T.C.
BURSA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ROS TABANLI MOBİL ROBOTUN YAPIMI VE YÖNETİMİ

YÜKSEK LİSANS TEZİ

Melih KORKMAZ

Akıllı Sistemler Mühendisliği Anabilim Dalı

NİSAN 2023

T.C.
BURSA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ROS TABANLI MOBİL ROBOTUN YAPIMI VE YÖNETİMİ

YÜKSEK LİSANS TEZİ

Melih KORKMAZ
(19324814025)

Akıllı Sistemler Mühendisliği Anabilim Dalı

Tez Danışmanı: Doç. Dr. İzzet Fatih ŞENTÜRK

NİSAN 2023

BTÜ, Lisansüstü Eğitim Enstitüsü'nün 19324814025 numaralı Yüksek Lisans Öğrencisi Melih KORKMAZ, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "ROS TABANLI MOBİL ROBOTUN YAPIMI VE YÖNETİMİ" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Doç. Dr. İzzet Fatih ŞENTÜRK** .....
Bursa Teknik Üniversitesi

Jüri Üyeleri : **Dr. Öğr. Üyesi Nurettin Gökhan ADAR** .....
Bursa Teknik Üniversitesi

Dr. Öğr. Üyesi Mete YAĞANOĞLU .....
Atatürk Üniversitesi

Teslim Tarihi : 8 Mayıs 2023
Savunma Tarihi : 11 Nisan 2023

20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, Bursa Teknik Üniversitesi'nin abonesi olduğu intihal yazılım programı kullanılarak Lisansüstü Eğitim Enstitüsü'nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

İNTİHAL BEYANI

Bu tezde görsel, işitsel ve yazılı biçimde sunulan tüm bilgi ve sonuçların akademik ve etik kurallara uyularak tarafımdan elde edildiğini, tez içinde yer alan ancak bu çalışmaya özgü olmayan tüm sonuç ve bilgileri tezde kaynak göstererek belgelediğimi, aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim.

Öğrencinin Adı Soyadı: Melih KORKMAZ

İmzası :

Aileme,

ÖNSÖZ

Bu yüksek lisans çalışmasında gün geçtikçe robotik sektöründe etkisini arttıran ROS (Robot Operating System) kullanılarak, mobil robot (Autonomous Mobile Robot - AMR) simülasyon ortamında ve gerçek ortamda yapılmıştır. Geliştirilen yöntemler ile robota sanal kumanda ile harita çıkartıldıktan sonra, robot arayüz üzerinden yönetilerek otonom bir şekilde görevlere yollanmıştır. Sistem akıllı/merkezi yönetim sistemlerine entegre olabilir hale getirilmiştir.

Öncelikle tez konusunu seçerken isteklerimi göz önünde bulunduran bana tez çalışmam sırasında kıymetli bilgi, birikim ve tecrübeleri ile yol gösterici ve destek olan değerli danışman hocam Doç. Dr. İzzet Fatih ŞENTÜRK'e teşekkürlerimi sunarım.

Çalışmalarım boyunca yardımlarını hiç esirgemeyen değerli çalışma arkadaşlarıma ve dostlarıma teşekkürü bir borç bilirim. Ayrıca her zaman yanımda olan aileme sonsuz şükranlarımı sunuyorum.

Nisan 2023

Melih KORKMAZ
(Mekatronik Mühendisi)

İÇİNDEKİLER

Sayfa

KISALTMALAR

AGV : Automated Guided Vehicle
AMR : Autonomous Mobile Robot
AUV : Autonomous Underwater Vehicle
BSD : Berkeley Software Distribution
CPU : Central Processing Unit
GPU : Graphics Processing Unit
IMU : Inertial Measurement Unit
LDS : Laser Distance Sensor
LIDAR : Laser Imaging Detection and Ranging
PWM : Pulse Width Modulation
RAM : Random Access Memory
ROS : Robot Operating System
SAIL : Stanford Artificial Intelligence Laboratory
SLAM : Simultaneous Localization and Mapping
UAV : Unmanned Aerial Vehicles
URDF : Unified Robot Description Format

SEMBOLLER

$l_{normalsidedist}$: Dış güvenlik bölgesi için uzunluk

$l_{safetyoffset}$: Güvenli alanın robottan nekadar geniş olacağı

$l_{sidedist}$: Orta güvenlik bölgesi için uzunluk

r_{hor} : Robotun boyu

r_{normal} : Dış güvenlik bölgesi için genişlik

$r_{slowdown}$: Orta güvenlik bölgesi için genişlik

r_{ver} : Robotun eni

x_{lidar} : LIDAR sensörünün robot merkezinden uzaklığı

cm : Santimetre

m : Metre

ŞEKİL LİSTESİ

Sayfa

ROS TABANLI MOBİL ROBOTUN YAPIMI VE YÖNETİMİ

ÖZET

ROS (Robot Operating System, Robot İşletim Sistemi) oluşturduğu standartta robotları kontrol etmeyi sağlayan, robotu çevresel bileşenlerle haberleştiren yayıncı-dinleyici mesajlaşma modeline sahip açık kaynak kodlu bir yazılım geliştirme kitidir. Gün geçtikçe desteklediği ürün, paket ve proje sayısı artan ROS, simülasyon araçları ile dilden bağımsız ve modüler bir yapıya sahip olduğundan robot dünyasında standart konumunu almaktadır.

Bu çalışmada ROS tabanlı AMR (Autonomous Mobile Robot, Otonom Mobil Robot) tüm mekanik, elektronik ve yazılım kısımları ile beraber yapılmıştır. Simülasyon ve gerçek ortamda robot çalıştırılmış ve geliştirilen sistemler ile robot yönetilmiştir.

Simülasyon araçları olarak ROS'un sağlamış olduğu Gazebo, RViz ve RQt kullanılmıştır. Tasarımı 3 boyutlu tasarım programında yapılan robot, simülasyon ortamına aktarılarak Gazebo'da oluşturulan gerçekçi bir dünya ile simüle edilmiş, RViz üstünden robotun ve fonksiyonların çıktıları gözlemlenmiş ve RQt üzerindeki birçok araç kullanılarak robotun hassas ayarları yapılmış ve sistemdeki veriler gözlemlenmiştir.

SLAM (Simultaneous Localization and Mapping, Eşzamanlı Konumlama ve Haritalama) fonksiyonu sayesinde LIDAR (Laser Imaging Detection and Ranging, Lazerli Görüntüleme, Algılama ve Aralık Ölçümü) verisi ile birlikte robotun hiç bilmediği bir alanı haritalaması sağlanır. Sanal kumanda ile manuel sürülerek çıkartılan harita ile birlikte ROS'un sağlamış olduğu paketler kullanılarak robot doğal navigasyon sayesinde ortamda hiçbir belirteç olmadan robot konumlama işlemini yapabilmektedir. Sabit ve hareketli engelleri de dikkate alarak dinamik, güvenli ve maliyeti en düşük yol planlamasını çıkartıp hedefe ulaşması AMR'leri ön plana çıkarmaktadır. SLAM ve navigasyon fonksiyonunun parametreleri optimize edilerek robot üzerinde çalışmaya hazır hale getirilmiştir.

Robotun içerisinde geliştirilen web-socket haberleşme protokolü tabanlı arkayüz yazılımı ile robotun dış kaynaklardan yönetilmesi ve izlenmesine olanak sağlanmıştır. Bu imkan ile robot merkezi/akıllı sistemler ile haberleşip süreçlere entegre olabilir. Geliştirilen görev yönetim sistemi ile birlikte robotun otonomi kabiliyeti arttırılmış ve çevre birimler ile birlikte çalışmasına olanak sağlanmıştır. Geliştirilen dinamik konum/hedef sistemi ile birlikte robotun alacağı görevler hızlıca revize edilebilir.

İnsan robot etkileşimi hedeflenerek geliştirilen kullanıcı dostu ön arayüz, robot ile web-socket üzerinden haberleşmektedir. Harita fonksiyonunu çalıştırma ve harita kaydetme, daha önceden çıkartılmış haritalar ile navigasyon fonksiyonunu çalıştırma,

konum/hedef kaydetme ve robota görev atama ve yönetme işlemleri arayüz üzerinden yapılabilmektedir. Robotun anlık olarak birçok verisi de arayüz üzerinden takip edilebilmektedir. Ek olarak geliştirilen web tabanlı sanal kumanda ile robot manuel olarak sürülebilmektedir.

Geliştirilebilecek merkezi/akıllı yönetim sistemleri ile birlikte robotlar birbirleri ile ve binalardaki kapı, asansör sistemleri ile haberleşebilir. Endüstri 4.0 altında makineler ile haberleşip üretim süreçlerine dahil olabilir. Birden çok robot olduğunda ise filo kontrol sistemleri geliştirilerek robotların trafik kontrolü yapılabilir. Veri izleme ve raporlama sistemleri ile takip, bakım, arıza ve verimlilik analizleri yapılabilir.

Anahtar kelimeler: Robot İşletim Sistemi (ROS), Otonom Mobil Robot (AMR), Eşzamanlı Konumlama ve Haritalama (SLAM), Navigasyon, Kullanıcı Arayüzü, Görev Yönetim Sistemi

CONSTRUCTION AND MANAGEMENT OF ROS BASED MOBILE ROBOT

SUMMARY

ROS (Robot Operating System) is an open source software development kit with a publish- subscribe messaging model that enables to control robots and communicates the robot with environmental components in its created standard. ROS, which increases the number of products, packages and projects it supports day by day, takes its standard position in the robot world as it has a language-independent and modular structure with simulation tools.

In this study, ROS-based AMR (Autonomous Mobile Robot) has been performed together with all its mechanical, electronic, and software parts. The robot has been operated in the simulation and real environment, and the robot has been managed by the developed systems.

Gazebo, RViz and RQt provided by ROS have been utilized as simulation tools. The robot, the design of which was made in the 3D design program, has been transferred to the simulation environment and simulated with a realistic world created in the Gazebo, the outputs of the robot and its functions have been observed through RViz, the precise settings of the robot have been implemented using many tools at RQt, and the data in the system have been observed.

With the help of the SLAM (Simultaneous Localization and Mapping) function, the robot is enabled to map an area, it has never known, together with the LIDAR (Laser Imaging Detection and Ranging) data. By using the packages provided by ROS together with the prepared map by manual driving with a virtual joystick, the robot is able to perform robot positioning without any markers in the environment thanks to natural navigation. Dynamic, safe, and cost-effective road planning and reaching the target by taking into account fixed and moving obstacles bring AMRs to the forefront. The parameters of the SLAM and navigation function have been optimized and made ready to work on the robot.

With the backend based on the web-socket communication protocol developed inside the robot, it is possible to manage and monitor the robot from external sources. With this possibility, the robot can communicate with the central/intelligent systems and integrate them into the processes. With the developed task management system, the robot's autonomy capability has been increased and it has been enabled to work together with peripheral units. The tasks to be taken by the robot can be revised quickly with the developed dynamic position/target system. In addition, the robot can be driven manually with the web-based virtual joystick developed.

With the help of central/intelligent management systems that can be developed, robots can communicate with each other and with the doors and elevator systems in the

buildings. It can communicate with machines and participate in production processes under Industry 4.0. When there are multiple robots, traffic control of robots can be performed by developing fleet control systems. With data monitoring and reporting systems, follow-up, maintenance, fault, and efficiency analyzes can be performed.

Keywords: Robot operating system (ROS), Autonomous mobile robot (AMR), SLAM, Navigation, User interface, Task management system

1. GİRİŞ

ROS (Robot Operating System) [1], 2007'den beri gün geçtikçe robotik sektöründe etkisini arttırmaktadır. Günümüzde de robot kollardan [2], su altı araçlarına (Autonomous Underwater Vehicle AUV) [3], insansız hava araçlarına (Unmanned Aerial Vehicles UAV) [4], tarım sektöründen [5], uzay keşif araçlarına (NASA - Robonaut 2) [6], robot süpürgelerden [7], endüstri robotlarına [8] kadar robotiğin her alanında kullanılmaktadır.

Mobil robotların kullanımı, başta endüstriyel alanlar olmak üzere hizmet, sağlık, savunma vb. sektörlerde giderek daha da yaygınlaşmaktadır [9-11]. Son zamanlarda yapay zekanın gelişmesiyle mobil robotların otonom şekilde hareket edebilmesini sağlayan sistemler üzerine odaklanılmıştır [12]. Mobil otonom robotlar (Autonomous Mobile Robot - AMR) ortamda hiçbir belirteç olmadan konumunu ve engelleri algılayıp dinamik olarak hedefine gitmesini sağlayan doğal navigasyonu sayesinde dinamik, güvenli ve maliyeti düşük yol planlamasına ve otonom sürüşe sahiptir.

2020 yılında THM Uygulamalı Bilimler Üniversitesi'nde ROS ve simulasyon ortamı Gazebo, deneysel bir mobil robot olan TurtleBot3 üzerinde test edilmiştir [13].

2020 yılında Ramrao Adik Teknoloji Enstitüsü'nde Turtlebot Kullanarak Otonom Navigasyon İçin ROS Tabanlı SLAM uygulaması adlı çalışmada SLAM ve navigasyon fonksiyonları çalıştırılmıştır [14].

2022 yılında Ostrava Teknik Üniversitesi'nde tarafından yayınlanan Akıllı Fabrikada Otonom Mobil Robot Uygulaması adlı çalışmada MiR (Mobile Industrial Robots) firmasının MiR100 modeli olan robot üniversitedeki üretim hattında uygulanmıştır. SLAM ve navigasyon fonksiyonları kullanılarak robot üretim hattından ürünleri atanan görev ile otonom bir şekilde almaktadır. Ve bu işlemler arayüz üzerinden yapılmaktadır [7,15].

2017 yılında Ulusal Savunma Teknolojisi Üniversitesi'nde yapılan Ros Tabanlı Uzaktan Robot Görev İzleme ve Kontrol Sistemi adlı çalışma ile operatörlerin uzaktan robotları çalıştırması ve güncel durumlarını izlemesi hedeflenmiştir [16].

2021 yılında Rusya'da Akıllı Robotik Sistemler Laboratuvarı'nda ROS verilerini ReactJS ve ROS'un hazır rosbridge paketleri kullanılarak web tarayıcıda izleme ve görselleştirme imkanı sunulan çalışma yapılmıştır [17].

2020 yılında Karlsruhe Teknoloji Enstitüsü'ndeki Akıllı Sensör-Aktüatör-Sistem Laboratuvarı'nda ROS verilerini Linux işletim sistemi haricinde bit platformda görmek için geliştirilen "iviz" adlı mobil uygulamadan bahsedilmiştir [18].

2021 yılında Bihać Üniversitesi'nde sanal kumanda olarak ROS robotunun uzaktan kontrolü için web-socket tabanlı haberleşen web uygulaması sunulmuştur [19].

2023 yılında Uluslararası Gelişmiş Robotik Sistemler Dergisi'nde yayınlanan çalışma ile pandemiden sonra kapalı ve kalabalık ortamlarda devriye atarak ortamı ultraviyole ışınlar ile temizleyecek robot yapılmıştır. ROS'un haritalama ve navigasyon fonksiyonları kullanılarak robotun otonom bir şekilde devriye atması sağlanmıştır [20].

1.1 Tezin Amacı

Tüm mekanik, elektronik ve yazılımsal olarak kısımları içeren mobil robotun simülasyon ve gerçek ortamda yapılarak, ROS ve geliştirilen yöntemler ile birlikte robotun bilmediği bir ortamda sanal kumanda ile manuel sürülerek ortamı öğrenmesi ve orada otonom bir şekilde çalışması sağlanmıştır. Tüm bu işlemler geliştirilen arayüz üstünden kolayca yapılabilir.

Robotun tasarımı tasarım programında yapılarak ROS'a aktarılmıştır. Haritalama için google'un desteklediği cartographer [21], navigasyon için ROS'un sağladığı navigasyon (move_base) paketi [22] bir çok parametresinde hassas ayar yapılarak optimize edilmiştir. Oluşturulacak simülasyon dünyasında detaylı optimizasyon işlemleri ve testler yapılmıştır. Geliştirilen görev yönetim sistemi ile robotun otonomisi kuvvetlendirilmiş, geliştirilen web-socket haberleşme altyapısı arkayüz yazılımı (backend) ile dış kaynaklara yönetme ve izleme imkanı sağlanmıştır. Dış kaynak olarak önyüz yazılımı (frontend) geliştirilmiştir.

Açık kaynağa katkıda bulunmak için robotun geliştirilmesinde elde edilen bilgi birikim paylaşılmıştır. Geliştirilen robotun tasarım dokümanları ve kaynak kodları internet üzerinden erişilebilir [23].

1.1.1 Robotun donanım bileşenleri

Donanım bölümünde robotun mekanik ve elektronik bileşenlerinden bahsedilmiş ve kullanılan bilgisayar hakkında bilgi verilmiştir. Mekanik bölümünde, mekanik tasarım, mekanik montaj ve robotun kinematik modelinden bahsedilmiştir. Elektronik bölümde ise motor, motor sürücü, mikrodenetleyici, pil ve batarya yönetim sisteminden ve LIDAR sensöründen bahsedilmiştir. Daha sonrada robotta kullanılan bilgisayar hakkında bilgi verilmiştir.

1.1.2 Robotun yazılım bileşenleri

Yazılım bölümünde yazılım geliştirme kiti olarak neden ROS (Robot Operating System) kullanıldığından bahsedilmiş ve terminolojisi hakkında bilgi verilmiştir. İnsan robot etkileşimi için yapılan kullanıcı dostu arayüzün yapımında kullanılan React dili ve robotla haberleşmesinde kullanılan websocket haberleşme yönteminden bahsedilmiştir.

1.1.3 Yöntemler

Robotun güvenlik amaçlı hız kontrolünü yapmak için geliştirilen LIDAR bölgeleri algoritması ile birlikte verilmiştir. İnsan robot etkileşimi için yapılan arayüzden bahsedilmiştir. Robotu sanal kumanda mantığında telefondan sürmek için yapılan arayüzden de bahsedilmiştir. Geliştirilen görev yönetim sisteminden bahsedilmiştir. Ek olarak robotun haritalama modu için kullanılan SLAM (Simultaneous Localization and Mapping) ve otonom sürüş için kullanılan navigasyon modlarından bahsedilmiştir.

1.1.4 Robotun yönetimi

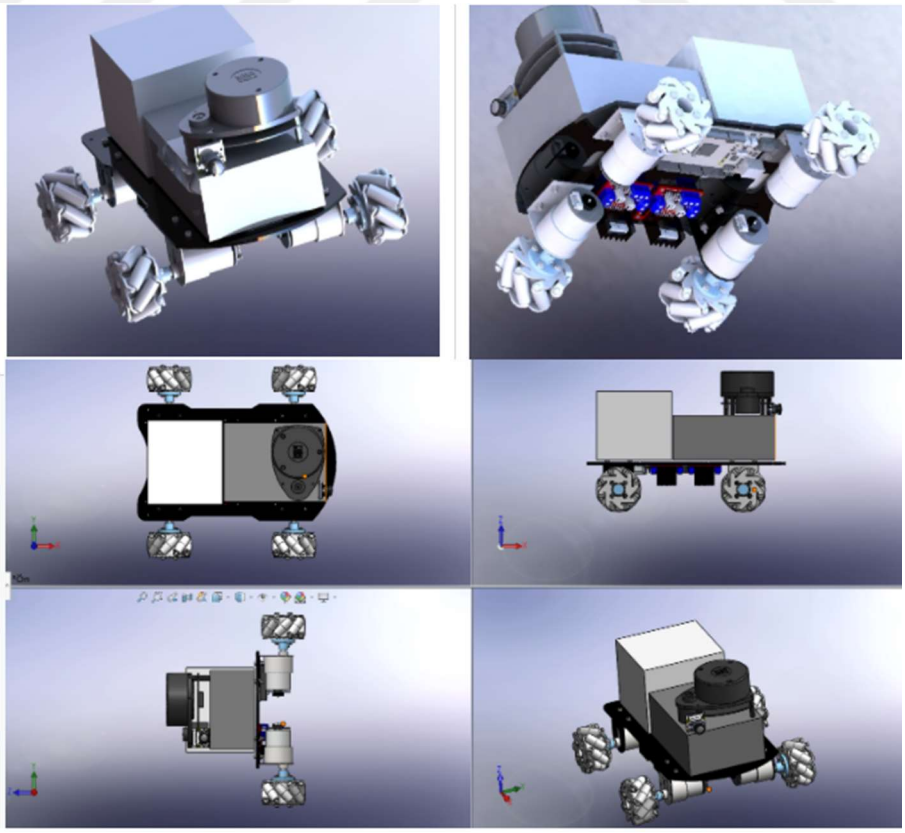
ROS'un sağlamış olduğu simülasyon ortamı Gazebo'da oluşturulan dünya, robotun sensör ve birçok verilerinin çıktıları RViz kullanılarak gösterilmiştir. Yapılan kullanıcı arayüzü kullanılarak haritalama ve hedef konum kaydetme işlemleri yapılmıştır. Görev yönetim sistemi sayesinde robota görev verilerek robot otonom modda çalıştırılmıştır.

2. DONANIM

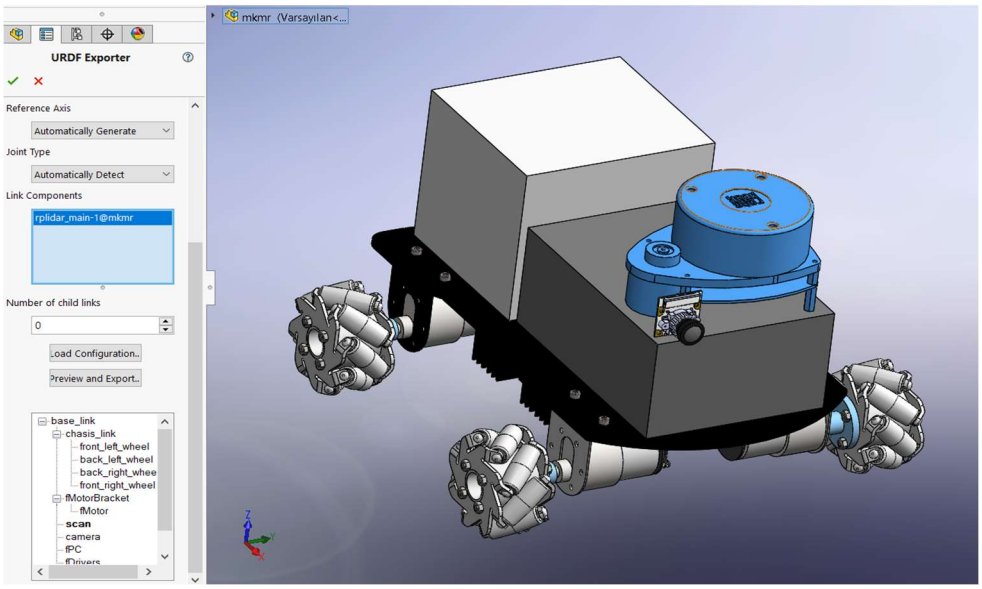
2.1 Mekanik

2.1.1 Mekanik tasarım

Robotun tasarımı (Şekil 2.1) 3 boyutlu tasarım programı olan Solidworks ile yapıldıktan sonra ROS'un sw_urdf_exporter paketi ile URDF (Unified Robot Description Format) dosyasına çevrilir (Şekil 2.2). Ayrıca robot model için 3 boyutlu dosya formatı (stl) olarak mesh dosyaları alınır. URDF dosyasında robotun tüm kinematiği mevcuttur ve ROS'ta bu transform (tf) olarak belirtilmektedir. Çevrilen URDF dosyasında robotun tasarımsal detaylarına, simülasyon ortamında kullanımına ve tasarımcının modüler bir yapıda tasarım yapmak istemesine göre gerekli düzenlemeler yapıldıktan sonra ROS ortamında kullanılabilir.

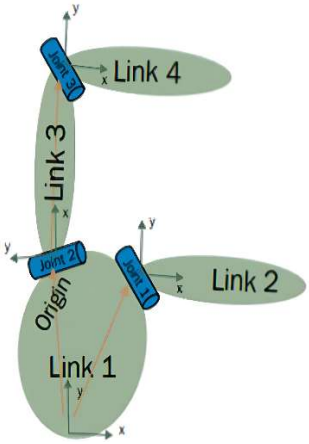


Şekil 2.1 : Robotun tasarım görselleri.



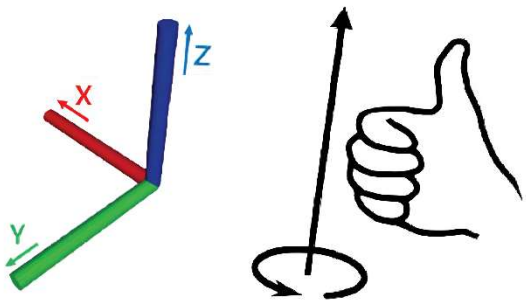
Şekil 2.2 : Robotun temel düzeyde URDF formatına çevrilmesi.

URDF dosyasındaki tf değerleri robotun tüm sabit ve hareketli eksenlerini, eksenlerin arasındaki uzaklık ve açı değerlerini, hareketli eksenlerin nasıl hareket edeceğini ve hareketi ile ilgili değerleri, hangi eksenlerin birbirine bağlı hareket edeceği gibi birçok bilgiyi içerir (Şekil 2.3).



Şekil 2.3 : Joint ve link gösterimi (tf) [24].

Tf'lerin x y z eksenleri sağ el kuralına (Şekil 2.4) göredir. Otonom sürüş esnasında hız değerleri ve sensör verileri tf değerlerine göre hesaplanacağından bu değerlerin hassas olması önem arz etmektedir [24].



Şekil 2.4 : Sağ el kuralı (tf) [24].

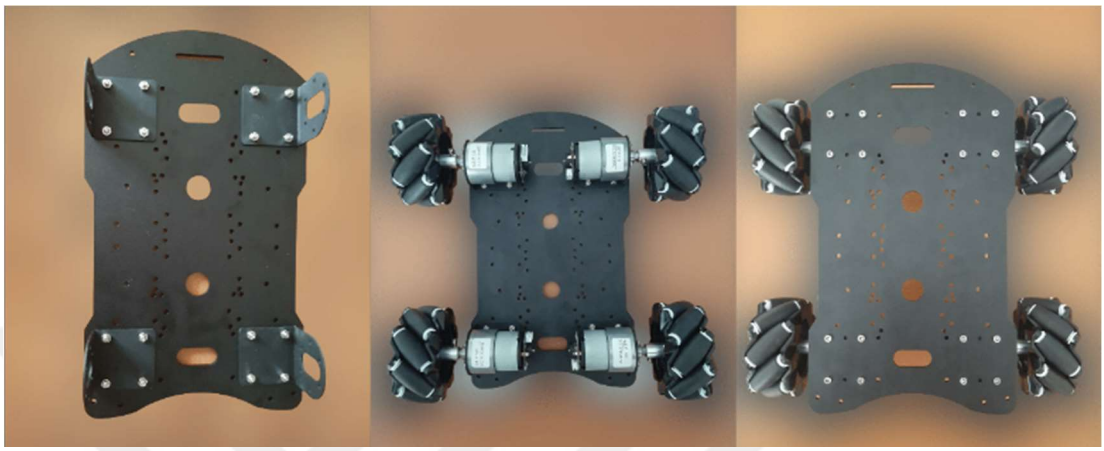
2.1.2 Mekanik montaj

Hazır alınan robot kiti üzerinden ilerlenilmiştir [28]. Robotun mekanik parçaları aşağıdaki gibidir. Görsel olarakta şekilde verilmiştir (Şekil 2.5).

- 1 adet plaka
- 4 adet dc motor
- 4 adet motor braketi
- 4 adet kaplinli mecanum tekerlek
- 4 adet M3, 10 mm düz vida
- 16 adet M3, 10mm vida
- 24 adet M3, 6mm düz başlı vida
- 16 adet M3 somun
- 4 adet motor kablosu
- 8 adet M3, 6mm altıgen başlı vida

Şekil 2.5 : Mekanik montaj malzemeleri.

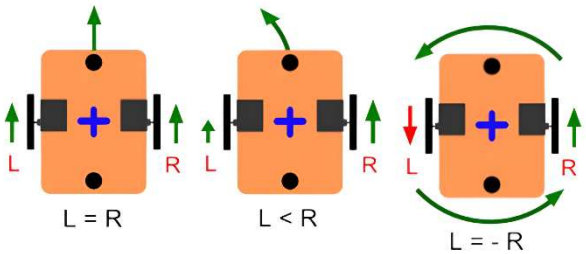
Montaj aşamasında ilk önce plaka ve motor braketlerinin bağlantısı yapılır. Daha sonra motorlar motor braketlerine takılır. Daha sonra tekerlere kaplinler takıldıktan sonra alyan yardımı ile motorlara sabitlenir. Böylelikle montaj aşaması tamamlanmıştır. Montaj görüntüleri (Şekil 2.6) verilmiştir.



Şekil 2.6 : Montaj görüntüleri.

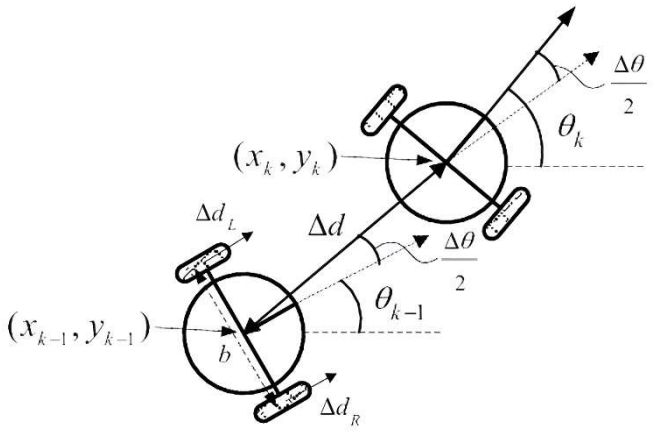
2.1.3 Kinematik model

Sürüş yöntemi olarak kullanılan diferansiyel sürüş yönteminde şekilde (Şekil 2.7) gösterildiği gibi tekerleklerin hız farkından robotun sürüşü, yön farkından ise kendi etrafında dönmesi sağlanır [25].



Şekil 2.7 : Diferansiyel Sürüş Yöntemi [26].

Şekil 2.8'de gösterildiği gibi odometri verisi zamana bağımlı olarak robotun hareketini karakterize eden veridir.



Şekil 2.8 : Odometri Verisi [27].

2.2 Elektronik

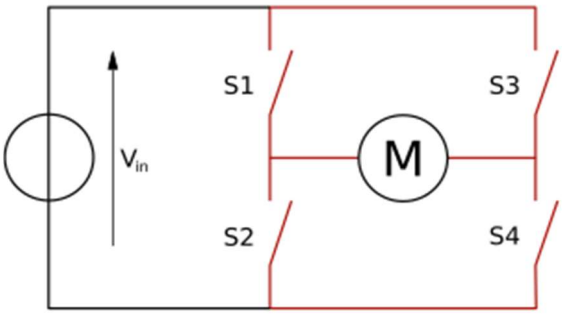
4 adet lifepo pil kullanılarak sistemin beslemesi 12V ve 5V olarak yapılmaktadır. 4 adet enkoderli dc motor ve 2 adet motor sürücü kullanılmıştır. Elektronik sistemin yönetimi için mikrodenetleyici olarak arduino mega kullanılmıştır. Sistemin elektrik çizimi Ek A da verilmiştir.

2.2.1 Motor ve enkoder

4 adet 12V 330 rpm DC motor kullanılmıştır.Ve üzerinde 5V beslemeli encoder mevcuttur. Encoder değerleri mikrodenetleyici üzerinden okunup bilgisayara aktarılmaktadır.

2.2.2 Motor sürücü

L298N Dual H Bridge Motor Sürücü, genellikle motorların hızını ve yönünü kontrol etmek için kullanılan bir motor kontrol kartıdır. Çift kanallı olduğundan 2 dc motor birbirinden bağımsız olarak sürülebilir. Şekil 2.9'da gösterildiği gibi H köprüsü, sürebilen bir devredir. S1 ve S4 anahtarları ile motor döndürülür, S2 ve S3 anahtarları açılarak ise tam tersi yönde akım verilerek motorun diğer yönde dönmesi sağlanır. PWM (Pulse Width Modulation) yani Sinyal Genişlik Modülasyonu ile de anahtarlama yapılarak motorun hızı kontrol edilebilir.



Şekil 2.9 : Motor sürücü H köprüsü [29].

2.2.3 Mikrodenetleyici

Arduino Mega 2560 , ATmega2560 tabanlı bir mikrodenetleyici kartıdır . 54 dijital giriş/çıkış pinine (15 tanesi PWM çıkışı olarak kullanılabilir), 16 analog girişe, 4 UART'a (donanımsal seri port), 16 MHz kristal osilatöre, USB bağlantısına, güç girişine, ve bir sıfırlama düğmesine sahiptir. Sistemde bilgisayar ve sürücüler arasındaki ara katmandır. Motorları süren sürücülere komutlar arduino üzerinden gitmektedir.

2.2.4 Pil ve batarya yönetim sistemi

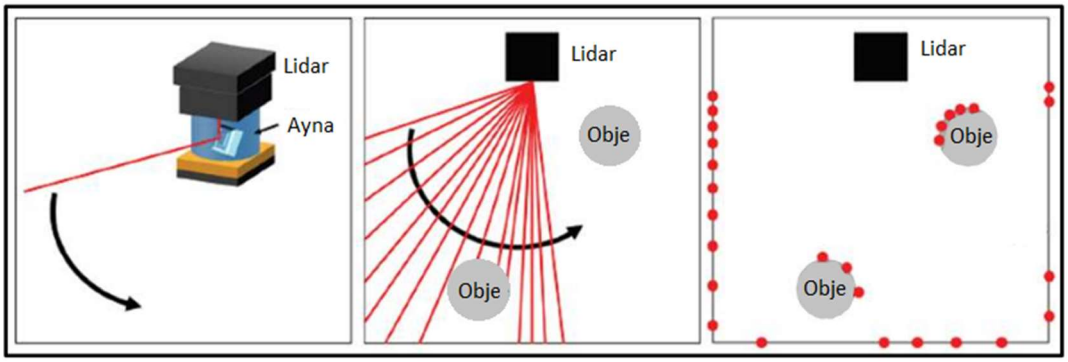
4 adet 3.2V LiFePo4 (Lityum Demir Fosfat) pil 4 seri 1 paralel (4S1P) şeklinde kullanılarak 12 V çıkış sağlanmaktadır. Her hücreyi dengeli bir şekilde kullanılmasını sağlamak için bms (battery management system) kullanılmaktadır. Dönüştürücüler ile sistemde 12V ve 5V sağlanmaktadır. Bilgisayar 5V luk kanaldan beslenmektedir.

2.2.5 LIDAR sensörü

Robotun çevreyi algılamasında ışık algılama ve ölçme sensörü (LIDAR-Laser Imaging Detection and Ranging) – Laser Distance Sensor (LDS) - kullanılmıştır. LIDAR, çevresindeki nesnelerin konumlarını algılamak için kullanılan radar benzeri bir sensördür. LIDAR sensörleri yüksek performans ve hızla gerçek zamanlı veri toplama avantajına sahiptir, bu nedenle mesafe ölçümü ile ilgili geniş bir uygulama yelpazesine sahiptir. LIDAR sensörü, nesnelerin ve insanların algılanması için robotik alanında, SLAM algoritmalarında ve gerçek zamanlı veri topladığı için insansız araçlarda yaygın olarak kullanılmaktadır.

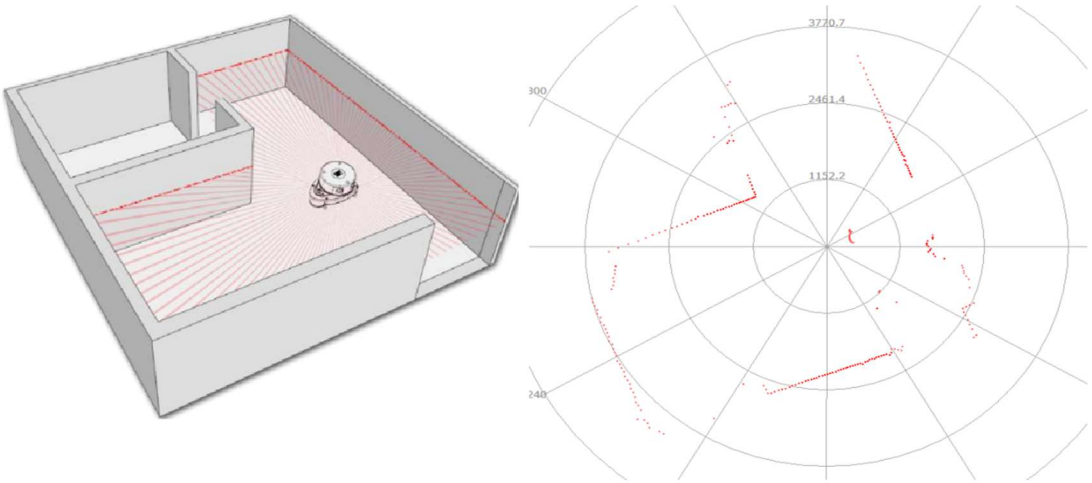
LIDAR sensörünün temelinde lazer, yansıtıcı bir ayna ve motor mevcuttur. Şekil 2.10'da solda LIDAR sensörü ve içindeki ayna gösterilmektedir. Motor aynayı

döndürür, sensör lazerin dönüş süresini dalga boyundaki fark ile ölçer. Ortadaki şekilde gösterildiği gibi LIDAR etrafını yatay bir düzlemde tarar. Sağ şekilde de gösterildiği gibi uzaklık arttıkça hassasiyet azalmaktadır. Cam, plastik şişe gibi malzemeler lazer ışığını dağıttığından sağlıklı veri elde edilememektedir. Sadece yatay bir düzlemi taradığından 2D veri olduğu unutulmamalıdır.



Şekil 2.10 : LIDAR sensörü ile mesafe ölçümü [30].

Bu projede Slamtec markanın RP-LIDAR A1M8-R5 LIDAR sensörü kullanılmıştır. Sensörün tarama mesafesi 0,15 – 12 metre, açısı 360 derece, frekansı 5.5 Hz'dir. Sensörün lazer ışınlarını yansıttığı durum Şekil 2.11'de verilmiştir.



Şekil 2.11 : LIDAR sensör tarama görseli [31].

LIDAR sensörü sağladığı LaserScan mesaj türündeki veri ile birlikte yazılımda kullanılır. LaserScan verisi sensörün hangi tf'te olduğu (frame_id), kaç derece tarama yaptığı (angle_min/angle_max), min ve max tarama mesafesi (range_min/range_max),

tarama frekansı ve çözünürlüğü ile birlikte her noktanın uzaklığını metre cinsinden içeren diziyi içermektedir.

2.3 Bilgisayar

Bu projede bilgisayar olarak NVIDIA Jetson Nano kullanılmıştır. Bilgisayara ubuntu 20.04 işletim sistemi kurulmuştur.

2.4 Gerçek Robot Görüntüleri

Tüm montaj işlemleri bitirilmiş robotun farklı açılardan görselleri verilmiştir (Şekil 2.12).

Şekil 2.12 : Robotun farklı açılardan görünümü.

3. YAZILIM

3.1 ROS

ROS (Robot Operating System) işletim sistemi değil, robotik uygulamalar için Berkeley Yazılım Dağıtımı (Berkeley Software Distribution - BSD) lisanslı ve açık kaynaklı bir yazılım geliştirme kitidir. Ticari ve araştırma amaçlı kullanım için ücretsizdir. ROS 2007 yılında Stanford AI Robot projesinin desteği ile Stanford Yapay Zeka Laboratuvarı'nda (Stanford Artificial Intelligence Laboratory - SAIL) geliştirildi. Daha sonra robotik araştırma enstitüsü olan Willow Garage, 2013 yılından itibaren ise Open Robotics tarafından geliştirmesi devam etmektedir. Birçok araştırma kurumu ve firma donanım ekleyerek ve kod örneklerini paylaşarak ROS'a katkıda bulunmaktadır. Bu çerçevede her geçen gün içerdiği proje sayısı ve desteklediği cihaz sayısı artmaktadır [32].

3.1.1 Neden ROS?

Tekerleği yeniden icat etmeden, ROS kullanıp üzerine eklemeler, geliştirmeler yaparak daha hızlı ve daha profesyonel ürünler, geliştirmeler yapılabilir. [33]. ROS ekosistemi aşağıdaki imkanları bize sunarak geliştirmelerimizi daha hızlı, profesyonel yapmamızı sağlar. ROS, küresel ve aktif bir topluluk desteği sayesinde sürekli olarak geliştirilmekte ve güncellenmektedir. Kullanımda kanıtlanmış bir platformdur ve birçok robotik uygulamada başarıyla kullanılmaktadır. ROS, açık kaynaklı bir platform olduğu için, kullanıcılar tarafından kolayca özelleştirilebilir ve geliştirilebilir. ROS, çoklu alanlarda kullanılabildiğinden farklı robotik uygulamalar için uygun bir seçenektir. Farklı yazılım dillerini desteklediği için, kullanıcılar tarafından tercih ettikleri dili kullanarak projelerini geliştirebilirler. ROS, sanayi desteği sayesinde, endüstriyel robotik uygulamalar için de uygun bir seçenektir. ROS'un simülasyon ortamı, robotların gerçek dünya koşullarında test edilmeden önce simüle edilerek, tasarım ve geliştirme süreçlerinde zaman ve maliyet tasarrufu sağlamaktadır.

Şekil 3.1 : ROS ekosistemi [34].

✔ Ara Katman Yazılımı

ROS özünde, genellikle "ara katman yazılımı" veya "tesisat" olarak adlandırılan bir mesaj iletme sistemi sağlar. Yazılım geliştirilirken donanım ile, yazılım paketlerinin kendi arasında, arayüzler ve merkezi yönetim sistemleri ile iletişim en önemli ihtiyaçtır. ROS'un sağlamış olduğu mesajlaşma sistemi, yayınlama/abonelik modeli aracılığıyla dağıtılmış düğümler arasındaki iletişimin ayrıntılarının kolay yönetilmesine ve hız kazanılmasına olanak sağlar.

✔ Araçlar

Robot uygulamaları oluşturmak zordur. Uygulamaları verimli bir şekilde oluşturmak için iyi geliştirici araçlarına ihtiyaç vardır. Simülasyon ortamı (Gazebo), robot görselleştirme (RViz) ve loglama, kayıttan oynatma, grafiğe dökme, çalıştırma ve görselleştirme araçlarını (RQt) içermektedir.

✔ Yetenekler

ROS ekosistemi, robot geliştiricileri için önemli bir imkandır. İster robot kol için kinematik hesaplarına, ister yapay zeka algoritmalarına, ister dört ayaklı araç için bir yürüyüş ve denge kontrol algoritmalarına veya mobil robot için bir haritalama sistemine ihtiyaç olsun, ROS'ta hepsi için bir şeyler vardır. Bir fikri olan herkes, temel donanım ve yazılım hakkında her şeyi anlamak veya erişmek zorunda kalmadan bu fikri gerçeğe dönüştürebilme imkanına sahiptir.

✔ Topluluk

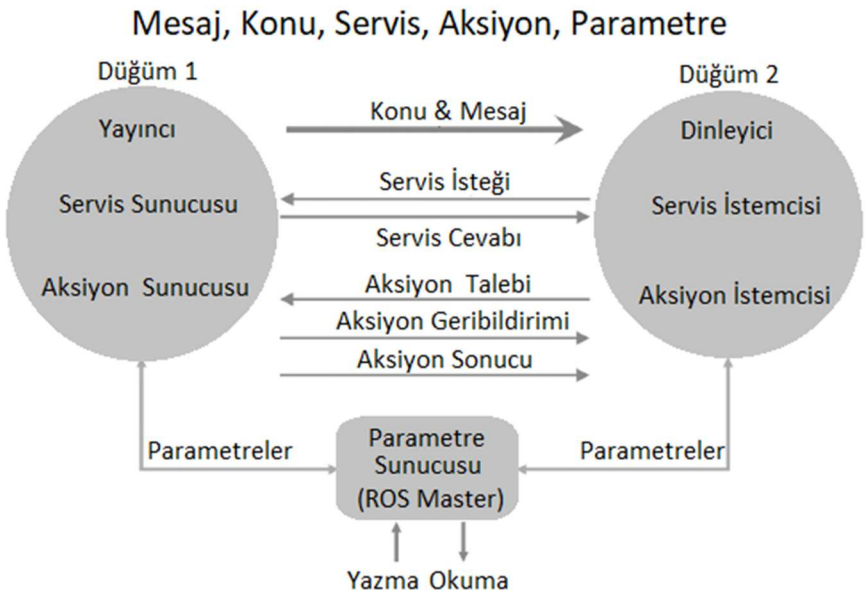
ROS topluluğu geniş, çeşitli ve küreseldir. Öğrencilerden ve hobi olarak ilgilenenlerden çok uluslu şirketlere ve devlet kurumlarına kadar her türden insan ve kuruluş ROS projesini devam ettiriyor.

Projenin topluluk merkezi ve tarafsız temsilcisi, paylaşılan çevrimiçi hizmetleri barındıran, dağıtım sürümlerini oluşturan ve yöneten ve temel yazılımın çoğunu geliştiren ve sürdüren Open Robotics'tir [34].

3.1.2 ROS terminolojisi

Tüm geliştiricilerin ve kullanıcıların ortak kullanacağı ROS terminolojisindeki terimler aşağıda verilmiştir.

1. Master: Düğümler arası bağlantı ve haberleşme için bir isim sunucusu gibi davranır. Master olmadan düğümler arası bağlantı ve haberleşme mümkün olmamaktadır. Roscore komutu, master'ı çalıştırmak için kullanılır ve çalıştığında standart olarak 11311 portunda çalışmaktadır.
2. Node (Düğüm): ROS içindeki en küçük işlemci birimini ifade eder. ROS terminolojisinde düğümlerin kullanımı şekilde gösterilmiştir (Şekil 3.2).

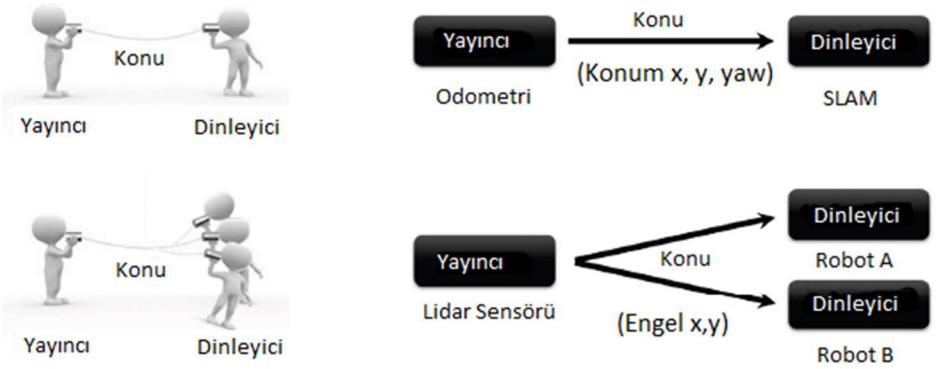


Şekil 3.2 : ROS terminolojisi [32].

3. Package (Paket): ROS'un temel birimidir. ROS uygulamaları paket bazında geliştirilmiştir. Paket içerisindeki node, message, service, action dosyalarının isimlerini ve bunlar için gerekli olan bağımlılıkları içeren CMakeList.txt ve package.xml dosyaları mevcuttur.

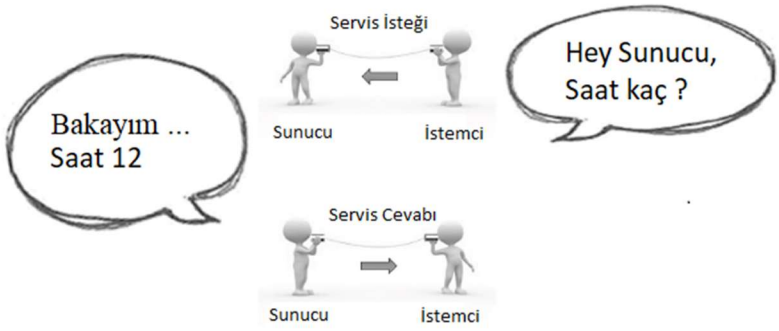
4. Message (Mesaj): Düğümler arasında veri gönderilirken veya alınırken mesajlar kullanılır.
5. Topic (Konu): Yayıncının ve dinleyicinin birleşeceği isimdir.
6. Publisher (Yayıncı): İlgili mesaj ve topiğe karşılık gelen verileri iletmektir.
7. Subscriber (Dinleyici): İlgili mesaj ve topiğe karşılık gelen verileri almaktır.

   Yayınlanan bir mesajı aynı anda birden fazla dinleyici alabilir. Şekil 3.3'te yayınlanan odometre verisini SLAM node'unun aldığı ve LIDAR sensörü verisini A ve B olmak üzere iki robotun aldığı gözlemlenmektedir.



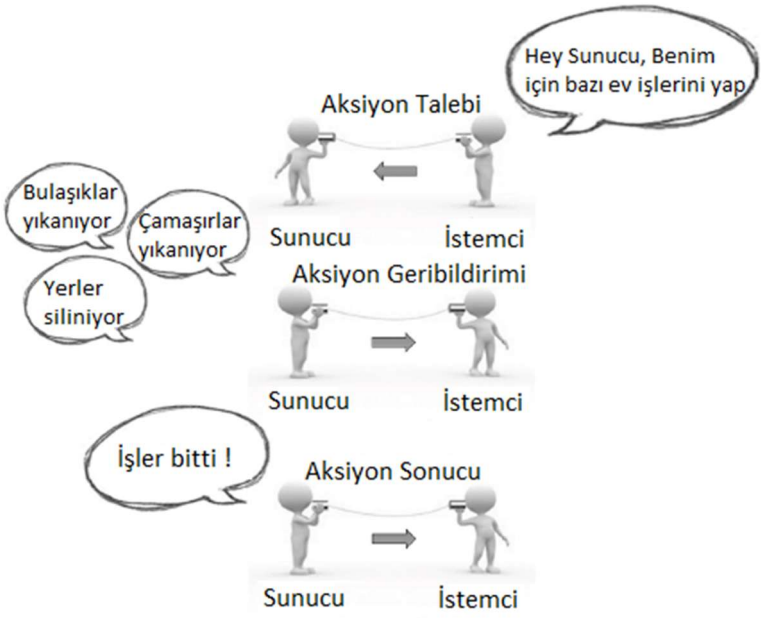
Şekil 3.3 : ROS yayıncı ve dinleyici [32].

8. Service (Servis): Client'lardan gelen isteklere server'ın cevap vermesidir. Basit bir sunucu ve istemci örneği Şekil 3.4'te verilmiştir.



Şekil 3.4 : ROS service [32].

9. Action (Aksiyon): Asenkron olarak çift yönlü kullanılan mesaj iletişim yöntemidir. Gelen istek sonuçlanıp sonuç döndürülene kadar ara mesajlarda döndürmektedir. Basit bir örneği Şekil 3.5'te verilmiştir.



Şekil 3.5 : ROS action [32].

10. Parameter (Parametre): Sistemdeki bir çok değişken değerin parametre olarak tutulmasıdır. Dinamik olarakta yapılabilmektedir.
11. Rosrun: Bu komut ROS'un temel çalıştırma komutudur. Tek bir düğüm çalıştırmak için kullanılır.
12. Roslaunch: rosrun komutunun aksine birden fazla düğümü tek seferde çalıştırmak için kullanılır. Bu komut launch uzantılı dosya kullanır. Bu dosyanın içerisinde çalıştırılacak düğümler olmalıdır.
13. Bag: ROS içerisinde mesajlar bag formatında kaydedilebilir. Bu mesajlar daha sonra tekrar kullanılabilir [32].

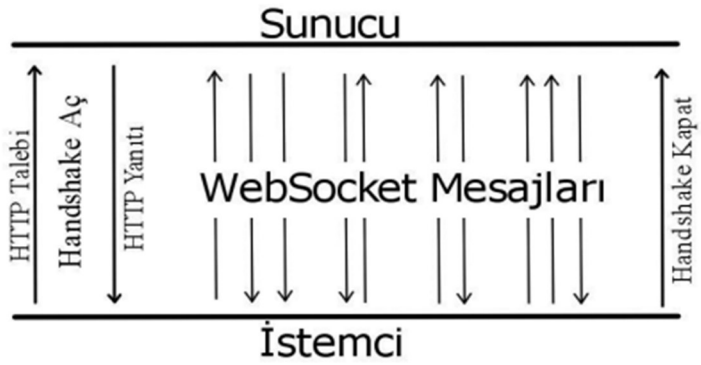
3.2 React

React, etkileşimli kullanıcı arayüzlerini geliştirmek için kullanılan bileşen tabanlı bir kütüphanedir. Javascript ve HTML birleşimi olarak reactjs (jsx) dili ile yazılmaktadır.

3.3 Web Socket

WebSocket protokolü, sunucu ve istemciler arasında gerçek zamanlı ve çift yönlü bağlantı sağlamak için geliştirilmiştir. TCP üzerinden gerçekleşen bu bağlantıda taraflar birbirlerine aynı bağlantı üzerinden gerçek zamanlı veri aktarımı yapabilirler.

WebSocket protokolünün çalışması temel olarak iki aşamadan oluşur. Bunlar mutabakat (handshake) ve veri transferi aşamalarıdır. Bir WebSocket bağlantısı HTTP talepleri üzerinden başlatılarak handshake sağlanır. Daha sonra iletilen mesajlar TCP üzerinden iletilir. Bu çalışma prensibi Şekil 3.6'da görülmektedir [35].



Şekil 3.6 : WebSocket protokolü [35].

4. YÖNTEMLER

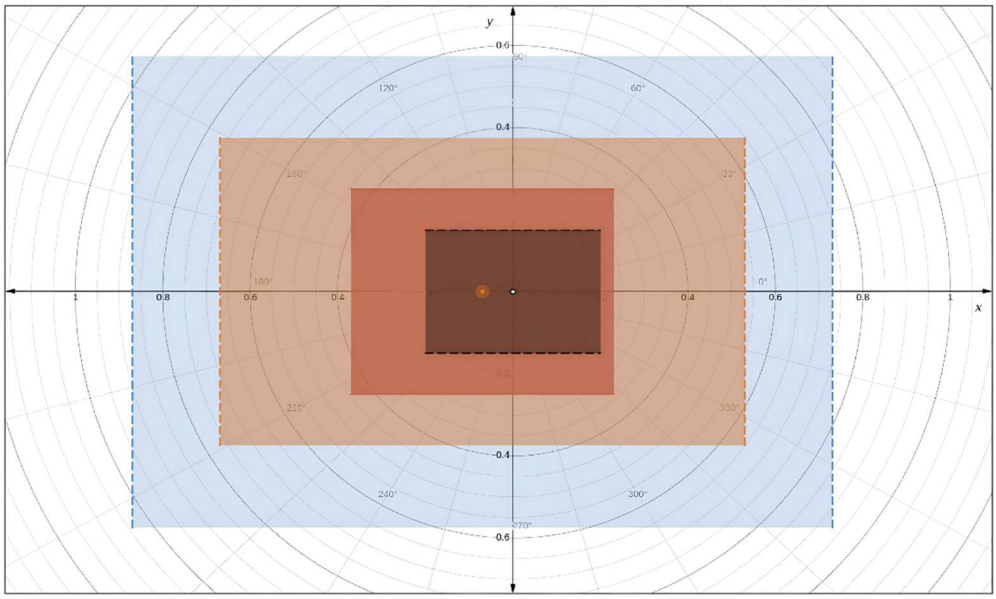
4.1 LIDAR Bölgeleri

Otonom sürüş yani navigasyon sistemi ortamdaki engellere ve hedefe yakınlığa göre dinamik olarak robotun hızını değiştirmektedir. Güvenlik amaçlı robota 2 adet uyarı ve 1 adet güvenli alan tanımlanmıştır. LIDAR verisi işlenip matematiksel işlemlerden geçirilerek bu bölgelerde herhangi bir sabit ya da hareketli nesne olup olmadığı saptanmaktadır. Uyarı bölgelerinde robotun hızı yavaşlatılmaktadır ve en son güvenli alanda ise robot durdurulmaktadır. Alanların genişliklikleri girilen uzunluk değerleri ile hesaplanmaktadır. Bu değerler tamamıyla dinamik bir şekilde değiştirilebilir. Robotun uzunluğu r_{hor} , genişliği r_{ver} ve LIDAR sensörünün robot merkezinden uzaklığı x_{LIDAR} olarak girilmektedir. Denklem 4.1'de güvenli alan olan en içteki bölgenin hesaplaması verilmiştir. Güvenli alanın robottan ne kadar geniş olacağı $l_{safetoffset}$ değeri ile girilmektedir. Denklem 4.2'de 2. uyarı bölgesi yani orta bölgenin hesaplaması verilmiştir. Bölgenin uzunluğu $l_{sidedist}$ ve genişliği için $r_{slowdown}$ değerleri girilmektedir. Denklem 4.3'te ise diğer uyarı bölgesi yani en dış bölgenin hesaplaması verilmiştir. Bölgenin uzunluğu $l_{normalsidedist}$ ve genişliği için r_{normal} değerleri girilmektedir. LIDAR güvenlik bölgelerinin görseli siyah robot, koyu turuncu güvenli alan, turuncu 2. uyarı bölgesi ve mavi birinci uyarı bölgesi olmak üzere Şekil 4.1'de verilmiştir.

$$\begin{gathered} -r_{hor} - l_{safetyoffset} \leq y \leq r_{hor} + l_{safetyoffset} \\ \{-x_{LIDAR} - r_{ver} - l_{safetyoffset} \leq x \leq l_{safetyoffset} + r_{ver} - x_{LIDAR}\} \end{gathered} \tag{4.1}$$

$$\begin{gathered} -r_{slowdown} - x_{LIDAR} < x < r_{slowdown} - x_{LIDAR} \\ \{-l_{sidedist} - r_{hor} \leq y \leq l_{sidedist} + r_{hor}\} \end{gathered} \tag{4.2}$$

$$\begin{gathered} -r_{normal} - x_{LIDAR} < x < r_{normal} - x_{LIDAR} \\ \{-l_{normalsidedist} - r_{ver} \leq y \leq l_{normalsidedist} + r_{ver}\} \end{gathered} \tag{4.3}$$



Şekil 4.1 : LIDAR güvenlik alan bölgeleri.

4.2 Kullanıcı Arayüzü

Robotun kurulumunun yapılması, yönetilmesi ve kullanılması için kullanıcı dostu bir arayüze ihtiyaç vardır. Teknik personeller harici kullanıcı seviyesine indirgenerek robot ile herkesin çalışması sağlanmaktadır. Seçilen robot ile websocket üstünden haberleşmektedir. Bu projede local ağda çalışma yapılmıştır ama sunucu seviyesinde kurulum yapılarak ve kullanım internete de çıkarılabilir. Tabi bu durumda güvenlik amaçlı kullanıcı ve rol tanımlama sistemi getirilmelidir.

Arayüzün ana hali Şekil 4.2'de verilmiştir. Sol üst kısımda proje numarası, robot ismi, robotun haritadaki x,y konumu ve açısı verilmiştir. Robotun anlık hız değerleri linear hız (m/s) ve açısal hız (rad/s) olmak üzere gösterilmiştir. Navigasyon modunda çalıştığı zaman çalışılan haritanın adı ve kat bilgisi gösterilmiştir. Eğer haritalama modunda ise buralar boş kalmaktadır.

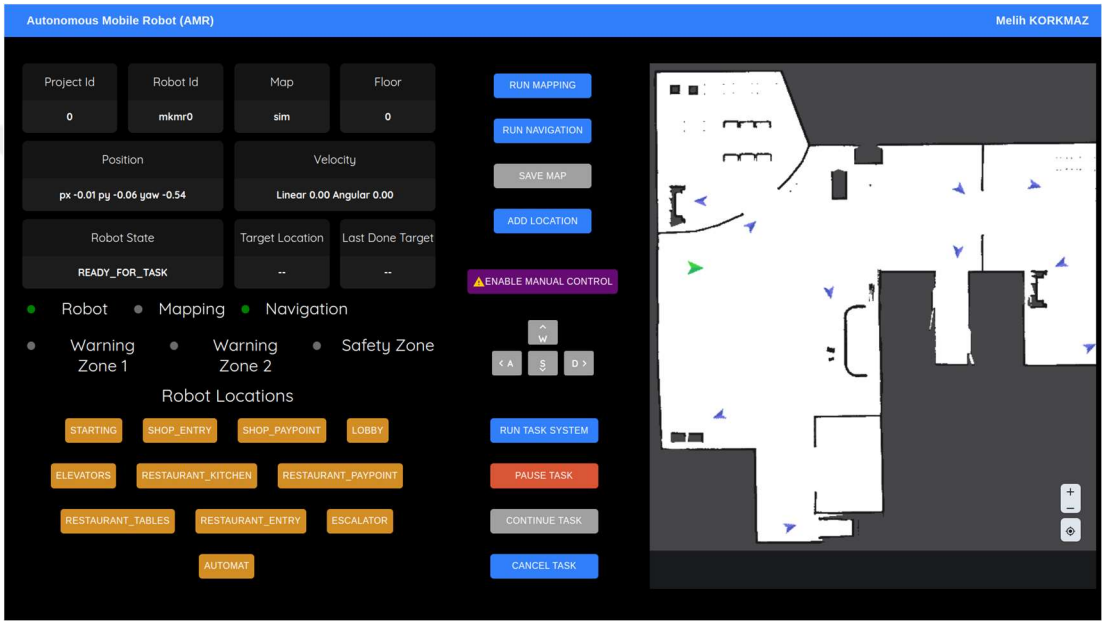
Ledler kısmında ise robot ile haberleşme durumu, robotun haritalama yada navigasyon modunda çalıştığı bilgisi verilmektedir. Alt satırda ise 2 adet uyarı ve 1 adet güvenlik bölgelerinin içerisinde herhangi bir şey olup olmadığının bilgisi de verilmiştir.

Orta üst kısımdaki butonlar ile haritalama ve navigasyon modu aktif edilebilir. Haritalama modunda iken harita ve kat ismi girilerek haritalama kaydedilebilir. Navigasyon modunda iken konum ismi girilerek konum/hedef kaydedilebilir. Mor olan manuel sürüş aktif etme butonu ile birlikte robot manuel olarak sürülebilir.

Robota w ile ileri, s ile geri, a ile sol dönüş ve d ile robot sağ dönüş hareketleri kazandırılabilir.

Sayfanın sağ tarafında robot içerisinde aktif olan harita ve o haritaya ait konumlar gösterilmektedir. Robotun anlık konumu yeşil ikon olarak gösterilmektedir.

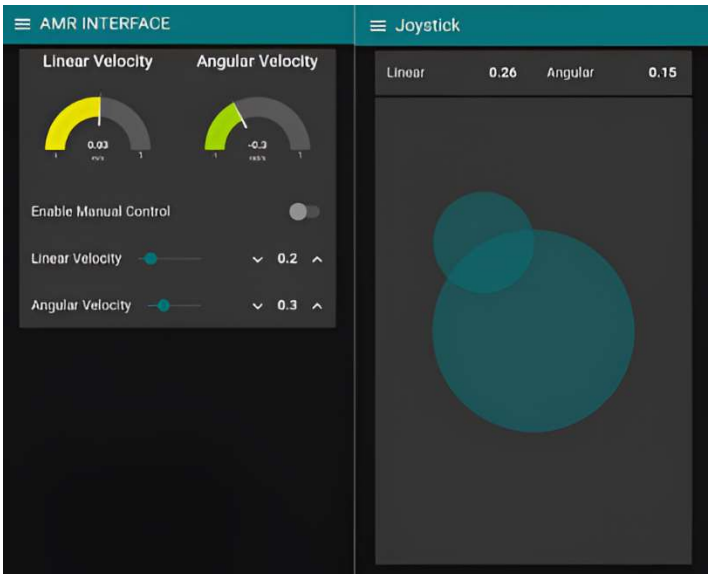
Görev yönetim sistemi başlatılabilir. Sol taraftaki hedef konumlar seçilerek robota konum verilebilir. Görev durdurulabilir, devam ettirilebilir ve iptal edilebilir. Sol orta tarafta ise robotun hangi aşamada olduğu, aktif hedef konumunu ve son başarılı hedef konumunu görebiliriz.



Şekil 4.2 : Kullanıcı arayüzü.

4.3 Telefon Sanal Kumandası

Robot manuel olarak sürülmek istenildiğinde her zaman bilgisayara erişim imkanı olmayabilir. İmkan olsa bile kolaylık olması açısından telefondan robotun ağına bağlı olduğunda erişilebilen sanal kumandası yapılmıştır. 2 sayfa olarak tasarlanmıştır. İlk sayfada hız limit değerleri değiştirilmekte, manuel mod devreye alınabilmekte ve anlık hız değerleri izlenebilmekte, 2. sayfada ise anlık hız değerleri izlenebilmekte ve sanal kumanda kullanılarak robot sürülebilmektedir (Şekil 4.3). Sanal kumanda verilerini koordinat sisteminde çıkarmaktadır. Y ekseni açısal hız değeri vermekte, X ekseni ise doğrusal hız değerini vermektedir.

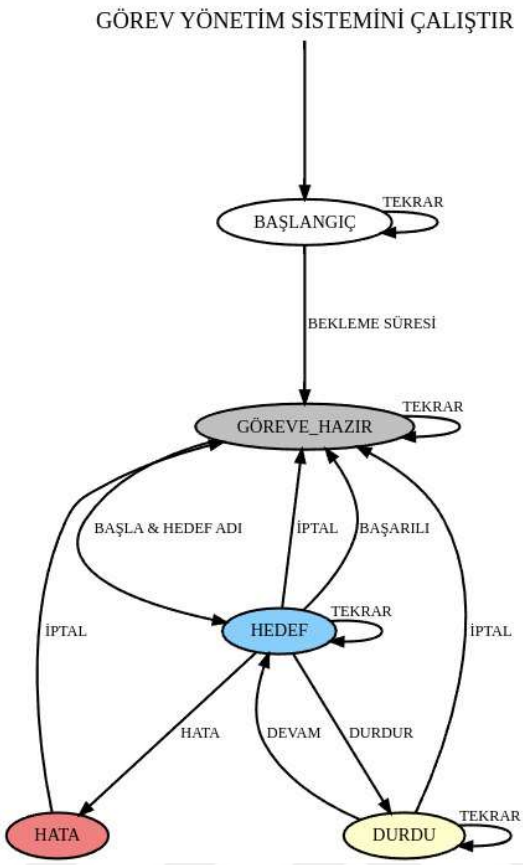


Şekil 4.3 : Kumanda arayüzü ayar ve sürme sayfası.

4.4 Görev Yönetim Sistemi

Geliştirilen görev yönetim sistemi sonlu durum makinesi (Finite-State Machine) mantığında yapılmıştır. Durumlar belirli bir işlemin yapıldığı ya da belirli bir işlem için beklediği ve akış diyagramında (Şekil 4.4) dairesel kutu içerisinde gösterilen elemanlardır. Durumlar arasında belirli şartlara göre gerçekleşen geçişler (transition) mevcuttur.

Akış diagramında (Şekil 4.4) olduğu gibi robot ilk açılışta boş durumda başlamaktadır. Görev yönetim sistemi arayüzden çalıştırıldığında BAŞLANGIÇ (STARTUP) durumuna geçmektedir. Gerekli şartlar ve süre sağlanınca GÖREVE_HAZIR (READY_FOR_TASK) durumuna geçmekte ve çalışmaya hazır halde beklemektedir. Arayüzden bir hedef seçildiğinde yani robota bir görev atandığında HEDEF (GOAL) durumuna geçmektedir. Arayüzden durdurulduğunda DURDU (PAUSED) durumuna geçmektedir. Devam denilerek robot kaldığı yerden HEDEF durumu ile devam etmektedir. Görev yönetim sistemi arayüzden verilen görev iptal edildiğinde GÖREVE_HAZIR durumuna ve herhangi bir hata durumunda HATA (ERROR) durumuna geçmektedir. Her bir durumdaki TEKRAR (REPEAT) geçişi ise o durumda iken while mantığında döngü şeklinde süreçi ilerleten ve gerekli şartların yani geçiş sinyallerinin gelmesini beklemek anlamına gelmektedir.



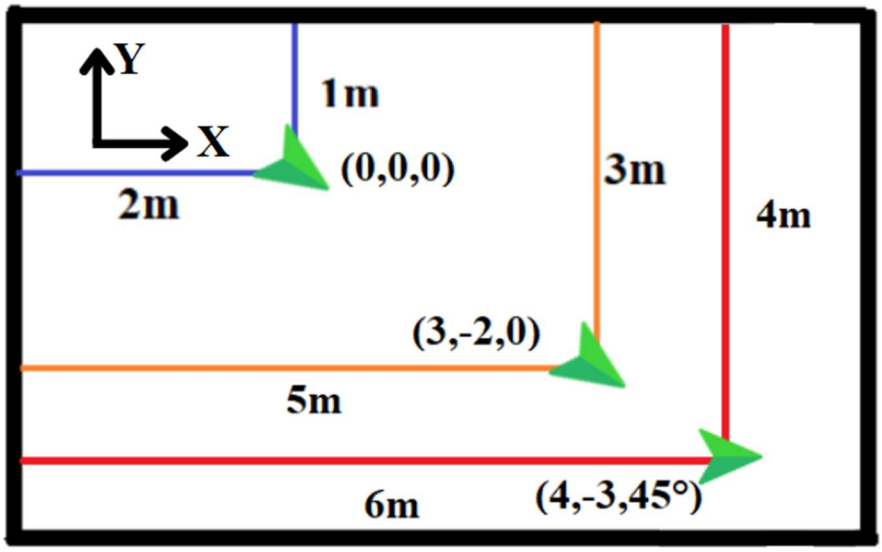
Şekil 4.4 : Görev yönetim sistemi mimarisi.

4.5 SLAM

SLAM (Simultaneous Localization and Mapping), robotun üzerindeki sensörler ile robotun pozisyonunu tahmin ederken bilinmeyen bir çevreyi keşfetmek ve haritalamak anlamına gelir. SLAM algoritması robotun odometri verisini yani hareket ve yerdeğişim değerleri ile LIDAR verisini kullanarak iterasyonlu bir şekilde harita verisini ortaya çıkarır.

Basitçe açıklamak gerekirse robot haritalama başlangıcında robotun haritadaki x konumu 0, y konumu 0 ve yaw açı değeri 0'dır. ROS'taki tf sistemine göre eksenler sağ el kuralına göre gelmektedir. Robot hiç hareket etmediğinden odometri değerleride sıfırdır. Sabit cisimlerden gelen ilk LIDAR değerleri ile harita verisi ortaya çıkmaktadır. Robot hareket ettirildikçe odometri verisi değişmekte robotun konumu haritada değişmektedir. Buradaki değişim miktarı kadar LIDAR sensör verisinde de değişim olmaktadır. Ve haritadaki engelin yeri ilk ve 2. durumda da aynıdır. Şekil 4.5'teki gibi robotun konumu başlangıç değeri olan (0,0,0) dan (3,-2,0) ve (4,-3,45°) olarak değişmiştir. SLAM algoritması çalıştırılırken verilen 5 santimetre çözünürlük

değerinden dolayı koordinat sistemi karelere bölünerek her piksel (grid) 5 santimere olacak şekilde çıkartılmıştır. Örneğin 30 santimetrelik bir kapı genişliği 6 karedir.



Şekil 4.5 : Haritalama başlangıcı koordinat sistemi.

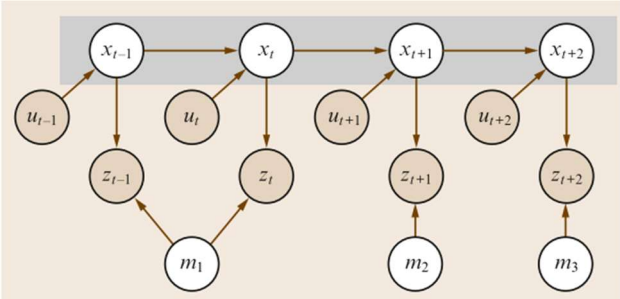
Hassas parametre ayarları ile algoritma optimize edilebilir. Örneğin uzun koridorlarda LIDAR sensöründe değişim az oranda kalacağından odometri değerinin ağırlığı (weight) arttırılabilir. Ya da LIDAR sensörü sisteme verilirken tarama alanı küçültülerek verilebilir ama bu durumda haritanın tamamlanması için robotun daha fazla gezdirilmesi ihtiyaç olabilir.

SLAM algoritması olasılık terminolojisinde denklem 4.4'teki gibi tanımlanır. Grafiksel modelde (Şekil 4.6) gösterildiği gibi zamana bağımlı olarak ardışık işlem yaparak harita verisini çıkarır. Olasılıksal çizge modeli olan bayes ağ grafiği olarak tasvir edilmektedir. Robot x_{t-1} konumundan x_{t+2} konumuna sürüş yöntemi tarafından odometri verisi u_t ile yönlendirilir. LIDAR sensöründen her gelen yeni veriye göre olasılık hesabı yapılır. Grid yani karesel ızgara yapısına eklenmektedir. Nesnenin yani isabetin olduğu kareler ve ıskalamanın olduğu kareler şeklinde her bir iterasyonda yenilenmektedir (Şekil 4.7). Örneğin haritalama esnasında robotla birlikte haritalayan kişi robota göre farklı konumlarda hareket ederse haritada sabit engel olarak çıkmaz. Kişinin sabit kalması için o gridlerde bir süre bulunması gerekmektedir bu işlemlerde olasılık terminolojisi sayesinde olmaktadır.

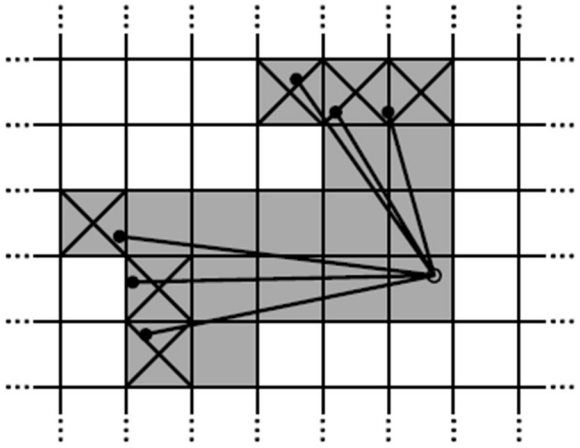
t: zaman

x_t: t anındaki robotun konumu
u_t: t anındaki odometri verisi
z_t: t anındaki LIDAR sensöründen ölçülen değer
m_n: harita

$$p(x_t, m / z_t, u_t) \tag{4.4}$$

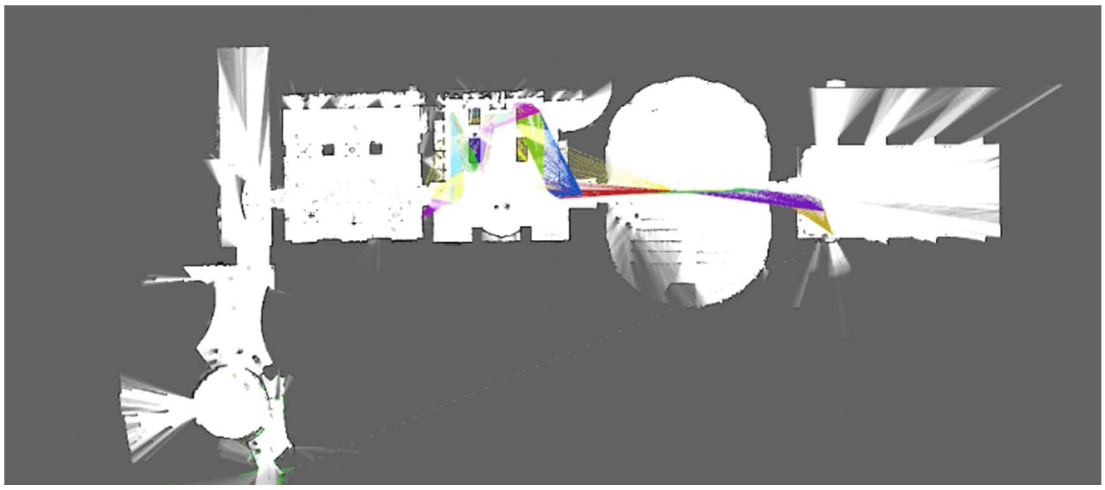


Şekil 4.6 : SLAM grafiksel model [36].



Şekil 4.7 : Haritalama isabetli veri (gri ve çizgili) ve iskalı veri (gri) [37].

Cartographer, birden çok platform ve sensör konfigürasyonunda 2 ve 3 boyutlu olarak gerçek zamanlı eş zamanlı yerelleştirme ve haritalama (SLAM) sağlayan bir sistemdir. Cartographer sunduğu birçok parametre ile sistemi kendi donanım ve sensörlerimize uygun hale getirip, gerçek zamanlı ve parametre dosyasında girdiğimiz (5 cm) çözünürlük değerinde haritalama gerçekleştirilmesine olanak sağlar. Haritalama esnasında küçük haritalar (submap) oluşturduğundan ve arka planda bu verileri gerçek zamanlı eşleştirdiğinden harita başlangıcı ve bitiş noktasını üst üstte oturtturmakta (loop closure) ideal sonuçlar vermektedir [37]. Şekil 4.8'de örnek bir harita verilmiştir.

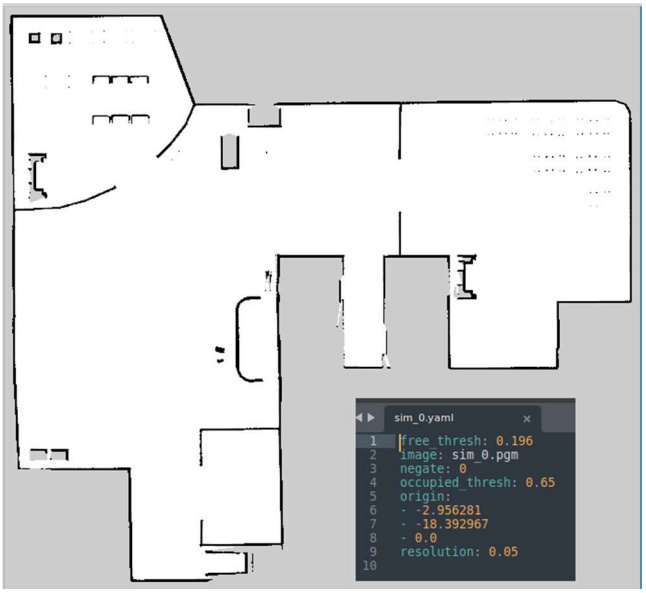


Şekil 4.8 : Örnek cartographer haritası [22].

Arka planda submapleri hem haritalarken hemde navigasyon esnasında işleme aldığından diğer yöntemlere göre biraz daha fazla kaynak harcamaktadır. Robotun konumunu bilerek bozduğumuzda (initial pose verilmesi) submapler üzerinden işlem yaparak kendini doğru yere almaktadır. Farklı odaya taşınsa bile kendini doğru odaya taşımaktadır. Ortamdaki objelere ve bilgisayarın performansına göre süresi 1 ile 20-30 saniye civarında olmaktadır.

Cartographer haritalama yöntemi diğer haritalama yöntemlerine göre ekstra alabileceği veriler mevcuttur. Örneğin birden fazla LIDAR verisini alabilir. Derinlik kamera verisini (point cloud) ve IMU verisini alarak hassas hareketlerde robotun atalet değişimlerini daha iyi algılayabilir. Ek olarak landmark (qr code benzeri tag) verisini alarak büyük ve değişkenliği fazla olan ortamlarda daha ideal konumlandırma yapabilir.

Navigasyon işleminde kullanılmak üzere kaydedilen harita pgm formatında görüntü ve yaml formatında parametre dosyalarından oluşmaktadır (Şekil 4.9). Parametre dosyasında pgm harita görsel dosyasının dosya yolu, haritanının çözünürlüğü, orijin (başlangıç noktası) gibi değerler mevcuttur. Harita dosyası aslında vektörel bir dosyadır. Her piksel değerinin bir karşılığı vardır. Bilinmeyen alan (unknown) -1 değerinde gri renkte, serbest alan (free) 0 değerinde beyaz renkte ve engel kapalı olan olan (obstacle) 100 değerindedir.



Şekil 4.9 : Örnek kaydedilen harita dosyaları.

Kaydedilen harita görselinin piksel değerleri 688 ve 632 pikseldir. Yaml yani config dosyasında çözünürlük 0,05 olara haritanın 34.4 x 31.6 metrede olduğu anlaşılmaktadır. Gerçek robot ile haritama yapıldığında da hesaplamalar bu şekilde olmaktadır.

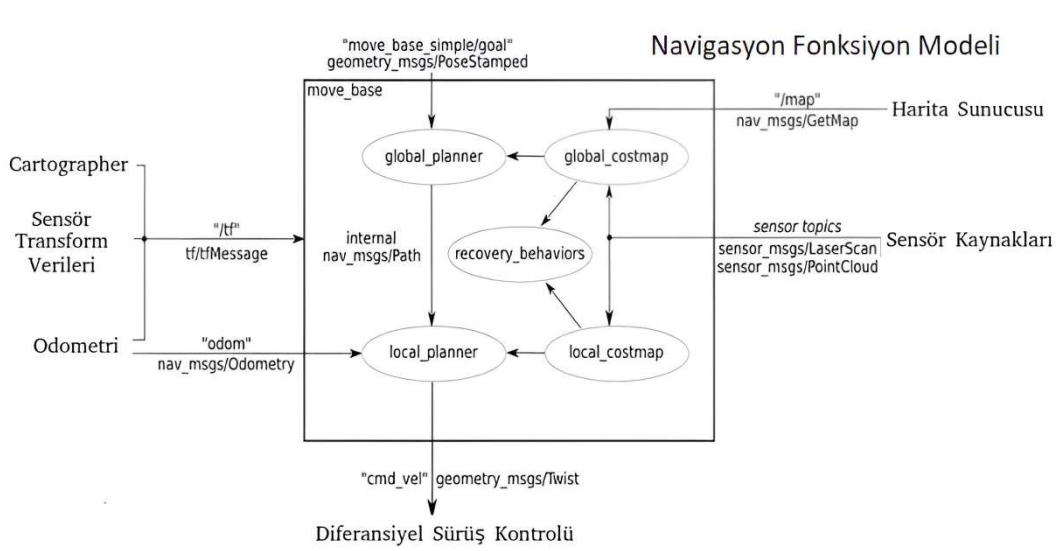
4.6 Navigasyon

Navigasyon fonksiyonu daha önceden çıkartılan haritanın yüklenmesi ve LIDAR verisinin kullanılmasıyla konumlama işlemini yaparak otonom bir şekilde hedef noktaya dinamik olarak gitmesidir. ROS'un navigasyon fonksiyonu için kullanılan move_base paketinin grafiksel arayüzü, Şekil 4.10'da gösterilmiştir.

Harita dosyasında her pixelin karşılık geldiği çözünürlük değeri ve vektörel harita dosyası navigasyon sistemine yüklendiğinde bu değer sayesinde koordinat sisteminde ölçüler ortaya çıkmaktadır ve parametre dosyasındaki orijin değerlerine göre koordinat sistemini ötelemektedir.

Sabit ve hareketli engelleri simüle eden 2 adet maliyet haritası (costmap) mevcuttur. Global costmap global yol planlaması için kullanılır. Global yol planlaması harita üzerindeki hedefe uzun vadeli yol planını yapar. Local costmap ise robotu merkezine alıp onunla hareket eden küçük bir haritadır ve local yol planlaması için kullanılır. Robotun hız değerleri local costmapteki engellere göre ve local yol planlamasındaki

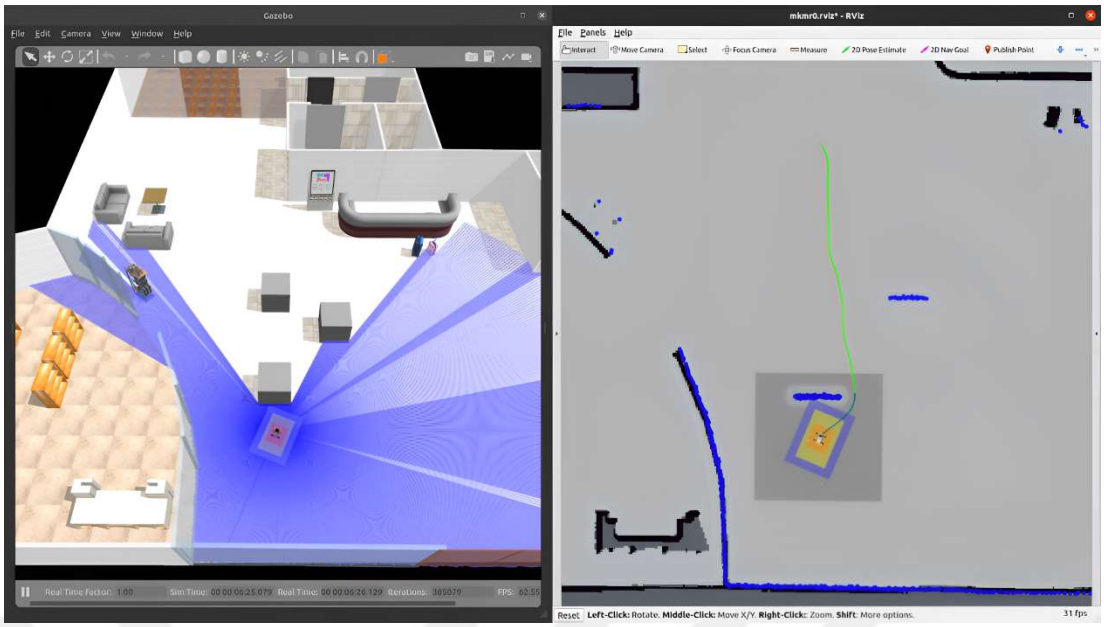
pozisyonlara göre çıkar. Engelleri algılamada 2D olarak LIDAR verisi, 3D olarak nokta yığınlarından oluşan point cloud (pcl) verisi kullanılabilir [38].



Şekil 4.10 : ROS navigasyon [38].

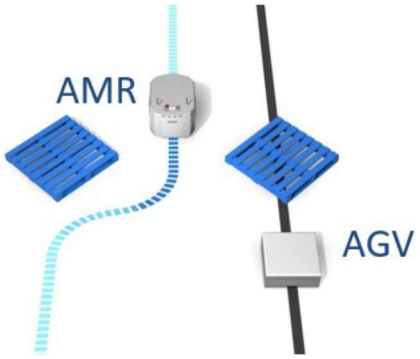
Navigasyon fonksiyonunun çalıştırılmış hali Şekil 4.11'de mevcuttur. Robot verilmiş hedefe sabit ve hareketli engelleri dikkate alarak otonom bir şekilde gitmektedir. Haritada robotun merkezi olduğu gri dörtgen ile local costmap, turkuaz ile global costmap, mavi ile local path gösterilmiştir. Local path'in takip ettiği global path ise yeşil renk ile gösterilmiştir. Local ve global path düz şekilde iken harita çıkartılırken herhangi engel olmayan yerde robotun önüne küp konularak yol planlamalarının dinamik olarak engele göre güncellendiği gözlemlenmiştir. Robotun otonom sürüşünü, hız değerlerini, yol planlamasını, engelleri algılamasını ve vereceği tepkimeleri optimize etmek için bir çok parametre mevcuttur.

Robota verilen hedef değeri ile robotun anlık konumu karşılaştırılmaktır. Verilen tolerans değerleri sağlandığında hedefe varıldı sinyali alınmaktadır. Bu projede xy tolerans değeri öklid olarak 0,1 metre, açısal tolerans değeri ise 0,1 radyan yaklaşık 5.73 derece olarak girilmiştir. Gerçek robot üzerinde yapılan çalışmalarda zemindeki belirteçler kullanılarak konumsal hassasiyet test edilmiştir.



Şekil 4.11 : Navigasyon çalıştırılması.

AMR'leri AGV (Automated Guided Vehicle)'lerden ayıran en önemli fark Şekil 4.12'de gösterildiği gibi ortamda herhangi bir belirteç olmadan otonom sürüşe sahip olması ve engel ile karşılaştığında yolunu dinamik olarak değiştirip hedefe gidebilmesidir.



Şekil 4.12 : AMR ve AGV Farkı.

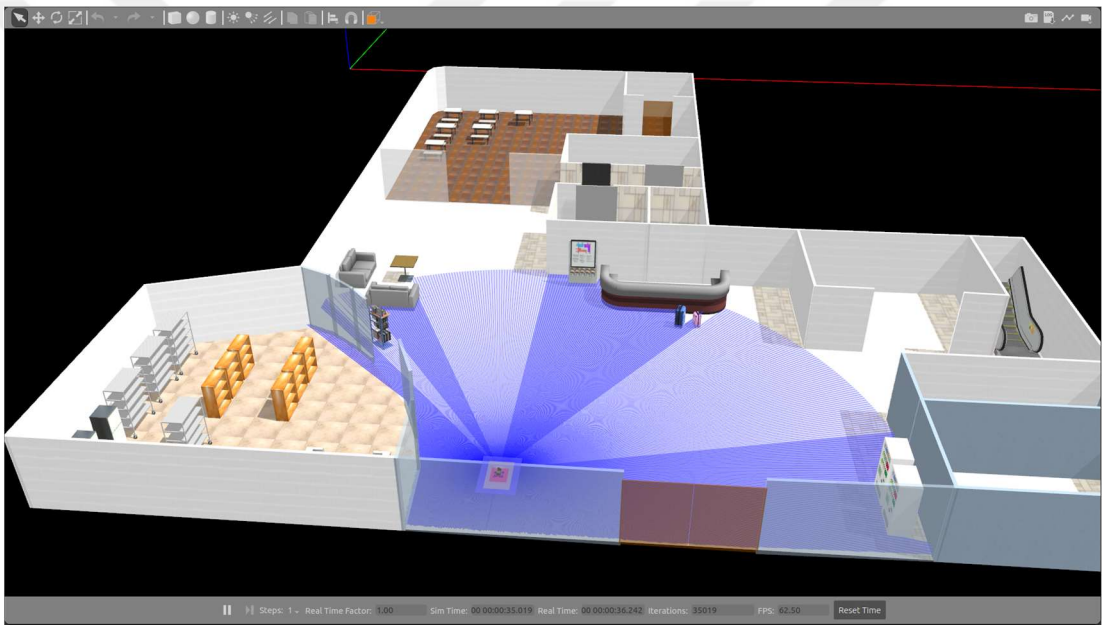
Robot LIDAR verisinden bir nesne yakaladığında o veri direk costmap yani maliyet haritasına eklenmektedir. Girilen parametreler ile engele ne kadar yakından ve uzaktan yol planlaması yapabileceği belirlenmektedir. Örneğin insan algılamalarında LIDAR sensörü kişilerin bacak seviyesinde olduğundan yol planlaması yaparken yanlış yönlendirme olabilir. Çünkü kişinin ayağına çarpılabilir. Bu yüzden 47 numara ayak yaklaşık 22 cm olarak ekstra offset değeri girilmiştir.

Algılanan nesne hareketli bir nesne de olabilir. LIDAR sensöründen gelen değere göre haritananın o kısmına engel eklenir. Hareketli nesne o konumdan gittikten sonra LIDAR verisi o değeri daha büyük verecektir. Navigasyon sistemi maliyet haritasında o konumu temizlemektedir. Engel kalktıktan sonra olan açıda LIDAR sensörünün tarama menzili (max_range) içerisinde başka bir engel olmadığında, LIDAR sensörü sonsuz (infinite) olarak özel değer döndürmektedir. Navigasyon sisteminde girilen parametre ile o değer algılanıp, o kısımda maliyet haritasında temizlenme olmaktadır.

5. ROBOTUN YÖNETİMİ

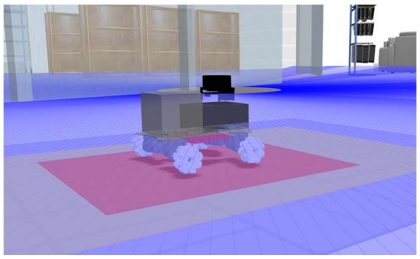
5.1 Simülasyon Dünyası (Gazebo)

Simülasyon testlerinin daha gerçekçi olması için gerçek hayattan uyarlanmış bir dünya (Şekil 5.1) oluşturulmuştur. Market, restaurant, boş oda ve lobi kısımları mevcuttur. İlerleyen süreçte bina testlerine uygun olması için kapı ve asansör kısımları da ayrılmıştır.



Şekil 5.1 : Gazebo dünyası.

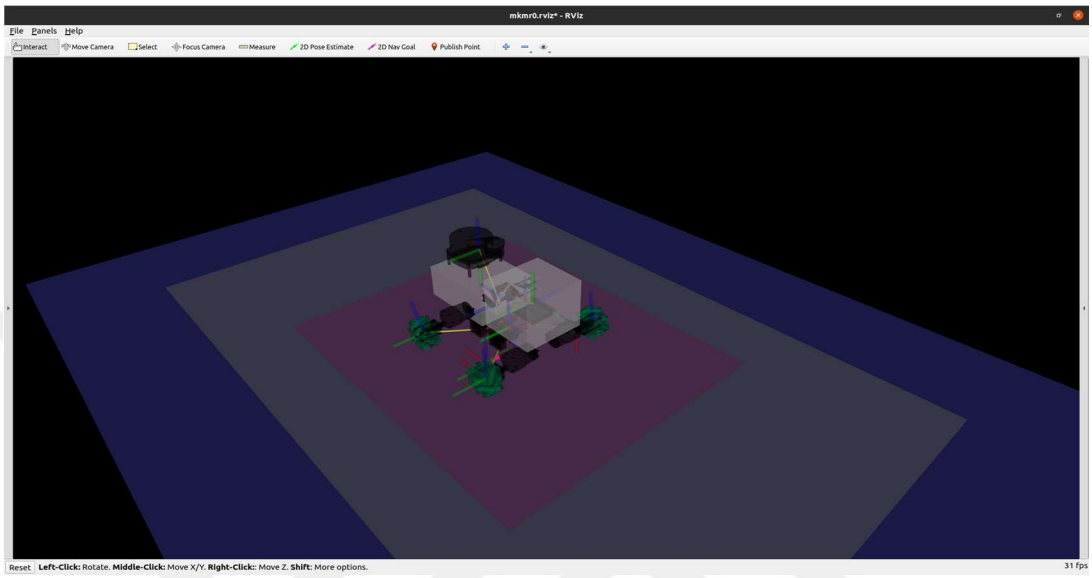
Robot (Şekil 5.2) tüm mekanik parçaları, kinematik modeli ve sensörleri ile simülasyon ortamına yüklenmiştir. Robottan çıkan mavi çizgiler LIDAR sensörünün ışınlarıdır.



Şekil 5.2 : Gazeboda robotun çıktısı.

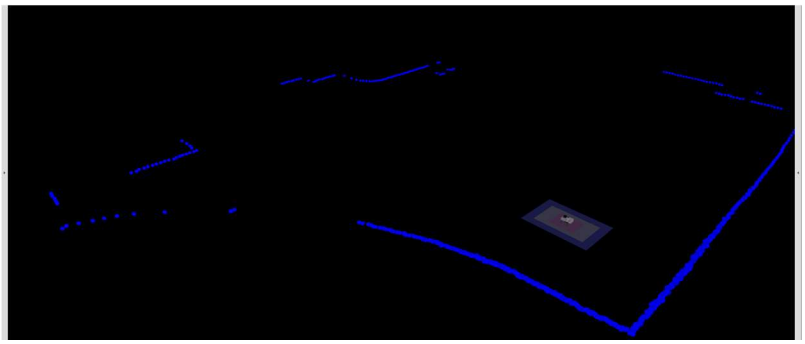
5.2 Robot Görselleştirme (RViz)

ROS'un sağlamış olduğu görselleştirme arayüzü RViz ile robotun çıktılarını görebiliyoruz. Robot model içerisinde robotun tüm mekanik parçalarını, motorlarını, tekerleklerini, sensörlerini ve güvenlik bölgelerini görebiliriz (Şekil 5.3).



Şekil 5.3 : RViz robot model.

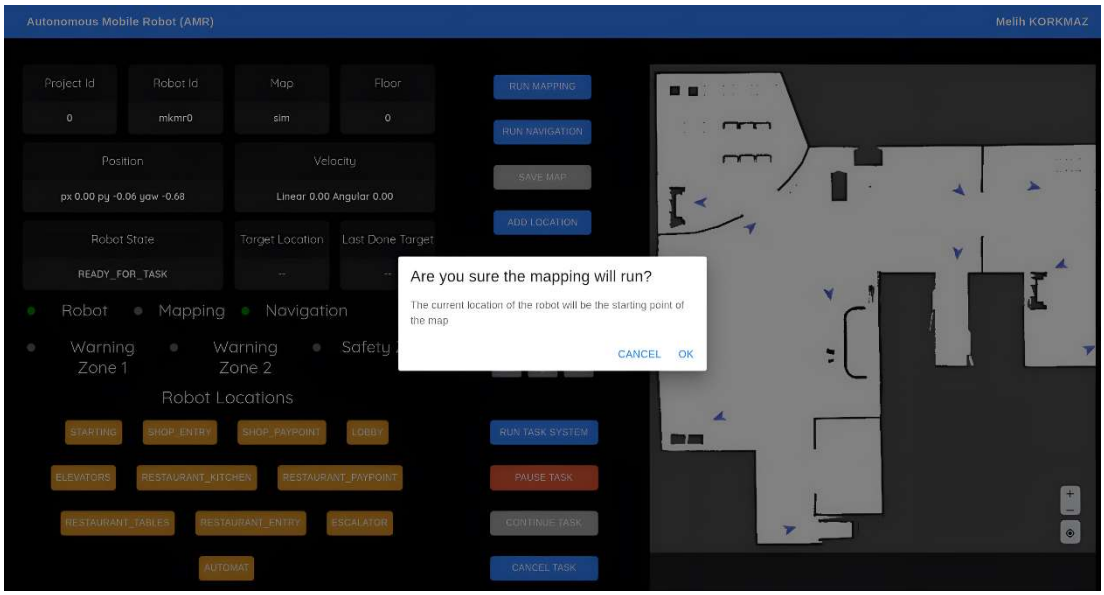
Robotun gazebodan gelen ya da gerçek robotta ki RP-LIDAR sensöründen gelen LIDAR verilerini gözlemleyip analiz edebiliriz (Şekil 5.4).



Şekil 5.4 : RViz **LIDAR**.

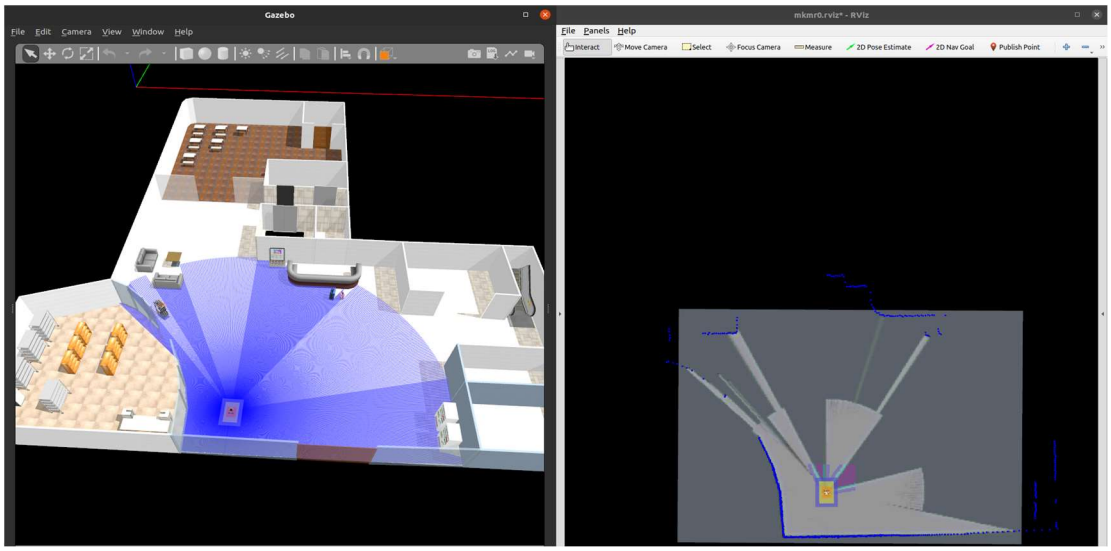
5.3 Haritalama Metodu

Kullanıcı arayüzünden haritalama fonksiyonu başlatılarak robotun bulunduğu ortamı haritalaması sağlanabilir (Şekil 5.5).

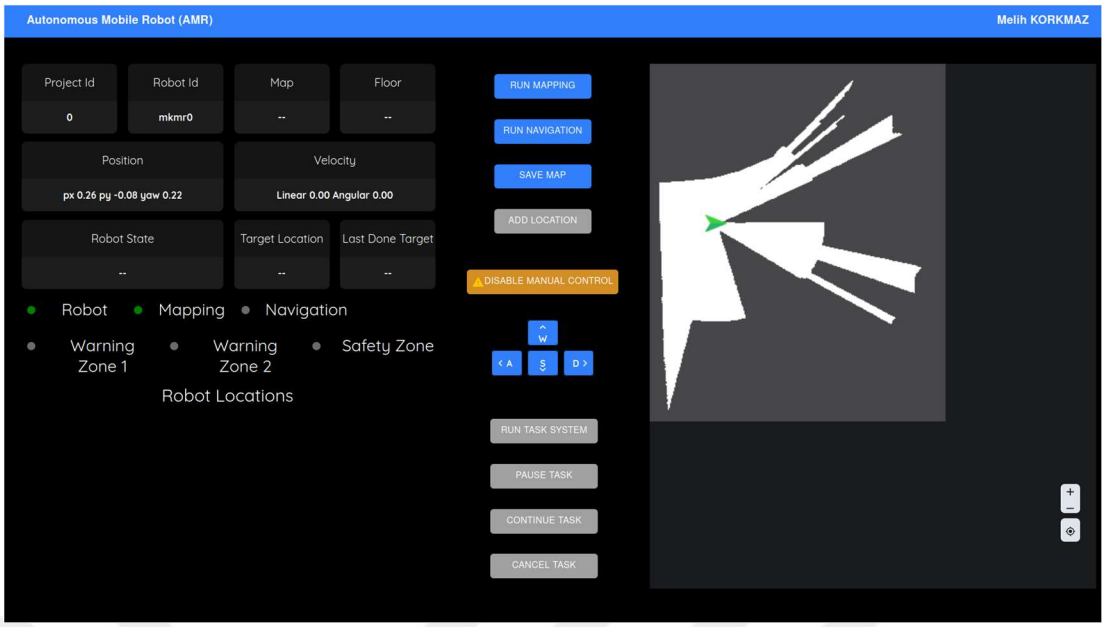


Şekil 5.5 : Arayüz haritalama başlatma.

Robot haritalama esnasında sanal kumanda ya da arayüz üzerindeki butonlardan manuel olarak sürülür. Haritalama fonksiyonu başlangıcında LIDAR'dan gelen ilk veriler kadar olan kısmın haritası çıkar. Gazebo ve RViz görüntüsü Şekil 5.6'da, kullanıcı arayüz görüntüsü ise Şekil 5.7'de verilmiştir. Robot gezdirildikçe harita genişlemektedir ve arayüzden takip edilmektedir.

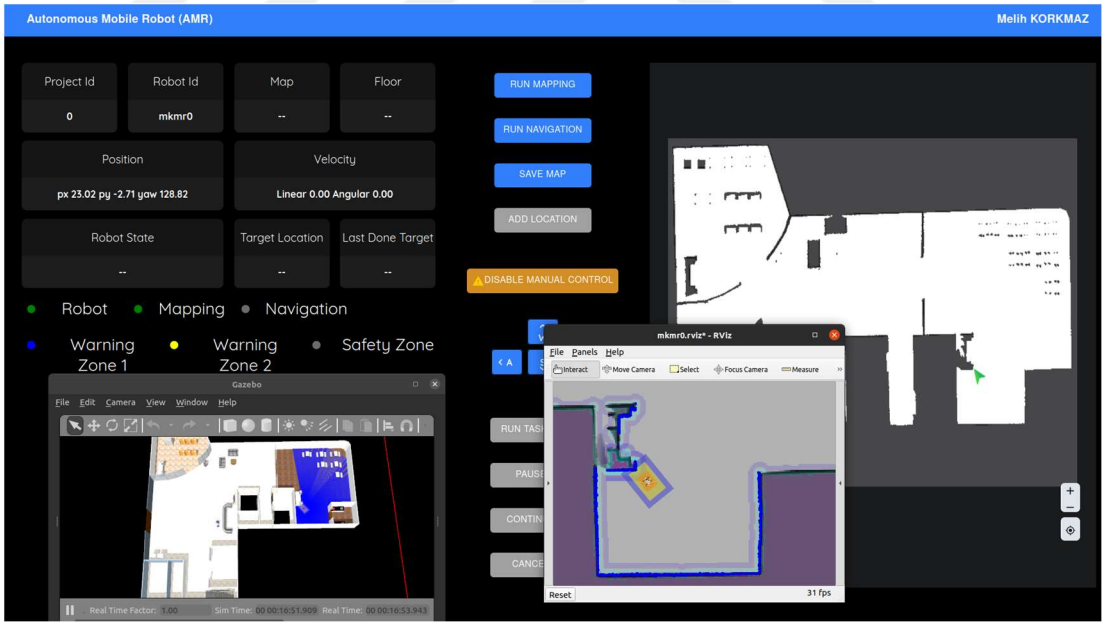


Şekil 5.6 : Haritalama başlangıcı gazebo ve RViz.



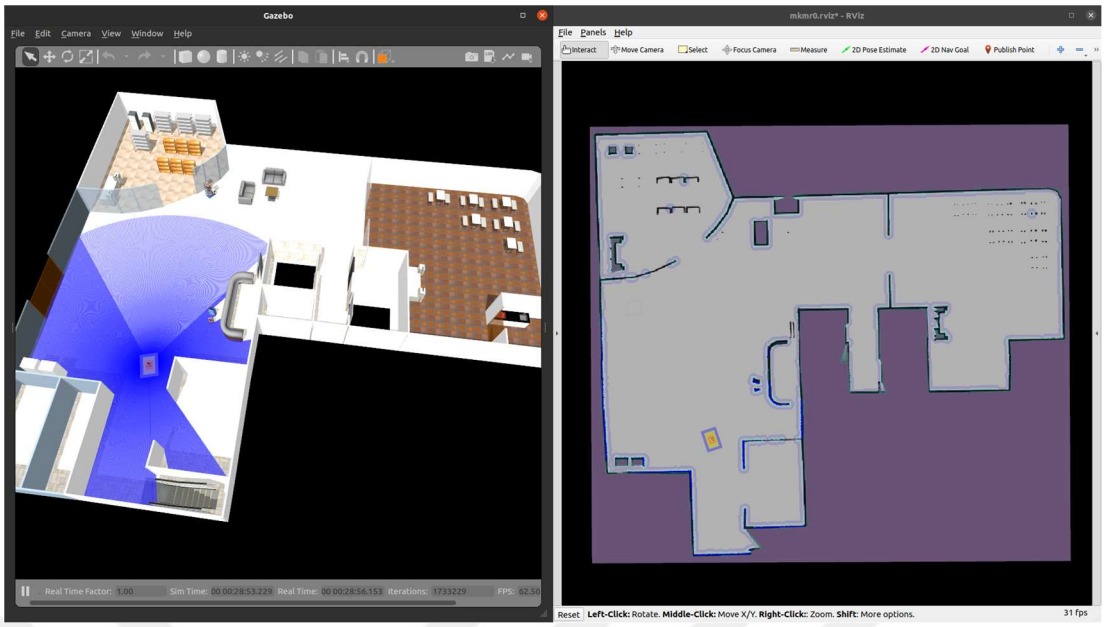
Şekil 5.7 : Haritalama başlangıcı arayüz.

Robot manuel gezdirilip haritalama işlemi yapılırken sabit bir nesneye yaklaştığından arayüzdeki güvenlik alan durum ledleri aktif hale geçmiştir (Şekil 5.8).



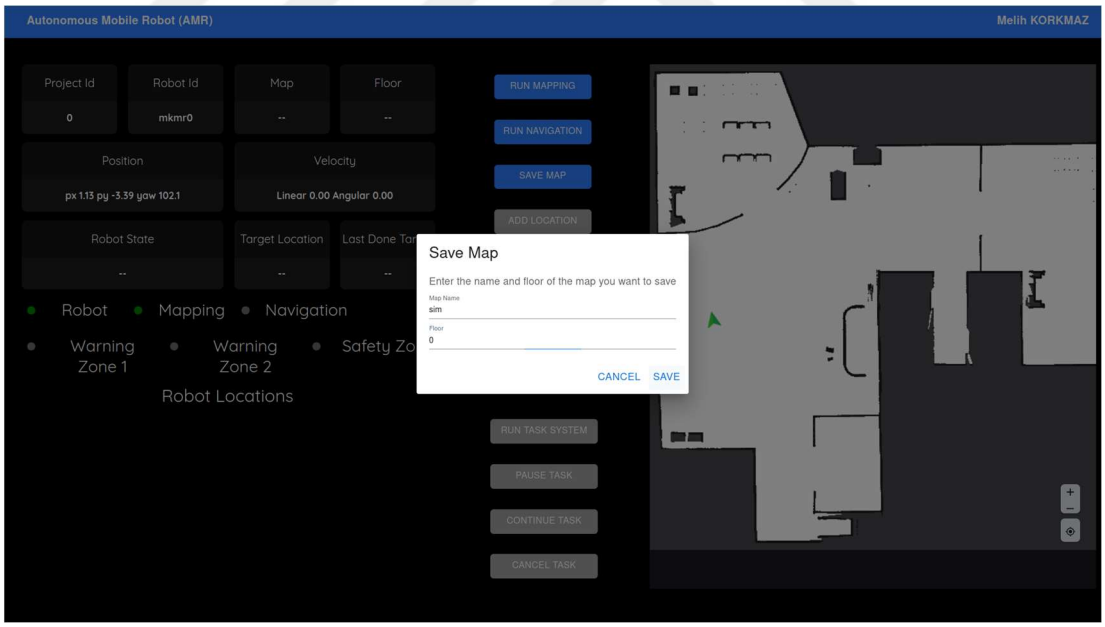
Şekil 5.8 : Haritalama esnasında robotun güvenli alanına nesne girmesi.

Haritalama işleminin sonucundaki gazebo ve RViz görüntüleri verilmiştir (Şekil 5.9).



Şekil 5.9 : Haritalama sonu gazebo ve RViz.

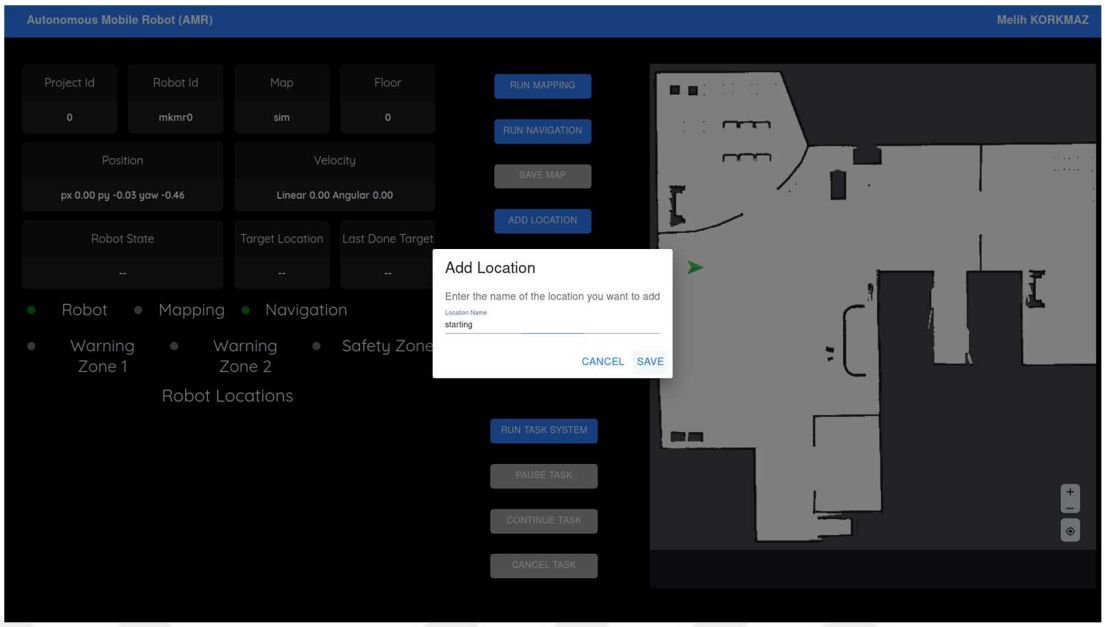
Haritalama işlemi bittikten sonra harita ve kat ismi verilerek harita kaydedilir (Şekil 5.10). Otomatik olarak kaydedilen harita ile navigasyon modu devreye girmektedir.



Şekil 5.10 : Arayüz harita kaydetme.

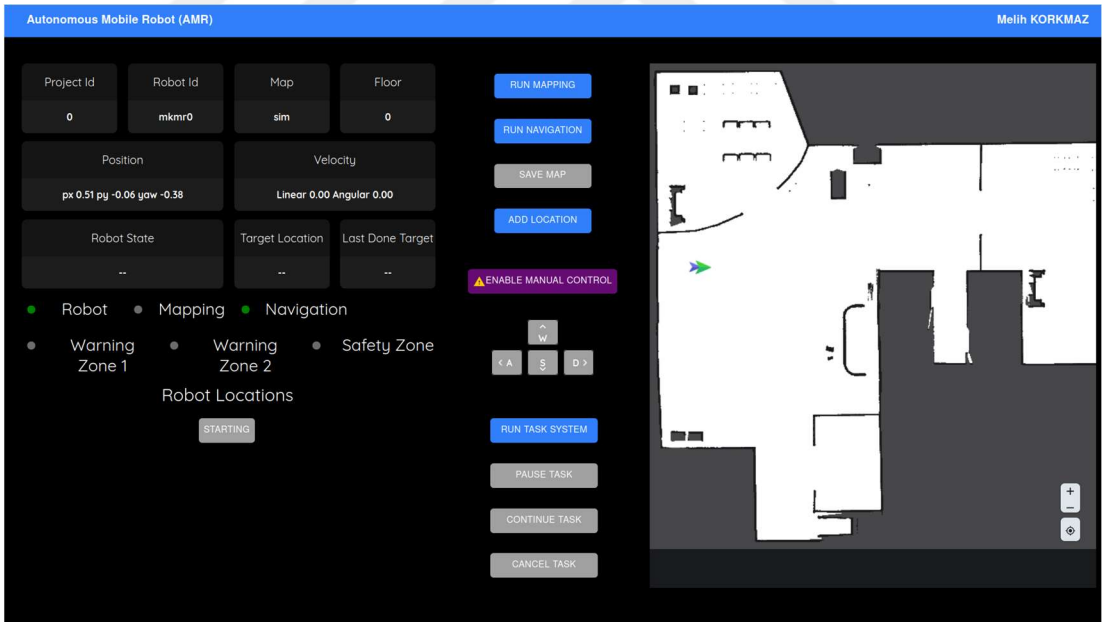
5.4 Konum Kaydetme

Kullanıcı arayüzünden konum ekle butonuna tıkladıktan sonra açılan pencerede konum ismi girilerek konum servisi çağrılmış olur (Şekil 5.11).



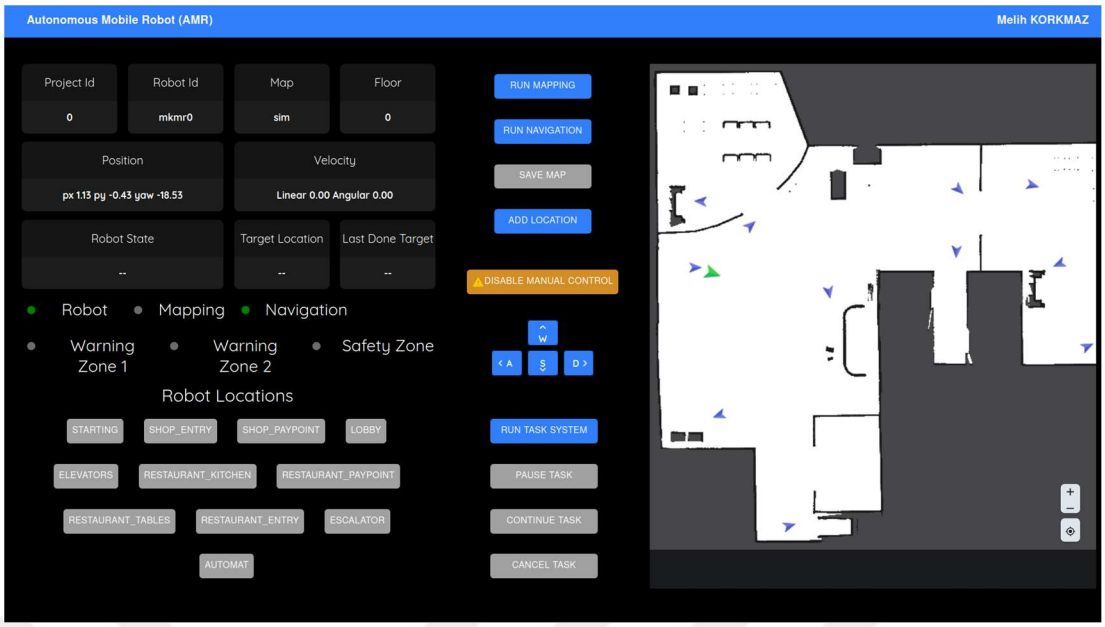
Şekil 5.11 : Arayüz konum kaydetme.

Robotun o anki çalışan harita ve konum bilgileri alınarak, kullanıcıdan gelen isim ile birlikte konum ekleme işlemi gerçekleşmiş olur (Şekil 5.12).



Şekil 5.12 : Arayüz ilk konum kaydetme sonrası.

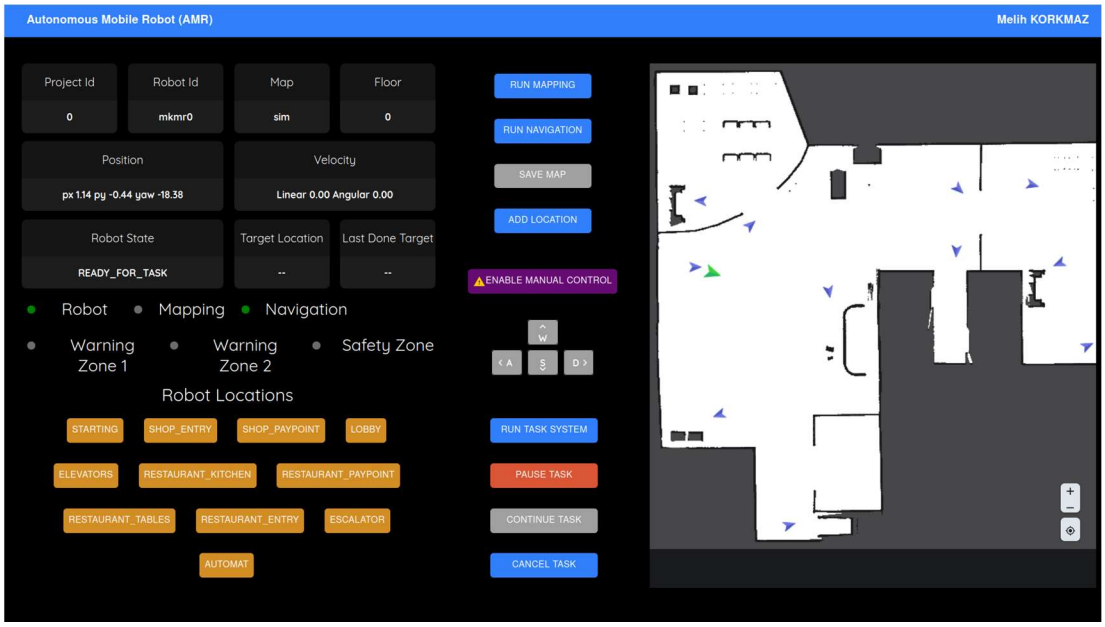
Konum ekleme işlemleri sonucundaki son durum şekilde verilmiştir (Şekil 5.13). Konum sistemi dinamik bir yapıda olduğundan otomatik olarak haritaya ikon olarak ve görev verilebilir konumların olduğu kısma gelir.



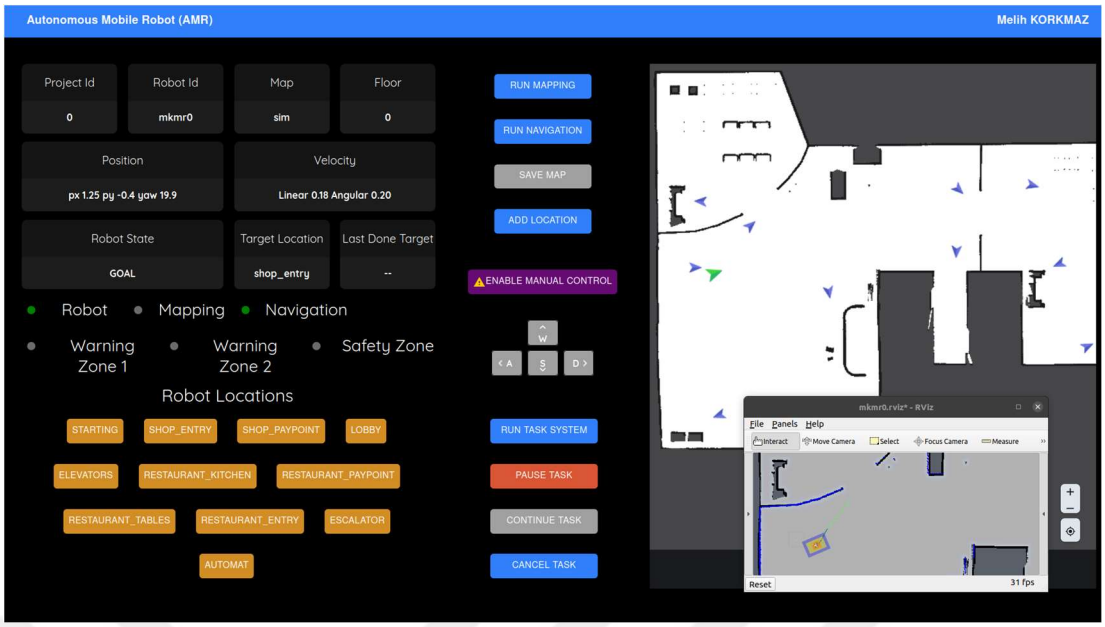
Şekil 5.13 : Arayüz tüm konumların kaydedilmiş hali.

5.5 Robota Görev Verme

Task yönetim sistemi çalıştırıldıktan sonraki ilk görsel verilmiştir (Şekil 5.14). Hedef konumun üstüne tıklanılarak görev verilebilmektedir (Şekil 5.15).

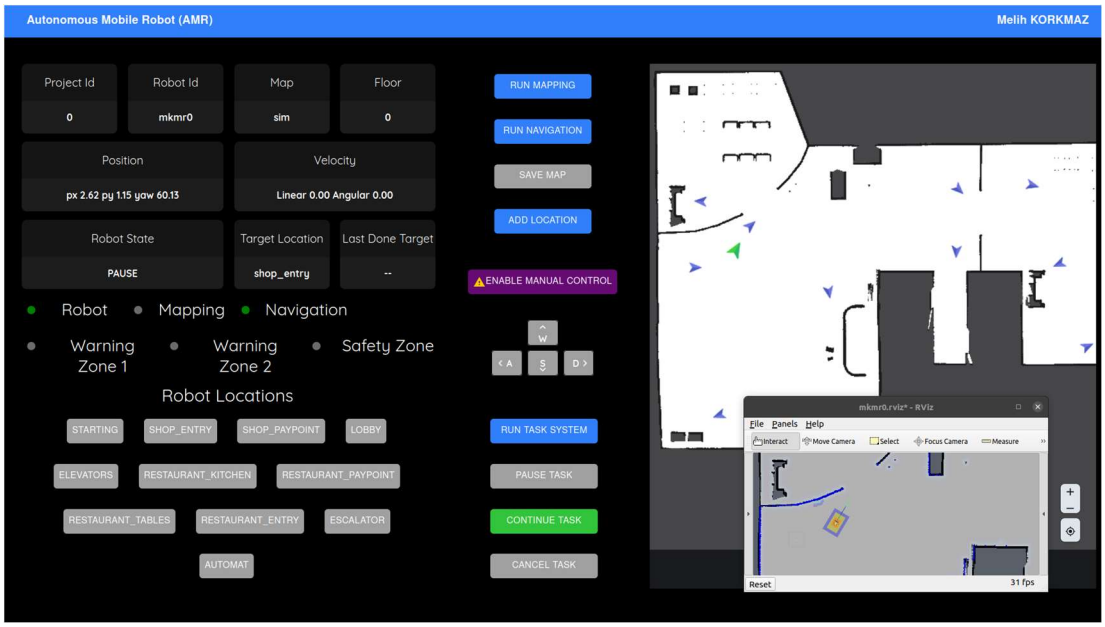


Şekil 5.14 : Arayüz görev yönetim sisteminin ilk çalıştırılmış hali.



Şekil 5.15 : Arayüz robota görev verilmiş görseli.

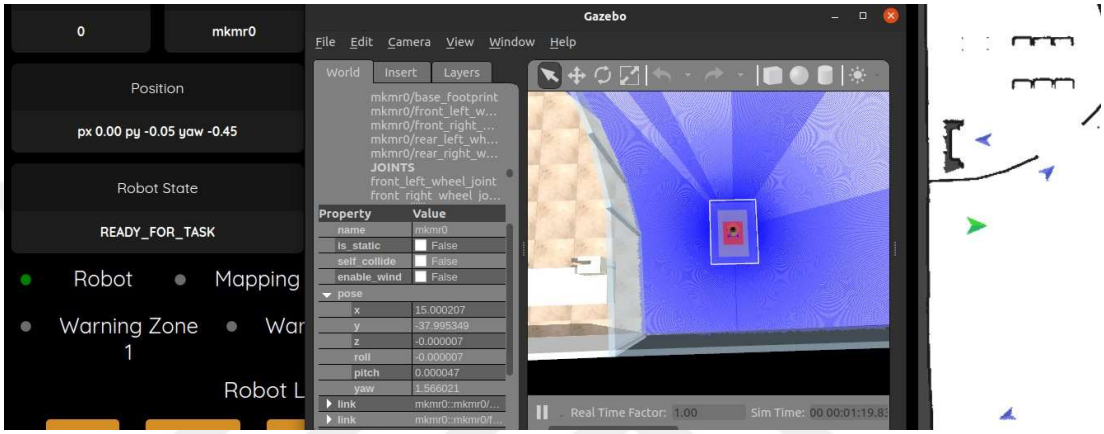
Daha sonra da yan kısımdaki butonlardan görev durdurulabilir (Şekil 5.16), devam ettirilebilir ve iptal edilebilir. Task yönetim sistemindeki robotun değişen durumu, hedef konumu ve son başarılı konumu arayüzden takip edilebilir.



Şekil 5.16 : Arayüz görevin durdurulmuş görseli.

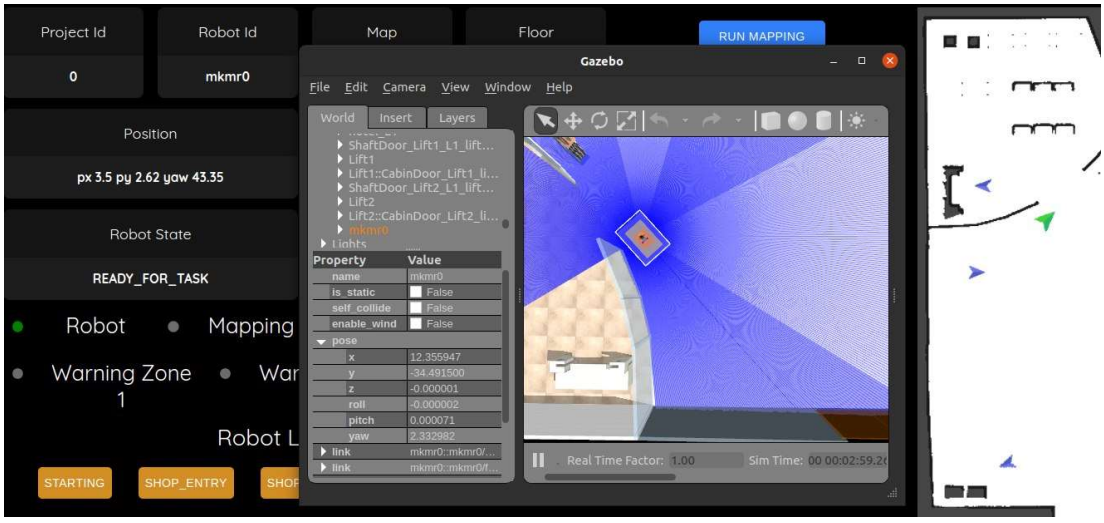
5.6 Simülasyon Sonuçları

Simülasyon ortamı olan gazebo da robotun konumu ve navigasyondaki konum karşılaştırılarak lokalizasyon hatası incelenmiştir. Gazebo dünyasının tasarımından dolayı eksenler arasında fark mevcuttur. X ve y eksenleri terstir. X ekseninde 15 m, y ekseninde ise -38 m offset değer farkı vardır. Robot hareket ettirilmeden önce gazebodaki değerler x: 15 y: -38 metredir. Navigasyon ise x: 0 y: -0,05 metredir (Şekil 5.17).



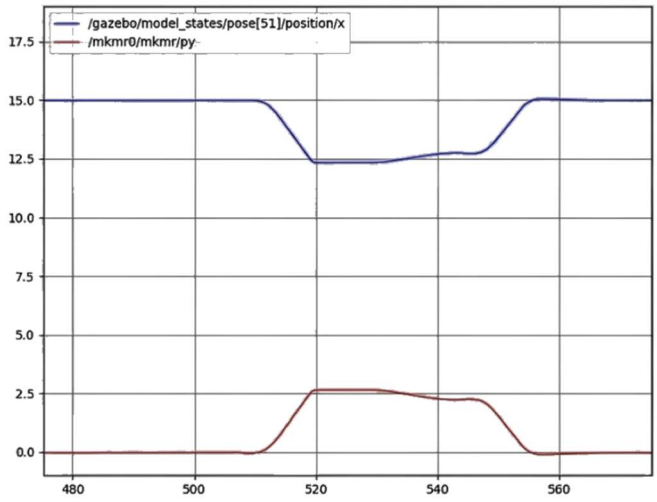
Şekil 5.17 : Simülasyonda x ve y ekseninde hareket öncesi.

Robot daha önceden kaydedilmiş konuma görev verilerek gönderilmiştir. Hareket sonrasındaki gazebodaki değerler x: 12.35 y: -34.49 metredir. Navigasyon ise x: 3.5 y: 2.62 metredir (Şekil 5.18).

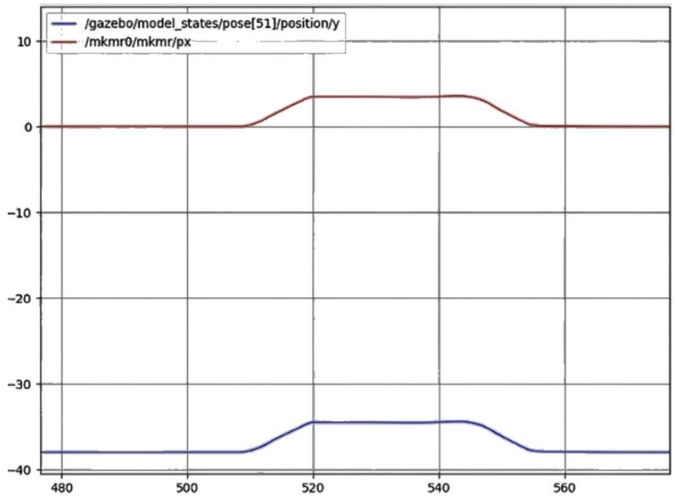


Şekil 5.18 : Simülasyonda x ve y ekseninde hareket sonrası.

Robota verilen görev süresince anlık konum değişimleri rostaki RQt aracı kullanılarak görselleştirilmiştir. Sol grafik navigasyonun y ekseni, sağdaki grafik ise x eksenidir. Y ekseninde yön farkı olduğundan simetriklik mevcuttur. Verilen görevden sonra robot tekrar başlangıç noktasına gönderilmiştir. Simülasyondaki x ekseni üst çizgi, navigasyondaki y ekseni ise alt çizgi olarak hareket değişimlerini içeren grafik Şekil 5.19'da verilmiştir. Simülasyondaki y ekseni alt çizgi, navigasyondaki x ekseni ise üst çizgi olarak hareket değişimlerini içeren grafik Şekil 5.20'de verilmiştir. Değerler incelendiğinde robot görev yerine gittiğinde x ekseninde 1 santimetre, y ekseninde 3 santimetre hata payı ile durmuştur.



Şekil 5.19 : Simülasyon (x) ve navigasyon (y) eksenindeki değişim grafiği

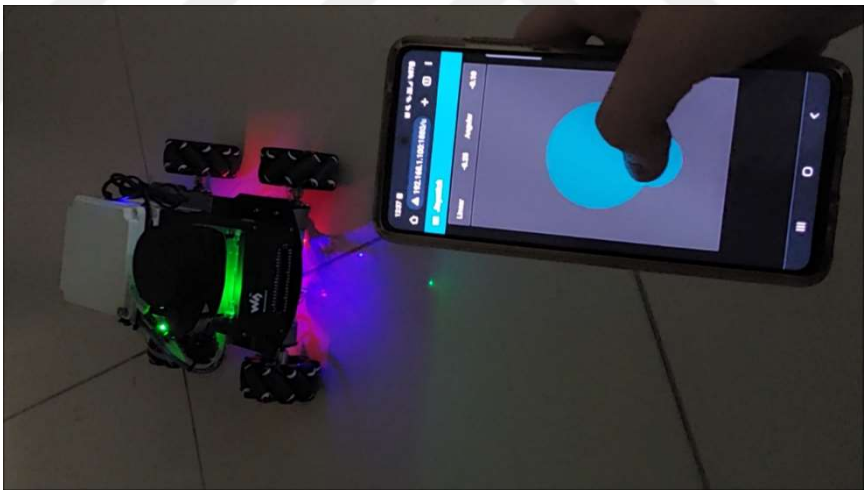


Şekil 5.20 : Simülasyon (y) ve navigasyon (x) eksenindeki değişim grafiği

Simülasyon ortamı için olan Gazebo ciddi oranda kaynak harcamaktadır. Performanslı sonuçların alınması için GPU, CPU ve RAM ortalamanın üstünde olmalıdır. Simülasyon için tam performans testi navigasyonun çalıştığı ve robotun otonom hedefine gittiği anda yapılmalıdır. Simülasyon dünyasının büyüklüğüne ve optimizasyonuna göre de sonuçlar değişebilir. Bu çalışmada 32 GB RAM, Intel i7 11. nesil CPU ve 6 GB Nvidia ekran kartına sahip bilgisayar kullanılmıştır. Çalışma esnasında ortalama olarak kullanım değerleri CPU %25 ve RAM %30 olarak gözlemlenmiştir. En çok kullanım gazebo paketlerinden olmaktadır.

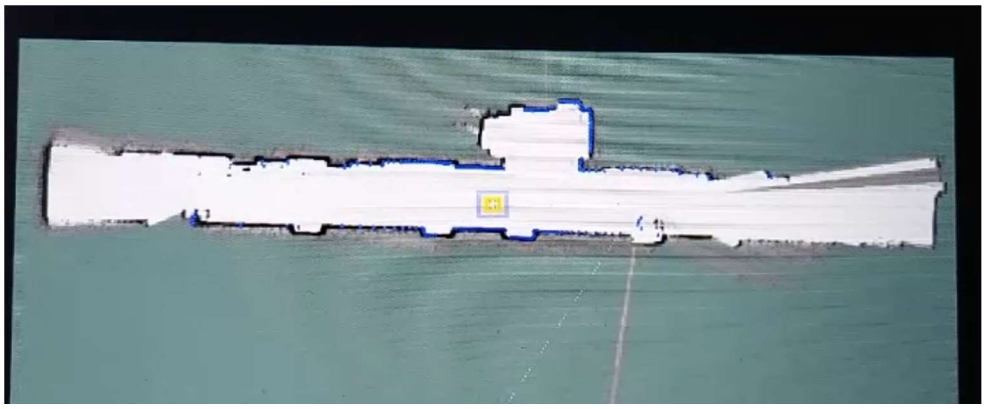
5.7 Robotun Gerçek Dünyada Çalıştırılması

Mekanik montajı, elektronik kablolaması ve tüm yazılım geliştirilmeleri yapılmış robot gerçek dünyada çalıştırılmıştır. Geliştirilen sanal kumanda ile robot başarılı bir şekilde sürülmüştür (Şekil 5.21). Daha sonra robot sanal kumanda ile sürülerek haritalama yapılıp, navigasyon modunda robota arayüz üzerinden görev verilmiş ve izlenmiştir.



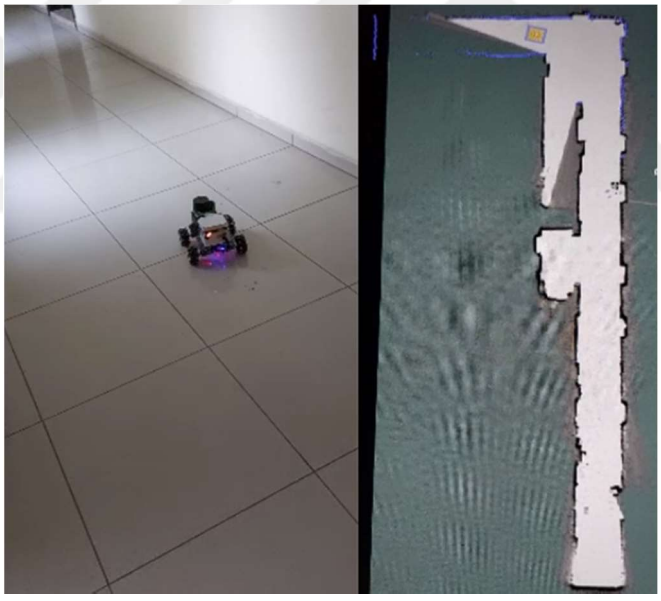
Şekil 5.21 : Robotun sanal kumanda ile sürülmesi.

Robot koridor şeklinde olan bir ortamda iken haritalama modu çalıştırılmıştır. İlk sonuçları Şekil 5.22'de verilmiştir.



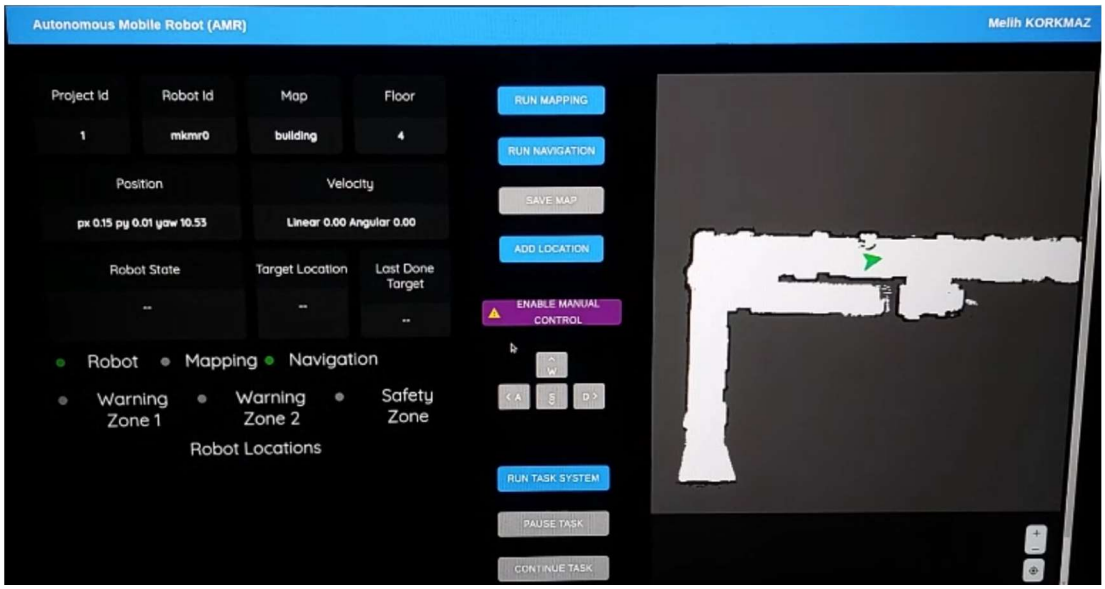
Şekil 5.22 : Gerçek robot ile haritalama başlangıcı.

Diğer bir koridora giriş esnasında robotun olduğu konum ve harita görseli verilmiştir. Kullanılan LIDAR sensörünün aralığının (12 metre) dışında kaldığından dolayı, olduğu koridorun sonuna kadar harita tamamlanmamıştır (Şekil 5.23). Robot ileri doğru sürüldükten sonra harita tamamlanmıştır.



Şekil 5.23 : Gerçek robot ile haritalamada yeni koridora giriş.

Çıkartılan harita arayüz üzerinden kaydedildikten sonra navigasyon moduna otomatik olarak geçilmiştir. Projenin numarası 1, haritanın adı building, robotun bulunduğu kat 4 olarak gerçek robot boş bir şekilde navigasyon modunda çalışmıştır (Şekil 5.24).

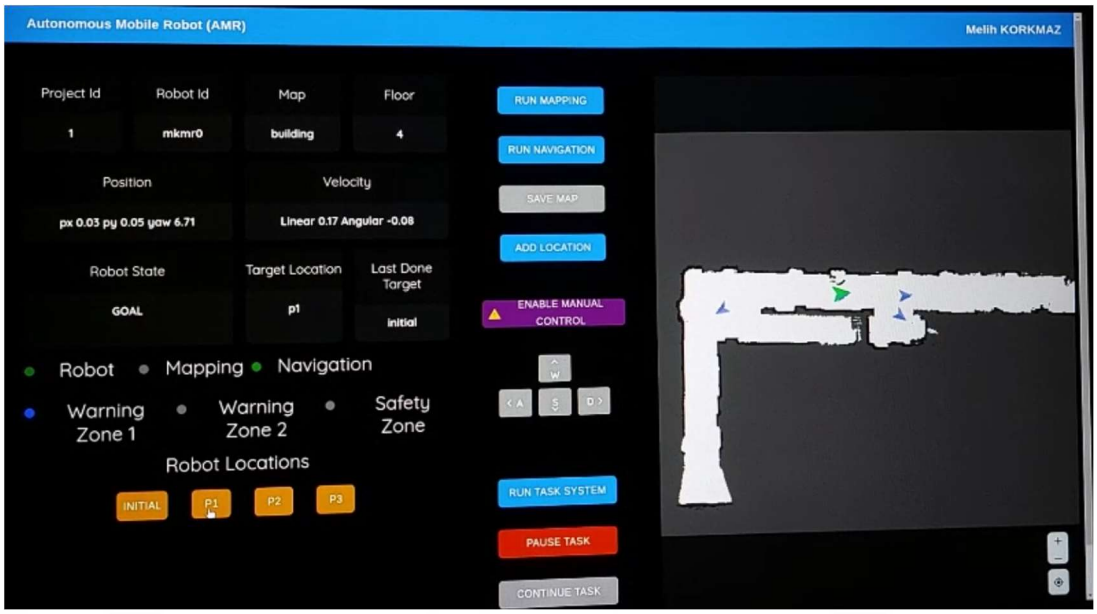


Şekil 5.24 : Gerçek robot navigasyon başlangıcı arayüz görseli.

Navigasyon modunda olan gerçek robotun RViz görüntüsü Şekil 5.25'te verilmiştir. Robot model, LIDAR güvenlik bölgeleri, LIDAR sensörü ve local ve global costmapler şekilde gözükmektedir.

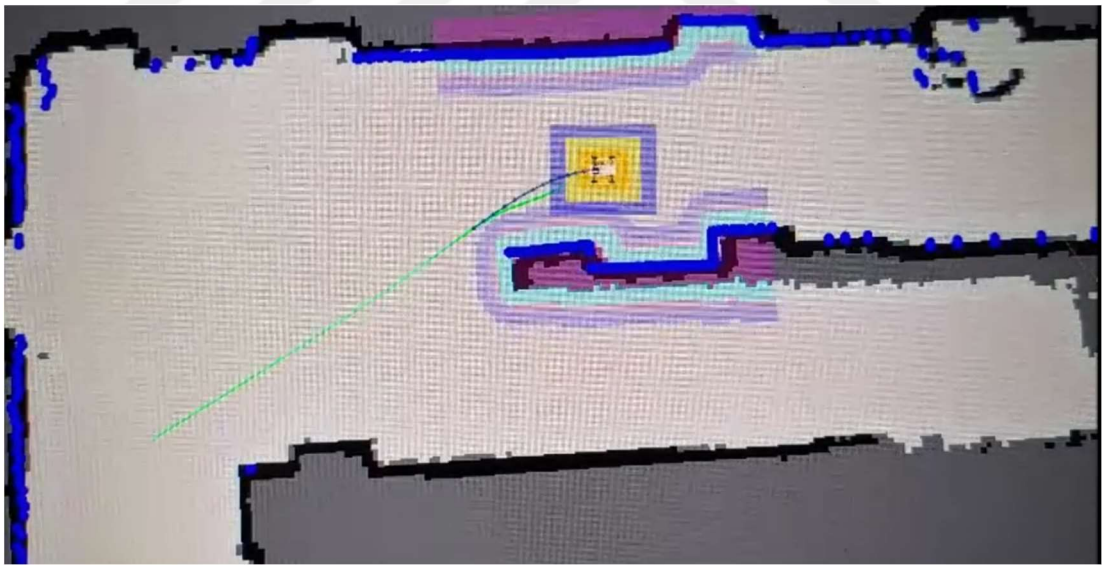
Şekil 5.25 : Gerçek robot navigasyon RViz görüntüsü.

Robotun sol tarafında engel olarak bir nesne durduğundan en dış LIDAR güvenlik bölgesinden sinyal alınmıştır, mavi led olarak gösterilmektedir. Bu yüzden robot hız limiti olarak bir alt limite geçmiştir. Haritanın başlangıç noktası ve 3 adet test konumu kaydedilmiştir. Görev yönetim sistemi çalıştırılmış ve robota P1 konumu için görev verilmiştir (Şekil 5.26) .



Şekil 5.26 : Gerçek robota görev verilmesi - arayüz görseli.

Global yol planlaması en kısa yol planlamasını yapmakta ama local yol planlamasına girilen hassas engel uzaklık parametreleri sayesinde robot dönemeçte daha açıktan almaktadır (Şekil 5.27).



Şekil 5.27 : Gerçek robota görev verilmesi - RViz görseli.

Simulasyon ve gerçek ortamdaki çalışmalar video olarak kaydedilmiştir [39]. Simülasyon ve gerçek ortamda bilgisayar gücüne göre farklılıklar olduğu gözlemlenmiştir, örneğin local costmap haritasının büyüklüğü. Ek olarak simülasyona göre gerçek robotta LIDAR sensörü verisinin ve odometri verisinin gürültülü olma ihtimali mevcuttur.

Gerçek ortamdaki bir alanda Şekil 5.27'deki gibi metresel olarak analiz çalışması yapılmıştır. Alandaki kapının genişliği ve kapının önündeki koridorun genişliği LIDAR'ın denk geldiği yükseklik hizasından ölçümlenmiştir. Çıkartılan haritanın piksellerine bakıldığında çizilen dikdörtgenin 17 x 35 olduğu gözlemlenmiştir (Şekil 5.28). Daha önceden bir pikselin yani karenin genişliğinin 0,05 metre yani 5 santimere olduğu belirtilmişti. Bu değere göre hesaplandığında kapı genişliğinin 0,85 m ve koridor genişliğinin 1.75 olduğu ortaya çıkmaktadır.

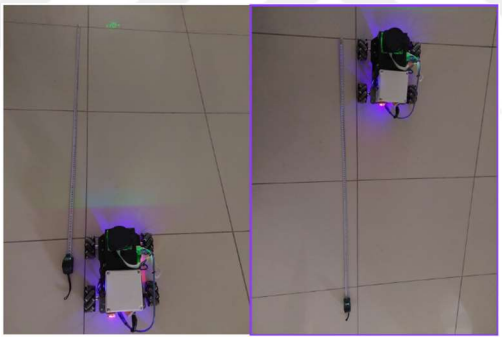
Şekil 5.28 : Haritadan piksel incelemesi.

Gerçek Alanda yapılan metre ile ölçümleme işlemlerinin sonuçları yaklaşık 0,83 m (Şekil 5.29) ve 1.73 m (Şekil 5.30) olarak çıkmaktadır. Hassasiyetin 0,05 m olduğu ölçümlenerek test edilmiştir.

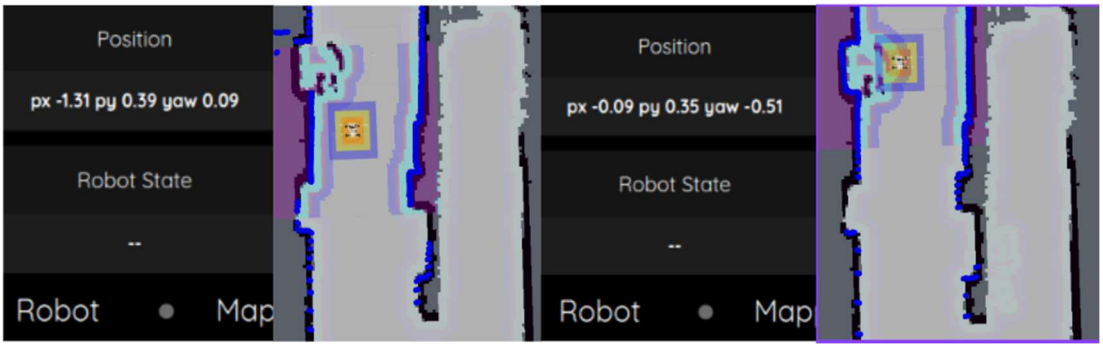
Şekil 5.29 : Gerçek ortamda kapının metre ile ölçülmesi.

Şekil 5.30 : Gerçek ortamda koridorun metre ile ölçülmesi.

Robot gerçek ortamda hakeret ettirilerek gerçek konum ve navigasyondaki konum karşılaştırılarak konum hassasiyeti analiz edilmiştir. Zemindeki fayanslar kare şeklinde ve ölçüsü 60 santimetredir. Robot x ekseninde 2 fayans 1.2 metre pozitif yönde hareket ettirilmiştir (Şekil 5.31). Navigasyonda ise x ekseni değeri -1.31 den -0,09 metreye çıkmıştır (Şekil 5.32). 1.22 metre yerdeğişi olmaktadır. Hata payı 2 cm olarak ölçülmüştür.



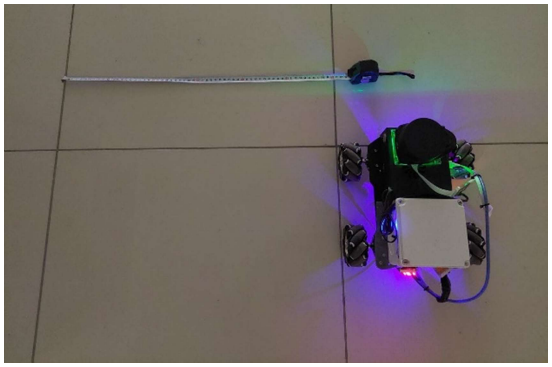
Şekil 5.31 : Gerçek ortamda x ekseninde hareket (robot).



Şekil 5.32 : Gerçek ortamda x ekseninde hareket (arayüz ve RViz).

Aynı işlemler hem x hem y ekseni içinde tekrarlanmıştır. Robot x ekseninde 1 fayans 0,6 metre pozitif, y ekseninde 1 fayans 0,6 metre negatif yönde hareket ettirilmiştir

(Şekil 5.33). Navigasyonda ise x ekseni değeri son olan -0,09 dan 0,51 metreye çıkmıştır, y ekseninde ise 0,35 ten -0,28 metreye düşmüştür (Şekil 5.34). X ekseninde değişim 0,6 m hata payı 0 m, y ekseninde ise değişim 0,63 m hata payı 3 santimetredir.



Şekil 5.33 : Gerçek ortamda y ekseninde hareket (robot).

Şekil 5.34 : Gerçek ortamda y ekseninde hareket (arayüz ve RViz).

Gerçek robotun otonom bir şekilde göreve giderken alınan performans değerleri incelenmiştir. Robotun bilgisayarında 4 GB RAM, çift çekirdek CPU ve 128 çekirdek Maxwell Nvidia ekran kartına mevcuttur. Çalışma esnasında ortalama olarak kullanım değerleri CPU %60 ve RAM %40 olarak gözlemlenmiştir. En çok kullanım navigasyon paketlerinden olmaktadır.

6. SONUÇ VE ÖNERİLER

ROS platformu hızla gelişmekte olup ve desteklediği marka ve proje sayısı giderek artmaktadır. ROS'ta elinizde gerçek robot yoksa robot ve dünyanın simüle edildiği Gazebo, çıktılarının simüle edildiği RViz ortamı kullanılır. Google'un desteklemiş olduğu cartographer SLAM algoritması sayesinde istenilen her ortamda hızlıca harita çıkartılabilir. Ek olarak çıkartılan harita robota bağımlı olmadığından istenilen her robotta kullanılabilme esnekliğini sağlar.

Çıkartılan harita kullanılarak ROS'un sağlamış olduğu navigasyon fonksiyonu çalıştırıldığında doğal navigasyon sayesinde ortamda hiçbir belirteç olmadan robot konumlama işlemini yapabilmektedir. Hareketli engelleri de dikkate alarak dinamik, maliyeti en düşük yol planlamasını çıkartıp hedefe ulaşması AMR'leri ön plana çıkarmaktadır.

Oluşturulan gerçekçi simülasyon ortamı sayesinde robot, tüm bileşenleri ve geliştirilen yazılımlar ile birlikte, SLAM ve navigasyon fonksiyonlarındaki parametreler hassas bir şekilde optimize edilmiştir.

Robotun içerisinde geliştirilen web-socket tabanlı arkayüz yazılımı ile robot dış kaynaklara yönetilmek ve izlenmek için açılmıştır. Robot bu imkan ile merkezi/akıllı sistemler ile haberleşip, süreçlere entegre olabilir.

Bu projede geliştirilen ön arayüz yazılımı web-socket altyapısı ile robotla haberleşmektedir. Arayüz ile ortamın haritalaması yapılıp hedef konumlar kaydedilebilir. Geliştirilen görev yönetim sistemi ile görevler verilip robot otonom bir şekilde çalıştırılabilir ve yönetilebilir. Ek olarak anlık robotun birçok verisi izlenebilir.

Robotun fiziksel özelliklerine, kullanılan sensörlerine, bulunduğu ortama göre yazılımda ve ilgili parametrelerde değişiklik yapılarak daha ideal çalışma durumu sağlanabilir.

Merkezi bir akıllı yönetim sistemi yapılarak binalardaki kapı ve asansörler ile entegre olunabilir. Endüstri 4.0 senaryosunda makineler ile haberleşilip görevleri makineler ve üretim hatları verebilir. Robotlar merkezi yönetim sisteminden trafik kontrol ile yönetilebilir. İstenilen zaman aralığında ve dosya formatında raporlama sistemi devreye alınabilir.

KAYNAKLAR

[1] **Quigley M.** (2009). ROS : an open-source Robot Operating System.

[1] **Sharma V. ve Bora K. S.** (2021). Development and Control of Modular 5 DoF Robotic Arm using ROS.

[3] **Natu A., Garg V. ve Gaur P.** (2020) Design and Development of an Autonomous Underwater Vehicle VARUNA 2.0

[4] **Janavi K ve Teja R.** (2022) Robot Operating Systems (ROS): The Fundamentals of ROS and Its Remarkable Performances in the World of Drones, Int. J. Res. Appl. Sci. Eng. Technol., vol. 10, no. 9, pp. 1844–1849, 2022, doi: 10.22214/ijraset.2022.46938

[5] **Lleras N. O., Brennan S., Murphy D., Jennifer Klena M., Garvey P. M. ve Sommer H. J.** (2016) Development of an open-source tractor driving simulator for tractor stability tests, *J. Agric. Saf. Health*, vol. 22, no. 4, pp. 227–246, 2016, doi: 10.13031/jash.22.11774

[6] **Baker W., Kingston Z., Moll M., Badger J. ve Kavraki L. E.** (2017). Robonaut 2 and you: Specifying and executing complex operations, *Proc. IEEE Work. Adv. Robot. its Soc. Impacts, ARSO*, 2017, doi: 10.1109/ARSO.2017.8025204.

[7] **Gangadharaswamy L. B.** (2019). Design of Autonomous Cleaning Robot, Yüksek Lisans Tezi no. Aralık, 2019, [Online]. Available: <https://trepo.tuni.fi/bitstream/handle/10024/118774/BangaloreGangadharaswamyLakshmi.pdf?sequence=2&isAllowed=y>.

[8] **Hercik R., Byrtus R., Jaros R. ve Koziorek J.** (2022). Implementation of Autonomous Mobile Robot in SmartFactory, *Appl. Sci.*, vol. 12, no. 17, 2022, doi: 10.3390/app12178912.

[9] **Christian Tamantini F. C., Francesco Scotto di Luzio, Giuseppe Pascarella L. Z. ve Felice Eugenio Agrò**. (2021). A Robotic Health-Care Assistant for the COVID-19 Emergency, *IEEE Robot. Autom. Mag.*, no. March, pp. 71–81, 2021.

[10] **Fragapane G., Ivanov D., Peron M., Sgarbossa F. ve Strandhagen J. O.** (2020). Increasing flexibility and productivity in Industry 4.0 production networks with autonomous mobile robots and smart intralogistics, *Ann. Oper. Res.*, 2020, doi: 10.1007/s10479-020-03526-7.

[11] **Bačík J.** (2020). Phollower—the universal autonomous mobile robot for industry and civil environments with COVID-19 germicide addon meeting safety requirements, *Appl. Sci.*, vol. 10, no. 21, pp. 1–16, 2020, doi: 10.3390/app10217682.

[12] **Fragapane G. , de Koster R., Sgarbossa F. ve Strandhagen J. O.** (2021). Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda, *Eur. J. Oper. Res.*, no. xxxx, 2021, doi: 10.1016/j.ejor.2021.01.019.

[13] **Raje S. D.** (2020). Evaluation of ROS and Gazebo Simulation Environment using TurtleBot3 robot.

[14] **Pramod Thale S., Mangesh Prabhu P., Vinod Thakur P. ve Kadam P.** (2020). ROS based SLAM implementation for Autonomous navigation using Turtlebot, *ITM Web Conf.*, vol. 32, p. 01011, 2020, doi: 10.1051/itmconf/20203201011.

[15] **Url-1 <***https://www.youtube.com/watch?v=GpsnmbaYQpU*>, MiR 2.0 Interface, erişim tarihi 5.03.2023.

[16] **Cheng Y.-Y. ve Shi D.-X.** (2017). Ros Based Remote Robot Task Monitoring and Control System, *ITM Web Conf.*, vol. 12, p. 01023, 2017, doi: 10.1051/itmconf/20171201023.

[17] **Ivanov A., Zakiev A., Tsoy T. ve Hsia K. H.** (2021). Online Monitoring and Visualization with ROS and ReactJS, SIBCON 2021 - Int. Sib. Conf. Control Commun., no. June, 2021, doi: 10.1109/SIBCON50419.2021.9438890.

[18] **Zea A. ve Hanebeck U. D.** (2021). iviz: A ROS visualization app for mobile devices, *Softw. Impacts*, vol. 8, no. August, 2021, doi: 10.1016/j.simpa.2021.100057.

[19] **Kapić Z., Crnkić A., Mujčić E. ve Hamzabegović J.** (2021). A web application for remote control of ROS robot based on WebSocket protocol and Django development environment, *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1208, no. 1, p. 012035, 2021, doi: 10.1088/1757-899x/1208/1/012035.

[20] **Thanh P. N., Hung N. ve Ha Q. T. N.** (2023). Visual application of navigation framework in cyber-physical system for mobile robot to prevent disease, International Journal of Advanced Robotic Systems, March-April 2023, 2023, doi: 17298806231162202.

[21] **Url-2** < https://google-cartographer-ros.readthedocs.io/en/latest/>, The Cartographer Authors Cartographer ROS documentation, erişim tarihi 11.04.2023.

[22] **Url-3 <***http://wiki.ros.org/navigation*>, navigation - ROS Wiki, erişim tarihi 5.03.2023.

[23] **Url-4 <** https://github.com/mkmr-software/Construction-and-Management-of-ROS-Based-Mobile-Autonomous-Robot >, ROS Tabanlı Mobil Otonom Robotun Yapımı ve Yönetimi projesinin dosyaları, erişim tarihi 12.04.2023.

[24] **Joseph L.** (2015). Mastering ROS for Robotics Programming, vol. 64, no. 6.

[25] **Fairchild C. ve Harman T. L.** (2016). ROS Robotics By ExamFairchild, C., & Harman, T. L. (2016). ROS Robotics By Example.ple.

[26] **Granát M.** (2018). Mobile robot control and guidance using computer vision, BRNO Univ. Technol.

[27] **Lee K., Jung C. ve Chung W.** (2011). Accurate calibration of kinematic parameters for two wheel differential mobile robots, *J. Mech. Sci. Technol.*, vol. 25, no. 6, pp. 1603–1611, 2011, doi: 10.1007/s12206-011-0334-y.

[28] **Url-5 <***https://www.amazon.in/Moebius-Mecanum-Chassis-Encoder-Raspberry/dp/B07VSW8VTB*>, Moebius 10KG Load 4WD 80mm Mecanum Wheel Robot Car Chassis Kit with DC 12V Encoder Motor for Arduino Raspberry Pi DIY Project STEM Toy : Amazon.in: Industrial & Scientific, erişim tarihi 3.05.2023.

[29] **Url-6 <***https://en.wikipedia.org/wiki/H-bridge*>, H-bridge - Wikipedia, erişim tarihi 05.03.2023.

[30] **Pyo Y., Cho H., Ryuwoon J. ve Lim T.** (2017). Robot Programming From The Basic Concept To Practical Programming and Robot Application.

[31] **SHANGHAI SLAMTEC CO., LTD.** (2019) RPLIDAR A1M8-R5 Low Cost 360 Degree Laser Range Scanner Introduction and Datasheet.

[32] **Martinez A. ve Fernández E.** (2013). Learning ROS for Robotics Programming.

[33] **Morgan Q., Brian G. ve William D.** (2015) Smart Programming Robots with ROS.

[34] **Url-7 <***https://www.ros.org/blog/ecosystem/*>, ROS: The ROS Ecosystem, erişim tarihi 5.03.2023.

[35] **Ebleme Ali** (2019). Nesnelerin interneti uygulama katmanı haberleşme protokollerinin başarım analizi, Yüksek Lisans Tezi.

[36] **Bruno K. ve Oussama Siciliano** (2008). Handbook Springer of Robotics.

[37] **Hess W., Kohler D., Rapp H. ve Andor D**. (2016). Real-time loop closure in 2D LIDAR SLAM, Proc. - IEEE Int. Conf. Robot. Autom. vol. 2016-June, pp. 1271–1278, doi: 10.1109/ICRA.2016.7487258.

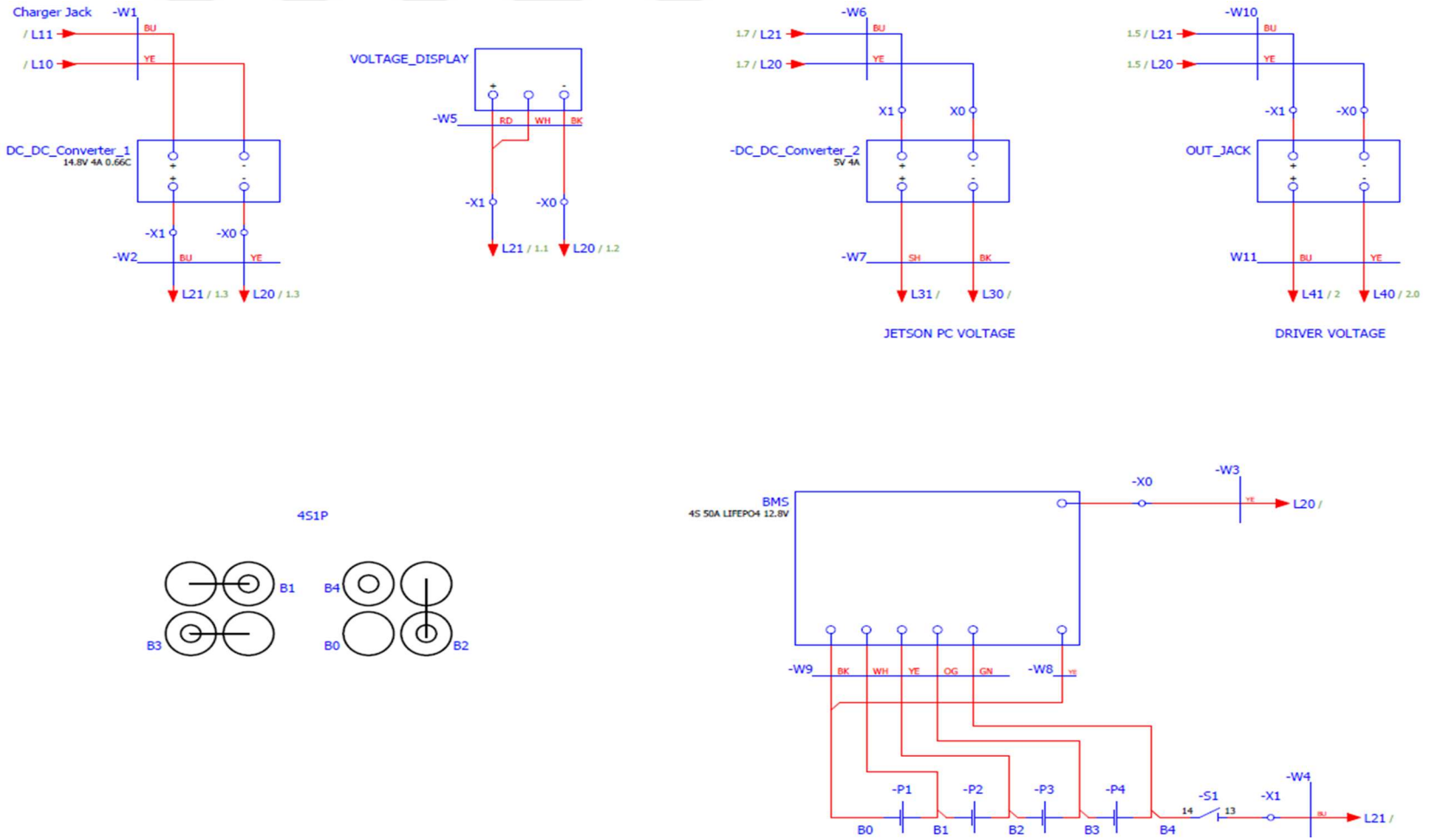
[38] **Guimarães R. L., de Oliveira A. S., Fabro J. A., Becker T. ve Brenner V. A.** (2016). ROS navigation: Concepts and tutorial, Stud. Comput. Intell., vol. 625, no. February, pp. 121–160, doi: 10.1007/978-3-319-26054-9_6.

[39] **Url-8 <***https://www.youtube.com/watch?v=H-SRVYuVXGc*>, (mkmr) CONSTRUCTION and MANAGEMENT of ROS BASED MOBILE AUTONOMOUS ROBOT, erişim tarihi 16.01.2023.

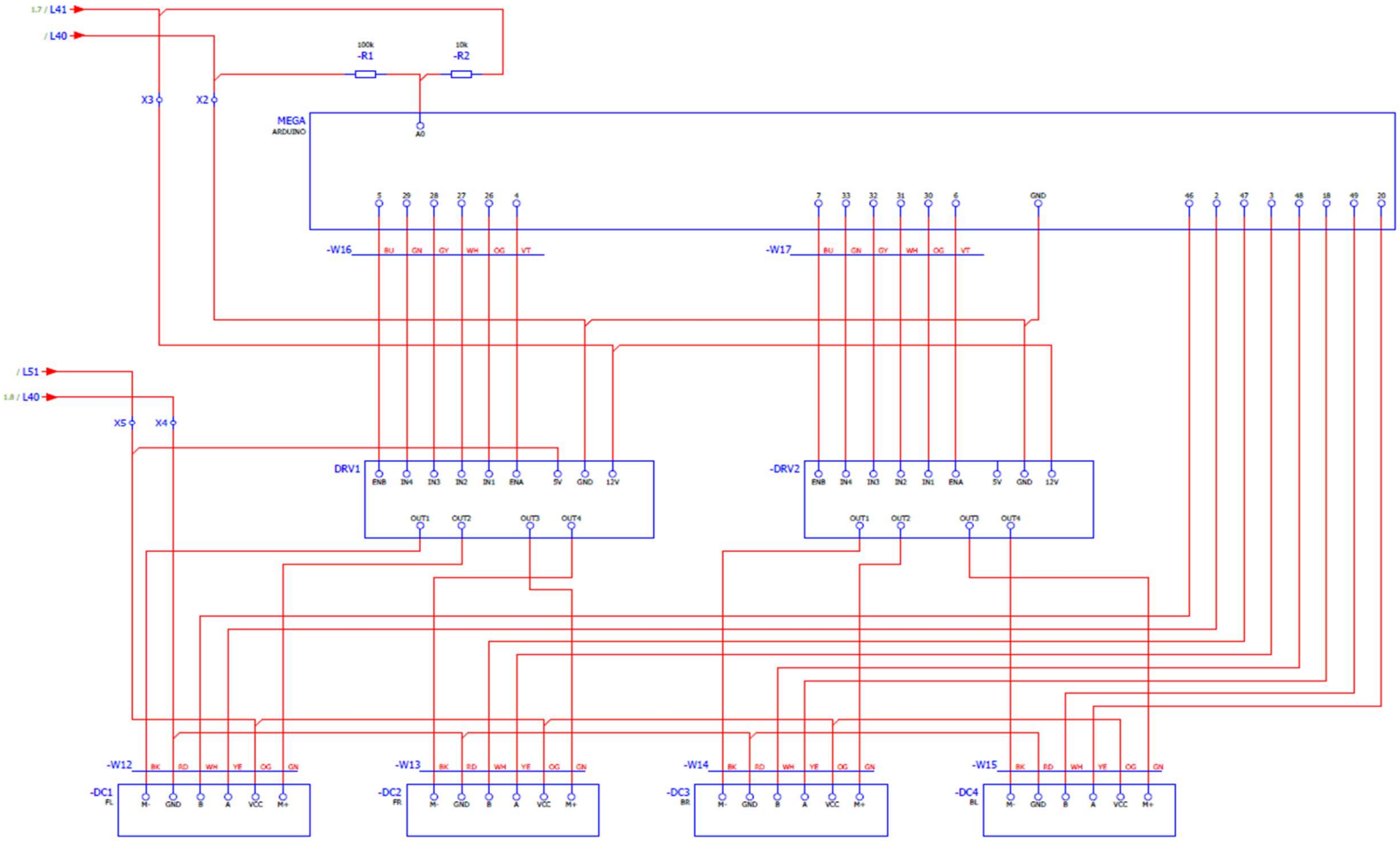
EKLER

EK A: Elektrik Şeması

EK A

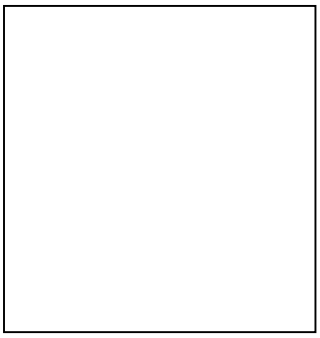


Şekil A.1 : Güç ve batarya elektrik şeması.



Şekil A.2 : Mikrodenetleyici ve motor sürücü elektrik şeması.

ÖZGEÇMİŞ



Ad-Soyad **:** Melih KORKMAZ

ÖĞRENİM DURUMU:

- **Lisans** **:** 2019, Bursa Teknik Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi, Mekatronik Mühendisliği

MESLEKİ DENEYİM VE ÖDÜLLER:

- Bursa Teknik Üniversitesi Mekatronik Mühendisliği Bölüm Birinciliği , Haziran 2019
- Proje ve Mekatronik Mühendisi, Bilgin Mühendislik, 2019 - 2021
- Robotik Mühendisi, Saha Robotik ve Teslimat Teknolojileri, 2022 - Halen

TEZDEN TÜRETİLEN ESERLER, SUNUMLAR VE PATENTLER:

- ROS Tabanlı Mobil Otonom Robot (AMR) Geliştirme, Uluslararası Katılımlı Mekatronik Öğrenci Konferansı (MEKON), 25 Haziran 2021 (https://github.com/mkmr-software/ROS-Tabanli-Mobil-Otonom-Robot-AMR-Gelistirme)

DİĞER ESERLER, SUNUMLAR VE PATENTLER:

- Windows IoT Cihazı ile Akılı Sistem Tasarımı , Lisans Bitirme , Mayıs 2019