T.C.
İNÖNÜ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

LAZER MESAFE ÖLÇER İLE OTONOM MOBİL ROBOTLAR İÇİN NAVİGASYON PLANLAMASI

ORKAN MURAT ÇELİK

YÜKSEK LİSANS TEZİ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

OCAK 2018

Tezin Başlığı : **Lazer Mesafe Ölçer İle Otonom Mobil Robotlar İçin Navigasyon Planlaması**

Tezi Hazırlayan : **Orkan Murat ÇELİK**

Sınav Tarihi : **19/01/2018**

Yukarıda adı geçen tez jürimizce değerlendirilerek Elektrik-Elektronik Mühendisliği Ana Bilim Dalında Yüksek Lisans Tezi olarak kabul edilmiştir.

Sınav Jüri Üyeleri:

Tez Danışmanı:	**Yrd. Doç. Dr. Murat KÖSEOĞLU** İnönü Üniversitesi	……..
	Prof. Dr. Müslüm ARKAN İnönü Üniversitesi	……..
	Doç. Dr. Ulaş EMİNOĞLU Gaziosmanpaşa Üniversitesi	……...

Prof. Dr. Halil İbrahim ADIGÜZEL
Enstitü Müdürü

ONUR SÖZÜ

Yüksek Lisans Tezi olarak sunduğum “Lazer Mesafe Ölçer ile Otonom Mobil Robotlar İçin Navigasyon Planlaması” başlıklı bu çalışmanın, bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurmaksızın tarafımdan yazıldığını ve yararlandığım bütün kaynakların, hem metin içinde hem de kaynakçada yöntemine uygun biçimde gösterilenlerden oluştuğunu belirtir, bunu onurumla doğrularım.

Orkan Murat ÇELİK

ÖZET

Yüksek Lisans Tezi

LAZER MESAFE ÖLÇER ile OTONOM MOBİL ROBOTLAR İÇİN NAVİGASYON PLANLAMASI

Orkan Murat ÇELİK

İnönü Üniversitesi

Fen Bilimleri Enstitüsü

Elektrik-Elektronik Mühendisliği Anabilim Dalı

XI + 80 Sayfa

2018

Danışman: Yrd. Doç. Dr. Murat Köseoğlu

Mobil robotlar gün geçtikçe gelişen teknolojileri ve giderek artan kullanım alanları ile robotik biliminin ayrılmaz bir parçasıdır. Artan kullanım alanlarına paralel olarak mobil robotlar daha akıllı hale getirilmekte ve otonom hareket edebilme kabiliyetleri ivmelenen bir şekilde geliştirilmektedir. Mobil bir robot için otonom olmanın temel koşullarından birisi, robotun bulunduğu ortamda kullanıcıdan bağımsız olarak navigasyonunu kendi kendine planlayıp icra edebilmesidir.

Mobil bir robot tarafından navigasyon planının doğru hazırlanması ve uygulanması için en önemli şartlardan biri, robotun bulunduğu fiziksel ortamı ve ortam parametrelerini doğru algılaması, buna bağlı olarak bulunduğu ortamı gerçeğe en yakın biçimde modellemesidir.

Bu tezde, mobil robotların üzerine yerleştirilebilen bir lazer mesafe ölçüm sensörü kullanılarak elde edilen veriler yardımıyla robotun bulunduğu ortamı haritalaması ve harita üzerinde kendini konumlaması sağlanmıştır. Ayrıca oluşturulan bu harita üzerinde kullanıcı tarafından verilen bir noktaya robotun kendi navigasyon planını yaparak kullanıcıdan bağımsız şekilde ve en kısa yolu izleyerek ulaşması sağlanmıştır.

Bu çalışma için belirli algoritmalar geliştirilmiş, bu algoritmaların koşturulacağı elektromekanik donanım üretilmiş, gerçek ortamda, gerçek zamanlı bilinen bir düzlemde test edilmiştir. Böylelikle nispeten daha uygun bir maaliyetle otonom mobil bir robot platformu geliştirilmiştir.

ANAHTAR KELİMELER: Otonom mobil robot, lidar, navigasyon, ROS, SLAM

ABSTRACT

MSc Thesis

NAVIGATION PLANNING FOR AN AUTONOMOUS MOBILE ROBOT BASED ON LASER RANGEFINDER

Orkan Murat ÇELİK

Inonu University
Graduate School of Natural and Applied Sciences
Department of Electrical-Electronics Engineering

XI + 80 Pages

2018

Supervisor: Assist. Prof. Dr. Murat KÖSEOĞLU

Mobile robots are inseparable parts of the robotic science due to their technologies developing day by day and ever-increasing area of utilisation. In parallel with their increasing area of usage, the mobile robots become smarter, and their autonomous moving capabilities are developed swiftly. One of the basic conditions for a mobile robot to be autonomous is the proper fulfilment of navigating by itself independent of the user in the current medium.

One of the most important requirements for the preparation and implementation of the navigation plan by a mobile robot is the accurate perception of the physical medium and medium parameters, and real-like modelling of the medium depending on these parameters.

In this thesis, the production of a map of the current environment by the robot and positioning itself on this map were performed with the aid of a laser distance measurement sensor which can be located on the mobile robots. In addition, the realization of the arrival of the robot to a point defined by the user on the produced map was provided by following the optimal path making its own navigation plan and independently of the user.

In this study, several algorithms were developed, an electromechanical hardware to run these algorithms was built, and this hardware was tested in real time in a real medium. So, an autonomous mobile robot platform was developed with a relatively cost-effective budget.

KEYWORDS: Autonomous mobile robots, lidar, navigation, ROS, SLAM

TEŞEKKÜR

Bu çalışma sırasında, bilgisinden ve tecrübesinden yararlandığım değerli tez danışman hocam Yrd. Doç. Dr. Murat KÖSEOĞLU başta olmak üzere tüm hocalarıma, çalışmaya başladığım günden beri bana desteğini esirgemeyen HAVELSAN Hava Elektronik Sanayi ve Ticaret AŞ.'deki mesai arkadaşlarıma ve son olarak beni hiç bir konuda yalnız bırakmayan sevgili aileme teşekkür ederim.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER ve KISALTMALAR

ROS	Robot İşletim Sistemi (Robot Operating System)
AGV	Otomatik Güdümlü Araç (Autonomous Guided Vehicle)
AMR	Otonom Mobil Robot (Autonomous Mobile Robot)
ARM	İleri RISC makineler (Advanced RISC Machines)
RISC	Azaltılmış komut kümeli Bilgisayar
GNU	GNU unix değildir (GNU's not unix)
GPL	Genel kamu lisansı (General public license)
SBC	Tek Kart Bilgisayar (Single Board Computer)
WiFi	Kablosuz Bağlantı (Wireless Fidelity)
GPIO	Genel Amaçlı Giriş Çıkış (General Purpose Input Output)
GHz	Giga Hertz
nA	Nano Amper
RTOS	Gerçek Zamanlı İşletim Sistemi (Real Time Operating System)
USB	Evrensel Seri Veriyolu (Universal Serial Bus)
IMU	Atalet Olçüm Sensörü (Inertial Measurement Unit)
MEMS	Mikro Elektromekanik Sensörler (Micro Electromechanical Sensors)
EKF	Genişletilmiş Kalman Filtresi (Extended Kalman Filter)
LIDAR	Işık Tabanlı Tarama ve Mesafe Ölçümü
Tf	Dönüşümler (Transforms)
CPR	Tur Başına Döngü Sayısı (Cycle Per Revolution)
PWM	Darbe Genişlik Modulasyonu (Pulse Width Modulation)
PID	Oransal İntegral Türevsel Kontrolcü (Proportional İntegrative Derivative Controller)
IOT	Nesnelerin İnterneti (Internet of Things)
SLAM	Eş Zamanlı Konumlama ve Haritalama (Simultaneous Localization and Mapping)

1. GİRİŞ

Bu bölümde bu çalışmaya konu olan mobil robotların zaman içerisinde evrilmesine ve otonomi kazanmasına sebep olan tarihsel ve bilimsel süreçler ile endüstriyel gelişmeler incelenerek çalışmanın nedeni açıklanmıştır.

1.1. Sanayi ve Üretim Tekniklerinin Evrimi

Endüstride robotların kullanımı 3. Sanayi Devrimi ile başlamış olup, 4. Sanayi Devrimini yaşadığımız bugünlerde daha da artış göstermektedir [1]. Robotların endüstride kullanımı ile beraber üretim hızı, verim ve kalite giderek artmıştır. Bu sayede hem endüstride hem de akademik dünyada robotik giderek daha fazla önem kazanmıştır [2].

3. Sanayi Devrimi ile üretimin her geçen gün daha fazla içine giren robotik bilimi giderek evrilen bir hal almıştır. Robotlar sanayide ilk kullanmaya başladığında amaç üretim bandındaki belirli bir işin belirli bir kalite düzeyi ile yine belirli bir zamanda gerçekleşmesini sağlamaktı. Bu yüzden günümüze kadar üretilen tüm endüstriyel robotların temel amacı verilen belli başlı bir işi gerçekleştirmek olduğundan robotların basit bir şekilde üretim hattına uygun olarak programlanması ve kullanılması yeterliydi. Ancak her alanda üretilen ürünlerin giderek daha karmaşık bir hal alması ile üretim tekniklerinin de bu karmaşaya uygun bir hal alması zorunluluk haline gelmiştir. Bu zorunluluk ile robotların yapılarında da değişiklikler ve gelişmeler meydana gelmeye başlamıştır. Bu sayede robotların gelişen endüstriye uyum sağlaması mümkün olmuştur.

Gelişen endüstri ve paralelinde gelişen robotik uygulamaların giderek birbiri ile daha ilintili hale gelmeye başlamasıyla, 3. Sanayi Devriminin temellerini oluşturan bu yapıların birbirleri ile haberleşmesinin ve entegrasyonun maksimum düzeyde sağlanması zorunlu bir hal almıştır [3]. Çünkü endüstride entegrasyonun tam olarak sağlanması ancak ve ancak üretim sırasında kullanılan üretim araçlarının birbiri ile tam uyumlu olarak çalışması ile mümkündür [4]. Bunun sağlanmasının etkin yolu da bu cihazlar arası haberleşmenin internet yolu ile gerçekleşmesidir. Bunun sebebi ise internetin günümüzde neredeyse her alanda etkin biçimde kullanılmasıdır. Her nesnenin internete bağlanabildiği bir dünyada (IOT) gerek üretimin gerekse sosyal yaşamın kolaylaşması söz konusudur.

Günümüzde bir robota basit programlar yazıp yazılan bu programa göre robotu operasyonda tutmak ve yalnızca belirli iş yaptırmak üretimin verimliliğini arttırmanın önünde engeldir. Bu yüzden endüstride kullanılan robotların, gelişen teknolojiye paralel olarak tek bir işi değil birden fazla işi otonom olarak yapması beklenmektedir [5].

1.2. Tezin Amacı

Robotların bahsedilen gelişmelere ayak uydurabilmesi için verilen görevleri planlayıp bu planlamaya göre yerine getirmesi gerekmektedir. Bir robotun yapacağı işin belirlenmesinden sonra, o işi nasıl yapacağını üzerinde çalışan algoritmalara göre planlaması ve yerine getirmesi otonomi olarak adlandırılır.

Bu çalışmanın amacı mobil bir robot geliştirerek bu robotun kendisine verilen görevi otonom olarak yapmasını sağlamaktır. Bu çalışmada mobil robota verilen görev robotun kendi bulunduğu ortamda kullanıcı tarafından işaret edilen bir noktaya otonom olarak hareket etmesidir. Bu tezde tasarlanan otonom mobil robot için gerçekleştirilen çalışmalar sırasıyla bölümlere ayrılarak açıklanmıştır.

1. bölümde, bu çalışmanın akademik ve endüstriyel anlamda ne amaçladığıile birlikte bu çalışmanın nasıl gerçekleştirildiği ile ilgili temel bilgiler sunulmuştur.

2. bölümde, mobil robotlar ile ilgili bazı kavramlar açıklanmıştır. Mobilrobotiğin gelişimi ile hem endüstriyel hem de akademik alanlardaki çalışmalardan bahsedilmiştir.

3. bölümde, mobil robotların lokalizasyonu ve hareket şekilleri ile ilgili temel kavramlar ve bu kavramların hangi amaçlarla nasıl kullanılabileceği açıklanmıştır.

4. bölümde, bu çalışmada kullanılan ve robotun çevresini algılamasını sağlayan sensörler ile robotun hareketini sağlayan aktüatörler açıklanmıştır. Ayrıca neden bu bileşenlerin seçildiği anlatılmıştır.

5. bölümde, robotun tasarımı ve entegrasyonunun nasıl gerçekleştirildiği ile ilgili açıklamalar yapılmış, bu çalışma için belirlenen gereksinimler olan,

- bulunduğu çevreyi algılayan,
- algıladığı çevrede kendini algılamaya eş zamanlı olarak konumlayabilen,
- algılamaya eş zamanlı olarak harita çıkarabilen,
- çıkardığı harita üzerinde işaretlenen bir noktaya hareket edebilen birotonom

mobil robot tasarımının nasıl gerçekleştirildiği ifade edilmiştir.

Bu tez içerisinde 5. bölümde detaylı olarak anlatılan bu süreçler kısaca şu şekilde gerçekleşmiştir: İlk olarak tasarlanacak robotta kullanılacak bileşenler belirlenmiştir. Bu bileşenlerin temininden sonra ilk olarak robotun mekanik tasarımı diğer akademik çalışmaların bazıları incelenerek belirlenmiş ve tasarlanmıştır. Daha sonra kullanılacak elektronik bileşenler için bir elektronik tasarım süreci gerçekleştirilmiştir. Elektronik tasarım sürecinin ardından sensörleri ve aktüatörleri kontrol edecek olan yazılım geliştirme süreci C dili ile tamamlanmıştır. Daha sonra bu çalışmanın temelini oluşturan ROS işletim sistemi bir dizüstü bilgisayar üzerine kurulmuş ve bu bilgisayar üzerinde robotun görev algoritmaları ve yazılımları C ve Python dillerinde yazılmıştır. Tasarlanan bu algoritma ve yazılımlar robotun üzerinde daimi olarak çalışacak olan bilgisayara yüklenmiştir. Robot ile robotun uzaktan kontrolünü sağlayacak olan bilgisayar için bir kablosuz ağ oluşturulmuş ve bu ağ üzerinden robot ile bilgisayar arası haberleşme sağlanmıştır. Daha sonra ise Robotun ihtiyacı olan enerjinin sağlanması için bir güç ünitesi tasarlanmıştır. Bu süreçlerin tammalanması ve entegrasyonu ile çalışma tamamlanmıştır.

6. bölümde, Robotun hareketi için ihtiyacı olan veriler, donanımlar ve özellikle sensörlerden gelen verilerin nasıl değerlendirildiği açıklanmıştır. Bu bölümde genel olarak matematiksel ifadelere yer verilmiş ve bu ifadelerin çalışma içerisinde nasıl kullanıldığı ifade edilmiştir.

7. bölümde, çalışma sırasında karşılaşılan sorunlar, bu sorunları çözmek için uygulanan yöntemler ve elde edilen deneysel sonuçlar yorumlarıyla birlikte sunulmuştur.

8. bölümde, çalışmada elde edilen sonuçlar kısaca özetlenmiş ve gelecek çalışmalar ile ilgili bilgiler verilerek tez sonlandırılmıştır.

2. MOBİL ROBOTLAR

Mobil robotlar günümüzde birçok alanda kullanılmaktadır. Ancak bir mobil robotun niteliğini belirleyen en önemli kavram otonomidir. Bu yüzden bu bölümde otonom mobil robotların diğer mobil robotlardan nasıl ayırt edileceği açıklanmıştır.

2.1. Mobil Robot Kavramı

Mobil robotlar, günümüzde hem bilimsel hem de endüstriyel açıdan popülerlik gittikçe artan ve robotik biliminde önemli yer tutan elektromekanik makinelerdir. Mobil robotlar, yerlerini kendi hareket yeteneği aracılığıyla değiştirebilen sistemlerdir [6]. Bu tür robotlar genellikle tekerlekli olmasına rağmen ayaklı ve paletli başta olmak üzere birçok farklı sürüş sistemi ile kontrol edilebilen sistemlerdir, yani bu sistemler birbirinden farklı mekanik konfigürasyonlara sahip olabilirler. Hangi mekanik konfigürasyonun tercih edileceğinde temel etken, çalışma bölgesi ve yapılacak olan iştir. Bu etkenler aynı zamanda robotun türünü de belirlerler [7]. Robotik biliminin amacı, temelde insanlığa hizmet eden elektromekanik cihazları daha efektif ve faydalı hale getirmektir. Ancak insanoğlunun sürekli bir değişim ve gelişim içinde olması ve bu durumun gün geçtikçe ivmelenen bir hal alması robotik biliminde çeşitli farklılaşmalara ve yeni robot sistemlerinin ortaya çıkmasına sebep olmaktadır [8]. Bu farklılaşmaların ve gelişmelerin en çok görüldüğü robotik alt dallarından biri de mobil robotlardır. Bu gelişmelerin paralelinde mobil robot kavramı günümüzde başlıca iki farklı başlık altında incelenmektedir. Bunlar, AGV(Autonomous Guided Vehicle) olarak bilinen otomatik yönlendirmeli araçlar ve AMR (Autonomous Mobile Robot) olarak bilinen otonom mobil robotlar şeklinde tanımlanabilirler [9]. Mobil robotlarda bu ayrımın teknik sebebi ise her iki türün mobil olmasına rağmen çalışma mekanizmaları ve çalışma ortamlarının farklı olmasıdır. Aralarındaki en büyük benzerlik ise ikisinin de otonom olarak bulunduğu ortamda hareket etmesidir.

2.1.1. Otonom yönlendirmeli araçlar (AGV)

Otonom ya da otomatik yönlendirmeli araçlar olarak bilinen AGV'ler mobil bir robot türüdür. Bu tür mobil robotlar bir bilgisayar veya kontrolcü barındırsalar da

bulundukları ortamda hareket etmek için ortamda var olan fiziksel yönlendiricilerden faydalanırlar. Endüstriyel bazı yük taşıma robotları ile çizgi izleyen robotlar bu kategoridedir. Kendi kendilerine navigasyon planlaması yapmaya uygun olmayan bu robotlar daha önceden belirlenmiş, tekrar eden ve/veya periyodik işleri yapmak için tasarlanmışlardır [10]. Bu nedenle bir AGV başka bir navigasyon planı gerektiren bir işte kullanılacaksa programcı tarafından yeniden programlanır. Yeni programa uygun olarak AGV navigasyon planını takip eder. Ancak bu robotların navigasyon planı ile ilgili kendi kendine karar verme yetisi bulunmadığından planda oluşabilecek herhangi bir hata istenmeyen sonuçlara neden olabilir. Bu nedenle AGV için yeni bir navigasyon planı yapılırken çok dikkatli davranılmalı ve tüm olası hatalara karşı önlem alınmalıdır.

Anlaşılacağı üzere AGV türündeki bir robotun, tekrar eden bir iş dışında kullanılması hem yeniden programlanmasını hem de ortamdaki fiziksel yol göstericilerin değiştirilmesi ile mümkündür. Bu yüzden AGV'lerin yeni bir işe adaptasyonu, sistemde yapılan değişikliğe göre ciddi bir efor ve süreç gerektirir ki bu durum iş süreçleri dinamik olan bir endüstriyel tesis için oldukça fazla zaman kaybı demektir. Uçüncü sanayi devrimi ile geliştirilmeye ve kullanılmaya başlanan AGV'ler günümüzde oldukça fazla kullanılsa da gelişen teknolojinin hızına ayak uyduramayan alt yapısı ile popüleritesini kaybetme eğilimindedir.

Arkin ve Murphy, üretim ortamında otonom navigasyonun önemine değinerek, otonom robot mimarisi üzerine karşılaştırmalı temel bilgiler vermiş, AGV lerin dezavantajlarını anlatmış ve gelecekte özellikle esnek üretim sistemleri açısından AMR lerin ne denli önem kazanacağına vurgu yapmışlardır [11].

2.1.2. Otonom mobil robotlar (AMR)

Otonom mobil robotlar, AGV'lerin yapısına benzer olarak kendi üzerinde bilgisayar veya kontrol ünitesi barındıran robotlardır. Ancak AGV'lerden farklı olarak AMR'ler barındırdıkları bu üniteler ile kendi navigasyon planını yapabilen ve yaptığı navigasyon planı ile bulunduğu ortamda herhangi bir fiziksel yönlendiriciye ihtiyaç duymaksızın hareket edebilen robotlardır [12]. Yani AMR'ler önceden bilinmeyen bir ortamda navigasyon planlayıp uygulayabilirler.

AMR'ler bulundukları ortamda navigasyon planlaması yapabilmek için çok sayıda sensör kullanırlar. AGV'ler ile kıyaslandığında kullanılan sensör ve algılanan

parametre sayısı daha fazladır. Bununla birlikte değerlendirilecek ve işlenilecek veri sayısı AGV'lere göre daha fazla olduğundan AMR'lerin üzerinde bulunan bilgisayar ve/veya kontrol ünitesinin işlem gücü daha yüksektir. Dolayısıyla sistemin kullandığı enerji miktarı da daha yüksektir. Bu durum AMR'lerin getirdiği yenilik ve faydalar düşünüldüğünde göz ardı edilebilecek kadar küçük bir ayrıntıdır [13].

Genel olarak AMR'ler bulunduğu ortama ait parametreleri, üzerlerinde bulunan sensörler ile algılayabilen ve ortamın modelini oldukça hassas şekilde çıkartabilen robotlardır. Ayrıca çıkarttıkları ortam modeli üzerinde kendilerini konumlayıp navigasyon planlarını bulundukları konuma göre yeterli hassasiyette yapabilirler. Bununla birlikte AMR'ler ortamı hassas şekilde haritalayıp eş zamanlı olarak kendilerini bu haritada konumlayabilirler (SLAM). AMR'ler bahsedilen teknolojiler ile ortamın görsel modelini de oluşturabilir, bu modeli de kullanarak hareket edebilir ve ortamda var olan statik veya dinamik engellerden kaçınabilirler. Bu sayede ortamda var olan herhangi bir yabancı maddenin robota verebileceği zararın önüne geçilir.

Yukarıda anlatılan nitelikler dikkate alınarak kıyaslanığında, AMR'lerin hem teknolojileri hem de çalışma sistemleri AGV'lere göre oldukça farklıdır. Bununla birlikte AMR'ler, AGV'lerden farklı olarak statik veya sürekli değişen iş planlarına göre navigasyon planı hazırlayıp hareket edebilirler. Kendi planlarını kendileri hazırladıklarından tekrar programlanmaya ihtiyaç duymazlar [14]. Bu sayede özellikle endüstriyel tesislerde hem zaman hem de iş gücünden tasarruf edilerek verim arttırılabilir. Bahsedilen bu nitelikleri nedeniyle özellikle endüstride son dönemde AGV'ler yerine AMR'ler tercih edilmeye başlanmıştır [11].

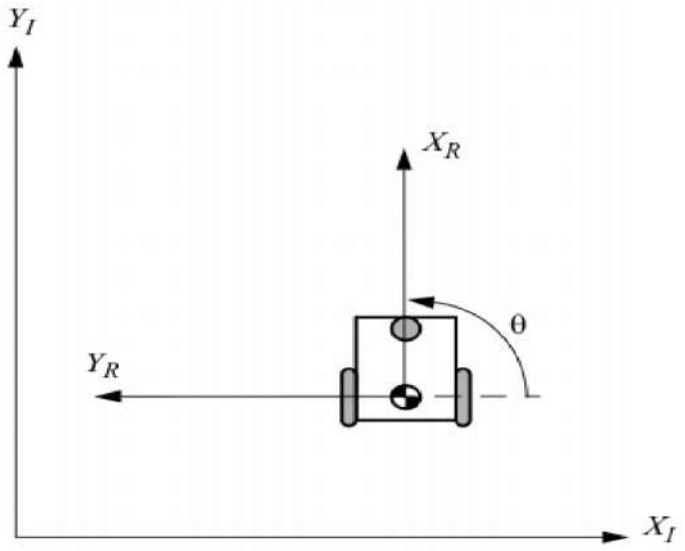
Bu çalışmada endüstriyel alanda giderek daha popüler bir hale gelen AMR tipi robotların tasarımı ve çalışma prensipleri üzerinde durulmuş, temel özelliklere sahip bir AMR tasarlanarak üzerinde bazı algoritmalar koşturulmuş ve karşılaşılan problemlere yönelik çözüm yöntemleri sunulmuştur.

3. OTONOM MOBİL ROBOTLARDA TEMEL KAVRAMLAR

Otonom mobil robotlar gelişen teknoloji ile evrildiklerinden ötürü çalışmalarının anlaşılması için bu bölümde, robotların çalışmasına etkiyen teknolojiler ve terimler açıklanmıştır.

3.1. Lokalizasyon

Bir mobil robot için lokalizasyon, robotun hareket edeceği çevre üzerinde kendini konumlaması anlamına gelir. Bunun için robotun ”neredeyim?” sorusunu cevaplaması gerekir.



Şekil 3.1. İki boyutlu koordinat düzlemine oturtulmuş olan diferansiyel sürüş sistemine sahip bir robotun pozisyonu

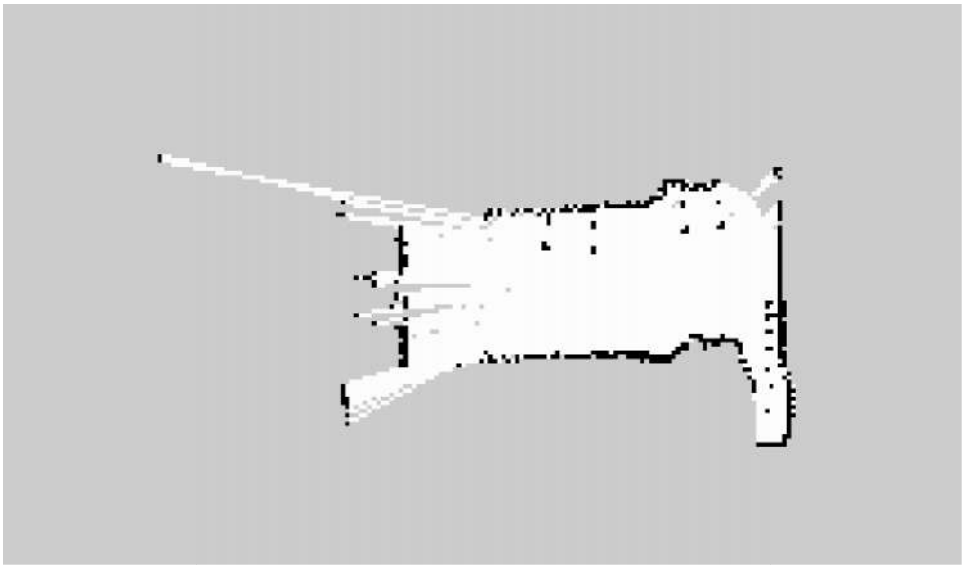
Robot biliminde, lokalizasyon önemli bir problemdir. Çünkü mobil robotun hareketini doğru bir şekilde planlaması ve uygulaması için lokalizasyonun doğru bir şekilde bulunmuş olması gereklidir. Buradan çıkarılacak sonuç lokalizasyonun otonom robotlar için düzgün hesaplanması gerekliliğidir.

Bu çalışmada tasarlanan robotta, robotun çevresini algılamakta kullanılan sensörlerin çıktıları birleştirilerek lokalizasyonda hatanın yok denilebilecek kadar az olması sağlanmıştır.

3.2. Konum Tahmini (Dead Reckoning)

Dead reckoning, hareket eden bir hareketlinin bir sonraki pozisyonunun bir önceki pozisyonuna bağlı olarak tahminen hesaplanması anlamına gelir. Özellikle hava ve deniz taşıtlarında önemli bir konum tahmin yöntemi olmasının yanı sıra mobil robotikte de sıklıkla kullanılır. T. Lee vd. yaptıkları çalışmada, gerçek zamanda mobil robotun konum doğruluğunu arttırmak ve daha hassas bir navigasyon sağlamak amacıyla, üç eksenli IMU, odometri ve aktif beacon sistemi gibi üç farklı sensörden gelen verileri alçak geçiren filtre ve Kalman filtresi kullanarak birleştirmişler, yaptıkları simulasyonlarda ve deneysel uygulamalarda robot navigasyonu ile ilgili daha doğru sonuçlar elde ettiklerini göstermişlerdir [15].

Buna göre bir cismin önceki hareketi sırasında elde edilen hız ve yön verisine göre bir sonraki konumunun tahminine Dead reckoning denir. Dead reckoning uygulanırken önemli olan elde edilen en son hız ve yön verisine göre tahminin yapılmasıdır [16]. Eğer daha önceki hız ve yön verileri kullanılırsa, robotun hesaplanan konumu ile gerçekte bulunduğu konum arasındaki fark giderek artar.



Şekil 3.2. Otonom mobil robot için üretilen harita örneği

3.3. Harita Tabanlı Lokalizasyon

Yukarıda lokalizasyon bölümünde anlatıldığı üzere, 2 boyutlu bir ortamda hareket eden bir robotun bulunduğu ortam kartezyen koordinat sistemi olarak kabul edilebilir ve bu şekilde robotun pozisyonu hesaplanabilir. Şayet robotun üzerinde

ortamı haritalamak için kullanılabilecek bir sensör (Lidar, Stereo Kamera vb.) var ise bu durumda ilgili sensörden gelen veri ile bir harita oluşturulup robot modeli bu harita üzerine konumlandırılabilir [17]. Buna harita tabanlı lokalizasyon denir. Şekil 3.2'de lidar tabanlı bir haritalama örneği verilmiştir.

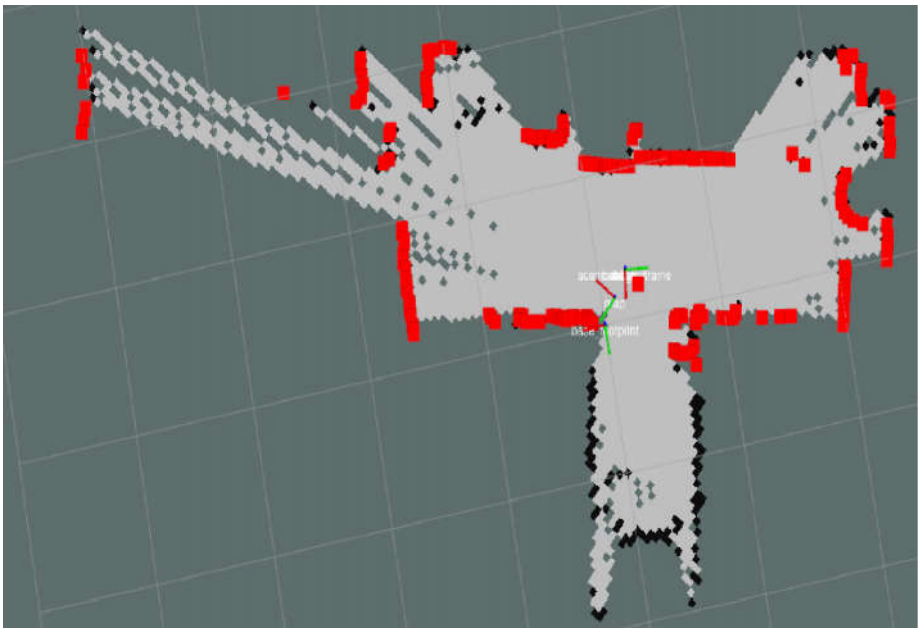
3.4. Lidar Tabanlı Lokalizasyon

Lidar, 2. bölümde anlatıldığı üzere lazer ışığının uçuş süresini kullanarak mesafe ölçmeye yarayan bir sensördür. Şayet lidar kartezyen koordinat düzlemi üzerinde düzleme paralel olarak dönüp uzaklık ölçümünü bulunduğu baş bilgisi ile birlikte verirse, bu durumda lidarın bulunduğu 2 boyutlu düzleme paralel ve lidarın atım ve algılama sayısına bağlı olarak, çevrede noktasal uzaklık ölçümleri ile harit'a oluşturulup bu harita üzerinde robot konumlandırılabilir. Buna lidar tabanlı haritalama denir. Şekil 3.3'te robot üzerine konumlanmış bir lidar yardımı ile elde edilen harita üzerinde lidarın anlık ölçümleri kırmızı ile gösterilmiştir.

Lidardan doğrudan elde edilen veriler kullanılarak çevre haritalanabilir. Ancak robotun düzlemde hareketine bağlı olarak değişen uzaklık ölçümleri sonucunda harita değişime uğrar ve bir önceki harita robotun hafızasında tutulamaz. Bu durumda hareket planlaması yapılamaz. Anlaşılacağı üzere Lidar tabanlı lokalizasyon tek başına yeterli değildir.

3.5. Haritalama

Haritalama, mobil robotikte kullanılan önemli özelliklerden biridir. Haritalama sayesinde kullanıcı, robotun sensörlerinden gelen verilere göre oluşturduğu çevresel modeli inceleyebilir. Gerektiğinde değişiklikler gerçekleştirebilir. Mobil robotlarda kullanılan haritalar statik ve dinamik haritalar olmak üzere ikiye ayrılır.



Şekil 3.3. Robotun üzerindeki lidar ile taranan çevresel modele ait harita

3.5.1. Statik harita

Statik harita, robota yapacağı görev öncesinde yüklenen haritalardır. Robot çevresel modeli kendisi sıfırdan oluşturmak yerine bu haritayı kullanarak görevini icra eder. Bunu yaparken kendi üzerinde bulunandan sensörlerden gelen verileri değerlendirir ve bu verilere göre karar verir. Ancak görev başında sistemine yüklenen haritada değişiklik yapmaz.

Bu tür haritaların kulllanıldığı robotlarda görev sırasında meydana gelen çevresel değişikliklere tepki verilemez. Bu nedenle çevrede ortaya çokan dinamik sebepler ile robotun yol planı değişikliği yapılması gerekirse robot bunu otonom olarak gerçekleştiremez.

3.5.2. Dinamik harita

Dinamik harita, robotun operasyonu sırasında kendi üzerinde bulunan sensörlerden gelen veri ile sıfırdan oluşturulabileceği gibi operasyon öncesinde yüklenen bir harita üzerinde robotun operasyonu sırasında kendi sensörlerinden gelen verileri kullanarak güncelleme yapması ile elde edilir.

Bu tür haritalar ile çevresel değişiklikler anında çevresel modele yansır ve robot hareketini yeni modele göre planlayabilir. Kısacası dinamik harita ile robot, çevresinde oluşabilecek değişikliklere karşılık verebilir. Bu tür haritaların

oluşturulabilmesi için robotun üzerinde haritalama için veri üretebilecek (Lidar, stereo kamera vb.) sensör bulunmalıdır.

3.6. Eş Zamanlı Konumlama ve Haritalama (SLAM)

SLAM bir mobil robotun bulunduğu ortamın çevresel modelini oluşturup, oluşturduğu modelde kendisini konumlaması ve oluşturulan çevresel modele göre etrafını haritalamasıdır [18]. Buna göre robot hareket halindeyken sürekli olarak çevresel modeli güncelleyip genişletir [19]. Yani SLAM ile oluşturulan harita dinamik bir haritadır. Şekil 3.4'te harita tasarlanan robotun kendi oluşturduğu harita üzerinde konumu görülmektedir.

Şekil 3.4. Tasarlanan robot kullanılarak SLAM yöntemi ile üretilen harita

SLAM problemi lokalizasyondan daha zor bir problemdir. Çünkü hem robotun hareketi süresince çevresel model oluşturulmaya çalışılırken hem de sensörlerden akan verilerdeki parazitlerin sönümlenmesi gerekir [20].

SLAM'ın matematiksel ifadesi şu şekilde açıklanabilir. Robotun her t anındaki pozisyonu x_t ile ifade edilirse, robotun gittiği yol şöyle ifade edilebilir:

$$X_T = \{x_0, x_1, x_2, \dots, x_T, \} \tag{3.1}$$

3.1 nolu eşitlikte ifade edilen T sonsuz olarak tanımlanmış olabilir. Burada dikkat edilmesi gereken x_0 pozisyonu başlangıç noktası olduğundan dolayı hesaplamanın en başında zaten bilinmektedir. Ancak aynı durum diğer noktalar için geçerli değildir.

Robotun $t-1$ ve t zamanları arasındaki hareketi u_t ile ifade edilirse hareket verisinin enkoder okumalarından veya motora verilen kontrol girdilerinden elde edildiği kabul edilirse, robotun hareketi zamana bağlı olarak şu şekilde yazılabilir:

$$U_T = \{u_0, u_1, u_2, \dots, u_T, \} \tag{3.2}$$

Buna göre gerçek harita M ile şu şekilde ifade edilebilir:

$$M_T = \{m_0, m_1, m_2, \dots, m_{n-1}, \} \tag{3.3}$$

3.3 nolu eşitlikte ifade edilen m_i, $i = 0 \dots n-1$ değerleri robotun çevresindeki nesneleri ifade eder. Sonuç olarak robotun bir anda, tek bir ölçüm aldığını kabul ederek tümölçümü şu şekilde ifade edebiliriz:

$$Z_T = \{z_0, z_1, z_2, \dots, z_T, \} \tag{3.4}$$

Yukarıda ifade edilen terminolojiye göre SLAM'ı, harita modelini (M) ve robotun geçtiği yolu, (X_T) odometri (U_T) ve gözlemlerden (Z_T) elde ettiği verilere göre iyileştiren bir yapı olarak kabul edebiliriz.

Robotun çevresindeki işaretler robotun üzerinde bulunan sensörlerin robot üzerinde bağlandıkları konuma göre görecelidir. Literatürde SLAM, Tam SLAM ve Online SLAM olarak ayırt edilir. Buna göre Tam SLAM X_T ve M boyunca sonraki ortak olasılığı 3.5 eşitliği ile gösterilir.

$$p(X_T, M \mid Z_T, U_T) \tag{3.5}$$

Online SLAM, Tam SLAM'a karşın x_t, M boyunca sonraki ortak olasılığı (3.6) gösterir.

$$(x_T, M \mid Z_T, U_T) \tag{3.6}$$

Anlaşılacağı üzere Tam SLAM robotun tüm geçtiği yolu (X_T) tahmin etmeye çalışırken, Online SLAM sadece bulunulan zamandaki pozisyonu (x_t) gösterir.

Bu çalışmada, sıfırdan bir SLAM algoritması yazmanın zorluğu dikkate alınarak, tasarlanan otonom mobil robot uygulamasında, ROS için hazır paket olarak bulunan Hector SLAM paketi, robota ve robot üzerinde kullanılacak sensörlere uygun olarak modifiye edilerek kullanılmıştır. Tasarlanan robot belirtilen SLAM paketini kullanarak bulunduğu ortamı günceller.

3.7. Navigasyon

Navigasyon bir mobil robotun belirli bir noktadan başka bir noktya bir navigasyon algoritması kullanarak veya navigasyon algoritması kullanmadan kullanıcı kontrolünde hareket etmesi anlamına gelir. Günümüzde mobil robotikte en önemli problemlerden biri robotun navigasyon algoritmasının doğru seçilmesi ve planın bu algoritmaya göre yapılmasıdır [21]. Navigasyon ile ilgili detaylı açıklama 6. bölümde yapılmıştır.

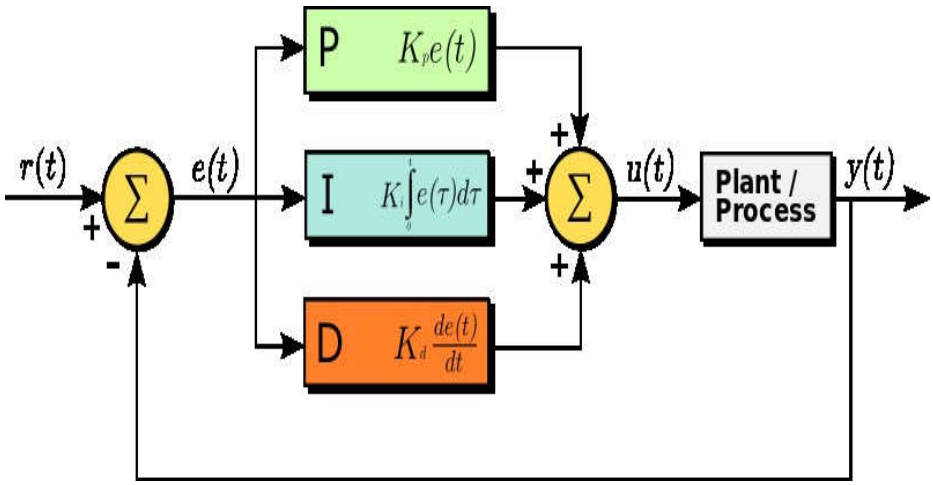
3.8. Hareket Kontrolü

Hareket kontrolü, bir robotun belirli bir navigasyon planı veya kullanıcı kontrolü dahilinde navigasyonu sırasında hareketini kontrol eden kontrolcüyü ifade eder. Bu çalışmada hareket konrolünün sağlanması için PID kontrolcü kullanılmıştır. Bu kontrolcü kullanılarak tasarlanan mobil robot için hız kontrolü sağlanmış ve bu şekilde robotun hareketi yönetilmiştir.

Günümüzde mobil robotlarda en fazla kullanılan kontrolcülerden birisi PID kontrolcü olsa da yerini yavaş yavaş adaptif kontrolcülere bırakmaktadır. Bunun nedeni adaptif kontrolcülerin bulundukları ortamın şartlarına göre kendilerini optimum kontrole ayarlayabilmeleridir. Bu çalışmada adaptif kontrolcü kullanılmamasının temel sebebi ise tasarlanan robotun iç ortamda tek düze koşullarda hareket etmesidir. Bu koşullara uygun PID kontrolcüsünün katsayıları da deneme yanılma yoluyla tespit edilmiş ve sisteme yüklenmiştir.

PID endüstride kullanılan en genel kontrol algoritmasıdır. Bunun nedeni kontrol edilmek istenilen neredeyse her sisteme uyarlanabilmesidir. PID kontrolcüsü basit olarak şöyle çalışır: Kontrol edilecek sisteme bağlı olan sensör okunur.

Sensörden gelen bu veriler ile istenilen sistem çıkış değerine göre oransal, integratif ve türevsel hesaplamalar yapılır. Sonrasında ise sistem çıkışına bu üç hesaplamanın toplamı verilir. Şekil 3.5'te PID kontrolcünün kapalı kontrol yapısı gösterilmiştir.



Şekil 3.5. PID kontrolcü yapısı

3.8.1. Oransal hesaplama (P)

Oransal hesaplama sistemin o anki çıkışı ile istenilen çıkış arasındaki farkı belirtir. Bu fark hata olarak adlandırılır. Bu hata oransal hesaplama için Oransal hesaplama katsayısı K_p ile çarpılarak çıkışa yansıtılır. K_p doğrudan sistemin tepkisini belirler. Eğer çok küçük olursa sistemin kararlılığa ulaşması zorlaşır. Bununla beraber K_p değeri çok büyükse bu durumda sistemin tepkisi çok yüksek olacağından, sistem osilasyona girer.

3.8.2. İntegratif hesaplama (I)

Kontrolcünün integratif bölümü tüm zaman boyunca haa sinyallerini toplar. İntegratif hesaplama değeri zaman içerisinde üretilen hata sinyali miktarına bağlı olarak yavaşça artar. İntegratif kontrolcünün sistem üzerindeki etkisi sistemin kararlı durumundaki hata miktarını sıfıra indirmektir. K_i, integratif hesaplama katsayısı olarak bilinir. Hesaplama sonucu K_i ile çarpılarak sistemin çıkışına aktarılır.

3.8.3. Türevsel hesaplama (D)

Türevsel hesaplamanın amacı hatanın zamana bağlı değişimini ölçmektir. Türevsel hesaplamanın amacı hatayı sistemin çalışması süresinde minimumda sabit tutmaktır. K_d, türevsel hesaplama katsayısıdır. Bu katsayı olabildiğince küçük tutularak sistem çıkışına aktarılır.

PID kontrolcüsü, Kapalı çevrim bir kontrolcüdür. Kontrolcünün bu özelliği yukarıdaki Şekil 3.5'de gösterilmiştir [22]. PID kontrolcünün tasarlandıktan sonra sisteme en uygun katsayılar ile sistem için ayarlanması (Tuning) gereklidir. Bu ayarlama için birçok yöntem mevcuttur. Bunlardan bazıları şunlardır:

- Ziegler-Nichols,
- Tyreus Luyben,
- Cohen-Coon,
- Angstrom-Hagglund,
- Try and Error (Deneme Yanılma)

Bu çalışma için tasarlanan PID kontrolcüde kontrolcü katsayıları (K_p, K_i, K_d) deneme yanılma yoluyla hesaplanmıştır. Bunun nedeni bu çalışmanın esas konusunun PID kontrolcü olmamasıdır.

4. KULLANILAN ROBOTİK SENSÖRLER ve AKTÜATÖRLER

Robotlar tasarlanırken çevreyi algılamayı amaçlayan sensörler ile donatılırlar. Farklı her robot türü için kullanılan sensörlerin çeşidi ve sayısı farklılık göstermekle birlikte, hassasiyetleri de algılanacak değişikliğin robot için önemine göre farklılıklar gösterir. Ayrıca sensörler kendi aralarında sınıflandırılabilirler. Mobil bir robot için kullanılacak sensörler ikiye ayrılarak incelenebilir.

4.1. Sensörler

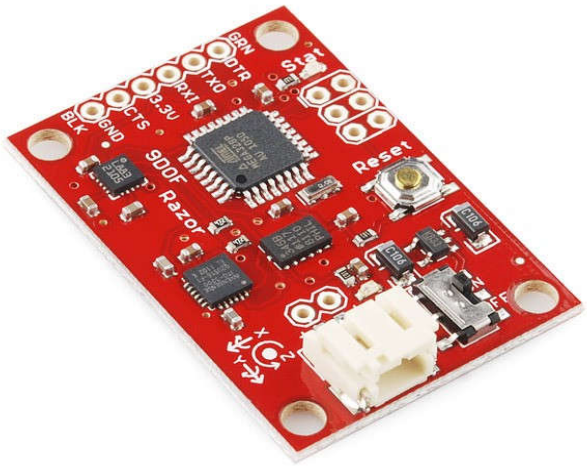
Sensörler bir robotun algılayacağı verinin türüne göre ikiye ayrılabilirler. Bu şekilde sensörlerin robot üzerindeki kullanım amaçları daha net olarak anlaşılabilir.

4.1.1. İç algı sensörleri

İç algı (proprioceptive) sensörleri, robotun kendisinden içyapısı ile ilgili değişiklikleri ölçen sensörlerdir. Robot, kendi hareketi sonucu, kendi içinde veya kendi üzerinde bulunan ve çalışmasında gerekli olan bir değişkeni ölçüyorsa bunu iç algısal sensörler aracılığıyla yapıyor demektir. İç algı sensörleri ile ölçülen değişkenler genel olarak robotun bulunduğu ortamdan bağımsız olarak ölçülür. Dolayısıyla çevrede oluşan herhangi bir değişiklik bu sensörler ile algılanamaz. Mobil robotlarda kullanılan başlıca iç algı sensörler şunlardır:

a) Atalet ölçüm sensörü (IMU) : IMU (Atalet Ölçüm Ünitesi), içerisinde ivme ölçer ve jiroskop bulunduran elektromekanik bir atalet ölçüm sensörüdür. Bazı IMU sensörlerinde ivmeölçer ve jiroskop ile birlikte yön bulmak için manyetik pusula vardır. IMU üzerinde bulunan bu sensörlerden aldığı ölçümleri birleştirerek analog ya da dijital bir atalet bilgisi ile yön bilgisi verir. IMU üzerinde, birden fazla sensör bulundurmasının temel nedeni, daha hassas ve doğru bir ölçüm yapabilmesini sağlamaktır. Birden fazla elektromekanik sensör barındırmasına rağmen IMU'lar boyut olarak çok küçüktür. Buna olanak sağlayan ise MEMS (Mikro elektromekanik sistemler) olarak bilinen sensör teknolojisidir [23]. MEMS teknolojisi, var olan elektriksel ve mekanik bazı sistemlerin mikron, boyutuna küçültülerek bu boyutlarda çalıştırılabilmesini mümkün kılan bir teknolojidir. Böylelikle MEMS kullanılan her sensör, aynı işi yapan diğer sensörlere nazaran daha küçük boyutlara sahiptir. Ayrıca

MEMS tabanlı sensörler dış dünyadan izole edildiklerinden, diğer sensörlerin maruz kaldıkları çevresel koşullardan etkilenmezler. Bu nedenle daha uzun ömürlü olurlar ve daha hassas ölçüm yaparlar. Günümüzdeki IMU ve benzeri sensörlerin büyük çoğunluğu MEMS teknolojisini baz alarak tasarlanmakta ve üretilmektedir. IMU üzerindeki sensörler teker teker incelenecek olursa IMU'nun çalışma prensibi daha kolay anlaşılabilir. Şekil 4.1'de bu çalışmada kullanılan IMU gösterilmiştir [24].



Şekil 4.1. Atalet ölçümü için kullanılan Sparkfun Razor IMU

İvmeölçer bir cismin kütlesine etki eden ivmeyi ölçen elektromekanik sensördür. İçerisinde bulunan kütleye etki eden ivmenin değerini ölçerek çıkışa yansıtır. İvmeölçerin çalışma mekanizması eylemsizlik kuvvetine bağlıdır. Yani cismin maruz kaldığı eylemsizlik kuvveti ivmeölçer tarafından değerlendirilerek sensör çıkışına bir ivme bilgisi verilir. İvme ölçümünde sensöre etkiyen referans vektör yer çekimi ivmesidir. Sensör yer çekimi ivmesini kullanarak anlık ivme ölçümü yapabilir. İvme sensörleri analog ya da sayısal çıkışlıdır. Aynı zamanda tek eksende ölçüm yapabilen sensörler var olduğu gibi 2 ya da 3 eksende de ölçüm yapanları da mevcuttur.

Jiroskop açısal hızı ölçen elektromekanik bir sensördür. Jiroskop temelde dönen bir diskten oluşan ve dönen diskin dönme momentine göre çalışan bir sistemdir. Bir başka deyişle jiroskop bir cismin bir eksen etrafında ne kadar hızlı döndüğünü hesaplamak için kullanılan bir sensördür. Bu sayede jiroskop hassas yörünge ayarlamada ve dönen bir sistemdeki stabilizeyi arttırmak için kullanılır. Bu

sensöre ihtiyaç duyulmasının bir başka nedeni yalnızca ivme ölçer ile yeterince hassas algılama yapılamamasıdır. jiroskoplar da ivme ölçerler gibi bir, iki veya üç eksende ölçüm yapabilirler.

Manyetik pusula dünyanın manyetik alanından faydalanarak yön tayini yapmaya yarayan bir sensördür. Yani manyetik pusula dünyanın etrafında oluşan manyetik alan çizgilerinden faydalanarak basit bir şekilde yön bulur. Elektromekanik manyeik pusulalar ise yine MEMS teknolojisine dayalı olarak üretilmiş sensörlerdir. Bu tür sensörlerin çıkışlarına ilettikleri veriler analog olabildiği gibi dijital de olabilir. Manyetik pusulaların en önemli dezavantajı ise klasik pusulalar gibi dünyanın manyetik alanını bozan etraftaki iletkenlerden etkilenmeleri ve yanlış sonuç vermeleridir. Bu yüzden manyetik pusula sensörlerinin kullanılacakları platform yapısında iletken olan bir platform ise sensörler bu iletkenden etkilenmeyecek kadar uzağa konulmalıdır.

b) Tekerlek dönüş sensörü (wheel shaft encoder) : Artımsal Enkoder dönebilen bir şafta bağlanan ve şaftın dönüş hızı ve dönüş yönü ile ilgili verileri toplayabilen bir elektromekanik sensördür. Genellikle optik enkoder olarak bilinse de son dönemlerde hall etkisi ile çalışanları mevcuttur. Optik enkoderler, bir ışık kaynağı, bir algılayıcı ve üzerinde yarıklar bulunan ve dönen şafta bağlanan bir disk kullanılarak yapılmıştır. Disk üzerinde bulunan yarıklar sensörün çözünürlüğünü belirler. Örneğin disk üzerinde 64 yarık varsa sensörün çözünürüğü şaftın turu başına 64 PPR' dır (Tur başına pals sayısı). Işık kaynağından çıkan ışık, disk üzerindeki yarıktan geçer veya engellenir. Algılayıcı ise yapısında bulunan bir foto transistörü yarıktan geçen ışığın tetiklemesi ve bu tetiklemenin sensör çıkışında gerilim oluşturması ile çalışır. Oluşan bu gerilim, dışarıdan bağlanan bir sayıcı ile sayılarak toplam dönüş sayısı hesaplanır. Hall etkisi ile çalışan enkoderler ise ölçümü ışık kullanmadan sadece hall etkisi ile ölçerler. Şekil 4.2'de bu çalışmada kullanılan enkoderin motor şaftına bağlı hali gösterilmiştir [25].

Şekil 4.2. Tekerlek dönüş sayısını ölçmek için kullanılan enkoder

Bir sistemde enkoder kullanılmasının esas amacı sistemde hassas hız ve konum kontrolü sağlamaktır. Enkoder temelde bir şafta bağlı bir disk üzerine işaretlenmiş olan kodu okuyup bu koda göre yön ve hız bilgisi verir. Bu işlemi disk üzerindeki bir kod dizisini bir kanaldan okuyarak yapabildiği gibi bir ya da birden fazla kod dizisini birden fazla kanal üzerinden okuyarakta yapayabilir. Tek kod dizisine sahip bir disk, tek kanallı bir algılayıcı tarafından okunursa, bu durumda sadece kod dizisinin algılayıcının önünden geçiş hızı algılanacağından, sensör tek kanal üzerinden yalnızca hız verisi okur. Eğer aynı disk birden fazla algılayıcı ile okuma sırasında algılayıcılar arasında faz farkı olacak şekilde yerleştirilirse bu durumda sensörün kanalları arasında oluşan faz farkından dolayı yön bilgisi de elde edilebilir. Bu da daha hassas hız ve konum kontrolü sağlar. İstenilen kontrol hassasiyetine göre sensör ve kanal sayısı seçilebilir.

Şekil 4.3. Tekerlek dönüş sayısını ölçmek için kullanılan quadrature enkoderin 1. ve 2. kanallarında oluşan sinyallerin osiloskoptaki görünüşü

Artımsal enkoderlerin en çok kullanılan türevlerinden biri quadrature (dört evreli) enkoderlerdir. Quadrature enkoderler bir disk üzerindeki yarıkları iki farklı algılayıcı tarafından algılamak üzere tasarlanmışlardır. Ancak algılayıcılar disk üzerindeki yarıkları algılarken algılama fazları arasında 90 derecelik faz farkı vardır. Diğer bir deyişle quadrature encoder diski okurken, 90 derecelik faz farkı olan 4 parça halinde okur ve bu dört parça sinyal birbirini takip eder. Bu tür artımsal enkoderlere quadrature denilmesinin sebebi budur. Şekil 4.3'de bu çalışmada kullanılan enkoderin her iki kanalının dönüş çıktılarının osiloskop üzerindeki görüntüsü verilmiştir [25].

4.1.2. Dış algı sensörleri

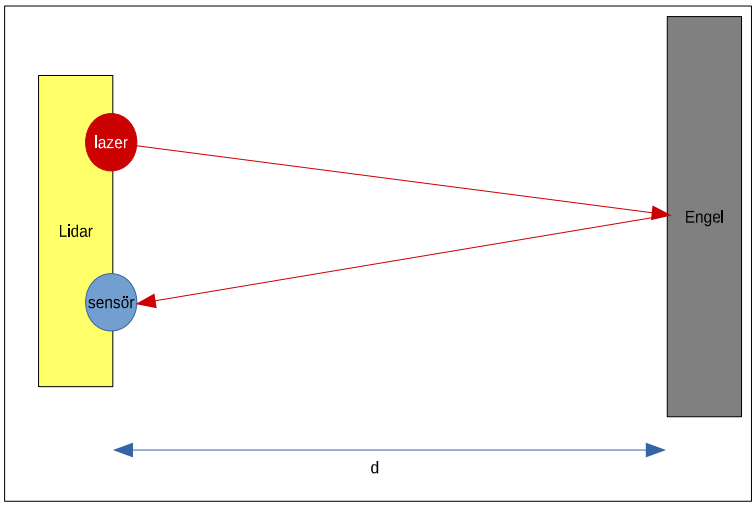
Dış algı (exteroceptive) sensörleri, robotun iç yapısı ile ilgili olmayan değişiklikleri ölçen sensörlerdir. Robotun, kendi hareketi ile ya da robotun hareketinden bağımsız olarak robotun bulunduğu ortamdaki değişiklikleri ölçüyorsa bu ölçümler dış algı sesörleri ile yapılıyor demektir. Dış algı sensörleri ile ölçülen değişkenler genel olarak robotun bulunduğu ortama bağımlı olarak ölçülür. Dolayısıyla çevrede oluşan herhangi bir değişiklik bu sensörler ile doğrudan algılanabilir [26]. Mobil robotlarda kullanılan başlıca dış algısal sensörler şunlardır.

a) Lazer mesafe ölçerler: Lidar (Işık tabanlı mesafe ölçümü), ışığa dayalı uzaktan mesafe ölçümü yapan bir elektromekanik sensördür. Basitçe açıklamak gerekirse lidar, ışık hızı ve ışığın havada uçuş süresini kullanarak mesafe ölçümü yapabilen bir sensördür. Ancak lidar herhangi bir ışık kaynağı kullanamaz. Kullanılan ışık kaynağının, dolayısıyla doğru ve hassas ölçüm için üretilen ışığın, çeşitli özellikleri taşıyor olması gereklidir. Bu özellikler:

- Kullanılan ışık kaynağının ürettiği ışık demetinin dağılmadan ilerlemesi gerekir.
- Üretilen ışık demetinin yönlendirilebilmesi gerekir.
- Üretilen ışık demeti monokromatik yani tek renkli olmalıdır.
- Işık kaynağı çok kısa sürede çok yüksek sayıda atım yapabilmelidir.

Tüm bu özellikleri bir araya topladığımızda lidar için kullanılan en doğru kaynağın lazer olduğu anlaşılır. Çünkü lazer ışığı eş evreli ve kısa dalga boylu olduğundan hem dağılmadan ilerler hem de kolay yönlendirilebilir. Ayrıca ışık eş evreli olduğundan yüksek enerjilidir ve uzak mesafelere iletilebilir [27]. Bu da daha uzun

mesafelerde daha hassas ölçüm yapılabilmesine olanak tanır. Bununla birlikte lazer kısa süreli atımlar gerçekleştirebildiğinden ölçüm yapılacak mesafe için çok kısa sürede oldukça fazla örnek alınmasına olanak sağlar.



Şekil 4.4. Lazer mesafe ölçer çalışma şeması

Lidar'ın çalışma mantığı ise yukarıdaki bilgileri göz önüne alarak şöyle açıklanabilir. Her bir lazer atımında oluşan eş evreli, kısa dalga boylu ışık demeti atıldığı yönde doğrusal yol boyunca ilerlerken bir engele çarparak yansır ve sensör üzerindeki algılayıcı tarafından algılanır. Işığın atım anından engele çarparak yansıması ve algılayıcı tarafından algılanmasına kadar geçen süre ışığın uçuş süresidir. Uçuş süresi ışığın hızıyla çarpılır ve mesafe ölçümü yapılır.

$$d = \frac{c \cdot t}{2} \tag{4.1}$$

Burada, *d* ölçülen mesafe (mm), *c* ışık hızı (m/s), *t* (s) ise geçen süre olarak ifade edilir. Şekil 4.5'de bu çalışmada kullanılan lidar gösterilmiştir [28].

Eğer lidar bir eksende 360 derece dönebilirse, sensör bulunduğu ortamda çevresinde olanlar ile arasındaki mesafeyi ölçmek için kullanılabilir. Buradan anlaşılacağı üzere lidar ve radar cihazları benzer şekilde çalışırlar. Ancak aradaki fark, algılama için kullanılan sinyalden kaynaklanır. Radar, lidar'dan farklı olarak radyo sinyalleri kullanır ve radyo sinyalleri bazı materyallerden yansırken bazı materyaller tarafından emilebilir ve yansımaz. Bu durum bugünkü hayalet uçak teknolojisinin gelişmesine olanak tanımıştır. Lazerler ise ışık demetleri kullanarak

ölçüm yaptığından böyle bir durum söz konusu değildir. ve ölçüm kabiliyeti radarlara oranla daha yüksektir.

Şekil 4.5. Ortam haritalaması ve uzaklık ölçümü için kullanılan lidar

4.2. Aktüatörler

Aktüatörler, diğer bir adıyla eyleyiciler ya da tahrik ediciler, robotlara hareket kabiliyeti sağlayan sistemlerdir. Bir robot, kendi üzerinde bulunan sensör veya sensörler ile çevreyi algılarken yine kendi üzerinde bulunan aktüatör veya aktüatörler ile çevrede hareket edebilir ve bu sayede çevrede oluşabilecek değişikliklere tepki sağlarken farkındalık düzeyini arttırabilir.

Mobil robotlar için en sık kullanılan aktüatörler elektrik motorlarıdır. Bunun sebebi ise elektrik motorlarının, bulundukları ortamın parametrelerine daha az bağımlı olması ve kolay kontrol edilebilmeleridir. Bunun yanı sıra aynı güçteki yakıtlı motorlara nazaran daha küçük boyutlara ve düşük ağırlığa sahip olması ile çalışma ömrünün daha uzun olmasıdır.

Elektrik motorları kullandığı elektrik enerjisi türüne bağlı olarak ikiye ayrılırlar. Bunlar AC ve DC elektrik motorlarıdır. AC motorlar daha yüksek verime ve güce sahip olmasına rağmen robotikte en çok kullanılan elektrik motorları DC elektrik motorlarıdır. Robotlar, istisnalar hariç, genellikle üzerlerinde bulunan DC bataryalardan beslenerek görevlerini yaparlar [29]. DC elektrik motorları da bu enerji kaynağı için en uygun seçenektir.

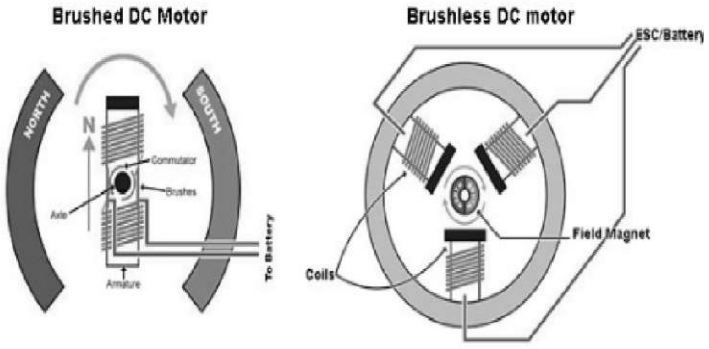
DC motorlar kendi içlerinde farklı türlere ayrılırlar; bunlardan bazıları şunlardır. Fırçalı motorlar (Brushed), Fırçasız motorlar (Brushless), Konum kontrollü motorlar (Servo), Adım motorları (Step) vs. Mobil robotlar için kullanılacak başlıca aktüatörler robotun bulunduğu ortamda hareketini sağlayacak olan fırçalı veya fırçasız motorlardır.

4.2.1. Fırçalı motorlar

Tipik bir fırçalı motor, yapısal olarak biri sabit olan üzerinde mıknatıslar bulunan stator ve elektrik enerjisi verilince dönmeye başlayan bir armatürden (rotor) oluşur. Armatürün dönmesine sebep olan güç ise elektrik enerjisi verilince bir mıknatısa dönüşüp elektrik alan yaratan sargılardır. Fırçalı motorun böyle anılmasına sebep olan yapı ise elektrik akımını dönen armatürün komütatörüne aktaran fırçalardır. Fırçalar, oluşan elektrik alanın kutuplarını dönüş yönüne bağlı olarak değiştirdiği için bu tip motorlarda hareketi sağlayan temel parçalardan biridir. Basit olarak ifade edilecek olursa,

$$P_{elektrik} = P_{\text{ısı}} + P_{mekanik} \tag{4.2}$$

Burada, $P_{elektrik}$ motora uygulanan giriş gücü, $P_{\text{ısı}}$ motorun ısıl kaybı, $P_{mekanik}$ harekete dönüştürülen giriş gücü olarak ifade edilir.



Şekil 4.6. Fırçalı ve fırçasız motorların yapıları

Fırçalı motorların avantajları ve dezavantajları bu çalışma yönünden avantajları ve dezavantajları Çizelge 4.1 ve Çizelge 4.2'de gösterilmiştir.

Çizelge 4.1 Fırçalı motorların avantajları ve dezavantajları

Avantajları	Dezavantajları
Üretimi kolaydır	Fırçalar kolay yıpranır
Ucuzdur	Ses düzeyi yüksektir
	Fırçalar motorun maksimum hızını sınırlar
	Sargılar motorun ortasında olduğundan soğutmak zordur
	Fırçalar motorun kutup sayısını sınırlar

4.2.2. Fırçasız motorlar

Fırçasız motorlar temelde DC motor olsalarda fırçalı motorlardan farklı bir yapıya sahiptirler. Fırçalı motorlarda sabit mıknatıslar stator üzerinde yer alırken fırçasız motorda sabit mıknatıs rotor üzerinde yer almaktadır. Aynı şekilde enerjilendiğinde elektromıknatıs görevi görecek olan sargılar da stator üzerine taşınır.

Çizelge 4.2 Fırçasız motorların avantajları ve dezavantajları

Avantajlar	Dezavantajlar
Sürtünme kayıpları yoktur	Fiyatları pahalıdır
Soğutulması kolaydır	
Daha hasssas kontrol edilirler	
Çok az ses çıkarırlar	

Böylelikle rotora elektrik akımı aktarmaya gerek kalmaz ve fırça kullanılmaz. Fırçaların oluşturduğu sürtünme kayıpları bu şekilde ortadan kaldırılırken aynı zamanda motorun maksimum hızını mekanik olarak sınırlayan yapı da ortadan kalkar. Kayıpların azalması motorun verimini arttırır, ayrıca motorda kullanılan sabit mıknatıs sayısı arttırıldığından fırçasız motorlar daha hasssas kontrol edilebilirler. Bununla beraber fırçasız motorları sürmek fırçalı motorlardan daha karmaşık bir sürücü devresi gerektirir.

Her ne kadar fırçasız motorlar daha yüksek verime sahip olsa da bu uygulamada gerek maaliyeti gerekse sürüş kolaylığı dolayısıyla fırçalı DC motor kullanılacaktır. Gerçekleştirilmek istenilen uygulamada kullanılacak motor önemli

bir farklılık yaratmayacağından bu karara varılmıştır. Şekil 4.7'de bu çalışmada kullanılan motor gösterilmiştir [30].

Kullanıacak motorun seçimi ise şu şekilde yapılmıştır. İlk olarak mobil robotun mekanik tasarımı bitirilmiş ve tahmini bir ağırlık belirlenmiştir. Uygulamada hız önemli bir parametre olmadığından, seçilen motorun dişli kutusu olması tercih edilmiştir. Bu şekilde yüksek tork değerleri elde edilebilmesi sağlanmıştır. Tasarlanan robotun odometri verisine ihtiyacı olması sebebiyle, robot için tercih edilecek motorların üzerinde artımsal quadrature enkoder bulunması tercih edilmiştir. Robot üzerinde artımsal enkoder olarak quadrature encoder kullanılmasının sebebi ise hem bu tip enkoderlerin daha hassas olması hem de enkoderden gelen veriye göre motorun dönüş yönünün tespit edilebilmesidir. Tercih edilen motorlar üzerinde bulunan enkoderler, optik enkoder olmayıp, Hall etkisi ile çalışan enkoderler olması tercih edilmiştir. Bu tercihin sebebi ise optik enkoderlerin çevresel şartlardan dolayı kirlenip yanlış veriler verebilmesidir.

Şekil 4.7. Bu çalışmada robota tahrik sağlamak için kullanılan motor

5. NAVİGASYON UYGULAMASI İÇİN MOBİL ROBOT TASARIMI

Mobil robotlar daha önce bahsedildiği gibi elektromekanik cihazlardır. Her elektromekanik sistemde olduğu gibi mobil robot tasarımında da hem elektronik hemde mekanik tasarım robotun kullanılacağı alana göre tasarlanıp üretilmelidir. Yani mobil bir robot alana özgü tasarlanıp üretilmelidir. Ancak bu şekilde robot kusursuzca görevini yerine getirebilir. Bu çalışmada kullanılan mobil robot, üzerindeki sensörler ve kullanılan algoritmalarla çevresini haritalayıp harita üzerinde kendisini konumlayabilme kabiliyetine sahip olup haritalamanın ardından kendi üzerinde bulunan aktüatörler yardımı ile kullanıcı tarafından harita üzerinde seçilen bir noktaya otonom olarak navigasyon planlayıp hareket edebilmektedir. Robotun görevini icra edeceği ortam ise iç ortam olup düz bir zemine sahiptir.

Tasarlanan robot daha önce de bahsedildiği gibi fiziksel olarak mekanik ve elektronik donanıma sahiptir. Tüm sistemin daha iyi anlaşılabilmesi için hem mekanik hem de elektronik tasarım ayrı ayrı açıklanacaktır.

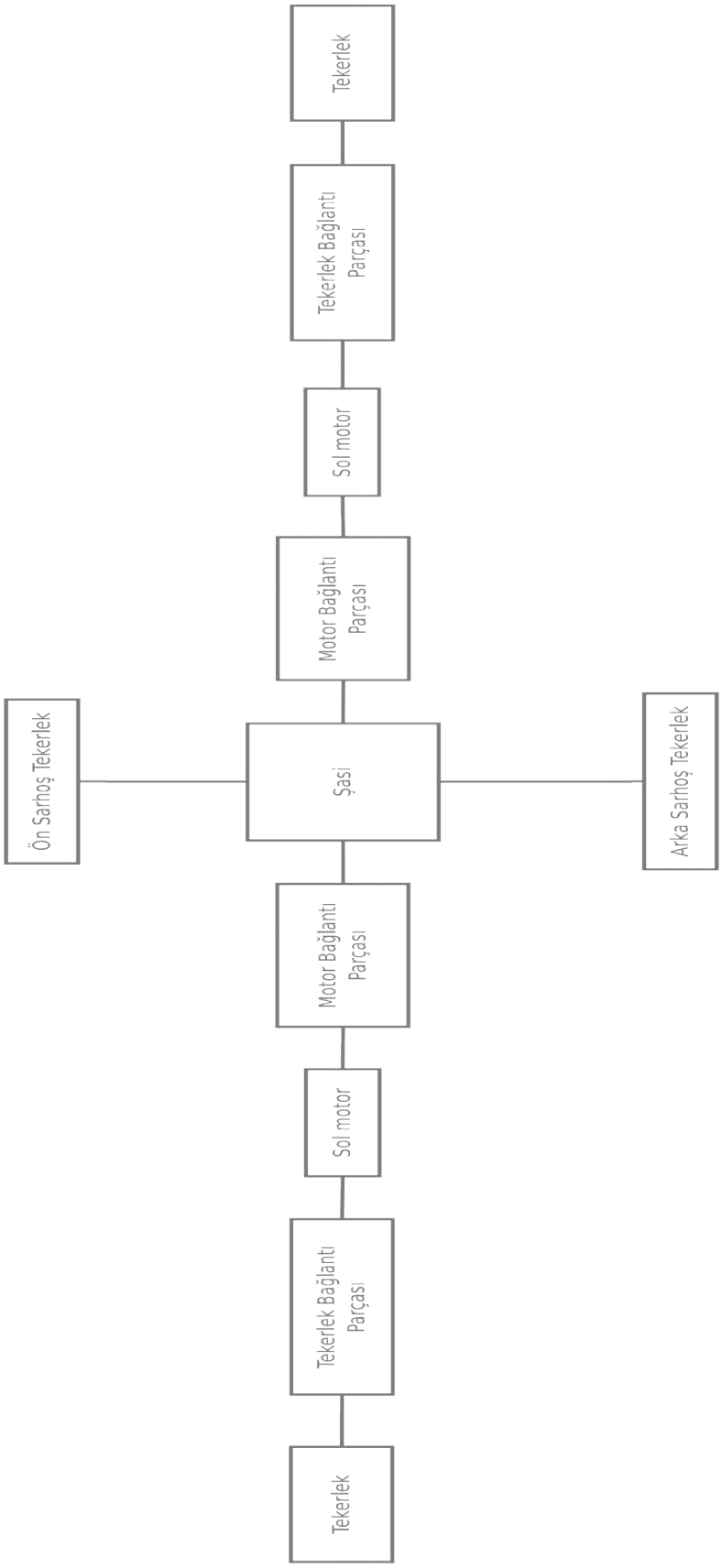
5.2. Mekanik Tasarım

Çalışmada kullanılan mobil robotun tasarımının ilk aşamasını mekanik tasarım oluşturmaktadır. Bunun nedeni robotun hareket edeceği fiziksel ortamın doğrudan robotun sürüş sistemini etkilemesidir. Bununla birlikte aşırı ağır tasarlanan mekaniğin robotun fazladan enerji tüketerek operasyon süresini kısaltacağı göz önünde bulundurulmalıdır.

Bu çalışmada kullanılacak robot iç ortamda kullanılacağından mekanik tasarımda dış ortamda çalışan robotlarada olduğu gibi karmaşık sistemler kullanılmamıştır. Mekanik sistemin tasarımında esas amaç robotun en kolay şekilde navigasyon yapabilmesi ve manevra kabiliyetinin en üst düzeyde tutulması olmuştur. Mekanik tasarım şase ve sürüş sistemi olmak üzere iki temel bileşenden oluşmaktadır. Geliştirilen robota ait mekanik bağlantı şeması Şekil 5.1'de gösterilmiştir.

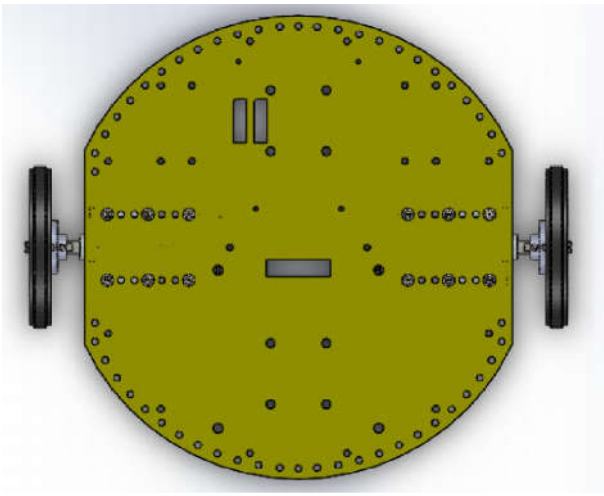
5.2.1. Sürüş Sistemi

Bu çalışmada kullanılacak robotun sürüş sistemi belirlenirken esas amaç robotun en yüksek seviye de hareket ve manevra kabiliyetine sahip olmasıdır. Robot, iç ortamda ve düz bir zeminde hareket edeceğinden, enerji tüketimini en az seviyede tutmak ve hızlı hareket sağlamak için sürüş sistemi tekerlekli bir sürüş sistemi olarak tasarlanmıştır.



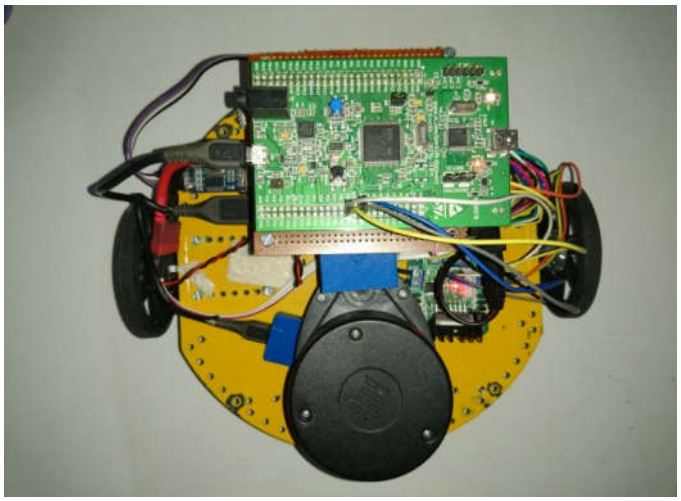
Şekil 5.1. Geliştirilen robota ait mekanik bağlantı diyagramı

Robotun bu ortamda önüne çıkabilecek engellerden en iyi şekilde kaçınabilmesi temel gereksinimlerden birisidir. Bu nedenle dönüş yarıçapının olabildiğince küçük olması gerekmektedir. Navigasyon sırasında robotun olduğu yerde geri dönmesi gerekebilir. Bu nedenle dönüş yarıçapı önemli hareket parametrelerinden biridir. Şekil 5.2 ve şekil 5.3'te bu çalışmada tasarlanan robota ait diferansiyel sürüş sisteminin robot şasesine montajlı hali gösterilmektedir.



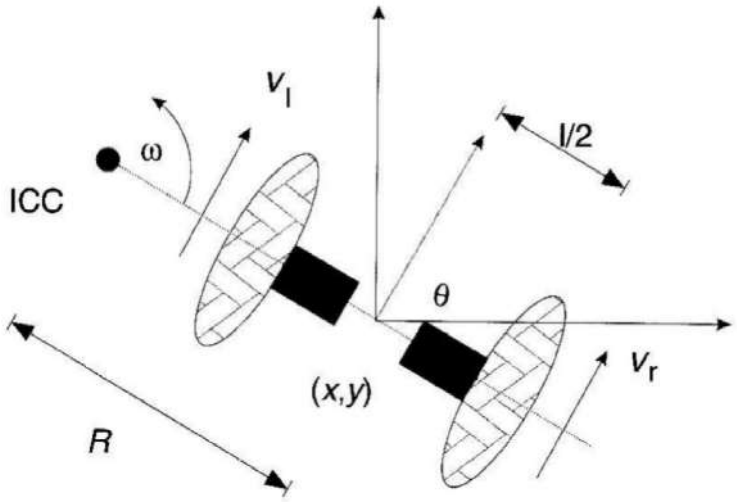
Şekil 5.2. Bu çalışma için tasarlanan sürüş sisteminin çizimi

Yukarıda bahsedilen özellikler göz önüne alındığında tasarlanan robotun en yüksek verim ve hızla hareket ederken en yüksek manevra kabiliyetine sahip olması için diferansiyel sürüş sistemi seçilmiştir.



Şekil 5.3. Tasarlanan sürüş sisteminin robot üzerindeki görünümü

Diferansiyel sürüş sisteminin seçilmesinini bir diğer nedeni robotik biliminde en çok kullanılan sürüş sistemlerinden birisi olmasıdır. Bu nedenle diferansiyel sürüş sistemi ile ilgili birçok bilimsel çalışma bulunmaktadır. Ayrıca diferansiyel sürüş sistemi için tasarlanan şase, sürüş sisteminin yapısı gereği basit bir yapıdadır. Bu da sürüş sisteminin mekanik montajını kolaylaştırmakla birlikte bu çalışmanın konusu olmayan birçok hesap yapılmasını önlemiştir. Tasarlanan mobil robot için diferansiyel sürüş sisteminin temel denklemi şekil 5.4'teki parametreler kullanılarak aşağıdaki şekilde ifade edilir.



Şekil 5.4. Mobil robot için diferansiyel sürüş kinematik görseli

$$\omega(R + l/2) = V_r \tag{5.1}$$

$$\omega(R - l/2) = V_l \tag{5.2}$$

Burada, V_r robotun sağ tekerlek hızını, V_l robotun sol tekerlek hızını, R robotun şasesinin tam orta noktasının dönüş merkezine olan uzaklığını, l robotun şasesinin ortasının sağ veya sol tekerleğin ortasına olan uzaklığını, ω robotun açısal hızını ifade eder.

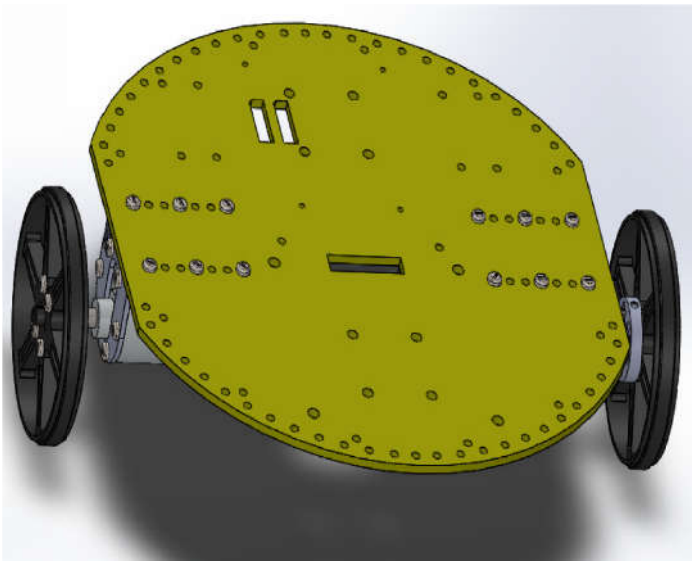
$$R = \frac{l}{2}\frac{V_l+V_r}{V_r-V_l}; \quad \omega = \frac{V_r-V_l}{l} \tag{5.3}$$

Denklemler incelenecek olursa diferansiyel sürüş sistemine sahip bir mobil robotun hareketi kısaca şöyle açıklanabilir:

1. Eğer $V_l = V_r$ olursa bu durumda robot düz bir hat üzerinde doğrusal hareket gerçekleştirir. Bunun sonucu olarak R değeri sonsuz olur ve ω sıfıra eşit olur.
2. Eğer $V_l = -V_r$ olursa bu durumda $R = 0$ olur. Robot tekerlek ekseninin orta noktasını merkez alacak şekilde olduğu yerde döner.
3. Eğer $V_l = 0$ olursa sol tekeri dönüş merkezi olur ve robot bu noktanın etrafında döner. Bu durumda $R = \frac{l}{2}$ olur. Bu durum $V_r = 0$ olduğunda da geçerlidir.

5.2.2. Şase

Şase, robotun sürüş sisteminin belirlemesinden sonra sürüş sistemini kolayca bağlayabilecek şekilde tasarlanmıştır. Ayrıca robotun iç ortamda ve düz zeminde hareket edeceği bilindiğinden çevresel koşullarda değişikliğe karşı robotu koruyucu herhangi bir önlem alınmamıştır. Robotun sürüş sisteminin diferansiyel sürüş sistemi olarak belirlendiği göz önüne alındığında şase, sistemin hareket kabiliyetini arttıracak şekilde tasarlanmıştır. Ayrıca robotun üzerine yüklenecek elektroniklerin ve diğer yüklerin ağırlığını taşıyabilmesi ve çarpmalara karşı dayanıklı olabilmesi için metalden üretilmiştir. Ayrıca ilerleyen zamanlarda robot üzerine eklenecek farklı donanımların bağlantılarını sağlamak amacı ile şasi üzerinde bağlantı noktaları oluşturulmuştur. Gerektiğinde motorların şasiden uzaklığının arttırılabilmesi için fazladan bağlantı noktaları eklenmiştir.



Şekil 5.5. CAD programı ile hazırlanan şase çizimi

Şase yapısı incelenecek olursa motorların şasenin ortasından eşit uzaklıkta iki tarafa yerleştirildiği ve bu şekilde diferansiyel sürüş sistemini oluşturduğu görülür. Motorların robotun tam ortasında iki yana yerleştirilmiş olması sebebiyle robot 360 derecelik dönüşü olduğu noktada, konumunu değiştirmeden yapabilmektedir. Bu da navigasyon sırasında robotun hareket kabiliyetine olumlu yönde katkı sağlamaktadır. Şekil 5.5'te bu çalışmada için tasarlanan robotun motor ve motora bağlı tekerlekleri gösterilmiştir.

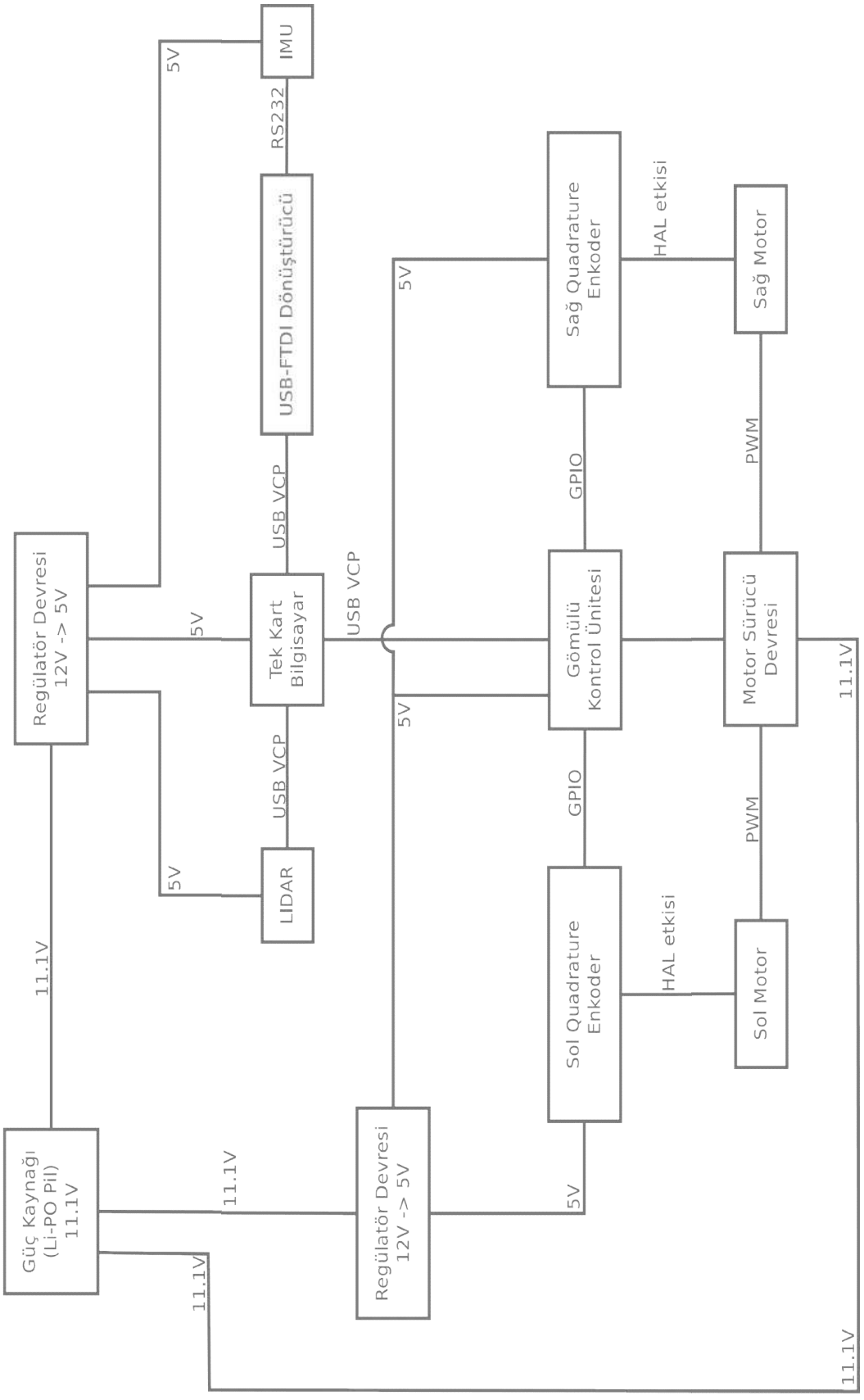
Şase üzerinde motorların yerleştirildiği konumlar gereği robotu dengede tutmak amacıyla şasenin önüne ve arkasına olmak üzere iki adet şarhoş tekerlek yerleştirilmiştir. Bu tekerleklerin herhangi bir tahrik gücü olmamakla birlikte tek görevleri robotu dengede tutmak ve motorlar üzerine binen ağırlığı hafifletmektir.

5.3. Elektronik Tasarım

Bu çalışmada gerçekleştirilen elektronik tasarım, robotun kontrol ve güç sistemlerinin tasarımı ile kullanılan sensör ve aktüatörlerin entegrasyonunu kapsamaktadır. Elektronik tasarım bu çalışmanın en karmaşık yapılarından birini oluşturmakla birlikte geliştirme süreci en uzun süren bölümüdür. Bunun nedeni robotun gerek duyduğu tüm alt yapıyı sağlaması ile maaliyet etkin bir yapı geliştirme gereksinimidir. Geliştirilen elektronik tasarım parçalara ayrılarak anlatılacak ve bu sayede entegrasyonun nasıl sağlandığı ifade edilecektir.

5.3.1. Kontrol sistemi elektronik alt yapısı

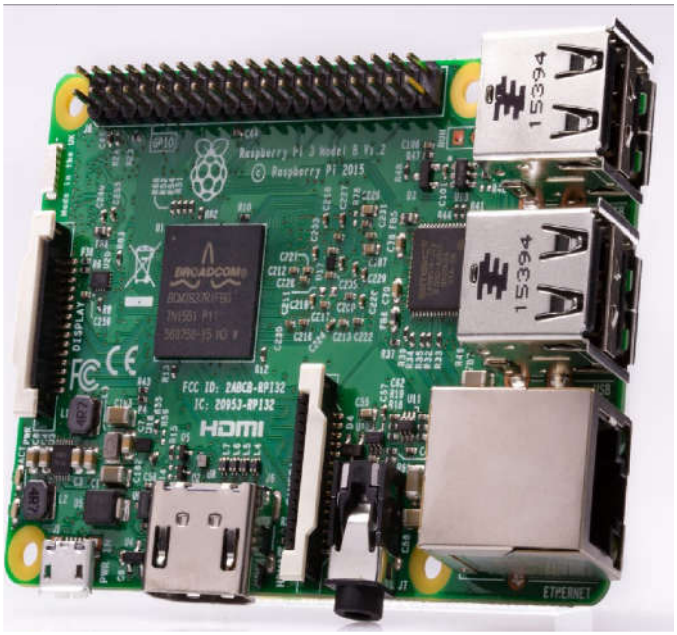
Mobil bir robotun navigasyon planlamasını yapıp planlanan navigasyonu harekete çevirmesi için gerekli birçok yazılımsal algoritma vardır. Bu algoritmaların çalışacağı fiziksel donanım bu çalışma için kontrol sistemi elektronik alt yapısı olarak ifade edilmektedir. Tasarlanan kontrol sistemi elektronik alt yapısı iki farklı kontrol ünitesi içermektedir. Bu ünitelerin her birinin hangi amaçla kullanılacağı daha sonra ifade edilecektir. Ancak bu yapılardan biri mikrodenetleyici bir kontrol ünitesi diğeri ise mikroişlemci tabanlı bir kontrol ünitesidir. Her iki ünite ayrı ayrı açıklanacaktır.



Şekil 5.6. Geliştirilen robota ait elektronik bağlantı diyagramı

a) Tek Kart Bilgisayar (Single Board Computer): Kontrol ünitelerinin biri mikroişlemci tabanlı bir tek kart bilgisayardır. Robot için tek kart bilgisayar seçilmesinin nedeni robot için kullanılacak yazılımların bazılarının bir işletim sistemi üzerinde çalışmasıdır. Özellikle robota otonom niteliği kazandıran yazılımların büyük çoğunluğu bu işletim sistemini kullanarak donanımlara erişen yazılımlardır. Şekil 5.6'te bu çalışma için tasaranan tek kart bilgisayar merkezli donanım tasarım şematiği gösterilmiştir.

Bir başka neden ise bu tür bilgisayarların, kişisel bilgisayarlardaki neredeyse her donanımı üzerinde barındırması ve maaliyetinin daha az olmasıdır. Robot üzerinde bulunan tek kart bilgisayar üzerinde sadece robotun kullanılma amacına özel uygulamalar çalıştırıldığından kişisel bilgisayarlardaki performansa ve kaynağa ihtiyaç yoktur. Ayrıca tek kart bilgisayar üzerindeki donanımlar kişisel bilgisayarlardaki kadar fazla kaynağa ve donanıma sahip olmadığından harcadıkları güç de daha azdır. Bu sayede robotun operasyon süresinin uzatılmıştır. Robot üzerinde kullanılan tek kart bilgisayar modeli günümüzde akademik ve endüstriyel uygulamalarda oldukça fazla kullanılan ve dökümantasyonu iyi olan Raspberry Pi3 modeli bir bilgisayardır. Şekil 5.7'de bu çalışmada kullanılan tek kart bilgisayar gösterilmiştir [31].



Şekil 5.7. Çalışmada kullanılan tek kart bilgisayar (SBC) Raspberry Pi 3

Bu çalışma için tek kart bilgisayarın üzerinde bulunan USB 2.0, WiFi ve GPIO donanımları kullanılmıştır. Bu donanımlar sistemin çalışması sırasında farklı görevleri birbiri ile bağlantılı şekilde yerine getirmektedir. Tek kart bilgisayar üzerinde kullanılan donanımları ve kullanım amaçlarını detaylı olarak şöyle açıklayabiliriz:

b) USB 2.0: USB 2.0, 2000 yılında kullanılmaya başlanmış elektronik cihazlar arasında kullanılan çok hızlı bir haberleşme protokolüdür. Bilgisayarlar başta olmak üzere günümüzde birçok cihaz bu veri yolunu kullanarak kolay bir şekilde birbirine bağlanıp haberleşebilir.

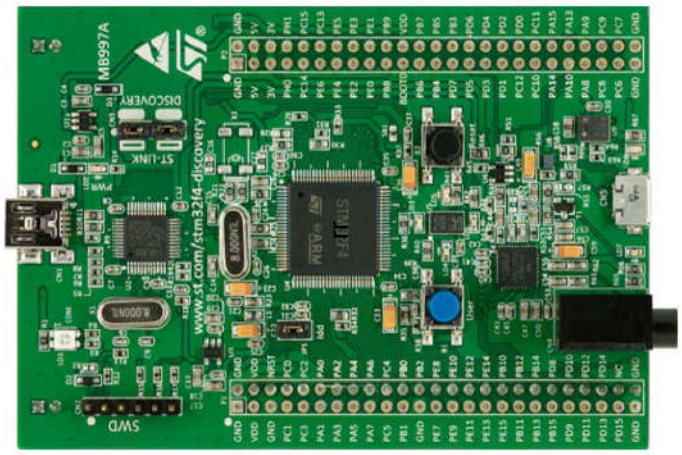
Bu uygulama için kullanılan Lidar, Gömülü Kontrol Unitesi ve IMU, tek kart bilgisayara USB 2.0 aracılığı ile bağlanmıştır. Bunun nedeni bahsi geçen tüm donanımların seri haberleşme kullanarak haberleşmesi ve bu haberleşmeyi USB 2.0'ın çalışma geriliminde yapabilmesidir. Anlaşılacağı üzere bu cihazlar USB 2.0'a (Communication Device Class) haberleşme sınıfında cihazlar olarak bağlanmıştır.

c) WiFi: Bu çalışma için tasarlanan sistem ortamda bulunan kablosuz ağa bağlanıp kullanıcı ile karşılıklı veri alışverişi sağlayabilir. Bu sayede sistemin durumu ve operasyonun ilerleyişi kullanıcı tarafından takip edilebilir. Ayrıca çalışma sırasında sistem üzerinde güncelleme yapılabilir. Sistemin kullanıcı ile haberleşmesi WiFi üzerinden ve kablosuz olacak şekilde tasarlanmıştır. Bunun nedeni WiFi üzerinden yapılan haberleşmenin yapının yüksek hızından (2.4 GHz - 5 GHz) dolayı gerçek zamanlıya yakın olarak gerçekleşmesidir. Bununla beraber WiFi kullanarak iç ortamda geniş bir satıh üzerinde haberleşme sağlanabilir. Robot, WiFi kullanarak kendi sistemi üzerinde koşması zaruri olmayan uygulamaları ağ üzerinde bir başka bilgisayarla paylaşabilir. Bu sayede robotun üzerindeki iş yükü azalırken sistemin hızı arttırılır.

d) GPIO: Bu çalışmada kullanılan tek kart bilgisayar birden fazla robotik algoritmasını çalıştırdığından ve çalışma süresince toplanan veriyi işleme tabi tuttuğundan bilgisayarın özellikle işlemcisinde ciddi miktarda ısınma problemi ortaya çıkar. Isınma problemi her elektronik sistemde aşılması gereken başlıca problemlerden biri olduğu gibi bu çalışmanın konusunu oluşturan robotun bilgisasyarı için de sorundur. Bu sistemde robot üzerinde bulunan gömülü bilgisayarın soğutulması için bir fan kullanılmıştır. Bilgisayarın üzerindeki yük ve ısınma durumuna göre bu fan ile soğutma işlemi yapılmıştır. Fan, az enerji tüketimi gerçekleştirse de verimliliğini arttırmak ve gereksiz enerji kaybını önlemek için artan

yük ile ısıya bağlı oransal bir kontrolcü(P) ile fanın hızı kontrol edilmiştir. Bu kontrol işlemi sırasında fana uygulanması gereken sinyal GPIO arayüzü aracılığı ile PWM (Pulse Width Modulation) sinyali olarak uygulanmıştır.

e) Gömülü Kontrol Ünitesi: Yukarıda da belirtildiği üzere bu çalışma için birden fazla kontrol ünitesi kullanılmıştır. Bunlardan biri olan gömülü kontrol ünitesi mikrodenetleyici tabanlı bir elektronik devre kartıdır. Bu ünitenin görevi tek kart bilgisayara bağlanamayan donanımların sistemle haberleşmesini sağlamak ve kontrol edilen çevresel birimlerin entegre çalışmasını sağlamaktır. Yapı olarak tek kart bilgisayardan farklı olan gömülü kontrol ünitesi içerisinde bulunan yazılım yardımı tek kart bilgisayar ile haberleşerek robotun hareketini yürütüp hareketi sağlayan sistemler ile sensörlerin koordinasyonunu yönetir. Şekil 5.8'de bu çalışmada gömülü yazılım ünitesinin kalbini oluşturan stm32 discovery kartı gösterilmiştir [32].

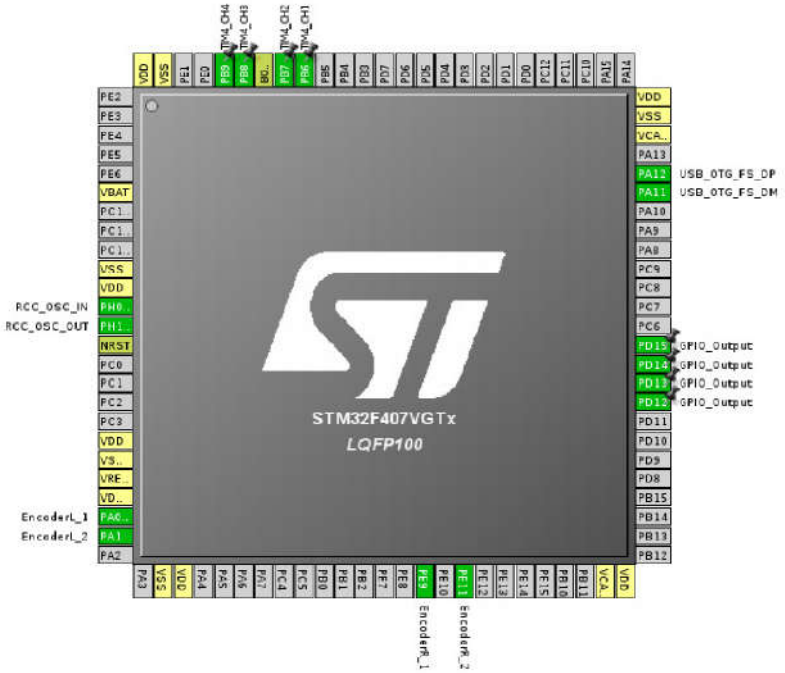


Şekil 5.8. Çalışmada kullanılan gömülü kontrol ünitesi STM32F4 discovery

Gömülü kontrol ünitesine bağlı olan donanımlar şunlardır: Motor sürücüler, enkoderler, motorlar, güç düzenleme ünitesi, tek kart bilgisayar. Tüm bu donanımların entegrasyonu için gerekli olan yazılımlı ve algoritmaları hazırlamak elektronik tasarım aşamasının en uzun süren bölümlerinden biridir. Bunun en büyük nedeni yönetilen donanımların bazılarının hassas bir zamanlama ile kontrol edilme gerekliliğidir. Bu gereksinimi sağlamak için bu ünite içerisinde buna yönelik yazılımlar kullanılmıştır. Şekil 5.9'da kullanılan mikro denetleyici pinleri hangi amaç için kullanıldığı ile birlikte gösterilmiştir.

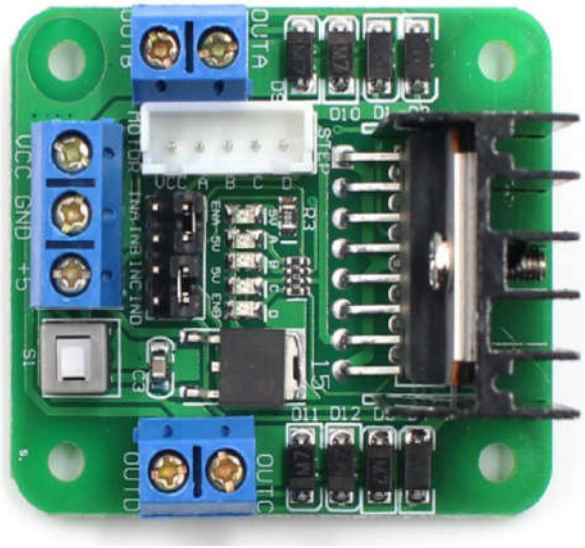
f) Motor Sürücü Devresi: Mikroişlemci ve mikrodenetleyicilerin fiziksel giriş lerindeki ve çıkışlarındaki sinyaller nano amper (nA) ile mili amper (mA)

düzeyindedir. Bu nedenle mikroişlemci veya mikrodeneleyicilerin kumandası ile motor sürülmek istenildiğinde bu çipler tarafından sağlanan akım motoru süremeyecek kadar küçük kalır. Bunun yanısıra doğrudan motorun gücünü sağlayacak şekilde motora bağlanması sonucu mikrokontrolcü veya mikroişlemci, motorun akım ihtiyacı dolayısıyla verebileceğinden daha fazla akım vermeye mecbur bırakıldığından zarar görebilir. Dolayısı ile kumanda ünitesinden gelen sinyalin bir yükselteç vasıtası ile motora uygulanması gerekir. Ancak bu şekilde motor sürülebilir. Motor sürücü devresi, mikroişlemci veya mikrodenetleyici tabanlı DC veya AC motor sürmek için tasarlanan tüm devrelerde kullanılan bir devredir. Bu devrenin görevi kumanda ünitesi tarafından sağlanan sinyalin motorun ihtiyaç duyacağı seviyelere yükseltimini sağlamaktır. Bu sayede motor stabil olarak sürülebilir.



Şekil 5.9. Gömülü kontrol bilgisayarı üzerindeki STM32F407 mikrodenetleyicisinin bağlantı şematiği

Motor sürücü devresi, sürülecek motora uygun olarak tasarlanır. Yani farklı motor türleri için farklı motor sürücülere ihtiyaç vardır. Her motor türü için farklı bir motor sürücüsü tasarlanmasının nedeni motorun iç yapısı ve güç ihtiyacıdır.



Şekil 5.10. Bu çalışmada kullanılan fırçalı motor sürücü devresi

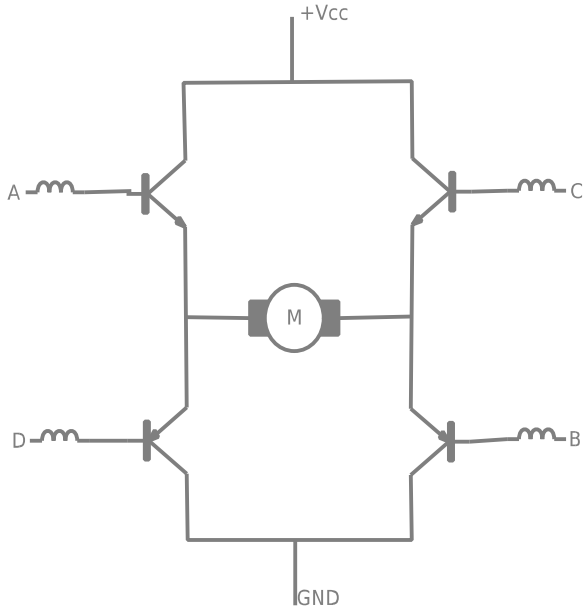
Bu çalışmada geliştirilen robotta kullanılan motorlar fırçalı motorlar olduğundan fırçalı motorlara uygun bir motor sürücü kullanılmıştır. Fırçalı motorları sürmek için en çok kullanılan motor sürücü H-Bridge olarak bilinen motor sürücüdür. Şekil 5.10'da bu çalışmada kullanılan motor sürücü devresi gösterilmiştir [33].

Şekil 5.11'de görüleceği üzere H-Bridge temelde dört transistörden oluşur. Bu transistörlerin anahtarlanma sıralamasına göre motora belli bir yönde gerilim uygulanır. Dolayısıyla motor dönmeye başlar. Şayet anahtarlanan transistör devrenin dinamiklerine uygun olarak değiştirilirse motor aksi yönde dönmeye başlar. Bu sayede motor farklı yönlerde sürülebilir ki bu durum mobil robotik için oldukça önemli bir durumdur.

H-Bridge devresi sadece yön kontrolü yapmakla kalmaz. Bu devre sayesinde motor istenilen yönde farklı hızlarda sürülebilir. Bunun için anahtarlanan transistörlere PWM (Pulse Width Modulation) yani darbe genişlik modulasyonu uygulanır. Uygulanan sinyalin doluluk oranına göre motorun dönme hızı değişir.

PWM, dijital sistemlerin yaygınlaşması ile kullanılmaya başlayan ve analogdan dijitale sinyal dönüşümünde en çok kullanılan yöntemlerden biridir. Bilindiği üzere dijital sistemlerin çalışması logic olarak "1" ve "0"a dayalıdır. Yani sisteme ait bir sinyal belirli bir anda ya "1" ya da "0" olabilir. Bu nedenle sisteme ait giriş veya çıkış sinyali bu değerlerin arasında bir değer alamaz. Bu da sistem tarafından üretilen bir sinyalin bu aralıkta olmasına engel teşkil eder. İşte dijital sistemlerde bu sorunu

çözmek ve sistemin belirli iki değerin dışında ve arasında sinyal üretmesini sağlamak amacıyla PWM kullanılır.



Şekil 5.11. Geliştirilen robotun motorlarını sürmek için kullanılan L298 entegresindeki H-Köprüsü benzetimi

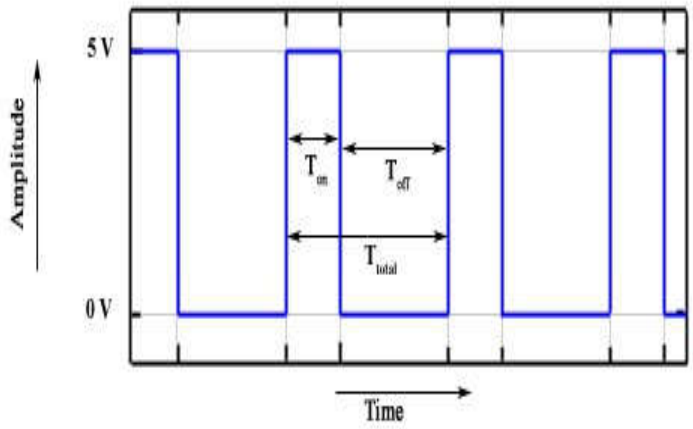
Dijital bir sinyal tipini tanımlayan PWM, dijital sistemler tarafından üretilen sinyallerin çıkış değerlerini analog değerlere yakınsamak için kullanılır. Kısaca açıklayacak olursak, PWM sinyal çıkışı olan dijital sistemlerin; çıkışlarının genliklerini 1 ve 0'lardan farklı hale getirerek bu iki değerin arasında genliğe sahip çıkış sinyalleri üretimini sağlar. Bu sayede bir sistemi kumanda edecek olan bir sinyal için optimum çıkış genliği değerleri sağlanmış olur. PWM mantığının anlaşılabilmesi için temel kavramların bilinmesi gerekir. Bir PWM sinyali için temelde iki kavram söz konusudur. Bunlar Frekans ve Doluluk oranıdır (Duty cycle). Frekans, PWM sinyali için sinyalin periyoduna bağlı olarak belirlenen bir parametredir. Bir sinyalin PWM kullanılarak modüle edilebilmesi için her şeyden önce sinyalin periyodik olması gerekir. Ancak bu şekilde sinyal istenilen oranda analoga yakınsanabilir. Periyodik bir sinyalde periyot süresi bellidir. Frekans ile periyot arasındaki ilişki ise bilindiği üzere,

$$F = \frac{1}{T} \tag{5.4}$$

denklemi ile ifade edilir. Burada, *F* frekans, *T* ise periyottur.

Bir sisteme uygulanacak PWM sinyali için en önemli kriter sinyalin uygulanacağı sistemin frekans aralığının bilinmesidir. Çünkü PWM sinyali şayet sistemin frekans aralığının dışında kalırsa bu durumda sinyal sistem üzerinde kullanılamaz, kullanılsa dahi istenilen sonucu vermez. Bu çalışmada PWM sinyali, aktüatör görevi gören motorları sürecek olan motor sürücülere uygulandığından dolayı PWM frekansı motor sürücülere uygun aralıkta belirlenmiştir.

Doluluk oranı, PWM sinyalinin çıkışını belirleyen en önemli faktördür. PWM sinyalinde çıkış, sinyalin bir döngüsündeki doluluğun tüm dalga boyuna oranı olarak ifade edildiğinden doluluk oranı değiştikçe sinyalin çıkışında da değişiklik gözlenir. PWM kontrolü bu şekilde gerçekleştirilmiş olur. Bir PWM sinyali [34] figür 5.1'de gösterildiği gibidir.



Şekil 5.12. PWM Sinyali

Şekil 5.12'deki PWM sinyalinde gösterildiği üzere sinyal periyodiktir.
Sinyalin periyodu:

$$T_{PWM} = T_{ON} + T_{OFF} \tag{5.5}$$

Burada, *T* Periyottur. Doluluk oranı, *D*, ise aşağıdaki şekilde ifade edilir:

$$D = \frac{T_{ON}}{T_{ON}+T_{OFF}} \tag{5.6}$$

Dikkat edilirse sinyalin doluluk oranı bir periyot boyunca sinyalin logic 1 olduğu sürenin tüm sinyal periyoduna oranıdır.

$$D = \frac{T_{ON}}{T_{TOTAL}} \tag{5.7}$$

Yukarıda anlatılan frekans ve doluluk oranı ile birlikte BİR PWM sinyalinin çıkış gerilimi sinyalin voltaj regulasyonuna bağlıdır. Bunu matematiksel olarak ifade edecek olursak:

$$V_{out} = D \text{ x } V_{input} = \frac{T_{ON}}{T_{TOTAL}} \text{ x } V_{input} \tag{5.8}$$

PWM sinyalleri yukarıdaki formüllerden anlaşılacağı üzere sinyalin dolu kısmının tüm sinyale ortalamasıdır. Bu yüzden herhangi sinyalin ortalama gerlimini hesaplamakta kullanılan ortalama değer formülü ile de sinyalin çıkış gerilimi ifade edilebilir.

$$V_{ortalama} = \frac{1}{T} \text{ x } \int_0^T V(t)dt \tag{5.9}$$

$$V_{ortalama} = V_{pwm} = \frac{1}{T_{pwm}} \text{ x } \int_0^{T_{pwm}} V(t)dt \tag{5.10}$$

5.4. Yazılım

Robotik Bilimi, birbiri ile çalışan birden fazla bilimi içeren multidisipliner bir alandır. Bu nedenle herhangi bir yapıya göre daha karmaşık bir yapısal bütünlüğe sahiptir. Yazılım ise günümüz robotiğinde karmaşık olan ve giderek karmaşıklaşan bu yapıyı yönetmekle yükümlü en önemli alandır. Günümüzde her ne amaçla tasarlanmış olursa olsun robotlar, yoğun yazılım alt yapısına ve algoritmalara sahiplerdir.

Bu çalışmada tasarlanan otonom mobil robot için mekanik ve elektronik tasarımın yanısıra yazılımsal bir tasarım süreci izlenmiştir. Bilindiği üzere çalışma için tasarlanan robot bir adet tek kart bilgisayar ve bir adet gömülü kontrol ünitesi olmak üzere iki farklı karar verme ünitesine sahiptir. Bu nedenle yazılım tasarımı ve

gerçeklenmesi bu iki üniteyi birlikte ve senkron çalıştıracak şekilde ayrı ayrı yapılmıştır. Ayrıca birbirinden farklı yazılım geliştirme süreçlerine ve yapılarına sahip olduğundan bu süreçler ayrı ayrı açıklanacaktır.

5.4.1. Tek kart bilgisayar için yazılım geliştirme süreci

Bu çalışmada kullanılan tek kart bilgisayar robotun karar verme merkezi olarak görev yapar yani sistemin kontrolü temelde bu birim üzerinde koşan yazılımlara ve algoritmalara bağlıdır. Bu ünite birbirinden farklı birçok algoritmayı ve yazılım parçacığını üzerinde barındırdığından süperloop olarak bilinen ve tüm yazılım parçacıklarının aynı döngüde çalıştırılmasına uygun değildir. Çünkü bu tür bir yazılımda döngü içerisindeki yazılım parçacıklarının sayısı arttırıldığında sistem gerekli hassasiyette kontrol ve kumanda sağlayamaz bu da sistemin hantallaşmasına ve yapının çökmesine sebep olur. Bu yüzden süperloop bu çalışma için uygun değildir.

Bu çalışmada tek kart yazılım üzerinde koşturulmak üzere tasarlanan tüm yazılımlar robotik için hazırlanmış özel bir ara işletim sistemi kullanılmıştır. Bunun sebebi robotik uygulamaları için tasarlanan özel yazılım paketleri olan ara işletim sistemlerinin karmaşık algoritmaların bir arada çalışmasına olanak tanıyan bir yapıya sahip olmalarıdır.

Robotik uygulamaları için tasarlanmış bu yazılım sistemerine ara işletim sistemi denilmesinin nedeni bu sistemlerin doğrudan, çalıştırıldığı bilgisayarın donanımsal birimlerine erişimemesi ve kontrol kumanda sağlayamamasıdır. Bu yüzden bu sistemler uyumlu bir işletim sistemi üzerine kurularak çalıştırılmalıdır. Yani çalışmaları gerçek bir işletim sistemi ile birlikte ve entegre yürütülebilir. Anlatılan ara işletim sistemlerinin mantığı ve çalışma biçimlerinin anlaşılması bakımından bu konuyu ayrı bir başlık altında incelemek gereklidir.

5.4.1.1. Robot uygulamaları için özel yazılım setleri

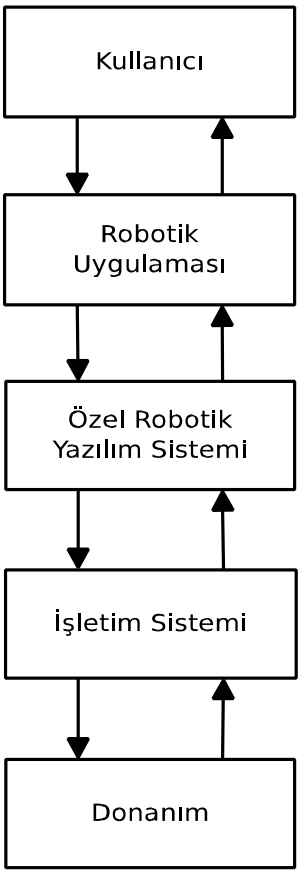
Robotik, günümüzde hem mekanik hem elektronik hem de yazılım alanlarında en son teknolojiyi kullanan çoğu zaman birçok yeni teknolojinin gelişmesine ön ayak olan ve kullandığı teknolojiler bağlamında giderek karmaşıklaşan bir bilim dalıdır. Gün geçtikçe karmaşıklaşan bu yapı çoğu zaman bir robotik sisteminin hatasız ve

hassas çalışmasına engel oluşturabilecek aksaklıkları beraberinde getirmektedir. Bir robotik sisteminin temelini oluşturan elektromekanik sistemler ile bu elektromekanik sistemi kontrol ve kumanda eden yazılımın kusursuz ve sorunsuz bir şekilde entegre edilmiş olması sistemin stabilitesini arttıran en önemli etmenlerden biridir.

Bir robotik sisteminin elektromekanik tasarımında sistemin çevreyi algılamasına yarayan çok sayıda sensör, bu algıyı değerlendirmesine yarayan kontrol ünitesi veya üniteleri ile çevreye tepkisini oluşturmaya yarayan aktüatörler bulunur. Bu elektromekanik sistem kendi içerisinde karmaşık bir yapı ile birbirine bağlıdır. Aynı şekilde bu sistem karmaşık yazılım algoritmaları ile kontrol edilmeye ihtiyaç duyar. Yazılım algoritmaları ise birbirine bağlı komplike bir yapıyı meydana getirir. Tüm bu durum robotun yapısında ortaya çıkabilecek bir hatayı veya problemin çözümünü zorlaştıracağı gibi geliştirme aşamasının uzamasına da sebep olur.

Günümüzde robotik algoritmaları her geçen gün biraz daha gelişmekte ve evrilmektedir. Bu yüzden güncel yazılım kütüphaneleri sürekli önem kazanmaktadır. Yani bir robotun tasarım ve üretim süreçleri bitse dahi her geçen gün yenilenen yazılım kütüphaneleri sayesinde robotun stabilitesi arttırılabilmektedir. Ancak bunun sistemli ve düzenli bir biçimde yapılması şarttır.

Yukarıda bahsedilen durumlar göz önüne alındığında robotik biliminin ulaşılmaz bir hızda gelişmesine paralel olarak robotik uygulamalarının hızlı şekilde gerçekleştirilebilmesi, gerek bilimsel araştırmalar gerekse endüstriyel kullanım için uygun hale getirilebilmesi amacıyla robotik için özel yazılım sistemleri geliştirilmektedir. Bunlar yukarıda bahsedilen hata ve problemlerin oluşmasına engel olmak, oluştuğu durumlarda kolaylıkla çözümlenebilmesine olanak sağlamak ve hızlı prototiplemeyi mümkün kılmak için tasarlanmış yazılımlardır. Bu yazılım sistemleri ayrıca donanımla en üst seviyede ki robotik yazılımlarının arasına bir katman çekerek kullanıcının donanımsal problemlerle uğraşmasını engellemektedir. Bunun sağladığı en büyük avantaj geliştirici veya kullanıcının donanım detaylarına takılmadan tasarım yapabilmesine ve yapılmış tasarımı uygulayabilmesine olanak sağlamasıdır. Ayrıca yazılımın kendi içerisindeki yapılar sayesinde robotun bileşenleri ile haberleşmek ve bu bileşenlerin birbiri ile haberleşmesini sağlamak daha kolaydır. Şekil 5.13'de robotik yazılım sistemi kullanılan bir robotik uygulamalar ile donanım arasındaki katmanlar gösterilmiştir. Bu çizimde ROS ve ROS'un diğer diğer bileşenler ile iletişimi gösterilmiştir.



Şekil 5.13. Özel robotik yazılım sistemi kullanılan bir robotta yazılım hiyerarşisi

Bu yazılımlar ayrıca kendi içerisinde simülasyon araçlarına sahiptir ve bu araçlar kulanılarak robotlar tasarım aşamasında test edilebilirken daha sonra robota uygulanacak özel algoritmaların simülasyonları yapılabilmektedir. Günümüzde oldukça fazla sayıda robotik için özelleşmiş yazılım paketi ve ara işletim sistemi vardır. Bunlardan bazıları Player Project, RT-Middleware Projects, Urbi, MIRO, Wasp Actuator Sensor Protocol, Orca2, OpenRDK, ROS ve Yarp'tır. Şekil 5.13'de Özel robotik yazılım sistemi ile robotik uygulamaların ve donanımın etkileşimi gösterilmiştir.

Günümüzde, yukarıda bahsedilen robotik yazılım paketlerinin en çok kullanılanı ROS'tur. Bunun en büyük nedeni dökümantasyonunun çok iyi hazırlanmış olması ile birlikte günümüzde kullanılan başlıca sensörler ile aktüatörler için hazır yazılım modülleri ve paketleri bulundurmasıdır. Var olan ROS ekosistemi oldukça büyük sayıda bir kullanıcı ve geliştirici kitlesine sahip olduğundan geliştirme aşamasında karşılaşılabilen bir sorun diğer kullanıcılara danışılarak hızlıca çözülebilmektedir.

5.4.1.2.ROS

ROS kelime olarak robot işletim sistemi anlamına gelmesine rağmen aslında bir işletim sistemi değil, bir yazılım sistemidir. Yani ROS tıpkı diğer özel robotik yazılım sistemlerinde olduğu gibi bir robotun ihtiyaç duyacağı temel her yapıyı ve yazılımı bünyesinde barındırır ve bu yapılar arasındaki iletişimi sağlar. Bunu yaparken tasarımcıyı ve kullanıcıyı yavaşlatan detaylarla uğraşmak zorunda bırakmaz. ROS robotik bilimi ile uğraşan kişiye modüler bir yapı sunar böylelikle kişi robotu modüller halinde tasarlar ve birbirine entegre eder. Böylelikle oluşabilecek hataların önüne geçilebildiği gibi sistemin test edilebilirliği arttırılır. ROS'un nasıl çalıştığını anlamak için kullandığı yapıları ve sisteminin anlaşılması gerekir.

Herşeyden önce ROS ve diğer tüm robotik yazılım sistemleri, işletim sistemi olarak tanımlanmış olmasına rağmen bu sistemler gerçek bir işletim sisteminden farklıdır. Bir işletim sistemi genel olarak bir bilgisayarın tüm donanım kaynakları ile birlikte sistem üzerinde çalışan tüm uygulamaları ve hafızayı yöneten bir yapıya sahiptir. Yani sistem kullancı, uygulama ile bilgisayar donanımı arasında bir köprü görevi görerek bilgisayarın işlem yaparken zamanı optimize kullanmasına olanak tanır. Böylelikle eş aynı donanım üzerinde eş zamanlı işlemler yapılmasına olanak tanırken bilgisayarın kullanımını ve programlanmasını kolaylaştırır.

Bilgisayarlar ilk geliştirdiklerinde işletim sistemleri henüz var olmadığından bilgisayar donanımı aynı anda yalnızca bir işi gerçekleştirirken kullanıcı bir yazılım geliştirmek için doğrudan donanımla meşgul olmak zorundaydı. Bu da bilgisayara sahip kişi ya da kurum için bilgisayarın işletilmesini pahalı ve zahmetli hale getirmekteydi. İşletim sistemlerinden sonra bu durum tersine döndü ve bilgisayarlar daha kolay programlanabilir ve idame ettirilebilir makineler haline gelmeye başladı. Getirdiği tüm bu olumlu gelişmeler nedeniyle günümüzde içerisinde bilgisayar veya kontrol ünitesi barındıran neredeyse her makine bir işletim sistemi kullanmaktadır. Amacına ve tasarlanma biçimine göre birçok işletim sistemi vardır. Bunlar çok kullanıcılı, tek kullanıcılı, çok görevli, tek görevli, gömülü, dağıtık ve gerçek zamanlı gibi işletim sistemleridir. Bu işletim sistemlerinini geliştirilmesinde esas, kullanıcının kullanım amacına ve kullanım sahasına göre farklılıklara çözüm oluşturmaktır. Bir işletim sistemi kullanım amacına ve tasarlanış biçimine göre iki veya üç birlikte çalışan bileşenlerden oluşur. Eğer işletim sistemi gömülü bir sistem

değilse bilgisayarın kullanıcı ile haberleşmesini sağlayan kullanıcı arayüzü (komut satırı veya grafik), kullanıcı arayüzünden alınan girdileri yorumlayan kabuk ve bilgisayarın tüm donanım ve yazılım proseslerini yöneten çekirdektir.

ROS veya başka bir robotik işletim sistemi ise özellikerinden bahsedilen işletim sistemlerinden farklı olarak bir donanımla haberleşen bir çekirdeğe sahip değildir. Yani bilgisayar donanımına doğrudan erişim hakkına sahip değildir. Bunun yerine bu tür robotik işletim sistemleri gerçek bir işletim sisteminin üzerine kurulurlar. Yani robotik uygulama ile gerçek işletim sistemi arasında bir katman olarak yer alırlar. İşletim sistemine benzeyen tarafları ise bu sistemlerin üzerinde çalışan tüm robotik uygulamaları gerçek bir işletim sistemi gibi kontrol edebilmesi ve birbiri ile ilişkilerini sağlamalarıdır.

Anlaşılacağı üzere ROS gerçek bir işletim sisteminin üzerine kurulur ve bu sisteme bağlı olarak çalışıp görevleri yerine getirir. Kullanılan gerçek işletim sistemi ise GNU/Linux tabanlı bir işletim sistemidir. Ayrıca ROS standart BSD-3 Lisansı ile lisanslanmıştır. Bu lisans sayesinde normalde açık olan bir robotik yazılımı kapatılarak ticari bir amaç için kullanılabilir. Yani ROS ticari uygulamalarda kullanılmaya müsaittir ve gün geçtikçe daha fazla kullanılmaktadır. ROS'un nasıl çalıştığını anlamak için yapısını bilmek gereklidir.

5.4.1.3. ROS'un yapısı

ROS neredeyse her robot türevi için yazılım desteği sunar. Bu nedenle kapsamı çok geniş ve ilk bakışta anlaşılması zordur. Bu zorluğun üstesinden gelmek için ROS kapsamlı bir wiki yani eğitim dökümanı sunar. Bu dökümanlar incelenerek ROS'un yapısı ve çalışma mantığı kolay bir şekilde anlaşılabilir. ROS'un çalışırken kullandığı yapılardan şunlardır:

a) ROSCORE (ROS Çekirdeği):

Bu yapı ROS tabanlı sistemlerin temelini oluşturan yazılım parçası (node) ve programların koleksiyonudur [35]. Kısaca açıklamak gerekirse ROS üzerinde çalıştırılacak yapıların temel gereksinimlerini ve birbirleri ile iletişimlerini ROSCORE sağlar. ROSCORE bir ağ yapısının merkezine benzer. Diğer tüm robotik uygulamaları bu yapının üzerinde çalışır ve bu yapı bir server (sunucu) görevi görür.

b) ROS Paketleri ve Stackleri:

ROS içerisinde yazılımlar iki farklı sınıfa ayrılır. Bunlar paketler ve stacklerdir. ROS içerisinde paketler bir ya da birden fazla node içeren küçük yazılımlardır. Genel olarak robota ve sistemle ilgili bir yada bir kaç küçük işi gerçekleştirmek üzere tasarlanmışlardır. bu küçük yazılım parçaları ROS içerisindeki yapıları kullanarak birbirleri ile haberleşebilirler. Stackler ise küçük paketlerin ve yazılım parçalarının tümleşmesi ile ortaya çıkarlar.

c) ROS Haberleşme Yolları:

ROS bir sistem olarak gerçekleştirilen işlemleri bir mesajlaşma alt yapısı ile birbiri ile haberleştirir. Bu mesajlaşma alt yapısı kendi içerisinde birden fazla farklı yapıya sahiptir.

- Topic:

Topic'ler node lar arasında mesaj alışverişi yapılmasını sağlayan yapılardır [36]. Bu yapılar sayesinde sistem içerisindeki yazılım parçaları birbirleri ile asenkron halde haberleşebilir. Bu yapı sayesinde multithread yani eş zamanlı çalışabilirler.

- Service:

Topic yapısının yazılım parçaları için senkron çalışan versiyonudur. Yazılım parçaları bu yapıyı kullanarak birebir haberleşebilir.

- Mesajlar:

ROS içerisinde birçok mesaj standardı tanımlanöıştır ve bu yapılar yenilerinin eklenmesine müsaittir. Anlaşılacağı üzere yazılım parçalarının iletişimi mesaj gönderimi üzerine kuruludur.

Bu yapılar ROS tabanlı tasarlanan robotik bir sistem için kullanılması gereken yapılardan en önemlileridir. Bu yapılar sayesinde robotik uygulama modüler halde tasarlanıp kullanılabilir. Ayrıca yine bu yapılar sayesinde sistemde oluşabilecek hataların büyük ölçüde önüne geçilmiş olur. Ayrıca sistem üzerinde uygulamaların bazıları yine bu sistem sayesinde devre dışı bırakılabilir ya da sisteme dahil edilebilir. Böylelikle sistem üzerinde gereksiz yük oluşturan yapılar kolaylıkla kapatılabilir.

Bu çalışmada kullanılan tek çekirdek bilgisayar üzerinde çalıştırılan işletim sistemi bir Linux dağıtımı olan ve Debian tabanlı Ubuntu işletim sistemidir. İşletim sisteminin Ubuntu olarak seçilmesinin esas sebebi sistemin en hızlı gelişen Linux işletim sistemlerinden biri olmasının yanı sıra kullanıcı sayısının oldukça fazla olması sebebiyle dökümantasyonunun ve yardımlarının gelişmiş olmasıdır.

Ubuntu'nun bir başka iyi tarafı da tek çekirdek bilgisayarların işlemci mimarileriyle (Çoğunlukla ARM mimarili işlemciler) uyumlu sürüme sahip olması ve bu sayede işletim sisteminin çekirdeğine müdahaleye ihtiyaç duymadan tek çekirdek bilgisayarların üzerinde çalışabilmesidir.

Gerekli yazılım alt yapısının sağlanmasından sonra çalışma için tasarlanan robotun kullanacağı algoritmaların ve yazılımların tasarımına başlanmıştır. Tasarlanan robotun amacı otonom olarak navigasyon gerçekleştirmesi olduğundan bu robotta kullanılan yazılım ve algoritmaların büyük çoğunluğu hareket ve konumlama algoritmalarıdır. Bununla beraber sensörler ve motorlar ile haberleşme sağlanması için çeşitli veriyolları kullanılarak haberleşme algoritmaları tasarlanmıştır. Tüm bu uygulamalar farklı yazılım parçaları halinde tasarlanıp sisteme tek tek eklenmiş ve test edilmiştir. Test süreci sonunda bazı iyileştirmeler gerçekleştirilmiş ve yazılımlar stabil hale getirilmiştir.

5.4.1.4. ROS için tasarlanan yazılım bileşenleri

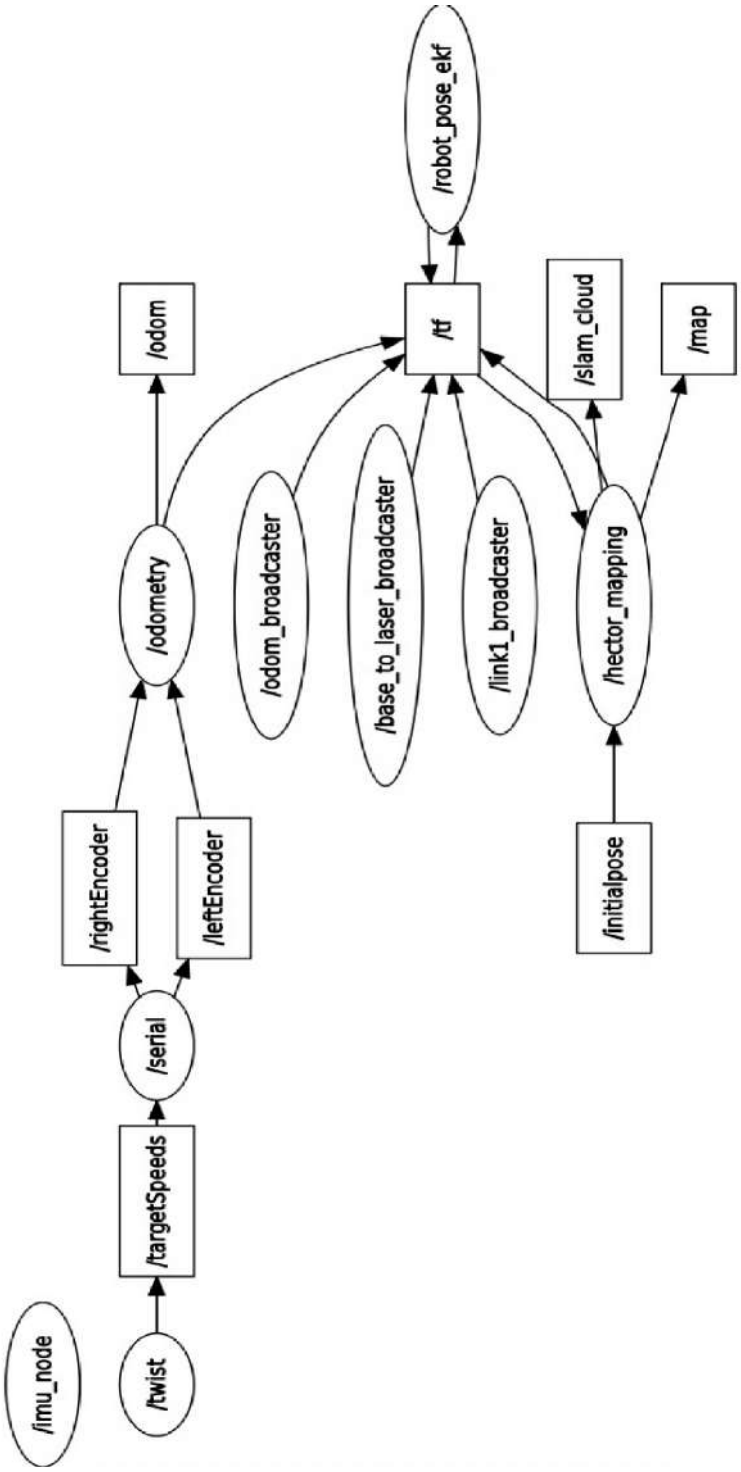
Yukarıda anlatıldığı üzere bu çalışmada kullanılmak üzere tasarlanan robotun çalışmasını belirleyen en temel yazılım bileşeni bir robotik ara yazılımı olan ROS'tur. ROS'un yapısı gereği robot için geliştirilen tüm algoritmalar ve yazılım parçacıkları modüller halinde tasarlanmış ve sisteme entegre edilmiştir. Bu şekilde sistem stabilitesi arttırılarak kararlılık ve hata ayıklama kolaylığı sağlanmıştır. Tasarlanan tüm modüller ROS üzerinde çalışacağından ve esas karar verme ünitesinin ROS üzerinde çalışan bu yazılım parçaları olacağından ötürü bu parçalar robotun temel özelliklerini ve kabiliyetlerini kontrol edecek şekilde gerçeklenmiştir. Bahsedilen bu yazılım parçacıklarını teker teker açıklamak konuyu daha anlaşılır hale getirecektir.

Şekil 5.14'te bu çalışmada tasarlanan robot için geliştirilen ROS yazılımları ve bu yazılımların sistemle ve birbirleriyle ilişiği gösterilmiştir. Bu yazılım parçacıkları kısaca şu şekilde açıklanabilir.

- /twist adlı ros uygulamasının (node) robotun hareketi için sağ ve sol tekerleklerin hızlarını seri port üzerinden gömülü yazılım bileşenine aktardığı görülür.
- /serial adlı uygulama USB üzerinden tekerleklere ait hız değerini alır. Alınan hız bilgileri odometri hesabı için /odometry uygulamasına gönderilir.

- /tf hesabı için tüm sensörlerden veri alınır. Alınan veriler bir EKF'den geçirilir. Hesaplanan tf verisi yayınlanır.
- /hector mapping uygulaması bazı sensör ve tf verilerini kullanarak ortamın haritalanmasını sağlar.

Şekilde görüldüğü gibi ROS üzerinde koşan tüm uygulamalar birbirleriyle ilişkilidir.



Şekil 5.14. Özel robotik yazılım sistemi kullanılan bir robotta yazılım şematiği

5.4.2. Gömülü kontrol ünitesi için yazılım geliştirme süreci

Bu çalışmada kullanılan mikrodenetleyici tabanlı gömülü kontrol ünitesi robotun karar verme merkezi olarak görev yapan tek kart bilgisayar ile koordineli olarak çalışan bir ünitedir. Bu kontrol ünitesi aracılığıyla sensörlerin ve aktüatörlerin (motorlar) tek kart bilgisayar ile iletişimi, kontrolü ve kumandası sağlanır. Bu amaçla gömülü yazılım ünitesi üzerinde çalıştığı mikrodenetleyicinin birbirinden bağımsız birçok çevresel birimini (GPIO, TIMER, USB) kullanır. Kullanılan bu çevresel birimler ile robotun farklı donanımları kontrol edilir. Bu işlemler yapılırken kritik olan gömülü kontrol ünitesi ile tek kart bilgisayarın senkron olarak çalışmasıdır. Çünkü bu robotun yaptığı işlemlerin doğruluğunu doğrudan etkileyen bir parametredir. Bunun sağlanması için gömülü yazılım ünitesinde zaman kritik bir yazılım tasarımı yapılması gerekir.

Yukarıda belirtildiği üzere zamanın kritik olarak yapılacak işlemlerin gerçekleştirilmesi için bir işlem planlayıcısına ihtiyaç vardır. Bu işlem planlayıcısı da ancak bir işletim sistemi ile mikrodenetleyiciye sağlanabilir. Bu yüzden bu çalışmada gömülü yazılım ünitesine bir işletim sistemi entegre edilmiştir. Bu işletim sistemi masaüstü bilgisayarların işletim sistemi ile benzerlikler gösterse de aslında yapıca ciddi farklılıkları vardır. Çünkü bu çalışmada yapılacak işlemlerin ve kontrollerin zamana bağlı olarak gerçekleştirilmesi istenildiğinden yani zaman kritik işlemlerin gerçekleşecek olması nedeniyle gerçek zamanlı bir işletim sistemi (RTOS) kullanılmıştır.

5.4.2.1. Gerçek zamanlı işletim sistemleri (Real time operating system-RTOS)

Gerçek zamanlı işletim sistemleri yani RTOS, normal işletim sistemlerinin aksine kontrol ve kumanda işlemleri sırasında ki zamansal gecikmelerin olabildiğince az olması gereken sistemlerde kullanılmak için tasarlanmış işletim sistemleridir. Örneğin uçakların kontrol yüzeylerinin kontrolü ve hava yastığı gibi sistemlerin çalışmaları sırasında oluşabilecek gecikmeler bu sistemlerin çalışmasını doğrudan etkiler ve sistemin gerektiği gibi çalışmasını engeller. Bu nedenler bu tür hassas zamanlamalı sistemlerde superloop olarak bilinen döngüye dayalı yazılımların çalıştırılması yerine zamanlamanın çok hassas olabildiği gerçek zamanlı işletim sistemleri kullanılır. Gerçek zamanlı işletim sistemlerinin zaman kritik işlemler

yapabilmesine olanak veren yazılımsal bazı araçlar (Thread, semaphore, Mutex, Queue) ve bu araçların öncelikleri (priority) bulunur. Zamanlamanın ve yapılan işlemin hassasiyeti bu araçlar ile sağlanır. Yapılacak işlemlerin hassasiyetine göre bu araçlar çeşitlendirilebilir.

Gerçek zamanlı işletim sistemleri zaman kritik olma durumlarına göre ikiye ayrılır. Bunlar hard realtime ve soft realtime olarak bilinirler. Hard realtime işletim sistemlerinde zamanlama nano saniye civarındayken soft realtime işletim sistemlerinde zamanlama mikro saniyeler ile ifade edilir. Yapılacak uygulamanın ciddiyetine göre hangi tür RTOS kullanılacağına karar verilir.

Bu çalışmada gömülü yazılım ünitesinde soft realtime bir işletim sistemi kullanımıştır. GNU GPL2 lisansına sahip olan bu işletim sistemi açık kaynak kodlu bir yazılım olan FreeRTOS'tur. Çalışma için FreeRTOS'un tercih edilmesinin temel sebebi bu işletim sisteminin açık kaynak kodlu ve ücretsiz olmasının yanı sıra çok geniş bir kullanıcı kitlesine sahip olması ile günümüzde kullanılan popüler mikrodenetleyici mimarilerinin birçoğu için destek sunmasıdır.

Bu çalışmanın gömülü yazılım ünitesinde gerçekleştirilen yazılım tasarım süreci bu üniteye bağlanacak sensörlerin ve aktüatörlerin belirlenmesi ile başlanmıştır ve kullanılan tüm donanımlara ait yazılım parçalarının entegresi ile sona ermiştir. Yazılımlar modüler olarak tasarlanmıştır. İstenilen yazılım modülünün sistemden çıkarılmasına veya yeni yazılım modülünün sisteme eklenmesine olanak tanınacak bir yapı inşa edilmiştir. Böylece bu çalışmanın sonlandırılmasının ardından bu robotun ve gömülü yazılım ünitesinin yeni bir çalışma için kullanılmasına olanak tanınmıştır.

Gömülü yazılım ünitesi için tasarlanan her bir modül FreeRTOS üzerinde çalışmaya uygundur. Bu modüllerin herbirinin ayrı ayrı sisteme eklenmesine ve istenilmeyenin çıkarılmasına olanak tanımak için her modül bir task (Thread) olarak tanımlanmıştır. Böylelikle FreeRTOS üzerinde öncelik sıralaması yapılarak hangi modülün önce çalıştırılması isteniliyorsa o modüle öncelik tanıyacak bir yapı elde edilmiştir. Bununla birlikte modüllerin çalışma planlaması yapılırken bazı modüllerin diğer bir modülden sonra çalışma gerektiğinden semaphore'ler kullanılmıştır. Ayrıca thread'lerin birbiri arasında veri taşımasına sağlamak için Queue'ler kullanılmıştır. Kullanılan bu yapılar aracılığıyla sistem zaman kritik ve stabil hale getirilmiştir.
Gömülü yazılım ünitesinde çalışan yazılım modülleri şunlardır:

- Encoder Yazılımı:

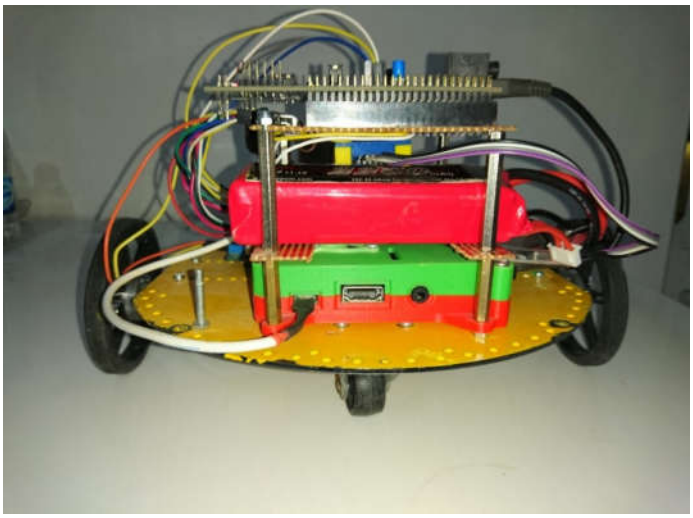
Bu yazılımın görevi motorlar üzerinde bulunan enkoder ticklerini saymak ve bu sayımı çalışma süresince sayılabilir limitlerde tutmaktır. Sağ ve sol motorlar üzerinde bulunan enkoderler için ayrı ayrı görev yapar.

- PID Yazılımı:

PID yazılımı hesaplanan enkoder tick'leri ile tek kart bilgisayar üzerinde çalışan robotik uygulamalardan gelen komutlara göre motorların hızını ayarlar.

- GPIO Yazılımı:

GPIO yazılımı gömülü yazılım ünitesinin kalbi olarak görev yapan mikrodenetletleyicinin GPIO işlemlerini yapan ve gerekli konfigürasyonları ayarlayan yazılımdır.



Şekil 5.14. Tasarlanan robotun elektronik ve güç sisteminin yerleşimi

- Motor Kontrol Yazılımı:

Motor kontrol yazılımı PID hesaplaması sonucu elde edilen sonucu motor sürücüye PWM ile aktaran bir yazılım parçacığıdır. Bu parçacık ayrıca motorların aşırı zorlamaya maruz kalmasını engelleyerek motorun yanmasını engeller.

Şekil 5.14 te tasarlan robota ait elektronik donanımın robot üzerine yerleşimi ve bağlantıları gösterilmiştir. Gömülü yazılım ünitesi USB aracılığıyla bir seri port (Virtual COM PortVCP) gibi davranarak tek kart bilgisayara bağlanmıştır. USB üzerinden gerçekleşti rilen bu haberleşme sayesinde haberleşme hızı 12 Mbps seviyesinde tutulmuştur. Ayrıca haberleşme USB 2.0 üzeriden sağlandığından veri kaybı olmamıştır. USB üzerinden haberleşmenin paketler ile gerçekleşmesi ve bu

paketlerin oluşturulup tekrar parçalanması amacıyla hem gömülü yazılım ünitesinde hem de tek kart bilgisayar üzerinde çalışabilecek bir serial parser oluşturulmuştur. Oluşturulan parser ile checksum yapılarak iletilen datanın eksiksiz olması sağlanmıştır.

6. NAVİGASYON METODU TASARIMI

Daha öncede belirtildiği gibi bu çalışmanın amacı otonom mobil bir robot yapmak ve bu robotun otonom olarak bulunduğu çevrede 2 boyutlu düzlemde otonom olarak yer değiştirmesini sağlamaktır. Bu amaçla daha önceki bölümlerde robotun bulunduğu çevre ile etkileşimini sağlayacak olan sensörler ile aktüatörler (Motorlar) anlatıldı. Robotun çalışmasını yönetecek olan kontrol ve kumanda üniteleri ile kullanılan yazılımsal gereksinimler ifade edildi.

Navigasyon, otonom mobil robotlar tarafından otonom olarak gerçekleştirilmesi gereken bir eylemdir. Bu eylemin kapsamı robotun bulunduğu ve daha önceden tanıdığı bir ortamda kullanıcı tarafından belirlenen ve gitmesi istenilen bir noktaya otonom hareketidir. Bu eylemin gerçekleştirilmesi için ROS üzerinde var olan ve stack olarak adlandırılan yazılım paketleri üzerinde değişiklik yapmak gereklidir. Çünkü her robotun kullandığı sensörler ve aktüatörler farklılık gösterir.

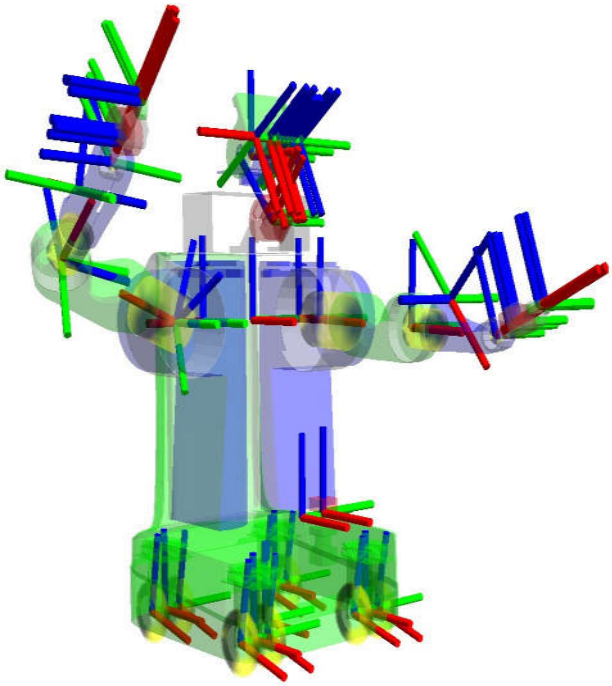
Navigasyon, kavram olarak sensörlerden akan verilerin ve tine bu verilere binaen oluşturulan odometri verisini esas alarak çalışır. Bu verileri kullanılarak hesaplanan hız ve yönlendirme komutları robotun kontrolcülerine iletilir. Bu çalışmada tasarlanan ve geliştirilen robot diferansiyel sürüş sistemi ile hareket eden bir robot olduğundan yönlendirme komutları da hız bilgisi olarak kontrolcülere iletilir.

Bahsedilen hesaplamalar ROS üzerinde çalışan yazılımlar tarafından hesaplandığından Tek kart bilgisayar üzerinde gerçekleştirilir ve daha sonra Gömülü yazılım ünitesine aktarılır. Aktarılan bu veriler daha sonra gömülü yazılım ünitesinde bulunan PID kontrolcü için istenilen hız değeri olarak kabul edilir ve PID sonucu motorlara aktarılır. Navigasyon, basit olarak bu şekilde ifade edilse de aslında biraz daha karışık bir çalışma yapısı vardır. Bu yüzden burada kullanılan terimler ve yapılar teker teker açıklanacaktır.

6.1. Dönüşümler (TF)

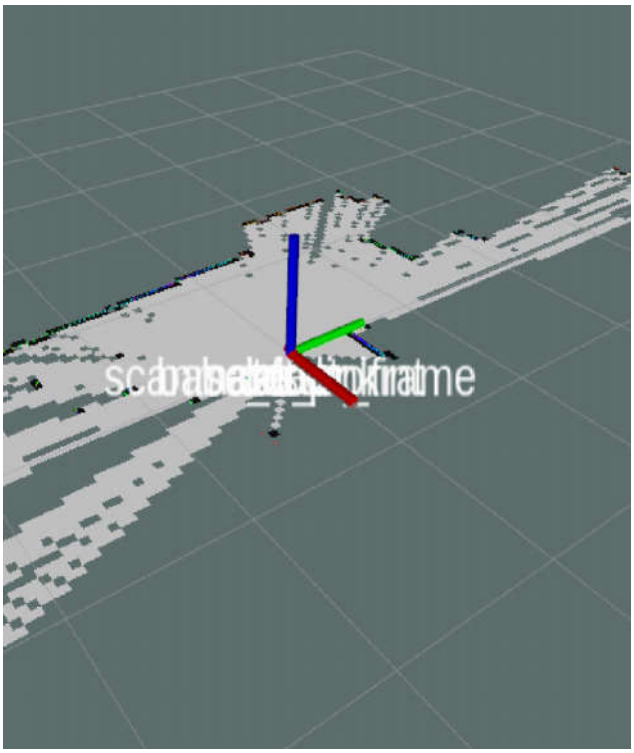
Bilindiği üzere robotlar birçoğu hareketli çok sayıda 3 boyutlu koordinat düzlemine sahiptir. Bunlara örnek verecek olursak robotun base frame, map frame bunların bir kaçıdır. Anlaşılacağı üzere bir robotun hareketli veya sabit her bir parçası bir frame ifade eder.

Frameler birbiri ile parent (evebeyn) ve child (çocuk) şeklinde ilişkilidir. Yani bu framelerden bazıları birbirine göre hareket eder ve bu göreceli hareketler frameler sayesinde tanımlanır. Frameler aslında robotun hareketli ya da statik fiziki parçalarının yazılımsal olarak tanımlanması ve ilişkilendirilmesi olarak anlaşılabilir. Yani frameler kullanılarak robotun aktüatörleri ile sensörlerin yerleşimlerinin robotun merkezine uzaklığı belirlenebilir. Böylelikle robot navigasyon planlaması yaparken kendi limitlerini hesaba katar. Şekil 6.1 incelenecek olursa humanoid bir robot üzerinde hareketli tüm parçaların hareket edebilme yönlerini gösteren kırmızı, yeşil ve mavi oklar ile tf gösterilmiştir [37].



Şekil 6.1. Humanoid bir robotta tf örneği

Tf ayrık bir sistem içerisinde görev yapar. Yani bir robotun koordinat düzlemleri hakkındaki bilgi tüm ROS uygulamalarına aktarılabilir. Buradan anlaşılacağı üzere dönüşüm bilgisi için merkezi bir sunucu yoktur. Ayrıca tf robotun tüm parçaları için tanımlanmış olacağından parçanın durumunun anlaşılması için mesaj yayını yapması veya diğer parçaların durumunu anlamak için yayınlanan tf mesajlarını dinlemesi gerekir. Bu çalışmada tasarlanan robot düzlem üzerinde hareket eder dolayısıyla tf görseli Şekil 6.2'deki gibidir.

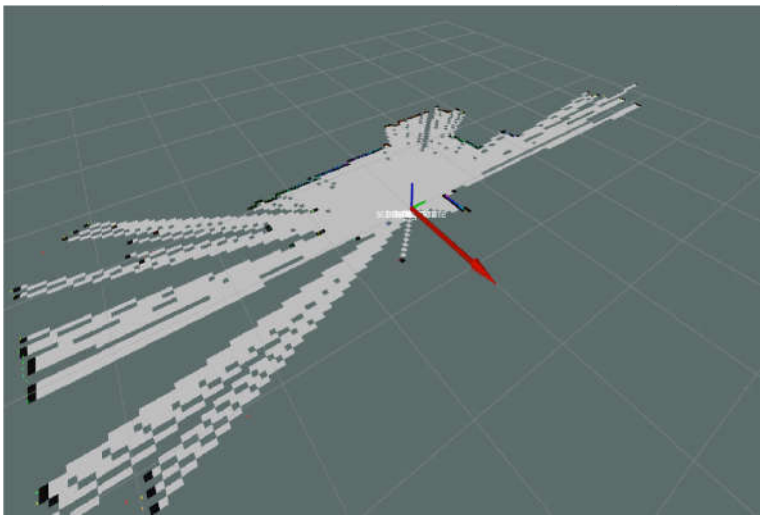


Şekil 6.2. Bu çalışmada tasarlanan robotun tf gösterimi

6.2. Odometri

Doğada bilinçli olarak hareket eden her şey konumunu bilmek zorundadır. Çünkü bilinçli bir hareket ve bu hareketin gerçekleştirilmesi esnasında yapılacak hareket ve navigasyon planlamaları için bu durum sistemin en önemli girdisini teşkil eder. İnsanoğlu hareket esnasında nerede olduğunu bilmek için gözlerini, hislerini ve aklını kullanır. İnsanlar sürekli çevreleri ile ilişkili olduklarından aslında hareket çevresindeki cisimlere göre görecelidir. Bu sayede çok hassas bir şekilde kendini konumlayabilirler. Ancak bunların birçoğundan yoksun olarak tasarlanan makineler olan robotlar için kendini gerçeğe yakın olarak konumlama ciddi bir problem haline gelir. İşte bu problemin çözümü için kullanılan yöntem odometridir [20].

Odometri, bir robotun sensörleri yardımıyla çevresindeki cisimlere göre yani göreceli olarak belirlenmesi olarak tanımlanabilir. Örneğin bir düzlem üzerinde konumlanmış olan bir tekerlekli mobil robotun tekerleklerinin çevresinin uzunluğu ve dönme sayısı biliniyor ise bu durumda tekerleklerin dönüş sayısı ve çevre uzunluğundan robotun ilk konumuna göre ne kadar yer değiştirdiği ve yol gittiği hesaplanabilir. Anlaşılacağı üzere bu hesaplama görecelidir. Şekil 6.3'te tasarlanan robota ait odometri verisi kırmızı ok ile gösterilmiştir.



Şekil 6.3. Bu çalışmada tasarlanan robota ait odometri gösterimi

Odometri ile bir robotun yer değiştirmesi hesaplanabilmesine rağmen genel olarak sensörlerden gelen verilerin toplanarak sonuç çıkartılmasına dayalı bir çalışma mantığı olması nedeniyle sensörlerden birinde meydana gelebilecek bir hata sonucu hesaplanan odometri sonucu giderek daha büyük bir yanlış halini alır vr doğruluğunu zamanla kaybeder. Bu yüzden geliştirme esnasında sensör sonuçlarının kısa bir periyotta gözden geçirilmesi gerekir.

6.3. Sensör Seçimi

Daha önce bahsedildiği üzere odometri verisinin sağlıklı olarak hesaplanabilmesi için kullanılan sensörler ve bunların entegrasyonu önem arz etmektedir. Bu yüzden donanımların anlatıldığı bölümde olmasına rağmen burada bir kez daha navigasyon sırasında en çok kullanılan sensörler ve algoritmalar içerisinde nasıl kullanıldıkları anlatılacaktır.

6.3.1. Enkoder

Daha önce bahsedildiği gibi bu çalışmada kullanılan sensörlerden biri quadrature enkoderdir. Her motorun miline bağlı bir adet olmak üzere iki motor için iki quadrature enkoder kullanımıştır. Bilindiği üzere tozlanma ve kirlenme ihtimaline karşı optik sensörler yerine Hall etkili enkoderlar tercih edilmiştir. Quadrature enkoder tercih edilmesinin nedeni bu tip enkoderlerin dönüş hızı bilgisinin yanı sıra dönüş yönü bilgisini de vermesidir. Ayrıca Hall etkili enkoderlerin çözünürlüğü daha

yüksektir. Bu sayede daha hassas konum değişimi ölçülebilir. Bu çalışmada quadrature enkoderler odometri verisinin hesaplanmasında kullanılmaktadır. Daha önce bahsedildiği üzere odometri bir robotun çevresi ile ilgili bilgi toplamadan sadece sensörleri ile topladığı verileri kullanarak kendini ilk konumuna göre göreceli olarak konumlaması olarak tanımlanır. Ancak çevre ile ilgili veriye dayalı konumlama yapılmadığından odometri verisi hesaplanırken girdileri esas alınan sensörün olası hataları sistem odometri hesabına doğrudan yansır ve odometri çıkışında hata görülür. Bu hata girdiye bağlı olarak zamanla büyür ve göreceli konumlama işlemi giderek daha yanlış bir hal alır. Bu yüzden odometri verisi hesaplanırken birden fazla sensör verisi farklı nicelikleri ölçer. Bu veriler birleştirilerek daha doğru bir odometri verisi elde edilir.

6.3.2. IMU

IMU daha önceki bölümlerde bahsedildiği üzere atalet ölçmek için kullanılan bir sensördür. Bu çalışmada IMU, robotun atalet ölçümünü gerçekleştirmek ve odometri hesaplamasının doğruluğunu arttırmak için kullanılmıştır. Yukarıda anlatıldığı üzere tek sensörün çıktısının odometri hesabında kullanımı, odometri verisinin hatalı olarak hesaplanmasına neden olduğundan, odometri hesabında doğruluğu arttırmak adına IMU'da kullanılmaktadır.

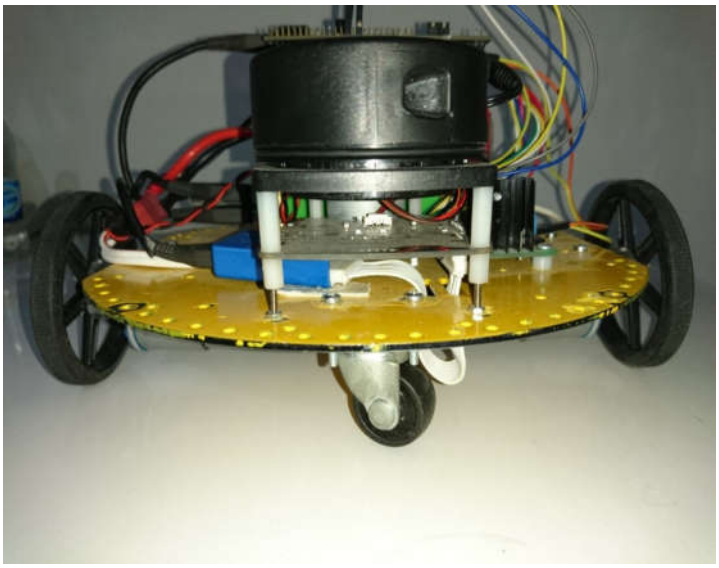
Enkoderlerden gelen pozisyon değişimi verisi ile atalet verisi birleştirilerek daha doğru bir konum değişimi hesabı yapılmasına olanak tanır. Ayrıca robota eklenen IMU yardımıyla robotun hareketi sırasında maruz kaldığı sarsıntı ve açı değişikliklerideide değerlendirilebilir. Burada belirtmekte fayda vardır ki IMU ve enkoderler robotun kendi iç parametrelerini ölçen sensörlerdir. Yani bu sensörler yardımı ile dış dünya ile ilgili bir veri elde edilemez. Bu yüzden bu sensörler çevre modellemesinde doğrudan kullanılamaz.

6.3.3. Lidar

Bu çalışmada kullanılan ve konum hesabında kullanılan bir diğer önemli sensör de lidardır. Daha önceki bölümlerde anlatıldığı üzere lidar, lazer ışığını kullanarak uzaklık ölçem bir sensördür. Bu yönüyle bir nevi radara benzetilebilir. Aradaki fark şudur: lidar mesafe ölçümü için lazer kullanır. Ancak radarl mesafe ölçümü için

radyo dalgaları kullanır. Lidar lazer kullandığından mesafe ölçümü için duyduğu enerji miktarı çok yüksektir. Bu nedenle radarlar kadar uzak mesafeleri ölçemez. Fakat lidar lazer tabanlı olduğundan ve eş evreli ışık ile ölçüm yaptığından mesafe ölçümü lidarın hareket bölgesinde çizgisel iken radyo sinyalleri ile ölçüm yapan radar için radar dalgalarının giderek daha geniş bir sathı taraması nedeniyle satıhsaldır [38]. Yani lidar bulunduğu düzlem üzerinde 2 boyutlu tarama yaparken radar 3 boyutlu tarama yapar.

Lidar çok hassas ölçüm yapabilen bir sensördür. Taradığı alanda yaptığı hata milimetreden küçük değerlerdedir. Bu yüzden bu çalışmada lidar kullanımı tercih edilmiştir. Şekil 6.4'te lidar sensörünün robot üzerindeki yerleşimi ve görünümü gösterilmiştir.



Şekil 6.4. Tasarlanan robotun önden görünümü ve lidarın yerleşimi

Bu çalışmada lidarın kullanım amacı robotun bulunduğu ortamda dış ortama göre konumlamasını sağlamaktır. Yani enkoder ve IMU'nun aksine lidar çevreyi taradığından robotun bulunduğu çevreye göre göreceli konumunu hesaplamak için kullanılır. Bu hesaplama sonucu yukarıda anlatılan odometri hesabı ile birleştirilir. Böylelikle robotun konumu ve konumunda yapacağı değişiklikler daha da iyi şekilde hesaplanır.

Lidarın bir başka görevi de haritalama için robotun çevresine olan uzaklığını ölçmektir. Burada hesaplanan uzaklıklar ile bir harita yaratılıp robotun bu harita üzerinde kendisini konumlaması sağlanmıştır. Böylelikle robotun konumu ve durumu

(hereketi, yönü vb.) grafik üzerinden takip edilir. Ayrıca bu tezin konusu olan otonom navigasyon da oluşturulan harita üzerinde seçilen bir noktaya robotun hareketi ile gerçekleştirilir.

6.4. Sensör Verilerinin Birleştirilmesi

Yukarıda anlatıldığı üzere tek bir sensöre dayalı konumlama sensörden veya algoritmadan kaynaklanabilecek hatalar nedeniyle her zaman için doğru değeri vermez. Bu nedenle oluşabilecek hatalar zamanla büyüyerek konumlamanın ve navigasyonun hatalı çalışmasına neden olur.

Tek bir sensörün kullanımından doğabilecek hatalar nedeniyle kullanılan birden fazla sensör ile hatalar azaltılabilir ve daha iyi konumlama gerçekleştirilebilir. Burada kritik olan konu ise ölçümü ve bu ölçüme göre yapılan konumlamanın doğruluğu için konumlama hesabında kullanılacak sensörlerin aynı parametreyi farklı yöntemlerle ölçmesidir. Örneğin enkoder hareketi motorun milinin dönüşü ile ölçerken IMU çizgisel ve merkezi ivmeyi ölçer [39]. Bu iki sensör farklı şekillerde ölçüm yapsa da aynı parametreyi ölçtüğünden bu sensörlerin çıkışlarının birleştirilmesi ile daha doğru bir konum değişikliği hesaplanabilir.

Birden fazla yöntem olmasına karşın burada bahsedilen sensör çıkışlarının birleştirilmesi kolay olamamaktadır. Bu ancak bazı özel algoritmalar ve filtreler ile mümkündür. Günümüzde bu algoritmaların en popüleri Genişletilmiş Kalman Filtresidir (EKF). Bulunabilen kaynak sayısının çokluğu ve uygulama kolaylığı nedeniyle bu çalışmada EKF kullanılmıştır.

6.4.1. Kalman filtresi

Kalman filtresi bir sistemin o anki durumunu kullanarak bir sonraki durumunu tahmin etmekte kullanılan bir filtredir. Bir kalman filtresi, sistemin o anki durumunu belirleyebilen veya belirleyebilecek olan doğrudan olmayan, doğru olmayan ve kesin olmayan ölçümleri kullanır. Bu ölçümlerin alındığı sürece filtre sürekli kendini çağırır. Olçümler bir robot için sensörlerden sağlandığından dolayı sensörlerden veri akışı sağlandıkça Kalman filtresi giderek daha tutarlı bir durum tahmini gerçekleştirir.

Eğer sistemde sensörlerden kaynaklı olan ve sistemi etkileyen gürültüler mevcutsa ve bu gürültüler gaussian ise kalman filtresi ortalama hatayı giderek küçültür. Buradan anlaşılacağı üzere kalman filtresi bir sistemin durumunu tahmin etmek için kullanılsa da aslında bir filtredir. Bunun nedeni kalman filtresinin sistemde var olabilen gürültüleri sönümleyerek en sade sonucu bulmak amacıyla kullanılabilmesidir.

Kalman filtresi birçok farklı sistemde kullanılır. Bunun nedeni filtrenin pratikte iyi sonuçlar vermesi, gerçek zamanlı işlem yapmaya uygun olması, formülize edilmesinin kolaylığı nedeniyle uygulamaya kolay geçirilmesi ve sistem ile ilgili ölçümler yapılırken bu ölçümlerin çevrime ihtiyaç duymamasıdır [40].

İlk bakışta Kalman filtresi katmaşık görülse de aslında basit matematiksel ifadelerin tekrarı ile oluşturulmuş başarılı bir filtredir. Matematiksel olarak ifade edilme şeklini inceleyecek olursak Kalman filtresi denklem 6.1'deki gibi gösterilir.

$$\hat{X}_k = K_k\ Z_k + (1 - K_k)\ \hat{X}_k - 1 \qquad (6.1)$$

Burada, t içinde bulunulan anı (Ayrık zaman ifadesi olarak düşünülebilir.), $\hat{X}_k$ anlık tahmini, K_k kalman Kazancını, Z_k ölçülen değeri, $\hat{X}_{k-1}$ bir önceki anlık tahmini ifade eder. Formülden anlaşılacağı üzere Kalman filtresi sistemin anlık durum tahminini yaparken sistemin bulunulan andaki durumu ile ilgili bilgi veren ölçümleri ve bir önceki sistem durum tahminini kullanır. Filtre her yeni durum tahmini için bu döngüyü tekrarlar [41].

Anlaşılacağı üzere Kalman filtresi ile yapılan tahminler nedensel bir sistemi ifade eder. Yani sistem durumu hem kendinden önceki tahminden etkilenir hem de kendinden sonraki durum tahminini etkiler. Bu filtrede esas amaç sistem için anlık durum tahmini yani $\hat{X}_t$'yi sistemin çalışmaya devam ettiği her t anı için bulmaktır. Bu çalışmada kalman filtresinin kullanım amacı sensörlerden gelebilecek hatalı verilerin otonom mobil robotun hatalı konumlamasını engellemek olduğu için sensörlerden akan bilgilerin (Z_t) filtrelenmesi gerekir. Bu yüzden sensörlerden akan veriler Kalman Kazancı (K_t) ile çarpılır. Aslında formüle bakacak olursak filtreyi etkileyebilecek en önemli parametrenin Kalman kazancı olduğu ve bu parametrenin bilinmez olduğu görülür. Kalman kazancının matematiksel ifadesinin bulunabilmesi için diğer tüm veriler Denklem 6.1'de yerine yazılır. Bu verilere göre kazanç belirlenebilir.

Standart Kalman filtresi, nedensel olan sistemlerin durum tahmininde kullanılır. Ancak bu filtrenin gerektiği gibi çalışabilmesi için uygulanacağı sistemin lineer olması gerekir. Nonlineer sistemlerde kalman filtresi, matematiksel eşitlikleri ile sistemin modeli gereği lineer sistemlerdeki gibi doğru sonuç üretemez. Bu durumda sistemin durum tahminini yapabilmek için temelde bir Kalman Filtresi olan Genişletilmiş Kalman Filtresi (EKF) kullanılır.

Genişletilmiş Kalman Filtresi, nonlineer sistem modelini her örnekleme zamanı için lineer bir sisteme dönüştürür. Tek değişkenli bir sistemde bu sistemin o anki değeri ve bu değerin türevi ile sağlanırken birden fazla değişkene sahip bir model için sistemin değeri ve jakobian matris eşitlikleri kullanılır. Nonlineer bir sistemin lineerleştirilmiş denklemleri standart Kalman filtresi ile benzer özellikler gösterir. Daha iyi anlaşılabilmesi için Genişletilmiş Kalman Filtresinin matematiksel ifadelerine bakmak gereklidir.

$$x_k = Ax_k - 1 + Bu_k + w_k - 1 \qquad (6.2)$$
$$z_k = Hx_k + v_k$$

Burada, k içinde bulunulan ölçüm sırasını (Ayrık zaman ifadesi olarak düşünülebilir.), x_k anlık sinyal değeri, x_{k-1} bir önceki anlık sinyal değerini, u_k kontrol sinyali, w_{k-1} işlem sırasındaki paraziti, z_k doğruluğundan emin olunmayan sensör ölçümünü, v_k ölçümden kaynaklı paraziti, ayrıca formülde bulunan A,B,H ise matrisleri ifade eder, ancak pratikte bu matrisler sayısal katsayıları ifade eder.

Denklem incelenecek olursa burada ölçümü ve tahmini en zor olan parametrelerin W_{k-1} ve v_k olduğu görülür. Çünkü ölçüm sırasında oluşan parazitler ile filtrenin uygulanması sırasında oluşan parazitlerin belirli bir formu yoktur. Bu nedenle bu parametreler sadece tahmin yöntemi ile doldurulabilir. Genişletilmiş Kalman Filtresinin uygulanması ise iki aşamalı olarak gerçekleştirilir. Bunlardan birisi tahmin aşaması diğeri ise doğrulama aşamasıdır. Tahmin aşaması, matematiksel olarak Denklem 6.3 teki gibi ifade edilebilir:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \qquad (6.3)$$
$$\hat{P}_k^- = AP_{k-1}.A^T$$

Burada, k içinde bulunulan ölçüm sırasını (Ayrık zaman ifadesi olarak düşünülebilir), $\hat{x}_k^-$ öncelikli durum tahmini, $\hat{x}_{k-1}$ bir önceki durum tahminini, u_k sisteme uygulanan kontrol sinyalini, $\hat{P}_k^-$ öncelikli hata kovaryansını, P_{k-1} bir önceki hata kovaryansını, ayrıca formülde bulunan A ve B ise matrisleri ifade eder.

Tahmin aşamasındaki eşitlikler incelenecek olursa filtrenin ilk döngüsü için $\hat{x}_k^-$ ifadesinin filtrenin doğrulama kısmına henüz geçilmemesi nedeniyle kabataslak bir tahmini ifade ettiği görülür. Aynı durum $\hat{P}_k^-$ için de geçerlidir. Bu aşamadan elde edilen sonuçlar doğrulama denklemlerinde kullanılacak ve filtrenin stabil bir çıktı elde etmesini sağlayacaktır. Eşitliklerde dikkat edilmesi gereken bir başka konu ise Burada ifade edilen A ve B matrisleri için gerçekleştirilen çarpma işleminin lineer cebir kurallarına göre gerçekleşmiş olmasıdır. Doğrulama aşaması matematiksel olarak 6.4, 6.5, 6.6 numaralı denklemlerdeki gibi ifade edilebilir:

- Kalman kazancının hesaplanması

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{6.4}$$

- Sensör ölçümleri ile tahminin güncellenmesi

$$\hat{x}_k = \hat{x}_k^- + K_k (Z_k - H\hat{x}_k^-) \tag{6.5}$$

- Hata kovaryansının güncellenmesi

$$P_k = (1 - K_k H) P_k^- \tag{6.6}$$

Buradaki matematiksel eşitliklere dikkat edilirse standart kalman filtresinde kullanılan eşitlikler olduğu görülür. Doğrulama aşaması Kalman filtresinin sonucu ürettiği ve doğruladığı aşamadır. Bu aşamadan sonra tekrar tahmin aşamasındaki matematiksel ifadeler çalışmaya başlar ve bu süreç döngü halinde devam eder. Filtre çalıştığı sürece sürekli tahmin ve doğrulama aşamaları birbirini takip ederek tekrarlanır. Yukarıdaki matematiksel eşitlikler incelenirse sensör çıktılarının birleştirilmesi için her bir sensörden sağlanan akış için ayrı bir değişken (Z) kullanılması gerekliliği görülür. Bu sayede her bir sensör için farklı katsayılar belirlenerek sensörlerden akan veriler arasında ağırlık belirlenebilir.

Bu çalışmada odometri ve buna bağlı konumlama verileri, genişletilmiş kalman filtresi aracılığıyla bahsedilen yöntem yardımıyla birleştirilen sensör çıktıları kullanılarak hesaplanmıştır. Yukarı da da bahsedildiği üzere eş zamanlı konumlama ve haritalama (SLAM) uygulamaları için hem konumlama hem de odometri verilerinin doğru hesaplanması en önemli etkendir. Çünkü bu hesaplardan kaynaklanabilecek bir yanlışlık ile robot aslında olmadığı bir konumdaymış gibi navigasyon hesapları yapıp yanlış bir yol izleyebilir.

6.5. Navigasyon Metodu Seçimi

Bir otonom mobil robot için navigasyon planlaması, robotun bulunduğu noktadan gitmesi istenilen noktaya gidinceye kadar yapacağı manevraları belirlemesi ve yolunda bulunan engellerden kaçınması anlamına gelir. Bir robot bahsedilen işleri yaparken bunu bir algoritmaya bağlı olarak yapmak zorundadır. İşte bu yüzden navigasyon metodundan kastedilen aslında bir algoritmadır. Robotik biliminde kullanılan birçok navigasyon algoritması vardır. Bunlardan bazıları şunlardır:

- Dijkstra Algoritması
- Yapay Potansiyel Alan Yöntemi (Artificial Potential Field Method)
- Görsel Grafik Metodu (Visibly Graph Method)
- Occupancy Grid Haritalama Yöntemi

Bu çalışmada Occupancy Grid Haritalama Yöntemi Bu yönteme göre mobil robotun bulunduğu çevreden sensörler yardımıyla elde edilen ve oluşturulan çevresel model ızgara şeklinde küçük bölgelere bölünür. Çevresel model üzerinden algılanan cisimler bu ızgara modellerinin üzerine işaretlenir. Bunun dışında robotun bulunduğu bölge ve robotun hareket etmesi istenilen yer de ızgara üzerindeki bölgeye işaretlenir [42]. Robotun navigasyonu için hesaplanan en kısa yol ise robotun bulunduğu bölgeden gidilecek bölgeye kadar başka cisimlerin işaretlediği bölgeler ile kesişmeyeceği şekilde hesaplanır.

Bu çalışmada ROS üzerine yazılmış paketler kullanılarak occupancy grid navigasyon algoritması kullanılmıştır. Tasarlanan robot iç ortamda ve düz bir zeminde gezindiğinden dolayı bu paketlerin yapısında çeşitli değişiklikler yapılmış ve iç ortama göre optimize edilmiştir.

6.6. Otonom Navigasyon Modu

Otonom navigasyon modu, tasarlanan robotun kendi üzerinde bulunan sensörlerden algıladığı çevresel parametreleri kullanarak modellediği çevre üzerinde Occupancy Grid Algoritmasını kullanarak gezinmesini ifade eden navigasyon modudur. Bu çalışmada tasarlanan robotun oluşturduğu harita üzerinde otonom gezinimi bu mod ile sağlanmıştır. Bu mod ile çevresel model üzerinde gezinen robot çevresel mod üzerindeki değişiklikleri algılayıp çevresel model ve harita üzerinde değişiklikler yapabilir.

6.7. Kullanıcı Kontrollü Navigasyon Modu

Kullanıcı kontrollü navigasyon modu, tasarlanan robotun kendi üzerinde bulunan sensörlerden algıladığı çevresel parametreleri kullanarak modellediği çevre üzerinde kullanıcı tarafından bir joypad, joystick veya klavye ile hareket ettirilmesini sağlayan navigasyon modudur. Bu mod aktifken robot üzerinde bulunan navigasyon algoritmasını kullanmaz. Sadece odometri verisini sensörlerden algıladığı verileri kullanarak hesaplar. Kullanıcı kontrollü navigasyon modu aktifken robot gezindiği çevre üzerindeki değişiklikleri algılayıp bu değişiklikleri çevresel model üzerinde güncelleyebilir.

6.8. Harita

Bu çalışmada tasarlanan robot kendi üzerindeki sensörleri kullanarak gezinirken veya durduğu yerde haritalama yapabilir. Bunu yaparken SLAM algoritmalarını kullanan robot harita üzerindeki verileri güncelleyebilir. SLAM ile ilgili açıklamalar 3. bölümde açıklanmıştır. Bu çalışmada SLAM algoritmalarının kullanımının temel amacı robotun daha önce bulunmadığı ortamları gezinme anında haritalanmansını sağlayarak haritanın elde edilmesidir.

7. SONUÇLAR ve TARTIŞMA

Bu bölümde çalışma sırasında karşılaşılan teknik sorunlar, bunların elimine edilmesi için kullanılan yöntemler ve elde edilen deneysel sonuçlar sunulmuştur.

7.1. Pratik Sorunlar

Bu bölümde çalışmanın bir parçasını oluşturan otonom mobil robot tasarımı sırasında ortaya çıkan donanımsal ve yazılımsal kaynaklı pratik sorunlar incelenmiş ve bu sorunları aşmak için geliştirilen yöntemler ile bu yöntemlerin tüm sisteme uygulanılmasından bahsedilmiştir. Bu bölümde anlatılan ve bahsedilen hatalar kendi içinde donanımsal veya yazılımsal hatalar olarak sınıflandırılmamıştır. Bunun yerine oluşan hata ile ilgili problemin donanımsal, yazılımsal veya her ikisinden mi kaynaklı olduğu detaylı olarak açıklanmıştır.

7.1.1. Odometri doğruluğu

Daha önce açıklandığı üzere odometri, bu çalışmada tasarlanan robotun bulunduğu ortamda hareketine başladığı noktaya göreceli olarak yer değiştirmesini ölçmek için kullanılan yöntemdir. Otonom mobil robotun verilen görevi istenilen şekilde gerçekleştirebilmesi için yer değiştirmesinin ve bulunduğu konumun en doğru şekilde ölçülmesi gerekir. Şayet yer değiştirme doğruluğu sağlanamazsa robotun gerçekleştirmesi istenilen görev tamamlanamaz. Bu çalışmada gerçekleştirilen görev bir noktadan başka bir noktaya otonom hareket olduğundan robot odometri verisini düzgün olarak hesaplayamazsa bu durumda hedeflenen noktaya hareket edemez ve sürekli bir sapma ile hareket eder. Tasarlanan robot ilk üretildiğinde robot bir noktadan diğerine hareket ettirildiğinde sürekli olarak sol tarafa doğru sapma göstermekteydi. Bu nedenle robot varması istenilen noktaya sürekli olarak fazladan manevra yaparak ulaşmaya çalışıyordu. Bu durum ile ilk karşılaşıldığında sorunun robotun hareketini sağlayan sağ ve sol motorlar üzerinde bulunan enkoderlerden gelen veri ile ilgili olduğu değerlendirildi. Bu nedenle robotun motorları aynı PWM sinyali uygulanarak sürüldü ve enkoder çıkışlarındaki verilerin eş olup olmadığı değerlendirildi. Bu sayede sorunun enkoderler ile ilgili olmadığı görüldü. Daha sonrasında ise ROS üzerinde yazılan odometri hesaplama

yazılımı “step by step” analiz edildi. Analiz sonucunda odometri hesaplama algoritmasının düzgün çalıştığı görüldü. Sonrasında ise odometri hesaplama yazılımına ait konfigürasyon dosyalarında motorlara bağlanan tekerleklerin çaplarına ait konfigürasyonların üreticinin sağladığı veri yerine kumpas kullanarak ölçüm yapılarak değiştirilmesi sağlandı. Bu durum robotun gerçeğe yakın odometri verisi üretmesini sağlasa da istenilen hassasiyet yinr sağlanamadı. Tüm bunların ardından odometri verisi için kullanılan tek sensör olan enkoderlere ek olarak sisteme bir IMU entegre edildi. Eklenen IMU ile enkoder verileri bir kalman filtresinden geçirilerek hesaplanan odometri verisi yeniden hesaplandı. Bu hesaplama sırasında EKF kullanılarak sensör verileri birleştirildi.

Sonuç olarak EKF kullanımı ile robot istenilen netlikte odometri verisi üretmeye başladı. Bu sayede odometri doğruluğu sağlandı.

7.1.2. Sistematik hatalar

Sistematik hatalar robotun üzerinde bulunan donanımlaradan birinden veya bir kaçından kaynaklanan ve sistem içerisinde başka hatalardan etkilenerek robotun çalışmasında sorunlar çıkaran hatalardır. Tasarlanan robot üzerinde bulunan elektronik ve elektromekanik donanımların neredeyse tamamı birbirinden farklı çalışma karakteristiklerine sahiptir. Bu yüzden robot için kullanılan güç sistemi birbirinden farklı bu donanımların tamamını çalıştırmak zorundadır. Bu robot için tasarlanan güç dağıtım şematiği daha önceki bölümlerde Şekil 5.2’de verilmiştir. Bu şematik incelenecek olursa, aynı 11.1 V gerilim sağlayan LiPo pilden 5 V’a bir regülatör yardımıyla gerilim düşümü sağlanmakta ve elde edilen bu gerilim ile tek kart bilgisayar ve gömülü kontrol ünitesi beslenmektedir. Ancak sensörlerin tamamı 5 V’luk gerilim ile beslendiğinden çalışmanın başında fazladan maliyetten kaçınılması amacı ile bu sensörlerden bazıları doğrudan tek kart bilgisayar üzerinde bulunan USB portları üzerinden beslenmekteydi. Sistemin modüler olarak tasarlanması sebebi ile her sensör bilgisayara ayrı ayrı bağlandığından bu besleme sensörlere yeterli gücü sağlamaktaydı. Ancak sistemde modüler tasarım tamamlandıktan ve tüm modüller robota entegre edildikten sonra tek kart bilgisayar zaman zaman sensörlerin çalışmak için ihtiyaç duyduğu gerilimi karşılamakta zorlanmaya başladı. Bu yüzden bilgisayar zaman zaman kendini resetlemeye başladı ve robotun çalışması kesildi. Bu durumun aşılması için tek kart bilgisayarlara bağlı

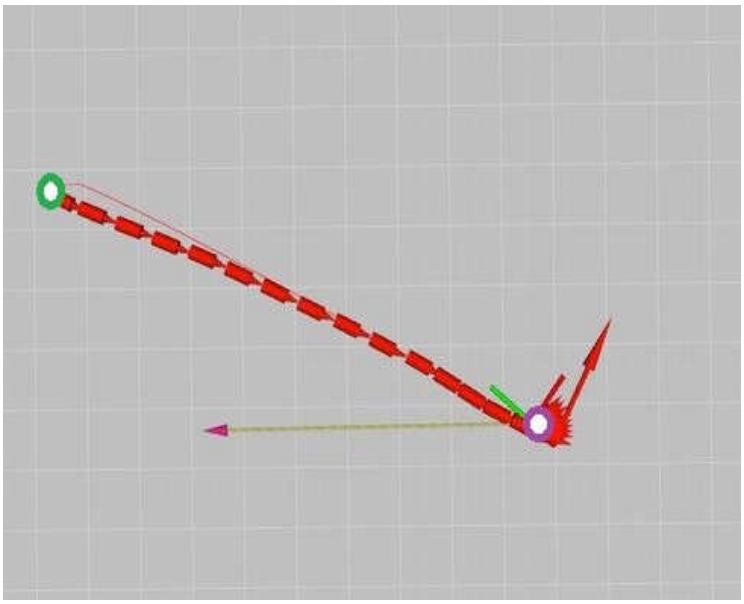
sensörleri bilgisayarın USB portlarından beslemek yerine, ayrı bir regülatör devresinin regüle ettiği gerilim ile beslemenin daha uygun olacağı kanaatine varıldı ve bağlantı bu şekilde düzeltildi. Sonuç olarak robot üzerinde yer alan bilgisayarın istem dışı resetlenmesi engellendi.

7.2. Deneysel Sonuçlar

Bu çalışmada tasarlanan mobil robot daha önceki bölümlerde anlatıldığı üzere herhangi bir düzlemsel ortamda çalıştırıldığında üzerinde bulunan sensörler yardımı ile ortamın çevresel modelini oluşturan, oluşturduğu bu çevresel modele göre bir harita üretip kendini eş zamanlı olarak bu harita üzerinde konumlayabilen bir robottur. Bunun yanı sıra bu robot kendi oluşturduğu ve kendini konumladığı harita üzerinde, kullanıcı tarafından belirlenen bir noktaya ROS üzerinde koşan navigasyon algoritmasına göre navigasyon planlaması yapan ve hareket eden bir robottur. Robotun bu özelliği kendisine otonomi özelliği kazandırmaktadır.

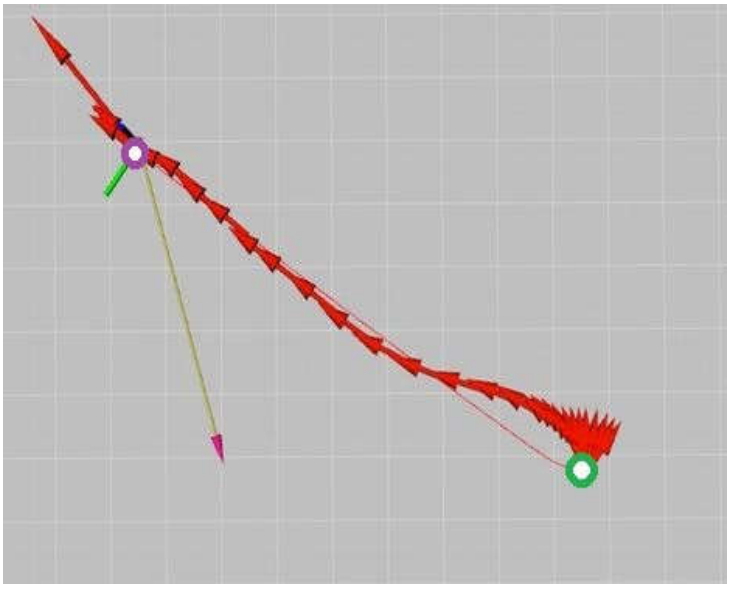
Kullanılan navigasyon algoritması daha önce anlatıldığı üzere occupancy grid haritalama algoritmasında türetilmiş bir navigasyon algoritmasıdır. Bu algoritmaya göre robotun bulunduğu çevresel model birbirine eşit kare ızgaralara bölünür ve çevresel model üzerindeki engeller bu algoritmaya göre bu model üzerine eklenir. Robot navigasyon planlaması yaparken bulunduğu noktadan gideceği noktaya olan uzaklığı ızgara üzerindeki engellere göre en kısa olacak şekilde planlar.

Yukarıda anlatılanlara göre bu çalışmada tasarlanan robot ile bir iki boyutlu bir düzlem üzerinde robota belirli hedefler verilmiş ve robotun verilen hedeflere gitmek için yaptığı navigasyon planları ile bu hedeflere giderkenki anlık konumları incelenmiştir. Aşağıda navigasyon çalışmalarına ait görseller verilmiştir. Şekil 7.1 incelenecek olursa robotun ilk bulunduğu konum yeşil nokta ile gösterilmiştir. Robota verilen hedef ise mor nokta ile gösterilmiştir. Buna göre robota hedefin verilmesi ile robot tarafından görsel üzerinde ince kırmızı çizgi ile navigasyon planlaması yapılmıştır. Bu planlamaya göre robotun hedefe hareketi esnasında bu yolu kullanması gerekmektedir. Ancak hareketin en başına bakıldığında robotun bu planlamadan saparak doğrudan hedefe yöneldiği görülmektedir. Bu durum robotun mekanik tasarım parametrelerinin, yazılımdaki parametreler ile tam olarak örtüşmediğini göstermektedir.



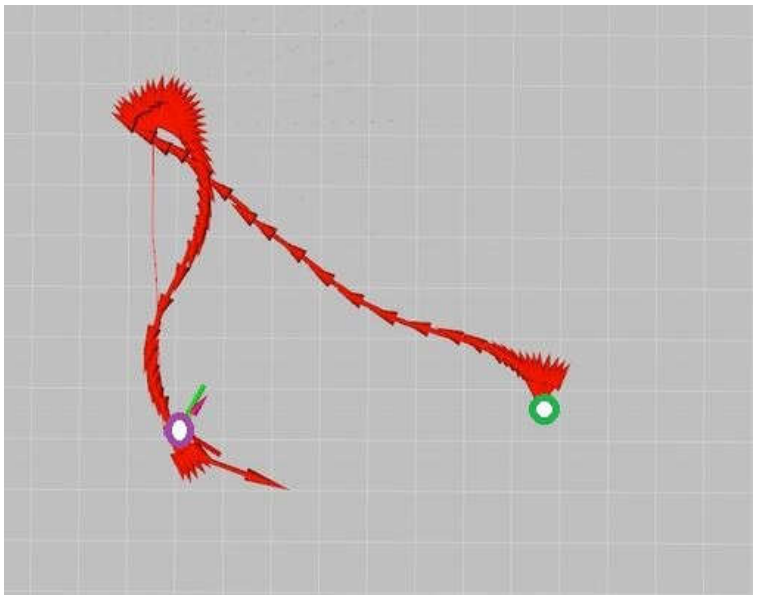
Şekil 7.1. Robota verilen bir hedefe robotun hareketi

Şekil 7.2 incelenecek olursa bu kez robota verilen hedef daha uzak bir noktadadır. Buna göre robota verilen hedef için robot tarafından oluşturulan navigasyon planı ve robotun hareketi incelenecek olursa, hareketin en başında robotun, yine bir önceki örnekteki gibi navigasyon planınından saptığı ve doğrudan hedefe yöneldiği görülür.



Şekil 7.2. Robota verilen daha uzun bir hedefe robotun hareketi

Hareketin ilerleyen zamanlarında navigasyon planını yakalamaya çalışan robot, bu kez plan üzerinde küçük bir osilasyona girerek hedefe ulaşır. Bu örnekte robotun bir önceki örneğe göre yaptığı toplam hata yani ideal yoldan sapma oranı daha yüksektir.



Şekil 7.3. Robota verilen ardışık iki hedefe robotun hareketi

Şekil 7.3 incelenecek olursa robota verilen ardışık iki hedef için robotun navigasyon planlaması ve hareketi gösterilmiştir. Ardışık iki ayrı hareket planlaması yapıldığından dolayı tüm hareketi iki parça halinde inceleyebiliriz. Robota verilen ilk hedef için navigasyon planlaması ve robotun bu hedefe hareketi incelenecek olursa, robotun önceki örneklerden farklı olarak navigasyon planına tam olarak uyarak hedefe ulaştığı görülür. Bunun nedeni verilen hedefin diğer hedeflere göre daha yakın bir hedef olmasıdır. Ancak hareketin ikinci yarısı incelenecek olursa bu kez tasarlanan navigasyon planından oldukça yüksek oranda bir sapma görülür. Bunun nedeni ikinci hedefin robotun arkasında kalmasıdır. Ancak esas sorun bu değildir. İdeal şartlarda robotun ikinci hedefe hareketi için soldan dönüp navigasyon planını uygulaması gerekirken robot sağdan dönmüş ve hedefe yönelmiştir. Bunun yazılım içerisinde eksik kalan bir yön algoritmasından kaynaklandığı düşünülmektedir.

Bu problem için algoritmada yön konusunda değişiklik yapılması gerekir. Bu değişiklik temelde robotun bulunduğu konum ve baş açısının verilen hedef ile kıyaslanmasını temel alan bir değişikliktir. Buna göre robotun bulunduğu baştan, gideceği hedefin konumunun robota göre sağdan ve soldan dönüş için kaç derecelik farklar içerdiği hesaplanmalı ve hangi taraf için fark küçükse robot o yöne dönmelidir.

8. SONUÇ ve ÖNERİLER

Bu çalışmada çalışmada birden fazla sensör verisi, genişletilmiş kalman filtresi ile birleştirilerek robotun navigasyonunun optimize edilmesi için kullanılmıştır. Elde edilen sonuçlar oldukça başarılı olmakla birlikte, bununla ilgili grafikler sunulmuştur. Robotun hızı arttırıldıkça robotun kontrol edilebilirliğinin azaldığı ve elde edilen haritalardaki bağıl hata değerlerinin arttığı gözlemlenmiştir. Bu durumun sensör çözünürlüğü ve işlemci hızıyla ilgili olduğu değerlendirilmektedir. Bu çalışma ile geliştirilen robot otonom niteliklerde olsa da sonuçlar bölümünde anlatıldığı üzere hala bazı eksikleri vardır.

Gelecekte yapılması planlanan çalışmalarda robotun bilinen eksiklikleri giderilmeye çalışılacaktır. Bu nedenle robotun hareketini kontrol eden PID kontrolcüsünün yerine Adaptif PID kontrolcüsü tasarlanacak ve robotun kontrolü bu şekilde gerçekleştirilecektir. Bunun dışında robotun 2 boyutlu düzlemde hareketini zorlaştıran robot şasesi değiştirilip geliştirilecektir. Diferansiyel sürüş sistemi değiştirilmeden sarhoş tekerlek sayısı bire düşürülecektir. Ayrıca lidarın dönüş merkezi tahrik sağlayan tekerlerin tam ortasındaki dönüş merkezine alınacaktır. Bu sayede robotun matematiksel modeli hazırlanırken lidarın konumunu belirleyen denklemler kolaylaşacaktır. Bunun dışında robotun işlem gücünü arttırmak ve eklenecek yeni sensör ve aktüatörlerin sistem üzerinde sorunsuz çalışmasını sağlamak için robot üzerinde kullanılan tek kart bilgisayar, performansı daha yüksek olan bir tek kart bilgisayar ile değiştirilecektir. Donanımsal olarak yapılacak bir başka değişiklik ise kullanılan lidarın örnekleme sayısı daha yüksek bir lidar ile değiştirilmesi olacaktır. Bunun nedeni ise kullanılan lidarın örnekleme sayısının düşük olması nedeniyle üretilen haritaların kalitesinin düşük olmasıdır.

9. KAYNAKLAR

[1] C. Heyer, IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, Oct. 18-22, (2010), pp. 4749-4754.

[2] J. Lee, B. Bagheri and Hung-An Kao. Keynote in International Conference on Industrial Informatics INDIN, Porto Alegre, Brazil, July 27-30, (2014), pp. 1-6.

[3] J. Rifkin, *The Third Industrial Revolution*, Palgrave Macmillan, New York, 2011, 26–27.

[4] J.M. Tien, *The next industrial revolution: Integrated services and goods*, **Journal of Systems Science and Systems Engineering**, 21:3 (2012) 257–296.

[5] L. Perez, I. Rodriguez, N. Rodriguez, R. Usamentiaga and D.F. Garcia, *Robot Guidance Using Machine Vision Techniques in Industrial Environments: A Comparative Review*, **Sensors**, 16:3 (2016) 335.

[6] U. Nehmzow. *Mobile Robotics: A Practical Introduction,* In the series Applied Computing, Springer-Verlag London, 2003,7-23.

[7] S. Dattaand, R. Ray and D. Banerji, *Development of autonomous mobilerobot with manipulator for manufacturing environment*, **The International Journal of Advanced Manufacturing Technology**, 38:5-6 (2008) 536–542.

[8] E. Garcia, M.A. Jimenez, P.G. De Santos and M Armada, *The evolution of robotics research*, **IEEE Robotics Automation Magazine**, 14:1 (2007) 90–103.

[9] I.R. Nourbakhsh, K. Crowley, A. Bhave, E. Hamner, T. Hsiu, A. S. Perez-Bergquist, S. Richards and K. Wilkinson, *The Robotic Autonomy Mobile Robotics Course: Robot Design, Curriculum Design and Educational Assessment*, **Autonomous Robots**, 18:1 (2005) 103–127.

[10] M. Koseoglu, O.M. Celik and O. Pektas, International Artificial Intelligence and Data Processing Symposium IDAP, Malatya, Sep. 16-17, (2017), pp. 1-5.

[11] R.C. Arkin and R.R. Murphy, *Autonomous navigation in a manufacturing environment*, **IEEE Transactions on Robotics and Automation**, 6:4 (1990) 445–454.

[12] M. Hagele, K. Nilsson and J.N. Pires, "Industrial Robotics", in B. Siciliano, O. Khatib (Eds.), Springer Handbook of Robotics, Springer-Verlag, Berlin Heidelberg, 2008, 963–986.

[13] M. Vagas, M. Hajduk, J. Semjon, L. Koukolova and R. Janos, *The view to the current robotics*, **Advanced Materials Research**, 463-464 (2012) 1711–1714.

[14] M. Hvilshoj, S. Bogh, O.S. Nielsen and O. Madsen, *Autonomous industrial mobile manipulation (AIMM): past, present and future*, **Industrial Robot: An International Journal**, 39:2 (2012) 120–135.

[15] T. Lee, J. Shin and D. Cho, IEEE International Symposium on Industrial Electronics ISIE, Seoul, July 5-8, (2009), pp. 1956-1961.

[16] C. Tsai, *A localization system of a mobile robot by fusing dead-reckoning and ultrasonic measurements*, **IEEE Transactions on Instrumentation and Measurement**, 47:5 (1998) 1399–1404.

[17] C. Hofner and G. Schmidt, *Path planning and guidance techniques for anautonomous mobile cleaning robot*, **Robotics and Autonomous Systems**, 14:23 (1995) 199–212.

[18] D.F. Wolf and G.S. Sukhatme, *Mobile Robot Simultaneous Localization andMapping in Dynamic Environments*, **Autonomous Robots**, 19:1 (2005) 53-65.

[19] H. Durrant-Whyte and T. Bailey, *Simultaneous localization and mapping: part I,* **IEEE Robotics Automation Magazine**, 13:2 (2006) 99-110.

[20] R. Siegwart, I.R. Nourbakhsh and D. Scaramuzza. *Introduction to Autonomous Mobile Robots*, Intelligent Robotics and Autonomous Agents Series, MIT Press, Massachusetts, 2011.
[21] C. Wu and C. Tsai, *Localization of an Autonomous Mobile Robot Based on Ultrasonic Sensory Information*, **Journal of Intelligent and Robotic Systems**, 30:3 (2001) 267–277.
[22] Anonymous. (2017). http://www.wikizero.org/index.php?q (on-line access on 22 Dec, 2017).
[23] D. W. Hodo, J. Y. Hung, D. M. Bevly and S. Millhouse, American Control Conference, New York, NY, July 9-13,(2007), pp. 2165-2170.
[24] Anonymous. (2017). https://www.sparkfun.com/products/retired/10736 (on-line access on 28 Oct, 2017).
[25] Anonymous. (2017). https://www.pololu.com/product/2827 (on-line access on 28 Oct, 2017).
[26] J. Clark and R. Fierro, *Mobile robotic sensors for perimeter detection and tracking*, **ISA Transactions**, 46:1 (2007) 3–13.
[27] S. Yun, Y. J. Lee and S. Sung, IEEE Aerospace Conference, Big Sky, MT, March 2-9, (2013), pp. 1-10.
[28] Anonymous. (2017). https://www.robotshop.com/en/rplidar-360-laser-scanner.html (on-line access on 28 Oct, 2017).
[29] I. Matijevics, 5th International Symposium on Intelligent Systems and Informatics, Subotica, Serbia, Aug 24-25, (2007), pp. 177-181.
[30] Anonymous. (2017). https://www.pololu.com/product/1107 (on-line access on 28 Oct, 2017).
[31] Anonymous. (2017). https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ (on-line access on 28 Oct, 2017).
[32] Anonymous. (2017). http://www.st.com/en/evaluation-tools/stm32f4discovery.html (on-line access on 28 Oct, 2017).
[33] Anonymous. (2017). https://www.makerfabs.com/L298-Dual-H-Bridge-Motor-Driver.html (on-line access on 28 Oct, 2017).
[34] Anonymous. (2017). http://www.8051projects.net/wiki/Pulse_Width_Modulation/ (on-line access on 5 May, 2017).
[35] Anonymous. (2017). http://wiki.ros.org/roscore (on-line access on 1 May, 2017).
[36] Anonymous. (2017). http://wiki.ros.org/Topics (on-line access on 1 May, 2017).
[37] Anonymous. (2017). http://www.ros.org/wp-content/uploads/2013/12/tf.png (on-line access on 16 Dec, 2017).
[38] Z. J. Chong, B. Qin, T. Bandyopadhyay, M. H. Ang, E. Frazzoli and D. Rus, IEEE International Conference on Robotics and Automation, Karlsruhe, May 6-10, (2013), pp. 1554-1559.
[39] L. Marin, M. Valles, A. Soriano, A. Valera and P. Albertos, *Multi Sensor Fusion Framework for Indoor-Outdoor Localization of Limited Resource Mobile Robots*, **Sensors**, 13:10 (2013) 14133–14160.
[40] M.B. Alatise and G.P. Hancke, *Pose Estimation of a Mobile Robot Basedon Fusion of IMU Data and Vision Data Using an Extended Kalman Filter*, **Sensors**, 17:10 (2017) 14133–14160.
[41] S.Y. Chen, *Kalman Filter for Robot Vision: A Survey*, **IEEE Transactions on Industrial Electronics**, 59:11 (2012) 4409–4420.

[42] N. C. Mitsou and C. S. Tzafestas, Mediterranean Conference on Control & Automation, Athens, June 27-29, (2007), pp. 1-8.
[43] Anonymous. (2017). http://www.slamtec.com/en/lidar/a1 (on-line access on 23 Dec, 2017).
[44] Anonymous. (2017). http://www.gensace.de/gensace2200mah111v25c3s1plipobatterypack.html (on-line access on 28 Oct, 2017).
[45] Anonymous. (2017). https://www.onsemi.com/pub/Collateral/LM2596-D.PDF (on-line access on 28 Oct, 2017).

10. EKLER

EK 1. Kullanılan Donanımlara Ait Teknik Bilgiler

a) RPLIDAR

Kullanılan lidara ait teknik veriler aşağıda verilmiştir [43].

Çalışma Gerilimi	5v DC
Tarama Açısı	360 derece
Tarama Frekansı	5-10Hz arası ayarlanabilir
Örnekleme Sayısı	2000 örnek/sn
Ölçüm Mesafesi	6 metre
Hassasiyet	%0.1

b) Razor IMU

Kullanılan IMU'ya ait teknik veriler aşağıda verilmiştir [24].

Çalışma Gerilimi	5-16v DC
Cayroskop	3 eksen Dijital çıkışlı
İvme Olçer	3 eksen 13 bit çözünürlüklü
Pusula	3 eksen dijital
Sensör Çıkış Tipi	Seri çıkışlı

c) Enkoder

Kullanılan enkoderlere ait teknik veriler aşağıda verilmiştir [30].

Çalışma Gerilimi	3.5-20v DC
Tipi	Quadrature
Hassasiyet	Tur başına 64
Kanal Sayısı	2
Çıkış Tipi	Kare Dalga

d) Motor

Kullanılan motorlara ait teknik veriler aşağıda verilmiştir [30].

Çalışma Gerilimi	12v DC
Hız	80 RPM
Tork	18 kg/cm
Yüksüz Çalışma Akımı	300mA
Stall Akımı	5A
Redüksiyon Oranı	131:1

e) Raspberry Pi 3

Kullanılan tek kart bilgisayara ait teknik veriler aşağıda verilmiştir [31].

Çalışma Gerilimi	5 V DC
İşlemci	BCM2837 4 Çekirdek 1.2GHz 64bit
RAM	1 GB
Kablosuz Haberleşme	Wi-Fi, Bluetooth
Giriş Çıkış Portları	40 pin GPIO
USB	4 adet USB 2.0
Video	1 adet HDMI
Depolama	Micro SD Kart

f) STM32 Discovery

Mikrodenetleyici tabanlı kontrol ünitesine ait teknik veriler aşağıda verilmiştir [32].

Çalışma Gerilimi	3-5V DC
İşlemci	STM32F407vgt6
RAM	192 kByte
Flash	1 Mb
Çalışma Frekansı	168Mhz (Maksimum)

g) Pil

Kullanılan Pile ait teknik veriler aşağıda verilmiştir [44].

Pil Gerilimi	11.1v DC
Pil Türü	lithium Polymer (LiPo
Hücre Sayısı	3
Bağlantı Şekli	3S1P
Nominal Pil Akımı	2200mAH
Maksimum Deşarj Oranı	25C
Maximum Sürekli Pil Akımı	55A
Maksimum Anlık Akım	110A

h) LM2596 Anahtarlamalı Gerilim Regülatörü

Kullanılan regülatöre ait teknik veriler aşağıda verilmiştir [45].

Maksimum Giriş Gerilimi	40v DC
Çıkış Gerilimi	1.23-37v DC
Maksimum Sürekli Çıkış Akımı	150kHz
Türü	Switch Mode

EK 2. Geliştirilen Robota Ait Kod Örnekleri

a) Örnek ROS Kodu (C++ Dili)

```cpp
#include <ros/ros.h>
#include <geometry_msgs/Twist.h>
#include <sensor_msgs/Joy.h>
#include <stdlib.h>
#include <stdio.h>

bool joyEnableStatus = false;

class TeleopTurtle
{
public:
  TeleopTurtle();

private:
  void joyCallback(const sensor_msgs::Joy::ConstPtr& joy);

  ros::NodeHandle nh_;

  int speed=0;
  int linear_, angular_;
  int joyEnable0, joyEnable1, velIncrement, velDecrement;
  double l_scale_, a_scale_;
  ros::Publisher vel_pub_;
  ros::Publisher vel_pub_turtle;
  ros::Subscriber joy_sub_;

  bool deadman_pressed_;
};


TeleopTurtle::TeleopTurtle():
  linear_(3),
  angular_(0),
  joyEnable0(0),
  joyEnable1(1),
  velIncrement(5),
  velDecrement(4),
  l_scale_(0),
  a_scale_(0)
{
  nh_.param("axis_linear", linear_, linear_);
  nh_.param("axis_angular", angular_, angular_);
  nh_.param("joyEnable0", joyEnable0, joyEnable0);
  nh_.param("joyEnable1", joyEnable1, joyEnable1);
  nh_.param("velIncrement", velIncrement, velIncrement);
  nh_.param("velDecrement", velDecrement, velDecrement);
  nh_.param("scale_angular", a_scale_, a_scale_);
  nh_.param("scale_linear", l_scale_, l_scale_);

  vel_pub_ = nh_.advertise<geometry_msgs::Twist>("cmd_vel", 1);
  vel_pub_turtle = nh_.advertise<geometry_msgs::Twist>("turtle1/cmd_vel", 1);
  joy_sub_ = nh_.subscribe<sensor_msgs::Joy>("joy", 10, &TeleopTurtle::joyCallback, this);
}

void TeleopTurtle::joyCallback(const sensor_msgs::Joy::ConstPtr& joy)
{

  geometry_msgs::Twist twist;

  if((joy->buttons[joyEnable0] == 1) && (joy->buttons[joyEnable1] == 1))
  {
    if(joyEnableStatus == false)
      joyEnableStatus = true;
    else joyEnableStatus = false;
  }

    if(joyEnableStatus == true)
    {
      if((joy->buttons[velIncrement] == 1) )
      {
        if(speed >= 0 && speed < 10)
          speed++;
        else speed = 10;
      }
      if((joy->buttons[velDecrement] == 1) )
      {
        if(speed > 0 && speed <= 10)
          speed--;
        else speed = 0;
      }
      l_scale_ = speed;
      a_scale_ = speed;
      twist.angular.z = a_scale_*joy->axes[angular_];
      twist.linear.x  = l_scale_*joy->axes[linear_];
    }

  vel_pub_.publish(twist);
  vel_pub_turtle.publish(twist);
}

int main(int argc, char** argv)
{
  ros::init(argc, argv, "teleop_orROT");
  TeleopTurtle teleop_turtle;

  ros::spin();
}
```

b) Örnek Mikrodenetleyici Tabanlı Kontrolcü Kodu (C Dili)

```c
#include "encoders.h"
#include "serial.h"


int32_t positionL_last=0;
int32_t positionR_last=0;
int32_t positionL=0;
int32_t positionR=0;
int16_t speedL=0, speedR=0;
int32_t encoderR=0;
int32_t encoderR_last=0;
int32_t encoderL=0;
int32_t encoderL_last=0;

TickType_t firstTick = 0;

void getEncoderCounts(TIM_HandleTypeDef *timL, TIM_HandleTypeDef *timR){
    struct encoderMsg *encoders;
        encoders = &encoder_t;

    encoderL = timL->Instance->CNT  ; //get left encoder data.
    positionL += encoderL - 32768;
    encoders->positionL = positionL;
    TIM1->CNT = 32768;

    encoderR = timR->Instance->CNT  ; //get right encoder data.
    positionR += encoderR - 32768;
    encoders->positionR = positionR;
    TIM2->CNT = 32768;

    getSpeed();

    xQueueSend(encoderQueueHandle, (void*) &encoders, (TickType_t) 0);
}

void getSpeed(){
    struct encoderMsg *encoders;
           encoders = &encoder_t;

    TickType_t now=osKernelSysTick();
    int time=(now-firstTick)/portTICK_RATE_MS;

    speedR=(int16_t)((positionR-positionR_last)/time);
    speedL=(int16_t)((positionL-positionL_last)/time);

    encoders->speedR_QPPS=speedR;
    encoders->speedL_QPPS=speedL;

    //printf("speedL=%d speedR=%d\n", speedL, speedR);

    positionR_last=positionR;
    positionL_last=positionL;
    firstTick=now;
}

void sendSpeeds()
{
    uint8_t speeds[4]={};

    if(speedL<0){
        speeds[0] = -1*speedL;
        speeds[1] = 1;
    }
    else {
        speeds[0] = speedL;
        speeds[1] = 0;
    }

    if(speedR<0){
        speeds[2] = -1*speedR;
        speeds[3] = 1;
    }
    else {
        speeds[2] = speedR;
        speeds[3] = 0;
    }

    send4Byte(GET_SpeedOfMotors, (uint32_t*)speeds);
}
```

ÖZGEÇMİŞ

Orkan Murat ÇELİK, 1991 yılında Malatya'da doğmuştur. İlköğretimini Adıyaman Gölbaşı Yavuz İlköğretim Okulunda, lise eğitimini Adıyaman Gölbaşı Anadolu Lisesinde tamamlamıştır.

2009-2014 yılları arasında İnönü Universitesi, Mühendislik Fakültesi, Elektrik Elektronik Mühendisliği Bölümü'nde lisans eğitimini tamamlayarak Elektrik Elektronik Mühendisi olarak mezun olmuştur.

2014-2017 yılları arasında, özel sektörde Gömülü Yazılım Mühendisi olarak çalışan Orkan Murat ÇELİK, 2017 yılı itibariyle HAVELSAN Hava Elektronik Sanayi AŞ.'de Test Analisti olarak çalışmaktadır.