilih file peluncuran (launch file) yang sesuai

BAB II

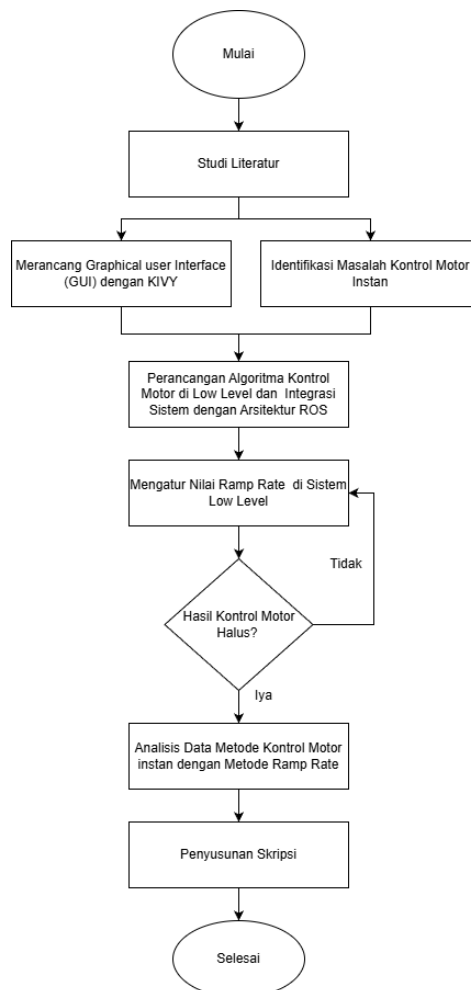
METODE PENELITIAN

2.1 Perancangan Umum

Penelitian ini dimulai pada bulan Mei 2025 hingga Januari 2026 dilaksanakan di Laboratorium Sosial Kognitif dan Robotika, Departemen Teknik Elektro, Fakultas Teknik, Universitas Hasanuddin, Gowa khususnya Gedung Centre of Technology (COT) Lantai 1. Dalam meningkatkan kualitas gerak motor digunakan metode ramp-rate dalam mengatur nilai PWM yang dikirimkan di motor untuk menghasilkan pergerakan yang lebih halus dan aman, sedangkan dari segi tampilan antarmuka digunakan framework kivy sebagai GUI dalam memudahkan pengguna sehingga tidak memerlukan pengetikan secara hardware dalam menggunakan waiter-bot.

2.2 Tahapan Penelitian

Penelitian ini dilakukan dalam beberapa tahap. Adapun diagram alir dari tahapan penelitian ini secara umum dapat dilihat pada gambar berikut:



Gambar 8 Diagram Alir Penelitian

2.3 Perancangan Pada Perangkat Lunak

Pada rancangan waiter-bot, dibagi menjadi 2 level yaitu : high-level dan low-level yang saling terhubung satu sama lain dalam mengendalikan robot khususnya gerak motor dan antarmuka pengguna

2.3.1 Perancangan di High-Level

Perancangan di High-Level Gerak Motor mencakup pengendalian Joystick, Kinect, PC (komputer), ROS. Mode controller (joystick) aktif dengan menjalankan node controller.launch, node ini terhubung dengan x360_joy_node sebagai kontrol dari joystick.

```
<node name="x360_joy_node" pkg="my_robot_pkg" type="x360_joy_node.py"
output="screen"/>
```

Node ini digunakan sebagai parameter untuk menentukan kecepatan joystick dengan mengatur twist, selain itu melakukan publish data cmd_vel yang nantinya akan terbaca di low level melalui roserial sebagai pergerakan.

```
def joy_callback(data):
    twist = Twist()
    if data.axes[7] == 1.0 :
        twist.linear.x = 0.3
    elif data.axes[7] == -1.0 :
        twist.linear.x = -0.3
    elif data.axes[6] == 1.0 :
        twist.angular.z = 0.3
    elif data.axes[6] == -1.0 :
        twist.angular.z = -0.3
    else :
        twist.linear.x = data.axes[1] * 1
        twist.angular.z = data.axes[0] * 1
    pub.publish(twist)
pub = rospy.Publisher('cmd_vel', Twist, queue_size=10)
rospy.Subscriber('joy', Joy, joy_callback)
```

Mapping mode aktif dengan menjalankan node mapping.launch, pada mode ini, mapping terdiri dari beberapa node seperti freenect.launch untuk pembacaan dari sensor kinect, RViz, dan SLAM Gmapping.

```
<include file="$(find freenect_launch)/launch/freenect.launch">
```

Dengan rviz, menu untuk pemetaan, model robot, sensor, dan node lainnya akan ditampilkan dan dengan menggunakan metode SLAM memungkinkan pemetaan secara real-time berlangsung

```
<node name="rviz" pkg="rviz" type="rviz" args="-d $(find
autonomus_mobile_robot)/config/amcl_nav.rviz"/>

<node pkg="gmapping" type="slam_gmapping" name="turtlebot3_slam_gmapping"
output="screen">
```

Untuk hasil dari pemetaan, disimpan di server dengan format .yaml dan .pgm

```
roslaunch map_server map_saver -f my_map
```

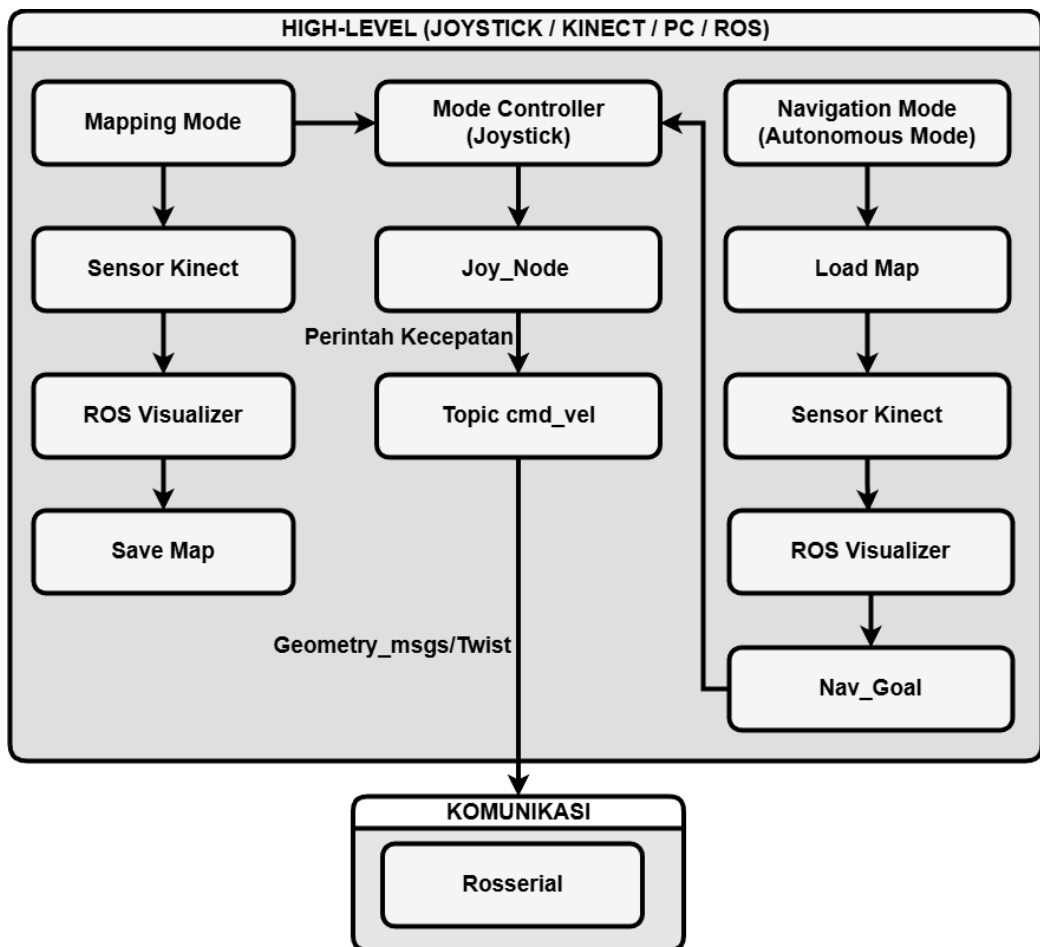
Untuk Navigation Mode (Autonomous), memiliki setting yang hampir sama dengan mode mapping, yang membedakan ialah pemilihan peta dilakukan di awal untuk navigasi

```
<node name="map_server" pkg="map_server" type="map_server" args="$(find autonomus_mobile_robot)/maps/your_map_name.yaml" />
```

Pembacaan dari sensor kinect dengan node freenect. Pada RViz ditampilkan posisi robot dan peta yang digunakan dan dengan menggunakan fitur 2d_nav_goals penentuan tujuan navigasi dari robot di aktifkan, digunakan metode navigation_stack untuk penentuan global_planner dan local_planner pada node move_base.

```
<include file="$(find autonomus_mobile_robot)/launch/move_base.launch">
```

Robot mampu melakukan navigasi jika mode controller (joystick) aktif, hal ini disebabkan perintah 2d_nav_goals dari ROS hanya melakukan publish kecepatan dari cmd_vel yang tidak terhubung dengan low-level sehingga motor belum menerima perintah pergerakan. Dengan mengaktifkan mode controller (joystick), nilai cmd_vel navigasi dari ROS diterima di low-level melalui rosserial sebagai perintah motor bergerak sesuai dengan nilai cmd_vel.



Gambar 9 Perancangan di High-Level Gerak Motor

Perancangan di High-Level Gerak Motor mencakup pengendalian Joystick, Kinect, PC (komputer), ROS yang saling terhubung.

2.3.1.1 Mode Controller Manual (Joystick)

Pada mode ini, pengendali robot dilakukan secara manual dengan menggunakan joystick sebagai input, pergerakan analog joystick dibaca pada `joy_node`, node ini akan mengirimkan nilai tersebut ke topik `cmd_vel`. Pesan inilah yang nantinya akan terbaca sebagai gerakan dan terhubung dengan proses di low-level khususnya di mikrokontroler yang terhubung dengan komunikasi serial (rosterial).

2.3.1.2 Mapping Mode

Mapping mode memungkinkan pengguna untuk membuat peta berdasarkan ruang atau lokasi yang ditentukan, pemetaan dilakukan secara manual dengan menggerakkan robot menggunakan joystick, selain itu dibutuhkan sensor kinect untuk melakukan pendeteksian lingkungan sekitar, dan dengan menggunakan metode SLAM, pembacaan dan penyimpanan peta secara real time di tampilkan di ROS Visualizer (RVIZ), Perlu diketahui titik awal ketika mapping dimulai akan menjadi posisi awal dari robot jika menggunakan mode navigasi dan posisi ini tidak dapat dipindahkan sehingga pengguna harus tahu posisi awal dari robot. Setelah pengguna merasa peta sudah sesuai, selanjutnya menyimpan hasil peta secara keseluruhan di server.

2.3.1.3 Navigation Mode

Mode navigasi, robot melakukan pergerakan secara otomatis. Di mode ini, robot akan memuat peta yang telah dibuat di mapping mode sebelumnya, dari hasil map inipun robot harus ditempatkan di titik awal mapping. Node Controller dan Sensor Kinect memiliki peranan yang sangat penting dalam pergerakan, menentukan jalur navigasi dan pembacaan objek seperti obstacle. RVIZ akan menampilkan posisi robot dan peta yang digunakan, dan dengan tampilan ini pengguna dapat menentukan goal atau tujuan navigasi dengan menggunakan fitur 2d nav_goals.

2.3.2 Perancangan di Low-Level

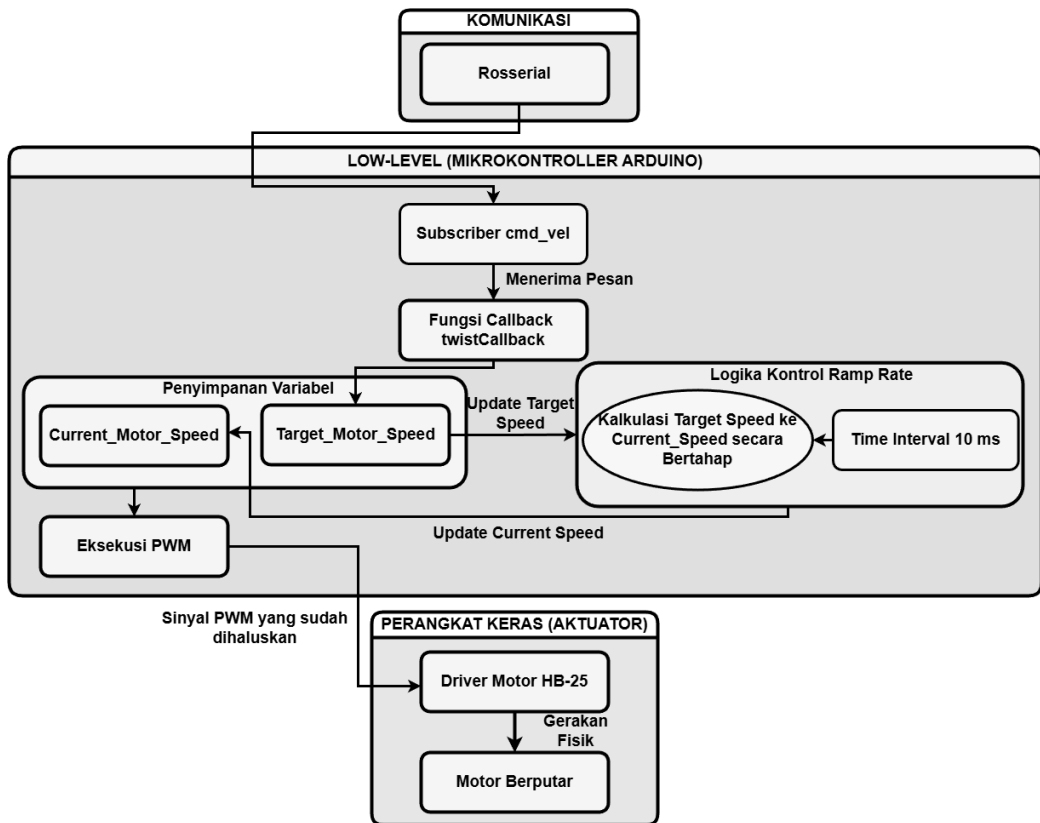
Di low-level, pesan yang dikirimkan ROS diterima dengan menggunakan komunikasi serial (rosterial), mikrokontroler (arduino) yang akan mengeksekusi perintah yang diberikan seperti perintah gerak, pembacaan sensor imu, dan encoder.

Rancangan low-level gerak motor terhubung dengan high-level (ROS) melalui komunikasi serial (rosterial). Topik `cmd_vel` yang ada pada ROS disubscribe di low-level sehingga mikrokontroler dapat menerima pesan seperti perintah gerak, pesan tersebut secara otomatis memicu `twistCallback` yang berfungsi untuk memperbarui tujuan atau target kecepatan motor (`target_motor_speed`) dan disimpan di variabel global selama perintah gerak diberikan. Logika kontrol ramp rate melakukan kalkulasi dalam mengatur `target_motor_speed` ke nilai `current_motor_speed` secara bertahap tiap interval 10ms. Hasil kalkulasi secara bertahap ini disimpan di global variabel (`current_motor_speed`) dan diubah menjadi nilai PWM, nilai inilah yang nantinya diterima driver (HB-25) dalam menggerakkan aktuator (motor).

```
ros::Subscriber<geometry_msgs::Twist> sub_twist("cmd_vel", &twistCallback);
```

Kontrol Ramp Rate diatur untuk melakukan Update kecepatan setiap 10 milidetik, dan tiap interval tersebut dilakukan kalkulasi nilai target ke nilai sekarang dengan setiap step sehingga nilai PWM yang masuk ke motor meningkat atau menurun secara perlahan mengikuti nilai target.

```
const int MOTOR_UPDATE_INTERVAL = 10;
const int SPEED_STEP = 2;
```



Gambar 10 Perancangan di Low-Level Gerak Motor

Implementasi algoritma penghalusan gerak pada penelitian ini mengadopsi prinsip Digital Ramp Generator dalam domain waktu diskrit, generator ini bekerja dengan mengakumulasi nilai Langkah (Δstep) untuk membentuk lintasan referensi kecepatan. Metode ini diformulasikan.

$$\text{ramp}_{out}(n) = \text{ramp}_{out}(n-1) + 2, \quad \text{jika } \text{ramp}_{out} < \text{target} \quad 7$$

$$\text{dan } (\text{ramp}_{out} + 2) < \text{ramp}_{max}$$

$$\text{ramp}_{out}(n) = \text{ramp}_{out}(n-1) - 2, \quad \text{jika } \text{ramp}_{out} > \text{target} \quad 8$$

$$\text{dan } (\text{ramp}_{out} - 2) < \text{ramp}_{min}$$

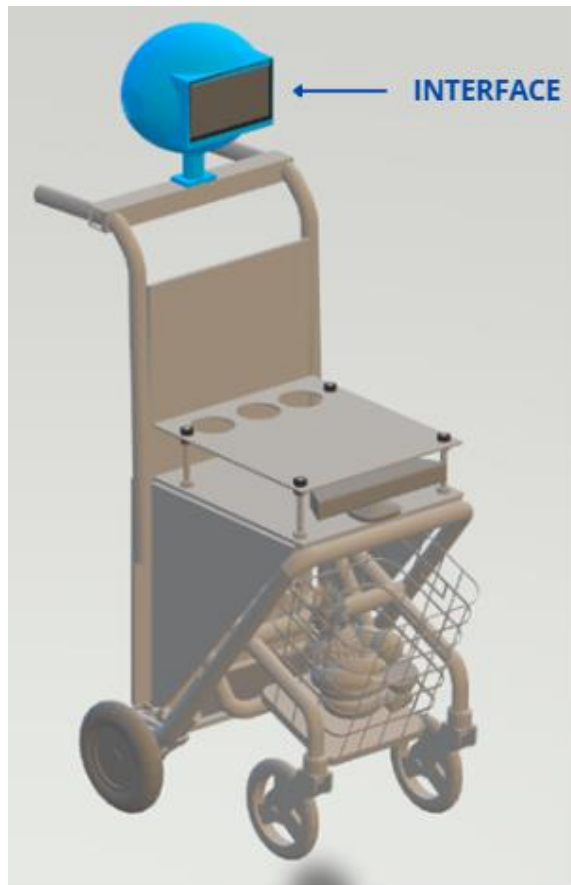
Persamaan di atas menunjukkan bahwa keluaran (ramp_{out}) akan mengejar nilai target dengan kenaikan atau penurunan sebesar 2 unit (atau speed_step) pada setiap iterasi, hingga mencapai batas saturasi (ramp_{min}) atau (ramp_{max}) Metode ini secara efektif membatasi lonjakan akselerasi (dv/dt), sehingga menghasilkan profil gerak yang linear dan meminimalisir sentakan (jerk) pada aktuator robot

2.4 Perancangan Perangkat Keras

2.4.1 Desain Mekanik Robot

Struktur dasar robot dimodifikasi untuk mendukung mobilitas di lingkungan indoor maupun outdoor. Menurut Muh Anshar (2025) ada lima komponen yang membangun robot: perwujudan, persepsi, aksi, otak dan kekuatan, dan otonomi. Namun dalam hal

ini, waiter-bot masih belum otonom sepenuhnya, robot masih dioperasikan dengan teleoperated dikendalikan Sebagian oleh manusia



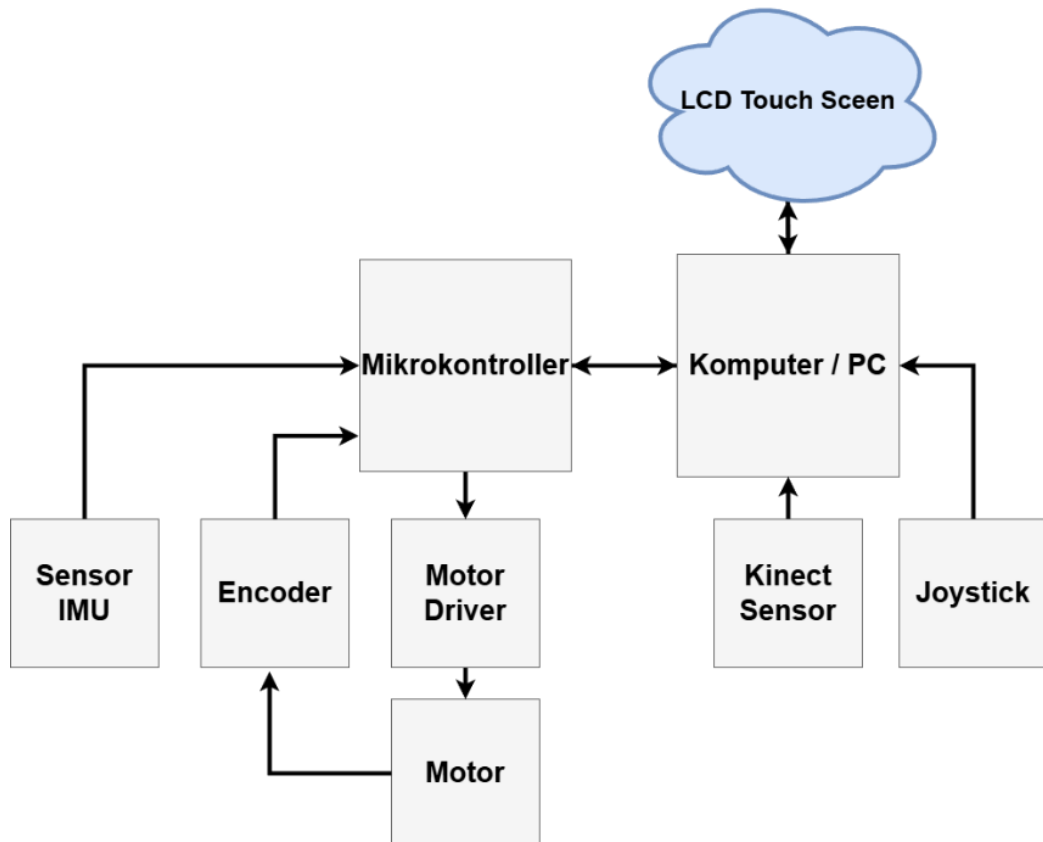
Gambar 11 Desain Waiter-Bot

Perwujudan, dapat diartikan sebagai tubuh, sebagai syarat awal menjadi robot, memungkinkan robot melakukan berbagai hal seperti berpergian suatu tempat dan menyelesaikan pekerjaannya.

Persepsi, berfungsi sebagai penginderaan robot untuk mengenali lingkungan, komponen ini terdiri dari sensor kinect dalam mendeteksi rintangan dan fitur visual untuk proses SLAM dan sensor IMU untuk memberikan data orientasi dan percepatan linear untuk mengoreksi kesalahan orientasi (angular drift) yang mungkin terjadi akibat guncangan mekanis saat robot bergerak.

Aksi, sebagai Penggerak (Aktuator dan Lokomosi) adalah mekanisme fisik yang memungkinkan robot untuk berpindah tempat dari satu lokasi ke lokasi lain dalam lingkungannya, komponen yang digunakan adalah encoder untuk memberikan umpan balik (feedback) dan dua motor DC dengan konfigurasi Differential Drive dengan driver motor. Konfigurasi ini terdiri dari dua roda utama yang terletak pada satu sumbu yang sama dan digerakkan secara independen. Mekanisme ini memungkinkan robot untuk melakukan tiga jenis gerakan dasar dengan memanipulasi kecepatan roda kiri dan roda kanan, seperti gerak maju/mundur, gerak berbelok, dan gerak putar arah ditempat.

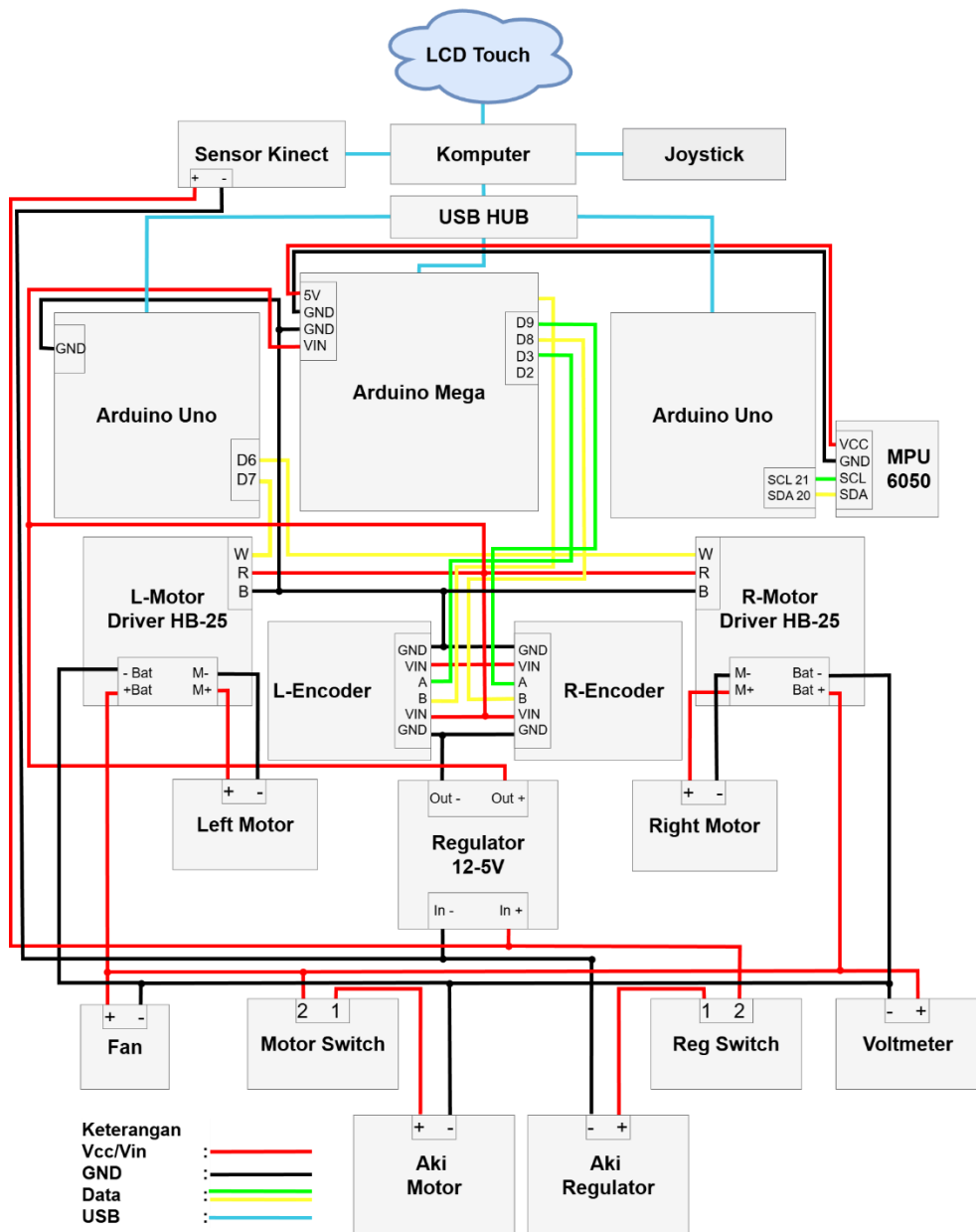
Otak dan kekuatan, robot membutuhkan otak agar dapat berfungsi dengan baik dalam hal ini komputasi, selain itu, pmbagin daya yang tepat sangat dibutuhkan agar robot bekerja dengan baik. Pada waiter-bot dibagi menjadi 2 yaitu high level controller (PC) sebagai otak utama yang menjalankan ROS, dan low level controller sebagai pengendali perangkat keras.



Gambar 12 Diagram Interkoneksi

2.4.2 Desain Elektronik

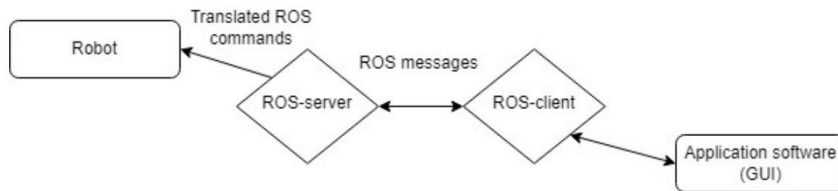
Seluruh sistem ditenagai oleh dua buah aki 12V. Satu aki digunakan untuk menyuplai daya besar ke motor DC melalui driver, sedangkan aki lainnya digunakan untuk menyuplai daya stabil ke mikrokontroller dan sensor melalui regulator tegangan. Pemisahan jalur daya ini bertujuan untuk mencegah terjadinya gangguan kinerja sensor sensitive dan mikrokontroller yang sangat penting untuk menjaga persepsi dari robot. Dilakukan penambahan interface seperti LCD menjadi bagian interaksi robot dengan operator.



Gambar 13 Diagram Skematik

2.4.3 Perancangan Graphical User Interface (GUI)

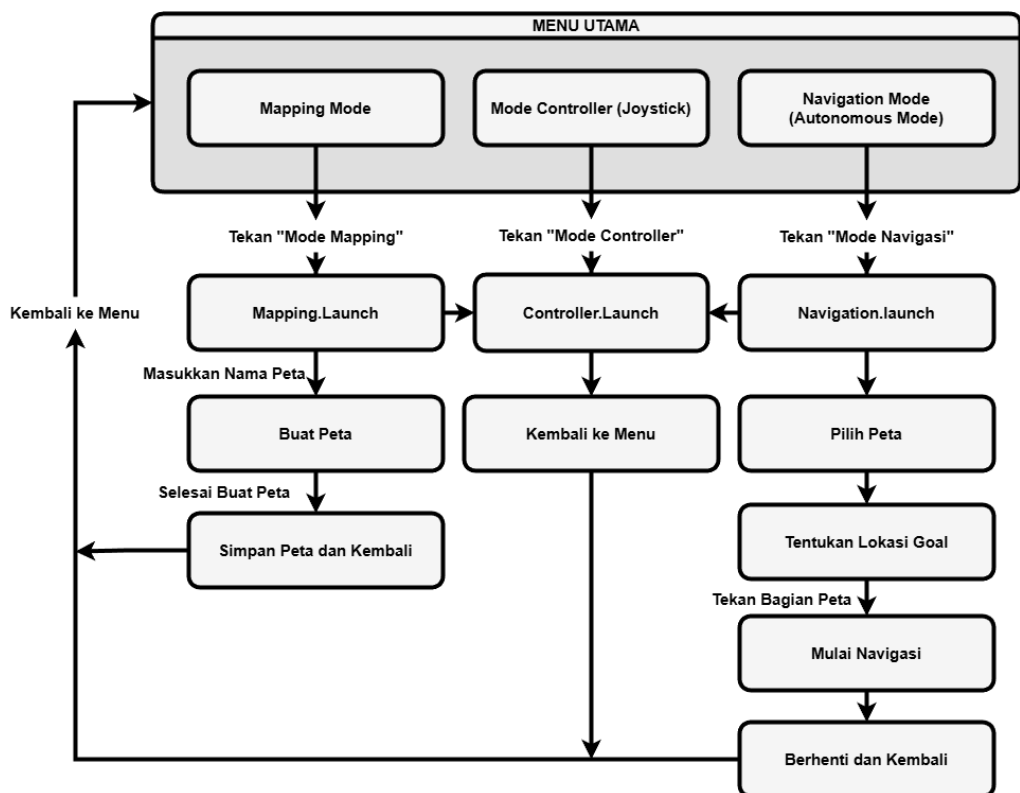
GUI merupakan tampilan antarmuka yang memungkinkan pengguna mampu berinteraksi dengan sistem komputer dengan berbagai elemen grafis seperti tombol, jendela, menu, gambar, dan lain-lain. Selain itu dengan merancang GUI, efisiensi penggunaan suatu perangkat lunak atau sistem memiliki dampak yang signifikan pada pengalaman pengguna



Gambar 14 Arsitektur Hubungan GUI ke robot

Kivy adalah sebuah framework open-source dengan menggunakan bahasa pemrograman python, menjadi salah satu solusi untuk membangun aplikasi ataupun tampilan antarmuka multi-sentuh diberbagai platform. Perangkat ini memungkinkan pengembangan yang sederhana dan fleksibel dalam membuat aplikasi yang kuat dan intuitif

Ros-server melakukan publish sebuah topik yang nantinya akan disubscribe oleh ros-client untuk menerima pesan. Di GUI, dengan menekan elemen yang diberikan perintah seperti menjalankan robot akan diterima sebagai pesan dari client dan ros-server akan menghubungkannya sebagai pergerakan di robot



Gambar 15 Perancangan GUI Waiter-Bot

Pengembangan antarmuka pengguna (GUI) pada sistem Waiter-Bot dirancang menggunakan pola arsitektur yang memisahkan antara logika tampilan (presentation

layer) dan logika pemrosesan data (application logic). Pendekatan ini bertujuan untuk meningkatkan modularitas dan responsivitas sistem. Berdasarkan kerangka kerja Kivy yang digunakan, struktur perangkat lunak dibagi menjadi dua komponen utama yaitu Front-End dan Back-End.

Sisi front-end bertanggung jawab atas visualisasi data dan interaksi langsung dengan pengguna. Menangkap event dari pengguna (seperti sentuhan pada layar atau penekanan tombol joystick virtual) dan menyajikannya dalam bentuk visual yang intuitif (Ben Shneiderman, 2016). Sisi Back-End diimplementasikan menggunakan bahasa pemrograman Python yang terintegrasi dengan pustaka rospy (ROS Python). Bagian ini berfungsi sebagai jembatan antara input pengguna dari Front-End dan sistem kendali robot. Back-End menangani pemrosesan data di balik layar.

2.4.4 Skenario Pengujian

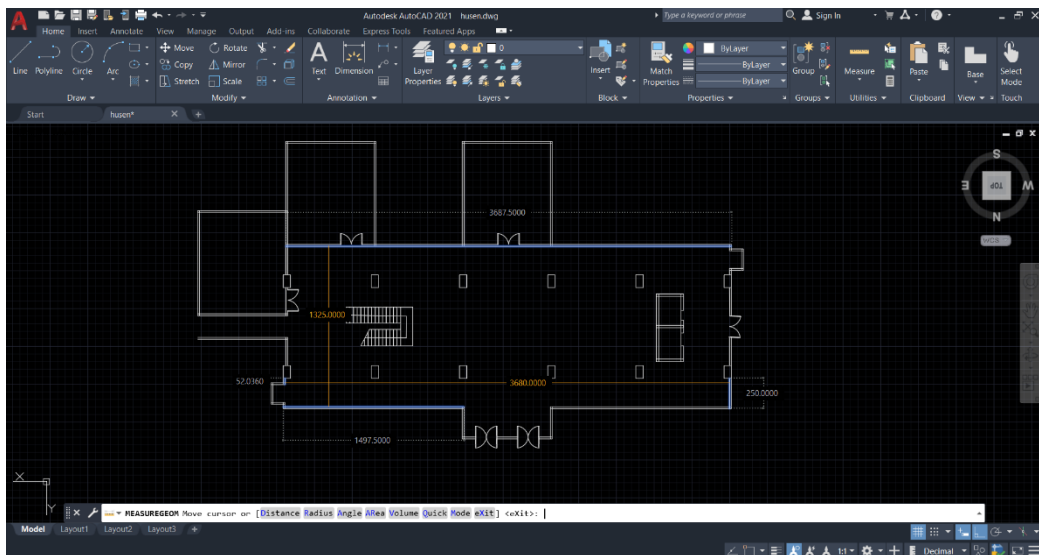
2.4.4.1 Uji Kecepatan Linear dan Angular

Uji kecepatan linear dan angular dilakukan dengan membandingkan kontrol motor konvensional yang digunakan pada penelitian sebelumnya dan kontrol motor dengan PWM Ramp Rate. Dari ROS perintah `msg.linear.x` untuk kecepatan maju (m/s) dan `msg.angular.z` untuk kecepatan putar yang diinginkan (rad/s), topik dari `/right_ticks` dan `/left_ticks` dari data encoder digunakan sebagai pembacaan putaran motor untuk melihat kecepatan pergerakan motor dengan parameter kecepatan yang digunakan 0.22, 0.5, 1, 1.5 m/s yang diujikan pada lintasan sejauh 2 meter dan untuk kecepatan angular 0.5 dan 1 m/s dalam melakukan 1 putaran penuh

$$Actual\ Linear = \frac{vR + vL}{2} \quad 9$$

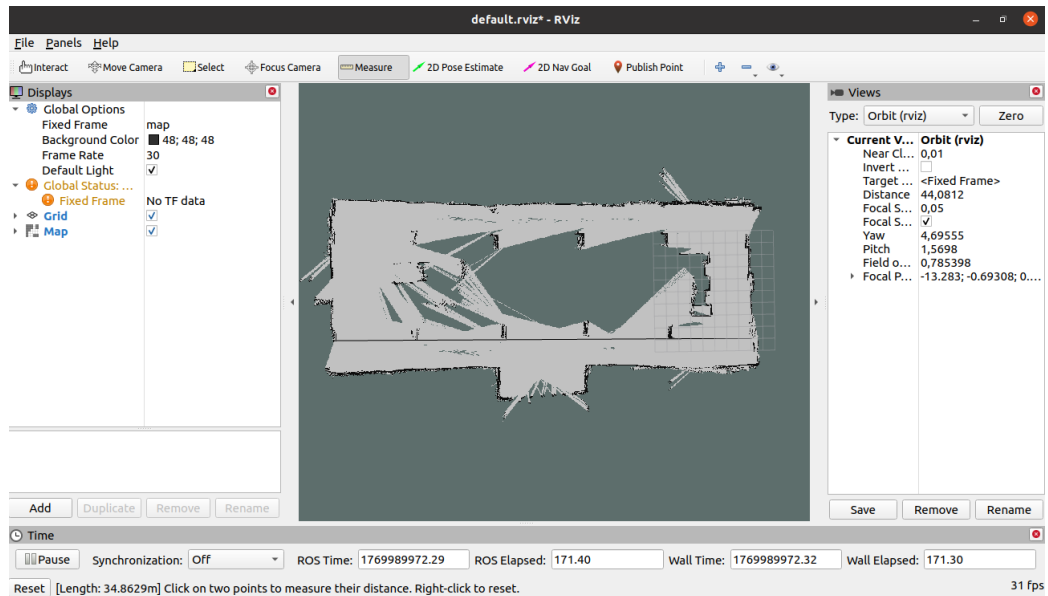
2.4.4.2 Uji Mapping

Uji mapping dilakukan dengan melakukan pengukuran ruangan pengujian secara langsung secara keseluruhan, hasil pengukuran kemudian di desainkan dengan perangkat Autodesk autocad untuk memudahkan pengukuran dan perbandingan untuk hasil mapping.



Gambar 16 Visualisasi Peta Aktual

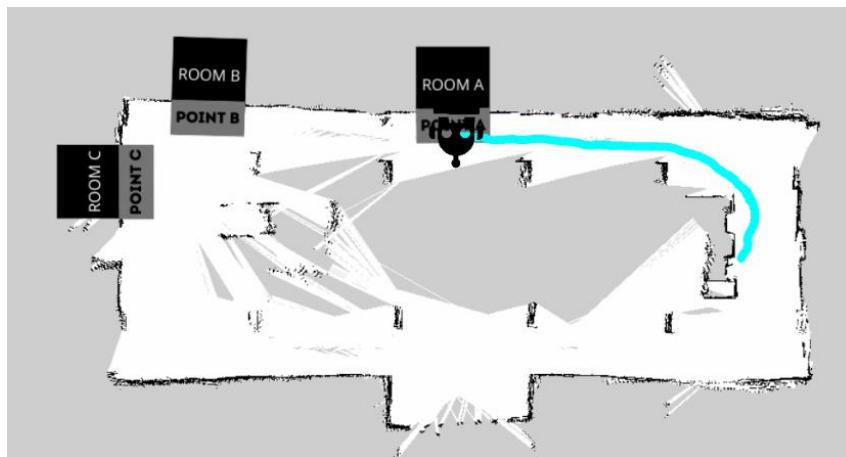
ROS telah dilengkapi dengan fitur Ros Visualizer (Rviz) sebagai dashboard yang menampilkan hasil pemetaan, selain itu terdapat fitur measure yang dapat digunakan untuk menghitung jarak dari satu titik ke titik lainnya. Fitur inilah yang digunakan untuk membandingkan hasil peta yang ada di dunia nyata dan yang dihasilkan waiter-bot.



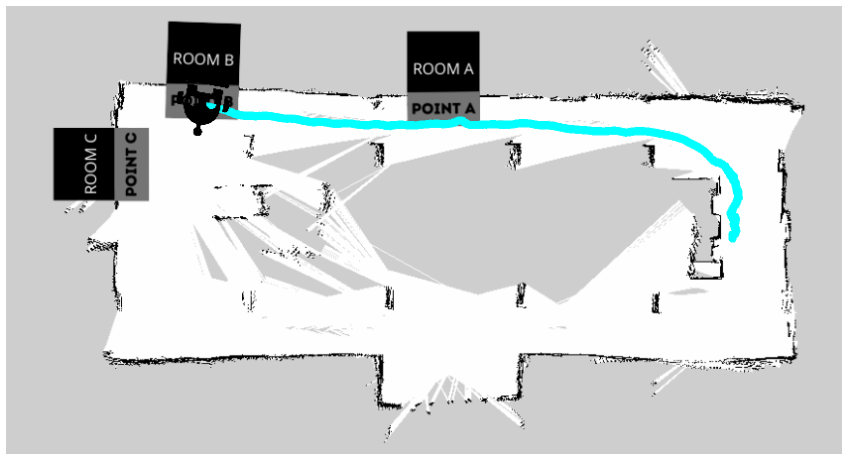
Gambar 17 Pengukuran Peta di RViz

2.4.4.3 Uji Navigasi

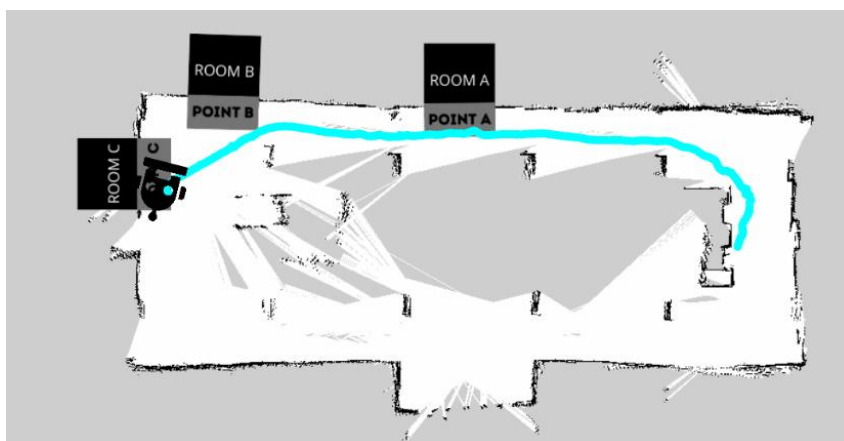
Uji navigasi dilakukan dengan menghitung pergerakan motor dengan parameter kecepatan yang digunakan 0.22, 0.5, 1 m/s. Parameter input navigasi berasal dari ROS dengan mengirimkan nilai dari `cmd_vel` yang berasal dari DWA Local Planner yang berupa `msg.linear.x` untuk kecepatan maju (m/s) dan `msg.angular.z` untuk kecepatan putar yang diinginkan (rad/s), topik dari `/right_ticks` dan `/left_ticks` dari data encoder untuk pembacaan pergerakan aktual motor dan topik `odom` untuk pembacaan koordinat estimasi Lokasi dan jalur pergerakan robot.



Gambar 18 Jalur Navigasi Point A



Gambar 19 Jalur Navigasi Point B



Gambar 20 Jalur Navigasi Point C