1. GİRİŞ

"Otomasyon" dediğimiz zaman akla ilk gelen sözcüklerden biri de robotlar ya da robotik teknolojiler. Objektif analiz kapasiteleri sayesinde insan kaynaklı hataları en aza indirmeleri beklenen robotlar, halen üretimde yaygın şekilde kullanılıyor. Dolayısıyla robot teknolojileri, Dördüncü Endüstri Devrimi'nin, yani Endüstri 4.0'ın etkisini de artırmak açısından gelecek vaat ediyor. Örneğin, akıllı fabrikalarda robotlar birbirini tanıyarak, iş bölümü yaparak, haberleşerek, analizler yaparak, değişikliklere daha hızlı uyum sağlayarak üretimi yönetebilir hale gelecek.

Gelişen teknoloji ile birlikte robotlar günlük hayatımızda, endüstride ve askeri uygulamalarda hızla kullanılmaya başlanmıştır. İnsan ile robotlar arası ilişkilerde robotların insanların ihtiyaçları doğrultusunda yönlendirilerek karmaşık kontrol işlemleri yerine kontrolünün basitleştirilmesi ve robota tanımlanan görevlerin geliştirilen algoritmalar tarafından otonom olarak gerçekleştirilmesi günümüzde rahatlıkla gerçekleştirilebilmektedir.

Robotların verilen görevleri yerine getirebilmesi, Endüstri 4.0'ın gerçekleştirilmeye çalışıldığı günümüzde; savunma sanayide, endüstriyel sanayide ve birçok alanda kullanılabilirliği açısından oldukça önemli bir konudur.

Dünya otonom araçlar ile ilk olarak 1982'de "Kara Şimşek" ile tanışmıştır. Bir vizyon ile başlayan bu serüven; 1990'daki bazı çalışmalar ve sonrasında 2014-2015 insansız araç çöl yarışları ve 2007 akıllı şehir içi araba yarışı ile önemli mesafe kat etmiştir. Özellikle 2007'deki yarışlardan sonra yük ve eşya taşımacılığında kullanılabilecek taşıtların otonom hale getirilmesi için aşılması gereken kritik aşamalar geçilmiştir. Aslında geliştirilen çevrenin algılanması, yapay zeka, karar verme, haritalama, konumlandırma vb. kritik teknolojiler, üretim sistemlerinde çok rahat şekilde uygulanabilecek hale gelmiştir. Geleceğin fabrikalarında olması beklenen gelişmeler den biride AGV, TGV gibi tanımlı bir rotate taşıma yapan sistemler yerine kend konumunu bilen ve etrafı algılayarak otonom olarak hareket eden robot taşıyıcılardır.

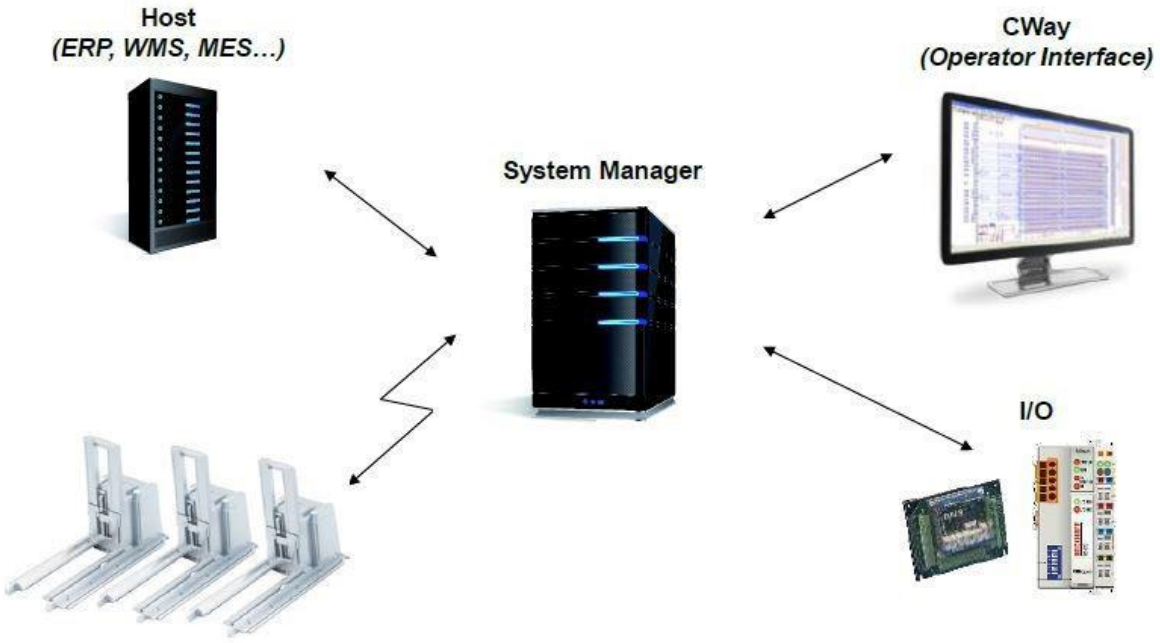
2.AGV SİSTEMLERİ

2.1. AGV SİSTEMİ NEDİR?

AGV (Automated Guided Vehicle) sistem tarafından kontrol edilen ve operatöre ihtiyaç duymayan bir dizi araç veya araçlar grubudur. Otomatik yönlendirmeli araçlar ya da otomatik güdümlü araçlar olarak adlandırılan AGV'ler merkezi bir kontrol ünitesi tarafından kontrol edilerek bir operatöre ihtiyaç duyulmadan otonom olarak çalışabilen, malzemelerin bir noktadan başka bir noktaya taşınması,istiflenmesi ve montaj hattındaki istasyonlara parça getirlmesi gibi işlevlerde kullanılan otomatik çalışan sistemlerdir.[1]

2.2 AGV SİSTEMLERİ NASIL ÇALIŞIR?

Şekil 2.1 de sistemin daha çalışma mantığını daha iyi anlamak için Agv sistemini oluşturan parçalar gösterilmiştir.



Şeki1 2.1: AVG Sistem Yönetimi

2.2.1 AGV Sisteminin Parçaları

- Araç,
- Sistem Denetleyicisi,
- Kablosuz iletişim,
- Kullanıcı arayüzü (CWay),
- Host Modül

2.2.1.1 Sistem Denetleyicisi

System controller tüm AGV sistemini yönetir. AGV sisteminin tüm yönetimi: rota planlama/ belirleme, trafik yönetimi, I/O denetimi ve iş emri verilmesi bu birimden sağlanır. İş emirleri/ sisteme 3 farklı kaynaktan verilebilir.

Bu Kaynaklar;

- Digitial I/O (sensör, buton vb.),
- Operatör arayüzü,
- Host system (ERP /MES/WMS).

2.2.1.2 Operatör arayüzü

AGV sisteminin realtime olarak izlenebilirliğini sağlayan arayüzüdür.

2.2.1.3 I/O

AGV sistemi çalışacağı tesis üzerindeki I/O birimleri ile haberleşme yeteneğine sahiptir. Bu yetenek; trafik kontrolü ve uygulamaya yönelik çözümler üretir.

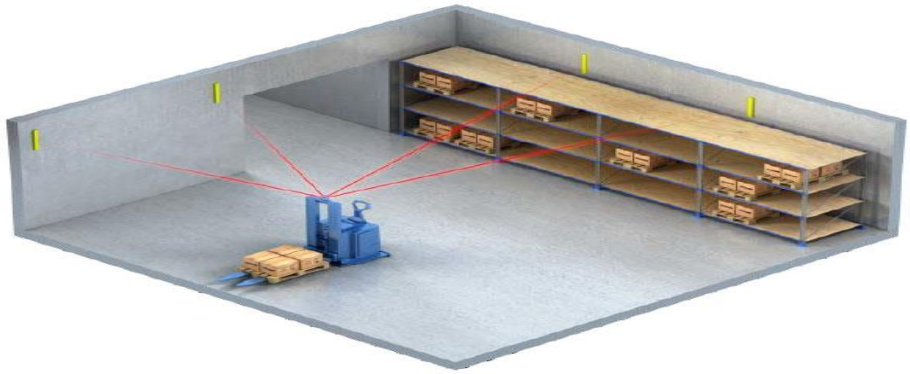
2.3 AGV SİSTEMLERDE NAVİGASYON TEKNOLOJİSİ

Araçların fabrikadaki yükleme ve boşaltma noktalarını bilmesi, AGV yolundaki diğer araç, makine ve personellere zarar vermeden bulması gerekir. Tek tip bir AGV yoktur. AGV'ler taşıdığı yüklere göre değişiklik gösterir. Ağır tonajlarda yük taşıyan ya da temiz ortamlarda hafif yük taşıyan AGV'ler mevcuttur.

2.3.1 LAZER NAVİGASYON

Arabalarla dolu bir şehirde olduğu gibi fabrikalarda da AGV'nin nereye gideceğine karar vermesi gerekir ve oluşacak trafik sıkışıklığından kaçınmak için fabrikanın da trafik kuralları olması şarttır. AGV sistemi bu kurallar bütünü ile taşıma proseslerini belirler.

Tesis içerisinde aracın taşıma işlemini gerçekleştireceği rota üzerine reflektörler yerleştirilir. Bu rota üzerindeki reflektörleri algılamak içinse bir lazer tarayıcı kullanılır. Reflektörlerden alınan yansımalar aracılığıyla araç konumunu sürekli güncelleyerek navigasyonunu sağlar. Şekil 2.2 de Lazer Navigasyon Sisteminin çalışma mantığını gösteren örnek bir resim verilmiştir.



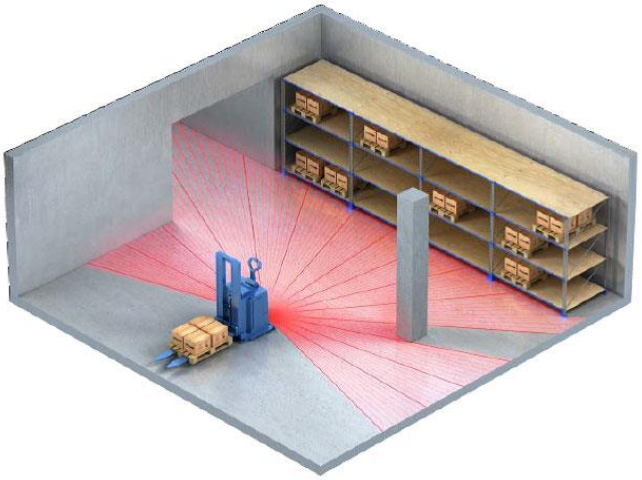
Şekil 2.2: Lazer Navigasyon Sistemi

Avantajları:

- Diğer navigasyon teknolojilerine göre daha yüksek hızları destekler. ±10mm hassasiyete sahiptir. Gelecekteki değişiklikler için esnektir.

2.3.2 Doğal Navigasyon

Lazer navigasyonda olduğu gibi AGV'nin önceden belirlenmiş rota üzerinde taşıma işlemini gerçekleştirebilmesi için yansımalara ihtiyacı vardır. Lazer navigasyonda bu yansımalar reflektörler üzerinden alınırken, doğal navigasyonda ise cisimler üzerinden yansıma alınarak navigasyon sağlanır. Şekil 2.3 de Doğal Navigasyon Sisteminin çalışma mantığını gösteren örnek bir resim verilmiştir.



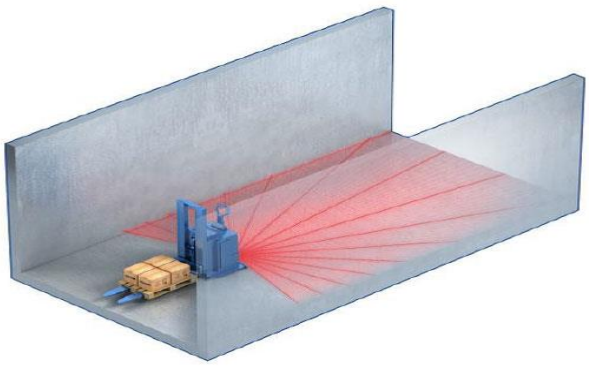
Şekil 2.3: Doğal Navigasyon Sistemi

Avantajları:

- Hassasiyeti ±10 mm ulaşılabilir.
- Reflektör gerekmez.
- Kurulum ve bakım maliyeti azdır

2.3.3 Range Navigasyon

Tesis içerisinde bulunan düzlemler (duvar vb.) aracılığıyla pozisyonunu tespit etmek için açı ve mesafe hesaplar. Dar çalışma alanlarında uygulanabilir ve uyumlu bir sensör kullanılırsa, ekstra donanım ihtiyacı olmaz. Genellikle otomatik araç yükleme operasyonlarında tercih edilmektedir. Şekil 2.4 de Range Navigasyon Sisteminin çalışma mantığını gösteren örnek bir resim verilmiştir.



Şekil 2.4: Range Navigasyon Sistem

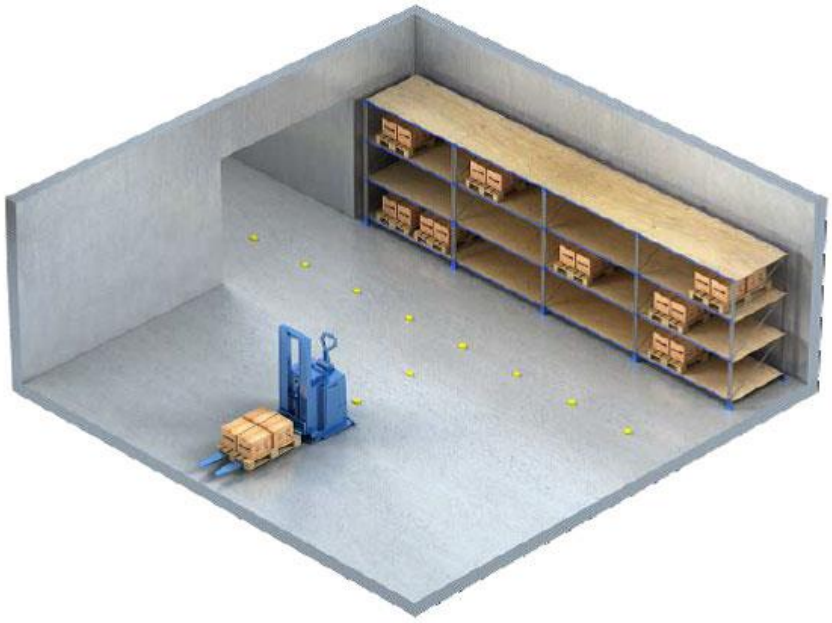
Avantajları:

- Orta düzeyedeki hızları destekler.
- Reflektör yerleşimi yapılamayan ve tesis düzeni uzun süre değişmeyen yerlerde kullanılır.

2.3.4 Spot Navigasyon

Spot navigasyon, Range navigasyonda olduğu gibi AGV'nin pozisyonu tespit etmek için zemine yerleştirilen, araçların belirli nokta ve boşlukları sayarak ilerlediği sistemdir.

Şekil 2.5 de Spot Navigasyon Sisteminin çalışma mantığını gösteren örnek bir resim verilmiştir.



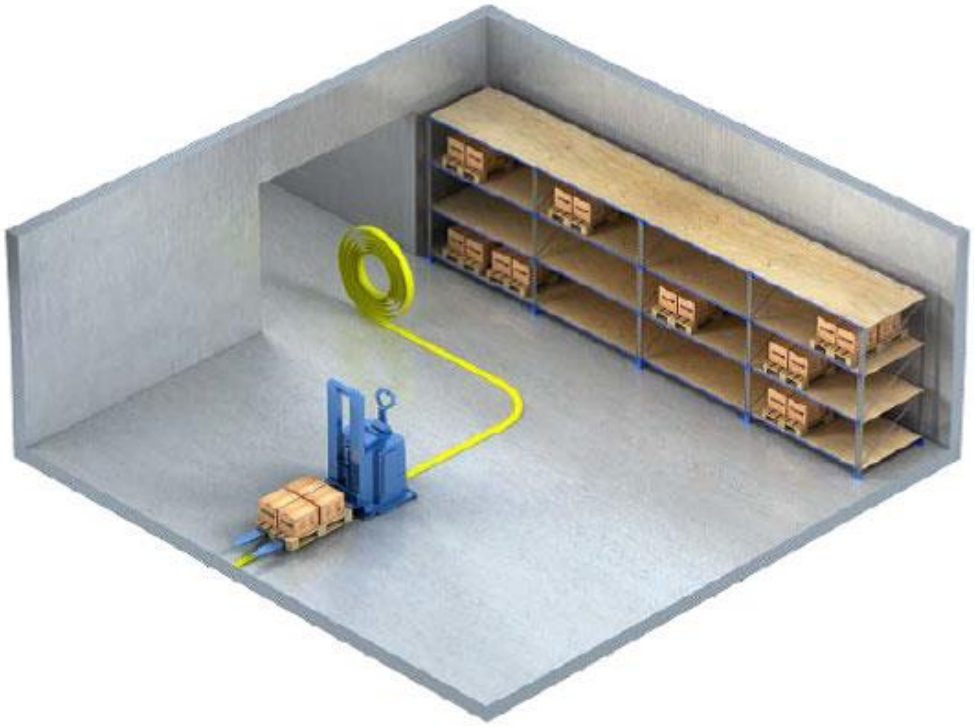
Şekil 2.5: Spot Navigasyon Sistemi

Avantajları:

- Orta düzeydeki hızları destekler.
- Farklı manevra kabiliyeti gerektiren araçlarda kullanıma uygundur.
- Reflektör yerleşimi yapılamayan ve tesis düzeni üzün süre değişmeyen alanlarda kullanılır.
- Derin istifleme yapılacak yerlerde kullanılır

2.3.5 Manyetik Bantlı Navigasyon

Manyetik bant navigasyonunda araç, zeminde manyetik bantlar kullanarak yönünü tayin eder. Bu gezinme yöntemi spot navigasyona ve endüktif telli navigasyona çok benzemektedir. Manyetik bant mekanik hasarlara karşı hassastır. Şekil 2.6 da Manyetik Bantlı Navigasyon Sisteminin çalışma mantığını gösteren örnek bir resim verilmiştir.



Şekil 2.6: Manyetik Bantlı Navigasyon Sistemi

Avantajları:

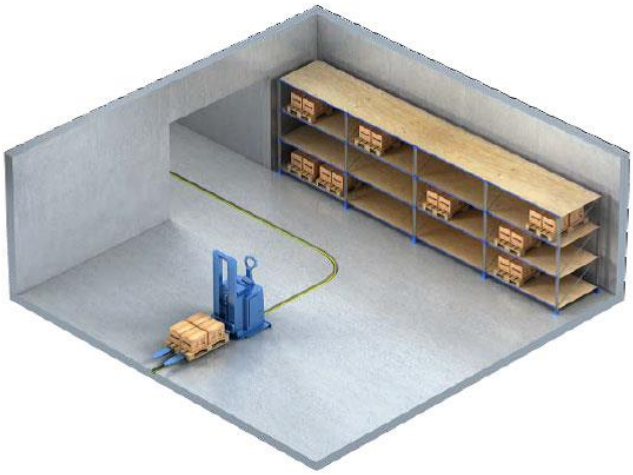
- Orta düzey hızları destekler.
- Basit uygulamalar için tasarlanan araçlara uygundur.

2.3.6 İnduktif Kablolu Navigasyon

İnduktif kablolu navigasyon sisteminde aracın izleyeceği yollara belirli işlemler sonucunda gömülü teller yerleştirilir. Ortamda özellikle reflektör yerleşiminde zorluk yaşıyor veya araçta bir lazer tarayıcı takılması zorsa bu sistem tercih edilmektedir. Bu navigasyon

yöntemi, manyetik bant navigasyonuna benzer ve teller yerde olduğu için mekanik hasar için hassaslığı daha azdır. Dar koridor ve Piggy Back türü AGV tiplerinde tercih edilir.

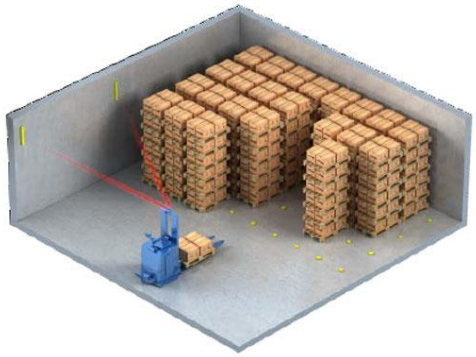
Şekil 2.7 de İndüktif Kablolu Navigasyon Sisteminin çalışma mantığını gösteren örnek bir resim verilmiştir.



Şekil 2.7: İndüktif Kablolu Navigasyon Sistemi

2.3.7 Multi Navigasyon

Araç sürüş sırasında farklı navigasyon yöntemleri arasında geçiş yapabilir. Taşıma işleminin yapıldığı rota üzerinde reflektörlerin görüş açısı azalıyor ya da kayboluyorsa (derin istifleme) Taşımanın başarı ile gerçekleştirebilmesi için tercih edilir. Şekil 2.8 da Multi Navigasyon Sisteminin çalışma mantığını gösteren örnek bir resim verilmiştir.



Şekil 2.8:Multi Navigasyon Sistem

2.4 AGV Şarj Sistemleri

2.4.1 Fırsata Bağlı Şarj

AGV'ler sürekli olarak akü durumlarını kontrol ederek sisteme bilgi aktarırlar. Bu sayede AGV üzerinde iş emri olmadığı takdirde (aracın iş emrİ alması sistem tarafından engellenir ve bu sayede AGV düşük akü nedeniyle yolda kalmaz.) fırsata bağlı AGV'yi şarj istasyonuna yönlendirir.

AGV'ler yeni iş emirlerini beklerken, otomatik olarak şarj istasyonuna yönelerek operatörlerden bağımsız şekilde şarj olurlar.

2.4.2 Operatöre Bağlı Şarj

AGV'ler sürekli olarak akü durumlarını kontrol ederek sisteme bilgi verirler. Bu sayede sistem AGV üzerinde iş emri olmadığı takdirde (aracın iş emrini alması sistem tarafından engellenir ve bu sayede AGV düşük akü nedeni ile yolda kalmaz.) AGV'yi şarj istasyonuna göndererek operatör tarafından akü değişimini sağlanır.

2.5 AGV Neden İyi Bir Yatırımdır?

- Gelişmiş üretkenlik
- Azaltılmış operasyon ve nakliye ücretleri
- Daha az insan ve daha az ürün zararı

Şekil 2.9 da AGV Sistemlerinin diğer sistemlerle karşılaştırılmasını gösteren bir grafik verilmiştir.

Şekil 2.9: Manuel Sistem ile AGV Sistem Karşılaştırması

3. PROGRAMLARIN TANITIMI

3.1 ROS (Robot Operating System)

ROS(Robot Operating System), isminden işletim sistemi gibi anlaşılsa da aslında değildir. ROS bilgisayar üzerinden robot bileşenlerini kontrol etmemizi sağlan BSD lisanslı bir yazılım sistemdir. Açık kaynak kodlu bir yazılımdır ve ROS'u kullanabilmemiz için Linux tabanlı bir işletim sistemine ihtiyaç vardır. Bu uygulamanın amacı robot ve programcı arasındaki ilişkiyi sağlamak ve belirli bir standart oluşturmaktır. ROS içinde barındırdığı standart kütüphanelerin ve desteklediği cihazların dışında, geliştiriciler tarafından yazılmış kütüphanelere ve sürücülere de destek verdiği için her geçen gün kapsamını arttırmakta ve robot dünyasında standart konumunu almaktadır.[2]

3.1.2 ROS Ne İşe Yarar?

ROS robotlarda yaygın olarak kullanılan bir yazılımdır. Kod üzerinden çok az değişiklik yaparak diğer robotlar üzerinde de çalışabilecek bir yazılım parçası geliştirmeyi hedefler. ROS sayesinde standart bir robot için yazılan standart bir algoritma yükle-kullan hızında çalıştırılabilir konumda olacaktır. Robota takılan sensörlerden alınan veri ROS ile bilgisayara ulaşacaktır, bilgisayarda işlenen veri robota bir görev olarak geri dönebilecektir. Aradaki bu haberleşme sistemi ROS arayüzündeki topicler ve mesajlar yardımıyla yapılacaktır. ROS, Gazebo simülasyon ortamıyla tam uyumlu olarak çalışmaktadır. Bu sayede projenin prototip aşamasında ROS Package haline getirilen geliştirilen veri yapısını test etmek için Gazebo simülasyon ortamı hızlı bir geliştirme için kullanılabilir.

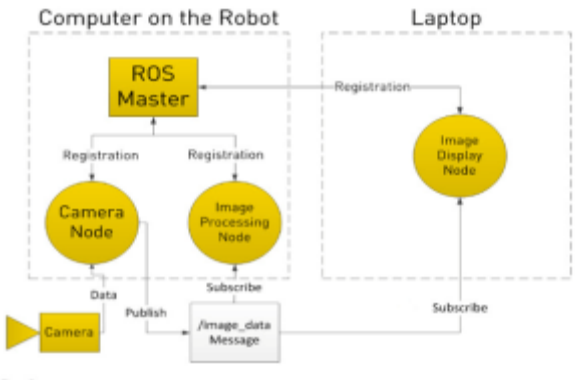
3.1.3 ROS'un Tarihçesi

ROS ilk olarak 2007 yılında Stanford Yapay Zeka Laboratuvarı'nda geliştirilmeye başlandı. Çalışmalar 2008 ve 2013 yılları arasında Willow Garage enstitüsünde devam etti. Bu süre içerisinden bu enstitüdeki mühendisler yirmiden fazla kurum ile görüşüp işbirliği yaptılar. Bu kurumlar, donanım ekleyerek ve kod örnekleri ile ROS'da proje geliştirmeye başlamışlardır. Daha sonrasında ise robot üreten şirketlerde ROS'da kullanılmak için ürünlerini ROS'a adapte etmeye başlamışlardır. 2013 yılının şubat ayında ROS yönetimi OSR Vakfı bünyesine geçiş yaptı. Yine aynı yıl Ağustos ayında Willow Garage, Suitable Technologies adlı şirkete devredildi. Willow Garage Tarafından

oluşturulan PR2'nin destek sorumlulukları da Clearpath Robotics tarafından alındı. Tüm bu değişim ve gelişmelerin ardından robotlarda kullanılan sensörler ve aktuatörler de ROS ile kullanılmak üzere tekrardan uyarlanmışlardır. Her geçen gün ROS destekli cihazların sayısı artmaktadır.

3.1.4 ROS Nasıl Çalışır?

Bir ROS sistemi yayınlama/abone mesajlaşma modelini kullanarak diğer düğümler ile iletişim kuran düğümlerden oluşur. Bir ROS sistemi içerisinden birçok düğüm bulunabilir. Düğümler, hesaplama yapabilen işlemlerdir. Örneğin; bir düğüm kameradan görüntü alır, bir düğüm görüntüyü işler, bir düğüm görüntüyü görüntülenmesini sağlayabilir. Bu düğümlerin birbirlerinden haberdar olup birbirleri ile haberleşebilmeleri için ROS Master'a ihtiyaç vardır. ROS Master merkezi XML-RPC sunucusudur yani mesajları içeren ağ tabanlı bir yapıdır. Biraz daha ayrıntılı şekilde bu konuya açıklık getirecek olursak; ROS, ROS Master ile başlar ve ROS düğümlerinin birbirlerini bulmasını ve birbirleriyle konuşmasını sağlar. Düğümler haberleşmeyi konular yayınlayıp bu konulara abone olarak gerçekleştirir. Diyelim ki robot üzerinden bulunan bir kameradan alınan görüntüleri robot üzerinde ya da başka bir bilgisayarda görmek istiyoruz. Kamera ile iletişim kurmak için bir kamera düğümü, görüntüleri işlemek için bir görüntü işleme düğümü ve görüntü gösterme düğümü vardır. Bu düğümlerin hepsi ROS Master'a kayıtlıdır. Kamera düğümü, ROS Master'a kayıt olurken /image_data adında bir konu yayınlayacağını belirtir. Diğer düğümler ise kameradan yayınlanan bu konuya abone olduklarını belirtir. Bu şekilde kamera düğümü kameradan alınan verileri /image_data konusu üzerinden mesajları diğer düğümlere iletir.[3] Not: Bir düğüm birden fazla konu yayınlayabilir ve birden fazla konuya abone olabilirler. Ros'un çalışma mantığına ilişkin görsel şekil 3.1 de verilmiştir.



Şekil 3.1: ROS Çalışma mantığı

3.1.5 ROS ve Komponentleri

3.1.5.1 ROS Core

ROS Master,

- Merkezi XML-RPC sunucusudur.
- Bağlantıların iletişimi sağlamaktadır.
- Sabit konfigürasyon parametreleri ya da diğer keyfi verileri kaydeder.
- Aslında, insanların anlayabileceği dilde mesajları barındırran ağ tabanlı bir yapıdır.

3.1.5.2 ROS Stacks & Packages

ROS kodu iki farklı seviye içerisinde gruplanmıştır;

- Packages
- Stacks

Packages, ROS'a çok küçük bir bağımlılıklar bağlanmış yazılım koleksiyonlarıdır. Stacks ise bu ufak parçaların bir araya gelerek oluşturduğu dağıtıma verilen isimdir.

Packages

- Kodlarınızı, build yapılandırmasınızı ve launch dosyalarınızı barındıran bir dizindir.
- Birden fazla node içerebilir.
- İçerisinde manifest.xml adında bağımlılıkların tanımlandığı bir konfigürasyon dosyası barındırır.
- Sadece paketin amacı ile ilgili kodları içerir.

Nodes

- Bazı fonksyionları yerine getiren yapılardır.
- Diğer Topic ve Servicesler ile haberleşebilir.
- Node isimleri benzersizdir.
- Node yapısı, iyi ayrılmış, skale edilebilir bir yapının oluşturulması için vardır.

Build System

- Kodların nasıl derleneceğinin belirtildiği CMakelist.txt dosyasına ihtiyaç duymaktadır.
- rosmake komutu paket ve bağımlılıkları derler.
- Eğer bağımlılıklar indirilmemişse, indirilir ve kurulur.
- Aynı anda birden fazla paket derlenebilir. ROS önce bağımlılıkları çözer.

3.1.5.3 Komut Satırı

ROS aşağıdaki bazı komutları içerisinde barındırır;

- roscd: ROS paketi dizinine geçer.
- rosls: Paket içeriğini listeler.
- rosmake: Paketi derler.
- roscreate-pkg: Yeni bir paket oluşturur.
- rosdep: Paket bağımlılıklarını listeler.
- roscp: Paketler arası kopyalama işlemi gerçekleştirir.
- rostest: Test dosyalarını çalıştırır.

3.1.5.4 İletişim Yolları

Topic (Pub/Sub)

- Asenkron iletişimler için kullanılır.
- İçerisinde güçlü mesaj tipleri barındırır.
- Callback fonksyionlara sahiptir, bu sayede mutli-thread çalışabilir.
- İstek/cevap etkileşimleri için uygun değildir.
- Çoktan-Çoka bağlantılar için uygundur.

Service (Higher Priority)

- Standart fonksiyonlar gibi, senkron olarak kullanılır.
- Güçlü mesaj tiplerini içerisinde barındırır.
- Bire-bir iletişim için uygundur.
- Birden fazla istemciler olabilir.

Actions

- Topic yapısının üzerine inşaa edilmiştir.
- Uzun işlemler gerçekleştirilmesi için kullanılır.
- İptal edilebilir.

Messages

- Node iletişimi mesaj gönderimi üzerine kuruludur.
- Mesajlar, özel tipli alanlardan oluşan bir veri yapısından ibarettir.
- ROS içerisinde birçok hazır mesaj standartı vardır ve özel olarak buna yenileri eklenebilir.

3.1.6 ROS'un Alternatifleri

ROS'a alternatif olabilecek başka robot yazılımları da mevcuttur. Bunlar Player, YARP, Orocos, Carmen, Orca, MOOS, Microsoft Robotic Studio.

Player

Robot ve sensör uygulamaları için ücretsiz yazılım araçları sunar. Amacı oyuncu projesi ile robot ve sensör sitemlerinden araştırma yapmayı sağlayan özgür yazılımı yaratmaktır. Ağ arabirimi sağlar, istemci/sunucu modeli mevcuttur. Herhangi bir programlama dili ile herhangi bir bilgisayardan robot ile ağ bağlantısı kurup eşzamanlı çalışmayı sağlar. Kullanılabilir son yayınlanan stabil sürümü 2010 yılında yayınlanmıştır.

CARMEN

CARMEN, mobil robot kontrolü için açık kaynak kodlu bir yazılımdır. Sensör kontrolü, engelden kaçınma, yerleştirme, yol planlama ve haritalama gibi temel prensipleri sağlamak için tasarlanmış modüler bir yazılımdır. Kullanılabilir son yayınlanan stabil sürümü 2008 yılında yayınlanmıştır.

ORCA

Orca, bileşen tabanlı robot sistemleri geliştirmek için açık kaynak kodlu bir yazılım sunmuştur. Kullanılabilir son yayınlanan stabil sürümü 2009 yılında yayınlanmıştır.

Microsoft Robotics Studio

Robotik uygulamaları oluşturmak için oluşturulmuş .NET tabanlı programlama ortamıdır. Akademik, hobi ve ticari geliştiricilere yöneliktir. Kullanılabilir son stabil sürümü 2012 de yayınlanmıştır. Windows 7 gerektirir. Windows 8 Consumer Preview'da test edildi ancak Windows 8'in son sürümüne kadar bile desteklenmedi.

3.1.7 Neden ROS?

Neden diğer X yazılımı değil de ROS? Bu soru birçok cevaba sahiptir. Kısa ve net bir cevap şu şekilde olabilir.

ROS (Robot Operating System) adının çağrıştırdığının aksine işletim sisteminden ziyade bir arayüz olarak tanımlayabileceğimiz bir uygulamadır. Bu uygulamanın amacı robot ve programcı arasındaki ilişkiyi sağlamak ve belirli bir standart oluşturmaktır. ROS içinde barındırdığı standart kütüphanelerin ve desteklediği cihazların dışında, geliştiriciler tarafından yazılmış kütüphanelere ve sürücülere de destek verdiği için her geçen gün kapsamını arttırmakta ve robot dünyasında standart konumunu almaktadır. ROS'un sunduğu bir diğer avantaj ise yazılan kodlar her robota ayrı olarak yazılmasının yerine bir kere yazılan kodun desteklenen bütün cihazlarda çalıştırılabilmesidir. Bu şekilde alınan standart bir robot için yazılan standart bir algoritma yükle-kullan hızında çalıştırılabilir konumda olacaktır. Robota takılan sensörlerden alınan veri ROS ile bilgisayara ulaşacaktır, bilgisayarda işlenen veri robota bir görev olarak geri dönebilecektir. Aradaki bu haberleşme sistemi ROS arayüzündeki topicler ve mesajlar yardımıyla yapılacaktır. ROS, Gazebo simülasyon ortamıyla tam uyumlu olarak çalışmaktadır. Bu sayede projenin prototip aşamasında ROS Package haline getirilen geliştirilen veri yapısını test etmek için Gazebo simülasyon ortamı hızlı bir geliştirme için kullanılabilir.

Diğer avantajları maddeler halinde aşağıda listelenmiştir.

- ROS'un amacı kodu desteklemek olduğu için yeniden kullanımlığı arttırır ve size her projede tekerleği sıfırdan icat etmeniz konusunda diretmez.
- İki işletim sistemi arasında çalışabilir. Örneğin; robot üzerinden oluşturduğunuz ROS Master'a Windows üzerinde kurulu olan MATLAB'tan robota kod gönderebilir ve robottan geriye veri döndürebilirsiniz. Bu sadece ROS sizi sadece Linux kullanmaya zorlamaz ve diğer işletim sistemlerinin sunduğu kolaylıklardan da yararlanabilirsiniz.

- Dil bağımsızdır. C++, python, lisp, java, lua gibi birçok farklı programlama dilini desteklemektedir ve sizi belirli bir programlama dilini öğrenmeye zorlamaz.
- Açık kaynak kodludur. Size sunduğu topluluk deposundaki güçlü araçları dilediniz gibi kullanılıp yeniden düzenleyebilirsiniz. Bunlardan ön önemlileri görselleştirme, simülatörler ve hata ayıklama araçlarıdır.
- Her geçen gün destekledi robot ve sensör sayısı artmaktadır.
- Platformlar arası çalışabilir. Oluşturduğumuz düğümleri farklı programlama dillerinde yazabiliriz. Bir düğüm C++ iken diğer düğüm java yada python olabilir.
- Modülerdir. Diğer robot uygulamalarının çoğunda bir sorun çıktığında bu sorun ana kodun kitlenmesine sebep olup robot uygulamasını durdurabilir. Fakat ROS'ta durum böyle değildir. ROS'ta her işlem için bir düğüm olduğundan dolayı bir düğüm bozulduğunda diğer düğümler çalışmaya devam eder.
- Her bir işlem için ayrı konu yayınlandığı için karmaşıklığı azaltır. Bu sayede hata ayıklamayı kolaylaştırır.
- Aktif bir topluluğa sahiptir. ROS ile ilgili birçok kaynak, doküman, kitap, makale, bildiri bulabilirsiniz.
- 2014 yılında yayınlanan İndigo sürümü 2019, 2016 yılında yayınlanan Kinetic Kame sürümü 2021 yılına kadar desteklenmektedir.
- ROS FastSLAM 2.0 algoritmasını GMapping paketinin altında desteklemektedir.

3.2 OPENCV

OpenCV (Open Source Computer Vision) açık kaynak kodlı görüntü işleme kütüphanesidir. 1999 yılında intel tarafından geliştirilmeye başlanmış daha sonra Itseez, Willow, Nvidia, AMD, Google gibi şirket ve toplulukların desteği ile gelişim sürecine devam etmiştir. İlk sürüm olan OpenCV alfa 2000 yılında piyasaya çıkmıştır. İlk etapta c programlama dili ile geliştirilmeye başlanmış ve daha sonra birçok algoritması C++ dili ile geliştirilmiştir. Open source yani açık kaynak kodlu bir kütüphanedir ve BSD lisansı ile altında geliştirilmektedir. BSD lisansına sahip olması bu kütüphaneyi istediğiniz projede ücretsiz olarak kullanabileceğimiz anlamına gelmektedir. OpenCV platformdan bağımsız bir kütüphanedir, bu sayede Windows, Linux, FreeBSD, Android, Mac OS VE İOS platformlarında çalışabilmektedir. C++, C, Python, Java, Matlab, EmguCV kütüphanesi aracılığıyla da Visual Basic.Net, C# ve Visual C++ dilleri ile topluluklar tarafından

geliştirilen farklı wrapperlar aracılığıyla Perl ve Ruby programlama dilleri ile kolaylıklar OpenCV uygulamaları geliştirilebilir.

27-05-2016 tarihli güncelleme ile OpenCV geliştirici Itseez firması Intel tarafından satın alındı. OpenCV geliştirilmesine intel çatısı altından devam edileceği duyuruldu.

3.2.1 OPEN CV BİLEŞENLERİ

OpenCv kütüphanesini daha iyi anlamak için mimarisinden ve OpenCV'yi oluşturan bileşenlerden bahsedelim.

- Core: OpenCV'nin temel fonksiyonları ve matris, point, size gibi veri yapılarını bulundurur. Ayrıca görüntü izerine çizim yapabilmek için kullanılabilecek metotları ve XML işlemleri için gerekli bilgileri bulundurur.
- HighGui: Resim görüntüleme, pencereleri yönetme ve grafiksel kullanıcı arabirimleri için gerekli olabilecek metodları bulundurur. 3.0 öncesi sürümlerde dosya sistemi üzerinden resim dosyası okuma ve yazma işlemlerini yerine getiren metodları barındırmaktaydı.
- Imgproc: Filtreleme operatörleri, kenar bulma, nesne belirleme, renk uzayı yönetimi, renk yönetimi ve eşikleme gibi neredeyse tüm fonksiyonları içine alan bir pakettir. 3 ve sonrası sürümlerde bazı fonksiyonlar değişmiş olsada 2 ve 3 sürümünde birçok fonksiyon aynıdır.
- Imgcodecs: Dosya sistemi üzerinden resim ve video okuma/yazma işlerimlerini yerine getiren metodları barındırmaktadır.
- Videoio: Kameralara ve video cihazlarına erişmek, görüntü almak ve görüntü yazmak için gerekli metodları barındırır.

3.2.2 Alternatif Görüntü İşleme Kütüphaneleri

Görüntü işleme projelerinde kullanacağımız kütüphaneyi amacımıza uygun olarak seçmemiz önemlidir. Bu seçimi yaparken ne yapmak istediğimize doğru karar vermeliyiz, örneğin sadece kameradan görüntü almak için projemize OpenCv entegre etmemize gerek yoktur. Bu gibi durumlarda OpenCV'nin neden iyi olduğunu anlayabilmek amacıyla alternatif görüntü işleme kütüphanelerinden bahsedelim.

- MATLAB: Matlab için bir görüntü işleme kütüphanesi olarak bahsetmek doğru değildir fakat içerisinde görüntü işlemeye yönelik temel algoritmaları barındırmaktadır. Dördüncü nesil ve çok amaçlı bir programlama dilidir. Akademik araştırmalardai performansın önemli olmadığı durumlarda temel görüntü işleme

işlemleri için tercih edilebilir. Matlab kullanarak OpenCV kütüphanesi ile etkileşimli olarakda uygulamalar geliştirmek mümkündür.

- Halcon: Endüstriyel projeler için tercih edilen, kendi içerisinde geliştirme ortamı yanı sıra çeşitli programlama dilleri için kütüphanesi bulunan, yapay görme(machine vision) odaklı ticari bir yazılımdır. İçerisinde birçok hazır fonksiyon bulundurur bu sayede hızlı uygulamalar geliştirilebilir. OpenCV açık kaynak kodlu bir kütüphanedir ve computer vision odaklıdır. Bu yönleri Halcon'dan ayrılmaktadır.
- Open Frameworks: Açık kaynak kodlu olarak geliştirilen bu kütüphane C++ programlama dili ile geliştirilen bu proje OS X, Linux, Embedded Linux(ARM), İOS, Android platformlarında çalışabilmektedir. OpenCV kütüphanesinin birçok algoritmasını kullanır ve temel çıkış amacı kolay ve hızlı uygulama geliştirmektir. Örneğin OpenCV ile 2t sürede gerçekleştirilen bi işlemi 1t sürede gerçekleştirebiliriz, bunun temel sebebi birçok fonksiyon aracılığıyla standart hale gelmiş olan işleri tek satır ile yapabilmesidir.(Nesne takibi, renk takibi, karşılaştırma vb.)
- CIMG: Açık kaynak kodlu görüntü işleme kütüphanesidir. Windows, Linux ve OS X platformu üzerinde çalışmaktadır. Birçok algoritmayı barındırmaktadır fakat OpenCV kadar performanslı ve geniş bir algoritma yapısına sahip değildir.

  Endrov, ImageJ, Lead tools, Pink, Image Magick, Boost İse alternatif kütüphanelerdir.

3.2.3 Raspberry Pi 3 OpenCV Kurulumu

Öncelikle daha önce opencv kurmaya çalıştık, başarısız olduk ve kalıntıları hala duruyor ise onları temizleyelim.

>>sudo apt-get remove libopencv*

Sudo apt-get autoremove

Sd kartımızın en az 16gb olmasına ve kurulumdan önce sistemi genişlettiğimizden emin olalım.

Şimdi sistemimizi güncelleyelim bu işlemi daha önce yapmadıysak veya güncel bir imaj yüklü değil ise bu işlem uzun sürebilir.

>>sudo apt-get update

sudo apt-get upgrade

sudo rpi-update

sudo reboot

Reboot edip sistemi tekrardan başlattıktan sonra konsala tekrar geçiş yapalım.

Ardından developer tool(Geliştirici araçlarımızın kurulumunu yapalım.)

>>sudo apt-get install build-essential cmake cmake-curses-guipkg-config

OpenCV kullanırken ihtiyacı olacak kütüphanelerin kurulumunu yapıyoruz.

>>sudo apt-get install

lipjpeg-dev

libtiff5-dev

libjasper-dev

libpng12-dev

libavcodec-dev

libavformat-dev

libswscale-dev

libeigen3-dev

libxvidcore-dev

libx264-dev

libgtk2.0-dev

Raspbian Jessie imajı ile birlikte python2.7 ve python3 yüklü şekilde geliyor, şimdi iste OpenCV'yi python3 için kuracağımız kısıma geçelim, bunun için öncelikle numpy ve python-dev kurulumlarını yapalım.

>>sudo apt-get install python3-dev python3-numpy

Şimdi OpenCV yi kurmaya başlayabiliriz. Öncelikle /home/pi klasörü altında OpenCV isimli bir dosya oluşturalum ve o dosyanın içine kurulumları yapalım.

>>mkdir /home/pi/opencv

Şimdi dosyanın içine girelim.

>>cd/home/pi/opencv

Wget https://github.com/opencv/opencv/archive/3.2.0.zip

-O opencv_source.zip

Wget https://github.com/opencv/opencv_contrib/arhive/3.2.0.zip

-O opencv_contrib.zip

Aynı dosya dizininde çıkarmadan, indirdiğimiz .zip uzantılı dosyaları çıkaralım.

>>unzip opencv_source.zip

Unzip opencv_contrib.zip

Artık OpenCV yi çalışabilir hale getirmek için derleme işlemlerini yapalım. Öncelikle dosya dizinimiz içerisinden opencv-3.2.0 içerisine girelim. Derleme için yeni bir dosya açalım bu dizinde.

>><cd opencv-3.2.0

Mkdir build

cd build

Build dosyası içerisinde aşağıdaki cmake ifadelerini yapıştıralım.

>>cmake –D CMAKE_BUİLD_TYPE=RELEASE|

-D CMAKE_INSTALL_PREFİX=/usr/local|

-D BUİLD_WITH_DEBUG_INFO=OFF|

-D BUILD_DOCS=OFF|

-D BUILD_EXAMPLES=OFF|

-D BUILD_TESTS=OFF|

-D BUILD_opencv_ts=OFF|

-D BUILD_PREF_TESTS=OFF|

-D INSTALL_C_EXAMPLES=ON|

-D ISNTALL_PYTHON_EXAMPLES=ON|

-D

OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib3.2.0/modules|

-D ENABLE_NEON=ON|

-D WITH_LİBV4L=ON|

../

Enter tuşuna bastıktan sonra bir süre işlem gerçekleştirecektir, burada kamera ile ilgili not found gibi satırlar alabliriz, çünkü kamera için gerekli işlemler henüz yapılmadı. Bu işlemin sonucunda aşağıdaki satırları görmemiz gerekiyor. Aksi takdirde bir yerde hata yapıyor olabilirz, işlemleri gözden geçirelim.

……..

- **- Configuring done**
- **- Generating done**
- **-Build files have been written to:/home/pi/opencv/opencv-3.2.0/build**

Şimdi aynı dizin işerisinde şu komutu çalıştıralım ve derleme işlemini başlatalım. Bu işlem 1-2 saat arası sürebilir.

>>make –j4

Burada 4 çekirdekte derleme yapacağımızı söylüyoruz, bazen aşırı ısınmalardan dolayı kilitlenmeler olabilir böyle durumlarda 2 çekirdekte dersleme yapabiliriz. Uzun işlemler sonucunca [100%]Build target… ifadesini göreceğiz kurulumu tamamlamak için son adım şöyledir:

>>sudo make install

Sudo ldconfig

Şimdi kurulum tamamlandı, dosyalarımızın olduğu konum:

>>/usr/local/lib/libcv*

/user/local/lib/python3.4/dist-packages/Cv*

/usr/local/include/opencv2/

/usr/local/bin/opencv_*

/usr/local/share/OpenCV/

Şimdi yükleyip yüklemediğini kontrol edelim. Öncelikle python3 idlesini açıyoruz, python3 açtığımızdan emin olalım çünkü kurulumu onun için gerçekleştirdik.

>>import cv2

>>>print(cv2.__version__)

3.2.0

>>>

Şimdi kamera ile ilgili driver işlemlerini yapalım.

/home/pi, default dizinimize geri dönelim. Öncelikle kamera kullanacağımız için sudo raspi-config ardından interfaceden kameraya izin vermemiz gerekiyor. Ardından yükleme işlemini yapalım.

>>sudo apt-get-y install libv4l-dev v4l-utils

v4l'yi aktif edelim

>>v4l2-ctl-list-devices

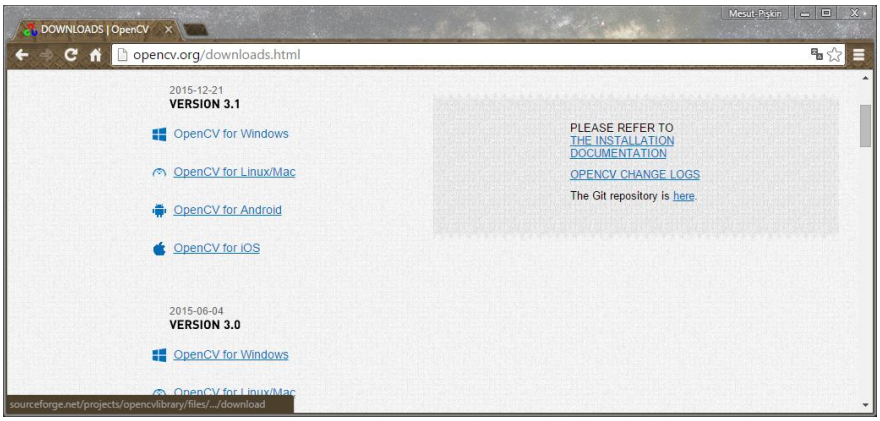
Komuttan sonra almamız gereken cevap şöyledir.

>>mmal service 16.1(platform:bcm2835-v4l2):

/dev/video0

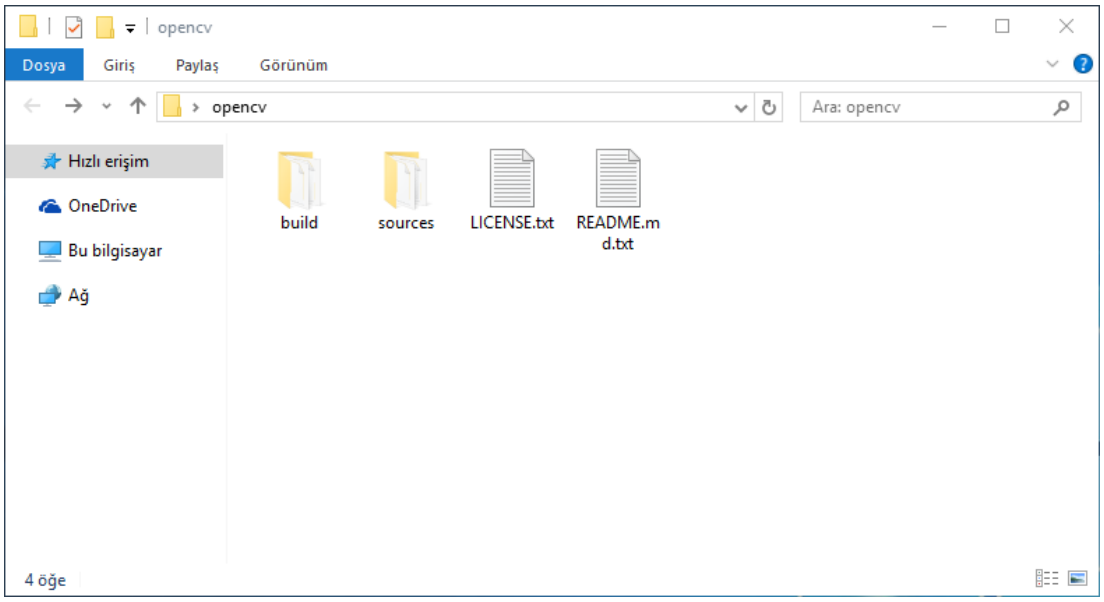
3.2.4 Windows İşletim Sistemi için OpenCV Kurulumu[5]

OpenCV'yi http://opencv.org/downloads.html adresine giriyoruz ve indirmek istediğimiz sürümün altındaki OpenCV for Windows linkine tıklıyoruz. Sürüm seçeceğimiz ekranın görüntüsü Şekil 3.2 de verilmiştir. İndirme bağlantısı sourceforge.net sitesine yönlendirecek ve indirme işlemi başlayacak.



Şekil 3.2: OpenCV İndirme Ekranı

İndirdiğimizde sıkıştırılmış olarak gelecektir, çalıştırdığımızda OpenCV dosyalarını çıkartmak için bir dizin isteyecektir, burada çıkartılmasını istediğiniz dosya dizini yolunu yazarak Extract butonuna tıklayalım. Dosyaları çıkarttığımız dizinde OpenCV klasörü içerisinde build ve sources diye 2 adet klasör bulunmaktadır. Bu klasörlerin ekran görüntüsü Şekil 3.3 de verilmiştir. Build klasörü içerisinde Windows platformu için derlenmiş olarak sistem native kütüphaneler ve programlama dilleri için kütüphaneler bulunmaktadır. Sources klasöründe ise OpenCV kaynak kodları ve örnek uygulamalar yer almaktadır. Buradaki kaynak kodları ve OpenCV'yi tekrar derleyebiliriz.



Şekil 3.3: Windows OpenCV Dosyaları

3.3 SolidWorks

SolidWorks yenilikçi, kullanımı kolay, Windows için hazırlanmış 3 boyutlu tasarım programıdır. SolidWorks her türlü makine, tesis, ürün tasarımında kullanıcıya Windows'un kolaylıklarını kullanarak hızlı bir şekilde çizim yapmasını sağlar.[6]

SolidWorks parasolid prensibinde çalıştığı için kullanıcıya, tasarımın her aşamasında müdahale şansı vererek, modelin boyutlarının, ölçülerinin ve ayrıntılarının istenilen şekilde değiştirilmesi imkanı vardır, saniyelerle ölçülebilecek zaman dilimlerinde teknik resim ve montajların yapılmasını sağlar. Feature tree (tasarım ağacı) ile yapılan işlemlerin sıraları ve yapıları değiştirilebilir. Üstelik yapılan değişiklikler sonucu varsa yapılmış olan montaj ve teknik resim anında güncelleşir. Böylece kullanıcıya teknik resimde veya montajda parçaya müdahale edebilme şansı doğar.

SolidWorks 2000'de ‘'smart mate" (akıllı montaj) adlı uygulama son derece hızlı montaj yapılmasını sağlar. Bunun dışında sac parçaların açılımını yaparak uzama miktarları hesaplanabilir. Ayrıca katı objeler kullanılarak kesme, yırtma veya şekillendirme işlemleri yapılabilmektedir.

Yapılmış olan tüm bu işlemleri IGES, DXF, DWG, SAT(ACIS), STL, STEP, VDA, VRML, parasolid dosyalarından kayıt veya transfer yapılabilir.

3.3.1 Neden Solid Works?

SolidWorks şu andaki CAD programları arasında kullanımı en kolay olan programdır. Ayrıca bünyesindeki ‘'Hole wizard ‘'(delik sihirbazı) ile metrik veya inch, havşalı veya faturalı tüm delikleri oluşturarak montaj ve tasarımda hızlılığı sağlar. Yine montajlara hareket verebilir, bu hareketlere çarpma kontrolü ‘'Collision Detection" yapalabilir, böylece tasarlanmış parçanın prototip maliyeti sıfıra indirilmiş olur. Tüm bu işlemler sonucu oluşturulan dosyaların büyüklükleri diğer programlara nazaran çok daha az yer kaplamakta ve açılım işlem zamanları çok kısa sürede olmaktadır. Montajların çok büyük olması bu özelliği etkilemez. Tasarımlarda foto gerçekçi görüntüler oluşturulabilir ve animasyonlar yapılabilir. Ayrıca SolidWorks bir çok çözüm olanağına sahiptir. Bir kaç örnek vermek gerekirse ; CAMWorks ile parça veya kalıbın takım yolları çıkarabilir, COSMOS/Works ile mukavemet analizleri yapılabilir, ToolBox ile standart makine ve kalıp elamanları kullanılabilir.

CAMWORKS : Parçaların otomatik işlem tanıtımıyla birkaç dakikada takım yolu çıkartabilir. Ayrıca takım yolları kontrol kütüphanesinde kontrolör tiplerine uygun formda çıkarılmaktadır.

Sonuç olarak SolidWorks 3 boyutlu tasarım için ideal çözümdür.

Kısaca SolidWorks aşağıdaki özellikleri bir arada bulundurmaktadır.

Katı modellerin avantajlarını kullanarak kolayca ve hızlı şekilde tasarım yapabilme imkanı sağlar.

• İstenildiği anda ölçü değiştirerek tasarımı kolayca düzeltebilme imkanı sağlar.

• Parça tasarımında diğer parçalara bağlı ölçü verebilme imkanı sağlar.

• Feature Tree (Tasarım Ağacı) sayesinde yapılan işlemlerin sırasını değiştirebilme imkanı sağlar.

• Nesne kütüphanesi ile sıkça kullanılan unsurları tekrar tekrar kullanabilme imkanı sağlar.

• Windows'ta kullanılan sürükle-bırak, kes-yapıştır uygulamaları SolidWorks'te de aynı işlevleri yürütür. Örneğin control tuşuna basarak bir unsuru bir yerden bir yere veya bir dosyadan başka bir dosyaya taşıyabilme.

• Bir parçadan değişik boyutlardaki konfigürasyonlarını Excel'de bir tablo oluşturarak otomatik olarak oluşturabilme imkanı sağlar.

• Sürükle-bırak tarzı ile tasarımın imalata hazır teknik resimlerini otomatik olarak oluşturabilme imkanı sağlar.

• Tasarımın herhangi bir anında yapılan değişikliklerde, teknik resimlerin otomatik olarak güncelleşmesi, istenirse teknik resimde yapılan değişikliklerde parçanın veya montajın güncelleşmesi imkanı vardır.
• Karmaşık sac parçaların açılımlarını elde eder. Saclar ister düz ister konik olsun SolidWorks'te açılımlarını elde edebilme, otomatik olarak teknik resmini çıkarabilme imkanı sağlar.
Montaj

• Binlerce parçadan oluşan montajlar yapabilme imkanı sağlar .
• Smart Mate"(Akıllı Montaj) adı verilen bir uygulama ile montaj parçalarını yerleştirirken (snap to fit) yakala-oturt özelliği ile daha hızlı bir şekilde montaj yapma imkanı sağlar.
• Montaj parçalarını, dinamik hareket mekanizmalarının çalışıp çalışmadığını kontrol eder.
• "Collision Detection"(Çarpma Kontrolü) ile montaja dinamik bir hareket verildiği zaman çakışan parçaların olup olmadığını görme imkanı sağlar.
• "Lightweight"(hafif yükleme) adı verilen bir sistem ile yüklü montaj dosyalarını daha hızlı bir şekilde açabilme imkanı sağlar.

4.MALZEMELERİN TANITIMI

4.1.Raspberry Pi

Raspberry Pi, İngilterde bulunan Raspberry Pi Vakfı tarafından desteklenen; öğrenci, amatör ve hobicilerin kullanımına sunulan kredi kartı büyüklüğünde, tek bir board'dan oluşan mini bilgisayardır. Raspberry Pi görüntüsü Şekil 4.1 de verilmiştir.[7]

Ürün 2009 yılından beri Raspberry Pi Foundation tarafından geliştirilmektedir. İlk satışı 29 Şubat 2012'de başlamıştır. A modeli 128 MB Ram olarak tanıtılmasına rağmen satış öncesi değişiklik ile 256 MB Ram ile piyasaya sürülmüştür. Raspberry Pi'nin A ve B olmak üzere iki modeli piyasaya sürülmüştür. B modeli 2 Adet USB, bir adet Ethernet girişine sahipken, A modelinde sadece 1 adet USB girişi bulunmaktadır. USB girişleri sayesinde her iki model de, standart tak-çalıştır USB mouse ve klavyeler ile sorunsuz çalışmaktadır.

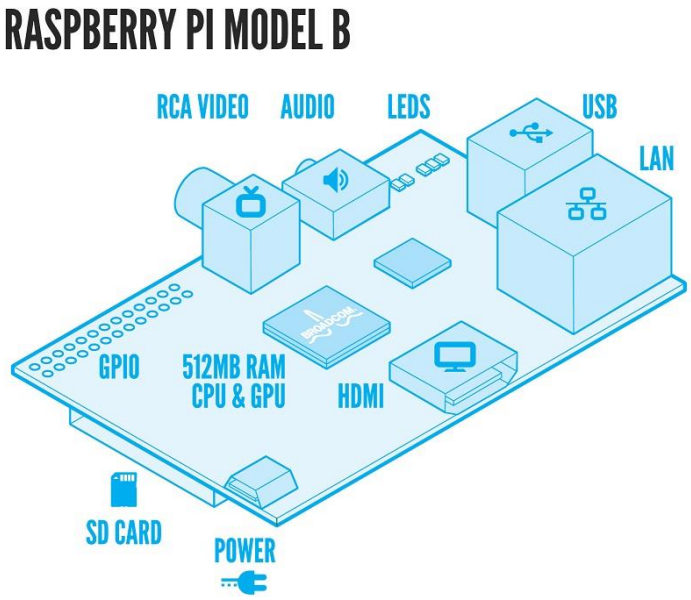
20 Nisan 2012 tarihinden itibaren Raspberry Pi'nin A ve B modelleri açık kaynak olarak kullanıma sunulmuştur. İsteyen herkes tüm devre çizimlerine ve teknik ayrıntılarına ulaşabilmektedir.

Şekil.4.1: Raspberry Pİ

Teknik Özellikleri:

- Broadcom BCM2835 (700MHz, ARM1176JZF-S tabanlı) İşlemci,
- Broadcom VideoCore IV (OpenGL ES 2.0, 1080p destekli) Grafik İşlemci,
- Model A'da 256MB ,Model B'de 512MB Ram,
- USB 2.0 (Model B'de 2 tane, Model A'da 1 tane bulunmaktadır),
- HDMI yuvası,
- SD Kart Okuyucu,
- 3.5mm ses jakı,
- RCA Video Çıkışı,
- CSI Bağlantısı
- 10/100 Ethernet (Model B'de bulunmaktadır),
- İşletim sistemi: Debian GNU/Linux, Fedora, Arch Linux ve türevleri,
- Düşük Seviye Çevre Birimleri: 8 adet GPIO, UART, I²C bus, SPI bus'la birlikte iki Chip Select, +3.3 V, +5 V, ground
- 45 gram ağırlığında,
- Model A 1.5W , Model B ise 3.5W güç tüketmektedir,
- Çalışma gerilimi: +5V DC

Raspberry Pı üzerine devre parçalarının yerleşimi Şekil 4.2 de belirtilmiştir.



Şekil 4.2: Raspberry Pi Model B

4.1.1 Raspberry Pi Modelleri

Raspberry Pi'nin çeşitli modelleri bulunmaktadır. Bu modeller temelde aynı olsalar da, yenilik, hız vb. açılardan farklılık gösterirler.

- **Model A:** Raspberry Pi'nin en temel sürümüdür. Üzerinde sadece 1 adet USB portu, 3.5 mm stereo ses çıkışı, kompozit video ve HDMI portu bulunur. Ethernet girişi bulunmaz. ARM v6 mimarisine sahip tek çekirdek işlemciye sahiptir. 256MB RAM belleği vardır. 26-pinli GPIO konektörü bulunur. Az güç tüketmesi sayesinde gömülü sistem projelerinde kullanılması için tasarlanmıştır.

- **Model A+:** Model A'nın güncellenmiş sürümüdür. Bu sürümde 26-pinli GPIO konektörü 40-pine çıkartılmış, kompozit video çıkışı kaldırılmış ve normal SD kart yerine mikro SD kart slotu kullanılmıştır. Kart boyutları ufaltılarak yer sıkıntısı çekilebilecek projelerde kullanılması hedeflenmiştir. En küçük boyutlu Raspberry Pi'dir.

- **Model B:** Raspberry Pi'nin en çok bilinen ve kullanılan modelidir. 2 adet USB portu, Ethernet girişi, 3.5 mm stereo HDMI ve kompozit video çıkışları bulunur. ARMv6 mimarisine sahip tek çekirdek işlemciye sahiptir. 512MB RAM belleği vardır. 26-pinli GPIO konektörü bulunur. En popüler Raspberry Pi modelidir.

- **Model B+:** Raspberry Pi Model B'nin geliştirilmiş ve kart tasarımı değiştirilmiş sürümüdür. Model B'den farklı olarak 4 adet USB portu, normal SD kart yerine mikro SD kart slotu ve 26-pin yerine 40-pinli GPIO konektörü bulunur. HDMI ve Ethernet bağlantıları yine bu kartta yer almaktadır. Kompozit video çıkışı bu kartta ayrı bir konektör olarak yer almamaktadır, 3.5 mm ses çıkış portundan 3'lü RCA tipi kabloyla bağlanır.

- **Raspberry Pi 2:** Model B+ ile aynı kart dizilimine sahip olmasına karşın, bu kartta ARMv7 mimarisine sahip 4 çekirdekli işlemci ve 1GB RAM bellek bulunmaktadır.
- **Raspberry Pi 3**: Raspberry Pi 2'nin devamı olan bu modelin en büyük farkı dahili Wi-Fi ve Bluetooth bağlantıya sahip olmasıdır. Ayrıca ARMv8 64-bit mimarisine sahip 4 çekirdekli işlemcisi, 1.2GHz frekansında çalışmaktadır ve 1GB RAM belleğe sahiptir.

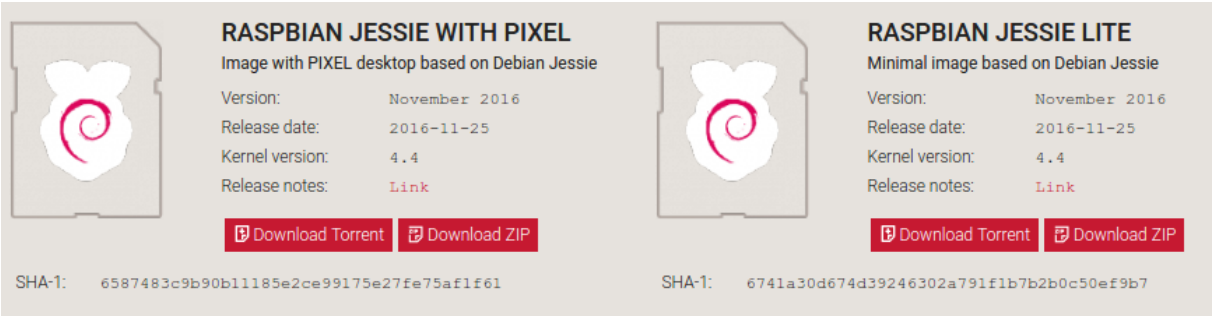
- **Raspberry Pi Zero**: En küçük boyutlu Raspberry Pi modelidir. Donanımı (işlemci ve belleği) Raspberry Pi Model B ile tamamen aynıdır. Boyutundan dolayı tam boy USB yerine mikro USB-OTG portuna sahiptir.

4.1.2 Raspbian Kurulumu

Raspberry Pi, GNU/Linux işletim sistemleriyle çalışması için tasarlanmıştır. Kullanabilmeniz için, en az 4 GB kapasiteli bir SD karta ihtiyacınız olacaktır. Raspberry Pi için özel olarak hazırlanmış bir Debian sürümü olan Raspbian, en popüler işletim sistemi tercihidir.

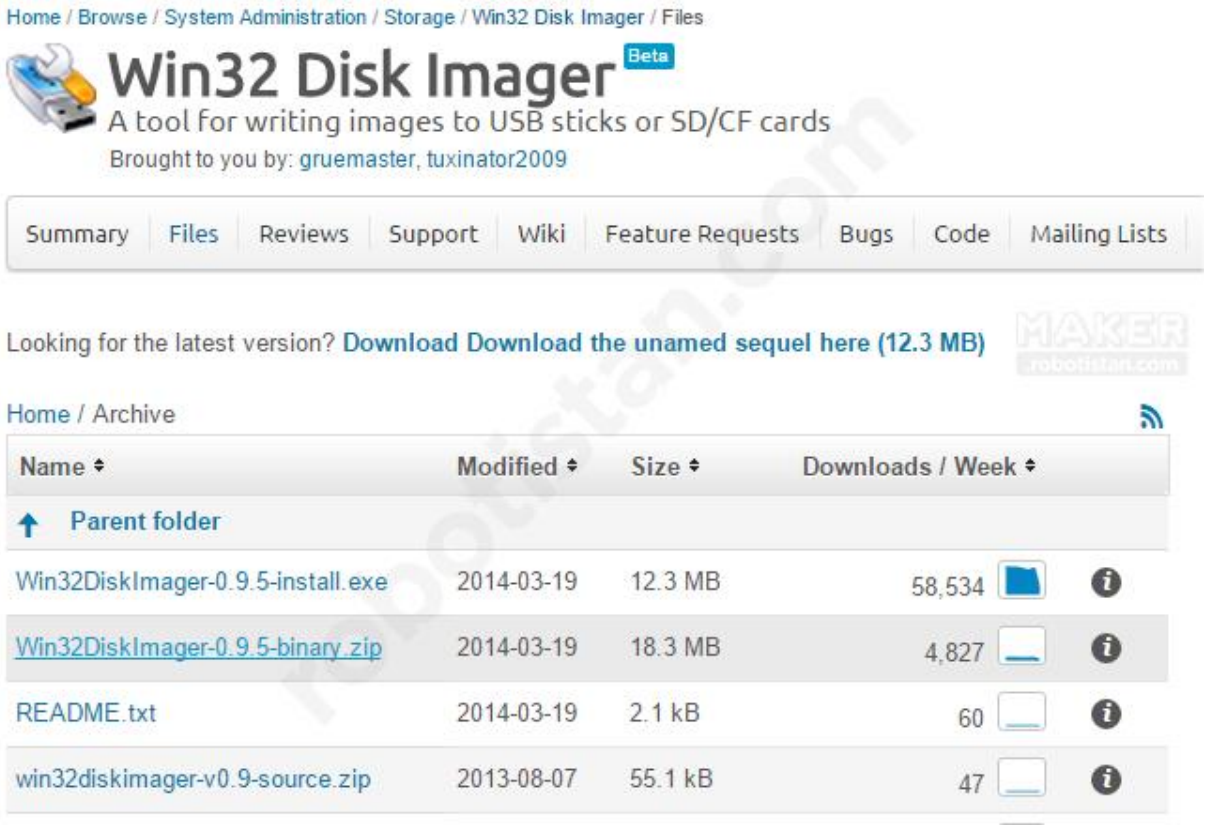
Raspbian kurulumu için https://www.raspberrypi.org/downloads/raspbian/ adresini ziyaret ediyoruz. Raspbian işletim sisteminin PIXEL masaüstü ortamına sahip tam sürümü ve masaüstü ortamı barındırmayan Lite sürümleri mevcuttur. Biz PIXEL masaüstü ortamına sahip sürümü kullanacağız. İşletim sistemini doğrudan .zip dosyası olarak indirmek için "Download ZIP" seçeneğini veya torrent istemcisi kullanarak indirmek için "Download

Torrent" seçeneğini tıklayabiliriz. Şekil 4.3 de indirme ekranında karşımıza çıkacak görüntü verilmiştir.



Şekil.4.3: Raspbian İndirme Ekranı

Raspbian'ı indirdikten sonra SD karta yazmamız için gerekli olan Win32 DiskImager-0.9.5-binary.zip dosyasını seçerek indiriyoruz. Buna ilişkin ekran görüntüsü Şekil 4.4 de verilmiştir.



Şekil. 4.4: Win32 Disk Imager

Her iki dosya da indikten sonra ZIP dosyalarını açıyoruz. SD kartımızı bilgisayarımıza takıyor ve Win32DiskImager programını çift tıklayarak açıyoruz. Daha sonra ZIP'ten çıkarttığımız Raspbian imaj dosyasını seçiyoruz. "Write butonuna tıklamadan önce "Device"

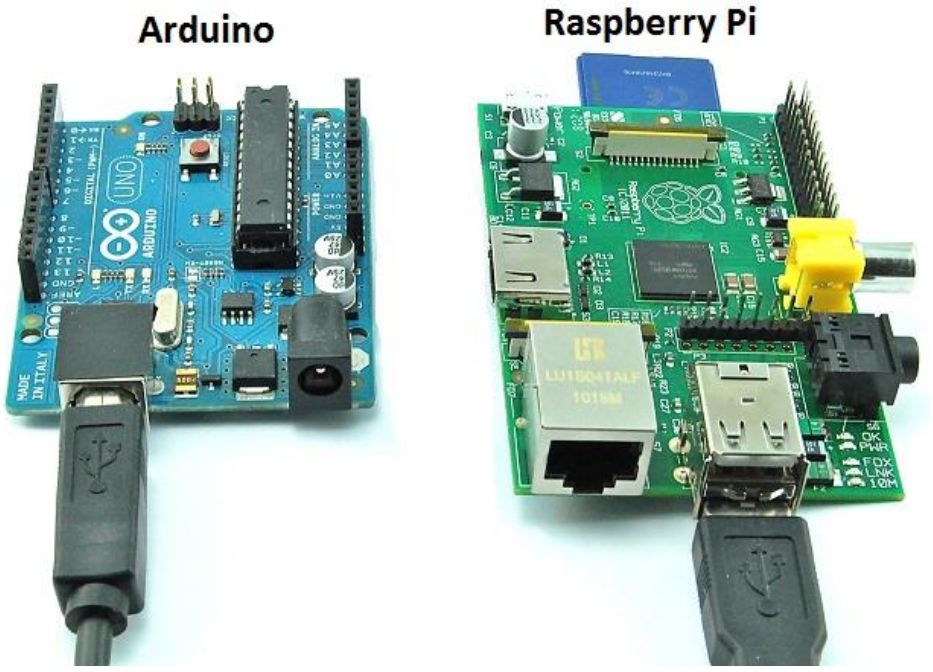
kısmından bilgisayarımızdaki SD kart okuyucunun seçili olduğundan emin olun.” Seçtiğimiz dosyadan ve yazmak istediğimiz sürücüden emin olduktan sonra “Write” butonun tıklayarak işlemi başlatıyoruz.

Yazma işlemi, kartımızın hızına ve imaj dosyasının büyüklüğüne göre biraz uzun sürebilir. İşlemin sonunda program “Write Succesful” şeklinde bize işlemin tamamlandığına dair bir mesaj verecek. Bu aşamada SD kartımızı bilgisayarımızdan çıkartıp Raspberry Pi’mize takabiliriz.

4.1.3 Arduino İle Raspberry Pi'nin Farkı

- Arduino, 8-bit'lik Atmega Mikrokontrolcü'ye sahipken, Raspberry Pi'de 32-bit ARM tabanlı Mikroişlemci bulunmaktadır,
- Arduino 8-16 Mhz saat hızına sahipken, Raspberry Pi de bu 1 Ghz'e kadar çıkmaktadır,
- Arduino 2-8 KB , Raspberry Pi ise 512 MB Ram kapasitesine sahiptir,
- Raspberry Pi de GPU (Grafik işlemci), Ses, USB ve Ethernet çıkışı bulunurken Arduino'da bunlar bulunmamaktadır,
- Arduino, Raspberry Pi'ye göre çok daha kolay programlama imkanına sahiptir. Raspberry Pi'yi iyi bir şekilde kullanabilmek için Linux komutlarına hakim olmak gerekmektedir.

Arduino ve Raspberry pi ye ait görüntüler Şekil 4.5 te verilmiştir.



Şekil. 4.5 Arduino ve Raspberry Pİ Karşılaştırması

Tüm bu bilgiler ışığında, Raspberry Pi'nin daha çok gömülü sistemler ve işletim sistemi uygulamalarında, Arduino'nun ise hobi devreleri ve basit ve orta düzeyde elektronik devre oluşturmada uygun olduğu görülmektedir.

4.2 Arduino

Arduino bir G/Ç kartı ve Processing/Wiring dilinin bir uygulamasını içeren geliştrime ortamında oluşan bir fiziksel programlama platformudur.

Arduino kartlarının donanımında bir adet Atmel AVR mikrodenetleyici (Atmega328, Atmega2560, Atmega32u4 gibi) ve programlama ve diğer devrelere bağlantı için gerekli yan elemanlar bulunur. Her Arduno kartında en azından bir 5 voltluk regüle entegresi ve bir 16MHz kristal osilator (bazılarında seramik rezonatör) vardır Arduino kartlarında programlama için harici bir programlayıcıya ihtiyaç duyulmaz, çünkü karttaki mikrodenetleyiciye önceden bir bootloader programı yazılıdır.

4.2.1 Arduino Bileşenleri

Arduino geliştirme ortamı(IDE), Arduino bootloader (Optiboot), Arduino kütüphaneleri, AVRDude(Arduino üzerinde mikrodenetleyici programlayan yazılım) ve derleyiciden (AVR-GCC) oluşur.

- Arduino yazılımı bir geliştirme ortamı (IDE) ve kütüphanelerden oluşur. IDE, Java dilinde yazılmıştır ve Processing adlı dilin ortamına dayanmaktadır. Kütüphaneler ise C ve C++ dillerinde yazılmıştır ve AVR-GVV ve AVR Libc. ile derlenmiştir.
- Optiboot bileşeni Arduino'nun bootloader bileşenidir. Bu bileşen, Arduino kartlarının üzerindeki mikrodenetleyicinin programlanmasını sağlayan bileşendir.

Arduino 'nun bu kadar çok tercih edilmesini sağlayan en önemli bileşen ise mikrodenetleyici konusunda detaylı bilgi sahibi olmayı gerektirmeden herkesin programlama yapabilmesini sağlayan Arduino kütüphaneleridir. Arduino kütüphaneleri, geliştirme ortamı ile birlikte gelmekte ve "libraries" klasörünün altında bulunmaktadır. Kodları inceleyerek mikrodenetleyicilerin nasıl programlandığını ve kütüphanelerin yapısını görmeniz mümkündür.

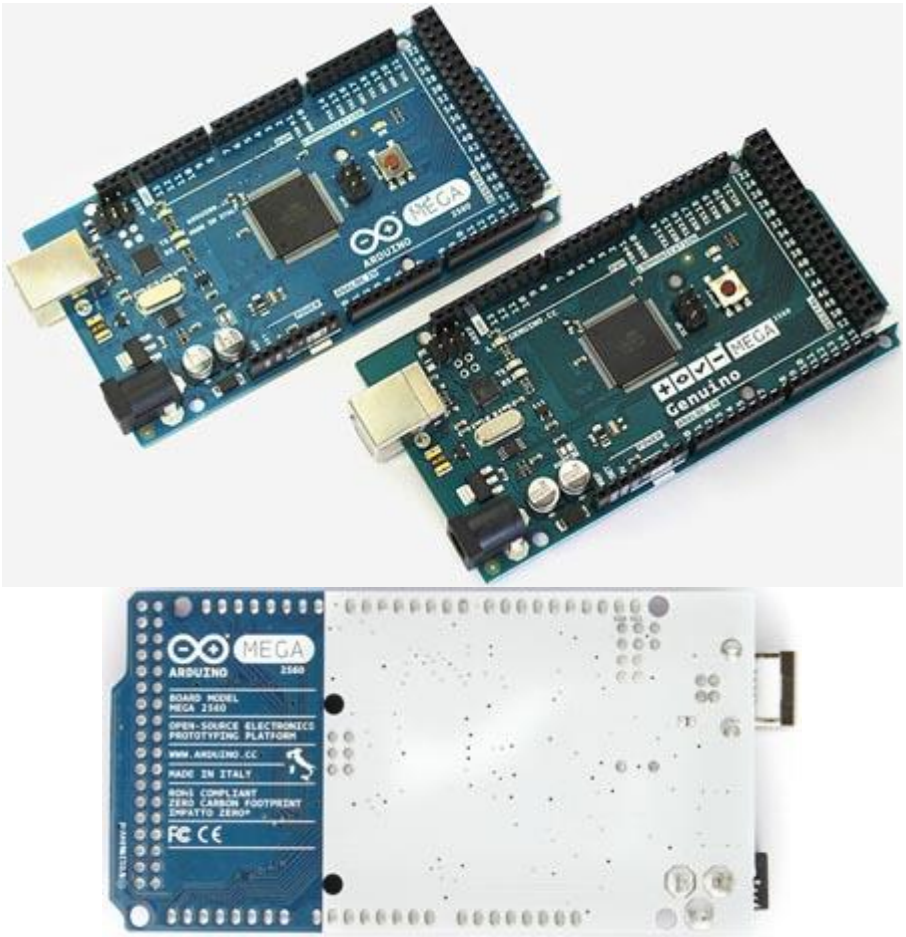
Son olarak AVRDude bileşeni iste derlenen kodları programlamak için kullanılır.

- **Arduino ile Neler Yapılabilir?**

Analog ve digital sinyalleri alarak işleyebiliriz. Sensörlerden gelen sinyalleri kullanarak, çevresiyle etkileşim içerisinde olan robotlar ve sistemler tasarlayabiliriz.

4.2.2 ARDUİNO / GENUINO MEGA 2560

Arduino Mega 2560 ATmega2560 mikrodenetleyici içeren bir Arduino kartıdır. Arduino Uno'dan sonra en çok tercih edilen Arduino kartı olduğu söylenebilir. Arduino 'nun kardeş markası olan Genuino markasını taşıyan Genuino Mega 2560 kartı ile tamamen aynı özelliklere sahiptir. Arduino Meha 2560 ‘ın önden ve arkadan görünüşü Şekil 4.6 da verilmiştir.



Şekil 4.6: Arduino Mega Görünüş

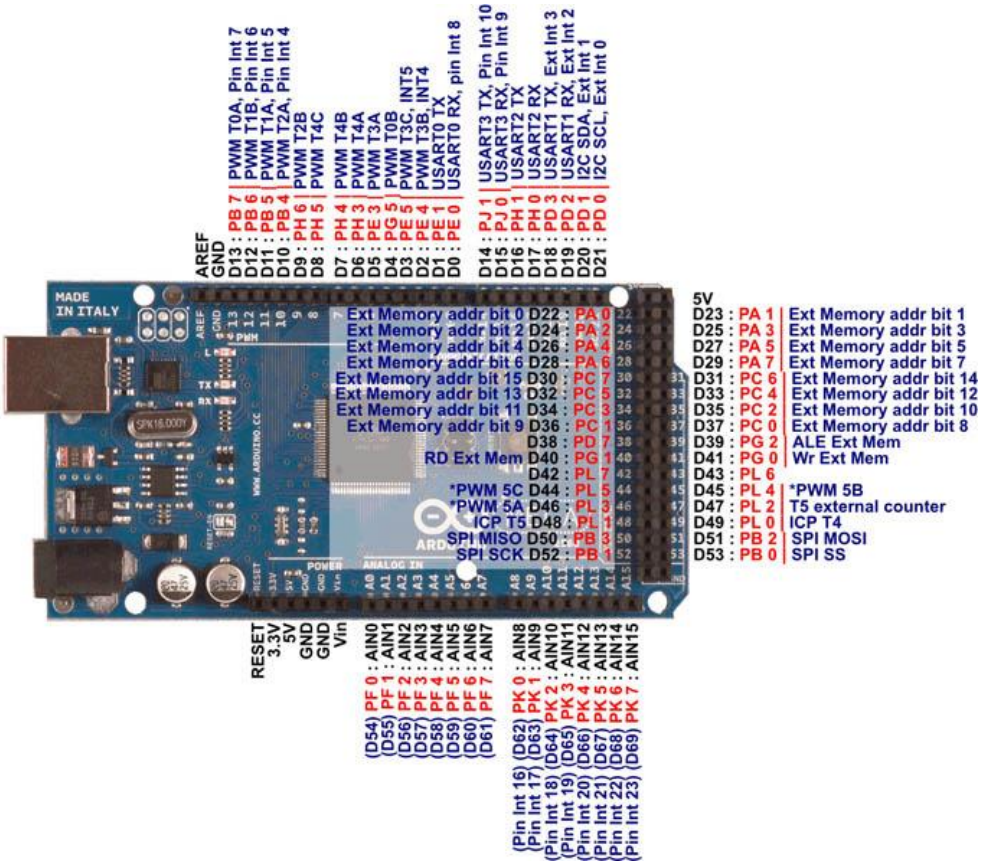
Arduino Mega 2560 'ta 54 tane dijital giriş / çıkış pini vardır. Bunlardan 15 tanesi PWM çıkışı olarak kullanılabilir. Ayrıca 16 adet analog girişi, 4 UART (donanım seri port), bir adet 16 MHz kristal osilatörü, USB bağlantısı, power jakı (2.1mm), ICSP başlığı ve reset butonu bulunmaktadır. Arduino Mega 2560 bir mikrodenetleyiciyi desteklemek için gerekli

bileşenlerin hepsini içerir. Arduino Mega 2560 bir bilgisayara bağlanarak, bir adaptör ile ya da pil ile çalıştırılabilir. Arduino Mega, Arduino Duemilanove ya da Diecimila için tasarlanan shield lerin çoğu ile kullanılabilir.

Arduino Mega 2560 R2 (revision 2) 8U2 HWB çizgisini toprağa çeken bir dirence sahiptir. Böylece DFU mode kullanmak kolaylaşır. Arduino Mega R3 (revision 3) ise şu ek özelliklere sahiptir;

- 1.0 pinout: AREF pininin yanına SDA ve SCL pinleri eklenmiştir. Reset pininin yanına iki yeni pin eklenmiştir. IOREF shield lerin karttan sağlanan voltaja adapte olmasını sağlar. İleride shield ler hem 5 V ile çalışan AVR kullanan kartlar ile hem de 3.3 V ile çalışan Arduino Due ile uyumlu olacaktır. İkinci pin ise herhangi bir yere bağlı değildir. İleride yapılacak geliştirmeler için eklenmiştir.
- Daha güçlü reset devresi
- Arduino Mega 2560 R1 ve R2 de kullanılan ATmega8U2 yerine R3 te ATmega16U2 kullanılmıştır.

Şekil 4.7 de Arduino Mega 2560 pinleri gösterilmektedir.



Şekil 4.7: Arduino Mega Pinler

4.2.2.1 Arduino Mega 2560 Teknik Özellikleri

- Mikrodenetleyici: ATmega2560
- Çalışma gerilimi: +5 V DC
- Tavsiye edilen besleme gerilimi: 7 - 12 V DC
- Besleme gerilimi limitleri: 6 - 20 V
- Dijital giriş / çıkış pinleri: 54 tane (15 tanesi PWM çıkışını destekler)
- Analog giriş pinleri: 16 tane
- Giriş / çıkış pini başına düşen DC akım: 40 mA
- 3,3 V pini için akım: 50 mA
- Flash hafıza: 256 KB (8 KB bootloader için kullanılır)
- SRAM: 8 KB
- EEPROM: 4 KB
- Saat frekansı: 16 MHz

4.2.2.2 Güç

Arduino Mega 2560 bir USB kablosu ile bilgisayar bağlanarak çalıştırılabilir ya da harici bir güç kaynağından beslenebilir. Harici güç kaynağı bir AC-DC adaptör ya da bir pil / batarya olabilir. Adaptörün 2.1 mm jaklı ucunun merkezi pozitif olmalıdır ve Arduino Mega 2560 'ın power girişine takılmalıdır. Pil veya bataryanın uçları ise power konnektörünün GND ve Vin pinlerine bağlanmalıdır.

Arduino Mega 2560 6 V-20 V aralığında bir harici güç kaynağı ile beslenebilir. Ancak 7 V altında bir besleme yapıldığında 5V pini 5 V tan daha düşük çıkış verebilir ve kart kararsız çalışabilir. 12 V üzerinde bir voltaj beslemesi yapılması durumunda ise regülatör fazla ısınabilir ve karta zarar verebilir. Bu nedenle tavsiye edilen besleme gerilimi 7 V-12 V aralığındadır.

- **VIN**: Arduino Mega 2560 kartına harici bir güç kaynağı bağlandığında kullanılan voltaj girişidir.
- **5V:** Bu pin Arduino kartındaki regülatörden 5 V çıkış sağlar. Kart DC power jakından (2 numaralı kısım) 7-12 V adaptör ile, USB jakından (1 numaralı kısım) 5 V ile ya da VIN pininden 7-12 V ile beslenebilir. 5V ve 3.3V pininden voltaj beslemesi regülatörü bertaraf eder ve karta zarar verir.
- **3.3V:** Arduino kart üzerindeki regülatörden sağlanan 3,3V çıkışıdır. Maksimum 50 mA dir.
- **GND:** Toprak pinidir.

- **IOREF:** Arduino kartlar üzerindeki bu pin, mikrodenetleyicinin çalıştığı voltaj referansını sağlar. Uygun yapılandırılmış bir shield IOREF pin voltajını okuyabilir ve uygun güç kaynaklarını seçebilir ya da 3.3 V ve 5 V ile çalışmak için çıkışlarında gerilim dönüştürücülerini etkinleştirebilir.

4.2.2.3 Giriş ve Çıkışlar

Arduino Mega 2560 'ta bulunan 54 tane dijital giriş/çıkış pininin tamamı, pinMode(), digitalWrite() ve digitalRead() fonksiyonları ile giriş ya da çıkış olarak kullanılabilir. Bu pinler 5 V ile çalışır. Her pin maksimum 40 mA çekebilir ya da sağlayabilir ve 20-50 KOhm dahili pull - up dirençleri vardır. Ayrıca bazı pinlerin özel fonksiyonları vardır:

- **Serial: 0 (RX) ve 1 (TX); Serial 1: 19 (RX) ve 18 (TX); Serial 2: 17 (RX) ve 16 (TX); Serial 3: 15 (RX) ve 14 (TX) :** Bu pinler TTL seri data almak (receive - RX) ve yaymak (transmit - TX) içindir. 0 ve 1 pinleri ayrıca ATmega16U2 USB-to-TTL Si çipinin ilgili pinlerine bağlıdır.
- **Harici kesmeler 2 (kesme 0), 3 (kesme 1), 18 (kesme 5), 19 (kesme 4), 20 (kesme 3) ve 21 (kesme 2) :** Bu pinler bir kesmeyi tetiklemek için kullanılabilir.
- **PWM: 2 - 13 , 44 - 46 :** Bu pinler analogWrite () fonksiyonu ile 8-bit PWM sinyali sağlar.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS) :** Bu pinler SPI kütüphanesi ile SPI haberleşmeyi sağlar.
- **LED 13:** Digital pin 13 e bağlı bir leddir. Pinin değeri High olduğunda yanar, Low olduğunda söner.
- **TWI: 20 SDA 21 SCL:** Wire kütüphanesini kullanarak TWI haberleşmesini destekler. (Bu pinlerin yeri Arduino Duemilanove ve Diecimila kartlardaki ile aynı değildir.)

Arduino Mega 2560 'ın 16 tane analog girişinden her biri 10 bit çözünürlüğü destekler. Varsayılan ayarlarda topraktan 5 V a kadar ölçerler. Ancak, AREF pini ve analogReference() fonksiyonu kullanılarak üst limit ayarlanabilir.

- **AREF:** Analog girişler için referans voltajıdır. analogReference() fonksiyonu ile kullanılır.
- **RESET:** Mikrodenetleyiciyi resetlemek içindir. Genellikle shield üzerine reset butonu eklemek için kullanılır.

4.2.2.4 Haberleşme

Arduino Mega 2560 bir bilgisayar ile, başka bir Arduino ile ya da diğer mikrodenetleyiciler ile haberleşme için çeşitli imkanlar sunar. ATmega2560 mikrodenetleyicisi TTL (5V) seri haberleşme için 4 donanımsal UART'a sahiptir. Kart üzerindeki bir ATmega16U2 seri haberleşmeyi USB üzerinden kanalize eder ve bilgisayardaki yazılıma sanal bir com portu olarak görünür. 16U2 standart USB com sürücülerini kullanır ve harici sürücü gerektirmez. Ancak, Windows 'ta bir .inf dosyası gereklidir. Kart üzerindeki RX ve TX ledleri USB den seri çipe ve USB den bilgisayara veri giderken yanıp söner.

SoftwareSerial kütüphanesi Arduino Mega 2560 'ın dijital pinlerinden herhangi biri üzerinden seri haberleşmeye imkan sağlar. Ayrıca ATmega2560 TWI ve SPI haberleşmelerini de destekler.

4.2.2.5 Programlama

Arduino Mega 2560 'ı programlamak için Arduino programını buradan indirmeniz gerekir. Programı indirip açtıktan sonra Tools > Board menüsünden Arduino Mega 2560 'ı seçiniz.

Arduino Mega 2560 üzerindeki ATmega2560 mikrodenetleyicisine önceden bir bootloader yüklenmiştir. Bu bootloader sayesinde Arduino 'yu programlamanız için harici bir programlayıcı donanımına ihtiyacınız olmaz. Orjinal STK500 programını kullanarak haberleşir.

Ayrıca Arduino ISP kullanarak Arduino 'nun bootloader 'ını devre dışı bırakabilir ve mikrodenetleyiciyi ICSP (In Circuit Serial Programming) pini üzerinden programlayabilirsiniz.

4.2.2.6 USB Aşırı Akım Koruması

Arduino Mega 2560, bilgisayarınızın USB portunu aşırı akım ve kısa devreden koruyan resetlenebilir bir çoklu sigortası bulunur. Çoğu bilgilsayarın portlar için kendi korumaları olmasına rağmen bu sigorta ekstra bir koruma katmanı sağlar. Eğer USB portuna 500 mA den fazla bir yük binerse, sigorta otomatik olarak bağlantıyı kısa devre veya aşırı akım durumu ortadan kalkana dek keser.

4.3 MOTORLAR

Motorlar Hakkında Bazı Terimler

Tork: Motorun dönme momentidir. Aynı devirde dönen iki motorda torku daha büyük olan daha çok ağırlık taşır ve daha güçlüdür. Özellikle sumo robot ve mini sumo robot projelerinde torku yüksek motorların tercih edilmesinin sebebi yüksek itiş gücü elde edebilmektir.

Devir: Motorun bir tam turudur.

Rpm: Motorun bir dakikada tamamladığı devir sayısıdır.

Fırçalı(Brush)motor: Motorda bulunan elektromıknatısların kutuplarını değiştirmek için fırça kullanılan motorlardır.

Fırçasız(brushless)motor: Motorda kullanılan elektromıknatısların kutuplarını değiştirmek için fırça yerine elektronik malzeme kullanan motorlardır.

Redüktör: Motorlara bağlı dişli sistemidir.

4.3.1 DC Motorlar

DC motorlar robotikte en çok kullanılan motorlardır. DC motorlarda step motorlardaki bobinler yerine mıknatıslar yer alır. DC motorlar piyasa da farklı çalışma voltajına ve rpm değerlerine sahip redüktörlü ya da redüktörsüz pek çok çeşitte bulunabilmektedir. DC motorların hız kontrolleri pwm ile yapılabilir. Robot motoru olarak kullanmak için ideal motorlardır. DC motorların robotlarda kullanımına dair temel özellikler aşağıda açıklanmıştır.

- **Yön:** DC motorlara bir güç kaynağı bağlandığında DC motorun dönüş yönü akımın yönüne bağlıdır. Akımın yönü terslendiğinde DC motorun dönüş yönü de terslenmiş olur.
- **Hız:** Bir motorun hızı rpm (rotations per minute - bir dakikada tamamlanan devir sayısı) ile ölçülür. Motorun hızı voltaja ve yüke bağlıdır.

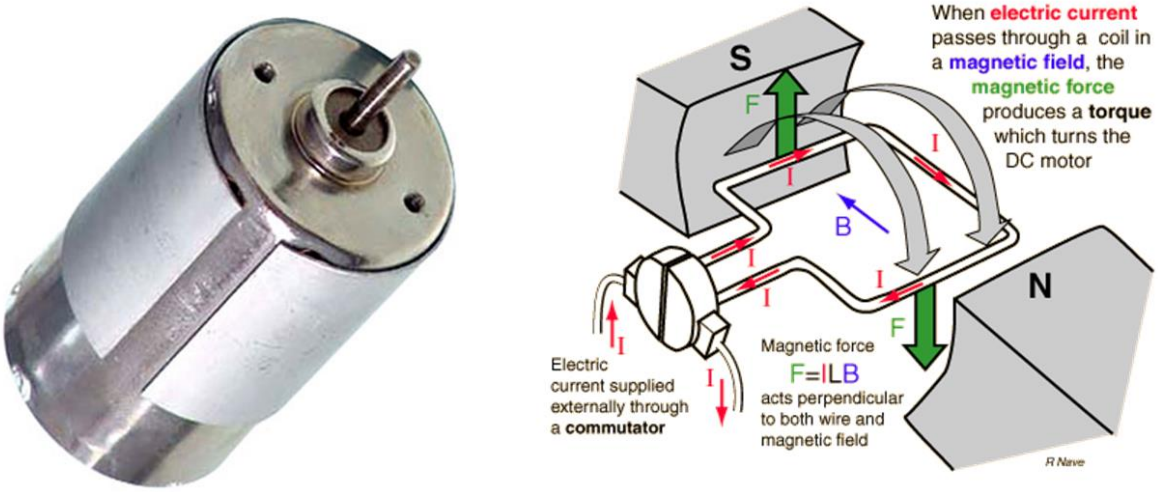
Bir DC motorun hızının voltaja ve yüke göre değişimini değerlendirmek için iki durum düşünülebilir. Bunlardan ilki; DC motora yük binmeyen ya da sabit bir yükün olduğu bir sistemdir. Böyle bir sistemde DC motorun hızı uygulanan voltaja bağlıdır ve voltaj arttıkça hız da artar. İkinci durum ise; DC motora binen yükün zamana ya da gerçekleştirilen göreve göre değiştiği bir sistemdir. Bu durumda DC motorun hızı yüke bağlı olacaktır. Yük arttıkça uygulanan güç de artar ve güç arttıkça hız azalır.

- **Voltaj:** Küçük DC motorlar 1,5 V ile 48 V arasında değişen voltaj değerlerine sahip olarak bulunabilirler. Her bir DC motor için belirtilen voltaj değeri, o DC motorun kendi verilen hız, güç ve akım değerlerinde stabil çalıştığı voltaj değeridir. Robotlarda ve diğer sistemlerde DC motorları kullanırken de bu voltaj değeri, DC motora verilecek maksimum çalışma voltajını belirlediği için önemlidir.
- **Akım:** Bir DC motor belirtilen voltaj değerinde çalıştırıldığında DC motorun çekeceği akım yüke bağlıdır. Yük arttıkça DC motorun çektiği akım da artar. DC motor, maksimum akım sınırının aşılacağı fazla bir yükle çalıştırılmamalıdır. Böyle bir durumda DC motor kısa devreye neden olur ve uygulanan güç ısıya dönüşür. Bu durum uzun sürerse DC motor yanabilir. Genellikle DC motorların uygulama akımı aralığı 50 mA den başlayıp 2A üzerine kadar çıkabilir.
- **Güç:** Güç bir motorun akımı ve voltajının çarpım değeridir. Ancak robot projelerinde ve mekanik sistemlerde bir motorun ürettiği kuvvetin tork (motorun dönme momenti) cinsinden değerlendirilmesi normaldir.
- Tork motorun dönme momentidir. Torku yüksek olan motor düşük olana göre daha güçlüdür. Tork motorun elektriksel ve mekanik karakteristiklerine ve motor şaftının yarı çapına bağlıdır. Bir motorun torku motora bağlanan dişli kutularıyla (redüktör) değiştirilebilir. Dişli kutuları hızın azaltılmasını ve gücün arttırılmasını sağlar. Örneğin; motor şaftının yarıçapının 10 katı yarıçapa sahip bir dişli motora eklendiğinde, motorun hızı 10 kat düşer ve gücü de 10 kat artar.

Robotikte, çeşitli boyutlarda ve redüksiyon oranlarında dişli kutuları motorun karakteristik özelliklerini isenilen işi yapabilecek düzeye getirmek için sıklıkla kullanılır. Bir motoru kullanırken torkunu bilmek önemlidir. Tork ve redüksiyon oranı bilindiğinde sistemin son çıkış gücü kolaylıkla belirlenebilir.

4.3.1.1 DC MOTOR ÇALIŞMA PRENSİBİ

DC motor, “Akım taşıyan bir iletken, bir manyetik alan içerisine girdiğinde kuvvete maruz kalır” prensibi ile çalışan bir elektrik makinesidir [9]. Şekil 4.8 de DC motorun çalışmasına dair görsel verilmiştir.



Şekil 4.8: DC motor örneği ve çalışma prensibi

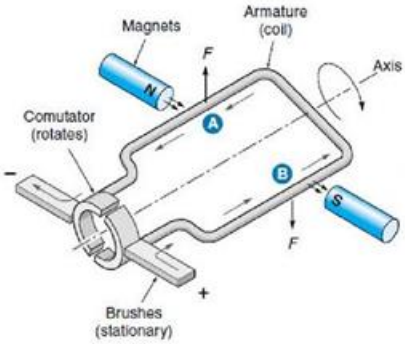
Uygulanan kuvvetin formülü: F=BIL

Denklemde F kuvveti, B manyetik alanı, I akımı ve L iletkenin uzunluğunu simgelemektedir. Kuvvet vektörel olduğu için aynı zamanda bir yön de belirtilmesi gerekmektedir. Kuvvetin uygulandığı yönü bulmak için Fleming'in sağ el kuralını kullanabiliriz.

DC Motorların ana elemanları, bobinler, mıknatıslar, rotorlar, fırçalar, stator ve doğru akım kaynağıdır. Armatür, mıknatıslar ya da bobinler tarafından üretilen manyetik alana yerleştirilmesi ve doğru akım kullanılarak döndürülmesi yukarıdaki örnekte bahsedildiği gibi bir mekanik kuvvetin oluşmasını sağlar. Bu kuvveti kullanarak istediğimiz işin yapılmasını sağlayabiliriz.

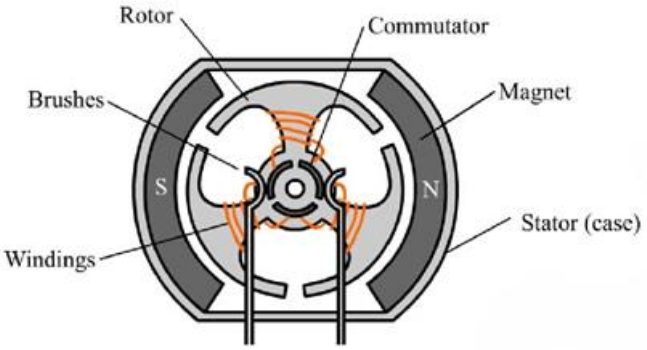
Armatür manyetik alan içerisinde döndüğünde ve manyetik kuvvet çizgilerini kestiğinde, sistem içerisinde bir emf (electromotive force/elektromotif kuvvet) indüklenir. Bu kuvvete ters emf (back emf) denir çünkü yönü armatür akımı ile ters yönlüdür. Bu emf nedeniyle sistemde eneri kaybı yaşanır. Ancak ters emf bir motorun yük altında düzgün çalışması için gereklidir. Ters emf olmadan motoru yükte çalıştırmak ve istediğimiz işleri yaptırmak bizim için oldukça zor olurdu. Şekil 4.9 da anlatıldığı üzere armatür manyetik alan içerisinde döndüğünde ve manyetik kuvvet çizgilerini kestiğinde, sistem içerisinde bir

emf(elektromotif kuvvet) oluşur. Bu kuvvete ters emf (back emf) denir çünkü yönü armatür akımı ile ters yönlüdür. Bu emf nedeniyle sistemde enerji kaybı yaşanır.



Şekil 4.9: Elektromotif Kuvvet Oluşumu

Şekil 4.10 da görüldüğü gibi DC motorun ana elemanları, bobinler, rotorlar, fırçalar, stator ve doğru akım kaynağıdır.



Şekil 4.10: DC Motor Elemanları

4.3.2 Redüktörlü Motorlar

Elektrik motorlarının yüksek dönüş hızları, makineler için gerekli olan dönüş hızlarına dönüştürmek için tasarlanan kapalı dişli sistemlerine redüktör denir. Redüktörler, gövde içine yerleştirilen miller, dişli çarklar, yataklar vs gibi elemanlardan oluşan sistemlerdir. Redüktörler güç iletmek, dönme yönleri elde etmek, az bir güçle büyük moment elde etmek, döndürülen iki eleman arasında bağımsız hareket sağlamaktır.

Redüktör hesabı, kullanıldıkları alana göre, küçük ve büyük boyutta yapılır. Redüktör tasarımı, her yandan bağlanabilen, değişebilir gövde sistemi ile montaj kolaylığı ve esnekliği sağlanmıştır. Flanş ve çıkış mil bağlantıları çok alternatifli bağlantılar için uygundur.

Redüktör Çeşitleri

- Delik milli redüktör
- Yatık tip redüktör
- E tipi redüktör
- K tipi redüktör
- M tipi redüktör
- N tipi redüktör
- D tipi redüktör

Redüktör seçimi, çıkış devri ve güç en önemli parametrelerdir. Bu iki parametre belirtildikten sonra motorlu motorsuz, flanşlı ayaklıi delik milli, yatay, dikey ayrıntıları belirleyerek redüktör belirlenebilir. Redüktör bağlantı şekilleri, bunu tahrik eden mekanizme arasında değişik bağlantı mevcuttur. Motor, fren, redüktör, ara bağlantı adaptörü, kaplin bağlantısı önemli bir değer ifade etmektedir. Redüktör bağlantı şekillerini şu başlıklar altında sınıflandırabiliriz:

1) Çıkış mişine göre

- Pararel miline göre
- Düz hatlı
- 90 yön değişimi

2) Mil bağlantılarına göre

- Düz mil bağlantılı
- Delik mil bağlantılı
- Delik milli ve sıkma bilezikli
- Spline mil bağlantılı

3) Dişli çeşidine göre

- Sonsuz dişli
- Planet dişli

- Helisel dişli
- Hipoid dişli

4) Bağlantı şekillerine göre

- Flanş bağlantılı
- Ayak bağlantılı
- Flanş ve ayak bağlantılı
- Moment kol bağlantılı

4.3.3 Encoderler

Encoder dönme hareketini ardışık sayısal sinyallere çevirerek dönme hızı ve dönme sayısı hakkında bilgi verir [10]. Manyetik veya optik olarak çalışır. Doğrusal ve döner olmak üzere ikiye ayrılır.

- **Rotary Encoderler**

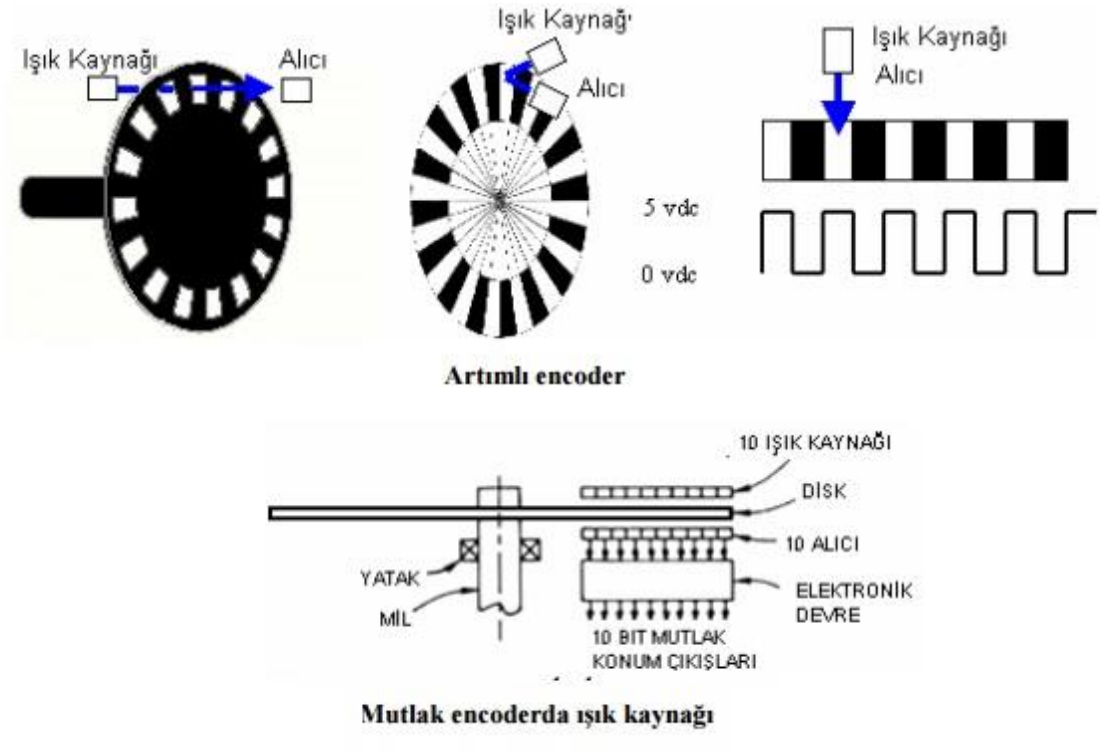
Döner encoderlar da konum bilgisinin veriliş tarzına göre artımlı ve mutlak olmak üzere ikiye ayrılır. Optik sensörlerde ışık kaynağı, alıcı ve üzerinde yarık ya da çizgiler kullanılır. Şekil 4.11 de Döner enkoderli DC motor örneği verilmiştir.

Şekil 4.11: Döner enkoderli DC motor

- **Incremental Encoder;**

Artımlı (incremental) encoderda disk yarıklı ise ışık, yarıktan geçerek alıcıya gelir. Yansımalı tipte ise ışık, üzerine siyah ve beyaz çizgiler çizilmiş disk üzerinden yansır. Artımsal encoderler konumlandırma ve servo gibi makina uygulamalarının en fazla tercih edilen model encoderlerindendir. Değişik hareketlerinin hem geri beslemelerinde hem de

pozisyon lamalarında oldukça iyi cevap verirler. Şekil 4.12 de arttırımlı enkoder iç yapısı gösterilmiştir.

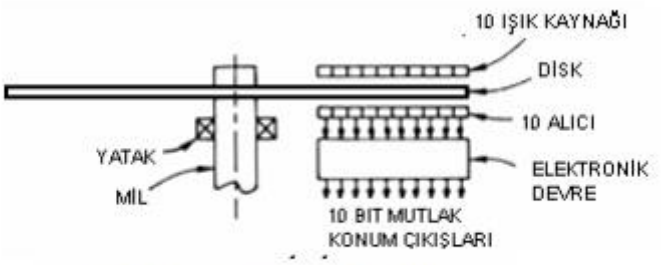


Şekil 4.12: Arttırımlı Encoder İç Yapısı

Dönüş yönü için iki sıra hâlinde dizilmiş yarıklar kullanılır. Alttaki yarıkla üstteki yarık arasında çeyrek açıklık vardır. Üstteki yarıktan gelen sinyale A, alltan gelen sinyale B kanalı denir. Mil döndükçe mil hızıyla orantılı bir frekansta bu kanallarda pulse üretilir. Bu iki sinyal arasındaki faz farkı dönme yönü konusunda bilgi verir. Örneğin A kanalı B'ye göre önde ise mil saat yönünde döner. Disk üzerindeki yarık sayısı encoderin çözünürlüğünü verir. Diskte 1024 yarık ya da iz var ise bu encoderın çözünürlüğü 1024 adım/tur olarak verilir. Her bir kanaldaki pulse sayısı ve çözünürlük bilinirse milin açısal konumu tayin edilir. Encoderlarda çoğunlukla üçüncü bir kanal daha bulunur. Z ya da indeks kanalı denilen bu kanal, her turda bir pulse üretirek milin dönme sayısı hakkında bilgi verir. Ayrıca mekanizmalarda başlangıç konumuna gelmek için bir referans sinyali olarak da kullanılır.

- **Mutlak Encoder**

Mutlak encoderda mil konumuna ilişkin tek bir sayısal sinyal üretilir. Milin her bir konumunun sayısal bir deseni vardır. İkilik düzendeki bu sayısal desendeki bit sayısı, çözünürlüğü verir.



Mutlak encoderda ışık kaynağı

Şekil 4.13: Mutlak Encoder Işık Kaynağı

Genelde binary kod yerine gary kod kullanılır. Bu durumda bu encoderler kullanıldığında gary kod binary koda çevrilmesi gerekebilir.

Mutlak encoder kullanmanın avantajları

Kalıcı Bellek;

Mutlak encoderler kalıcı olan konum doğrulama cihazlarıdır. Güç kesintisi olsa bile pozisyonunu koruyabilir.

Güvenlik;

Bazı uygulamalarda, bir pozisyon kaybına neden olabilen durumlar meydana gelebilmektedir. Makine zararı veya operatörün yaralanmasına bile sebep

olabilecek zararlara mutlak konum bilgisi verebilen absolute model encoderler daha sağlam cevap verebilir.

Elektriksel parazit ve gürültüye çözüm;

Kesin kodlayıcılar konumunu belirlemek için sürekli bir sinyal kodu verirler. Kaçak darbe veya potansiyel parazit olacak uygulamalarda okunacak bilgi hemen hemen değişikliğe uğramaz.

- **Analog Encoderler**

Uygulamaya göre analog girişe de ihtiyaç olabilir. encoderden 0-15 volt veya 4-20 mA gibi analog çıkışlar isteyebilirsiniz. Bu türlü otomasyon uygulamalarında analog çıkışlı encoder sistemleri işinizi hem kolaylaştırır hem de malzemeden tasarruf yapmanızı sağlar. Şekil 4.14 de Analog enkoder örneği gösterilmiştir.

Şekil 4.14: Analog Encoder

- **Rezolverler (Analog Kodlayıcı)**

Rezolveller endüstrinin hemen her branşında kullanılan kodlayıcılardır. İster hafif bir uygulama isterse servo motorun arkasına bağlayın fark etmez. Encoder sistemi gibi dış görünüme sahip olsalar da aslında çok farklı bir iç donanıma ve çalışma prensibi ile çalışırlar. Encoderler dijital bilgiler verirken, rezolverler analog bilgi verir. Bu nedenle çözünürlük problemi yoktur. Şekil 4.15 de Rezolver örneği gösterilmiştir.

Şekil 4.15: Rezolver Örneği

- **Lineer encoderler (düz hareket ölçümleme)**

Ölçülecek veya pozisyon bilgisi alınacak her mekanizma döner olmak zorunda değildir. Düz hareketlerin kontrolünde en sağlıklı ölçüm yapabileceğiniz seçeneklerden biri de lineer encoderlerdir.

4.3.4 DC MOTOR SÜRÜCÜLERİ

DC motor sürücüleri, bütün motor sürücüler gibi mikrodenetleyici pinleri tarafından sağlanan akım ve voltajların motor gibi elektriksel açıdan büyük yükleri sürebilmek için yeteri kadar büyük olmaması nedeniyle kullanılırlar. Fırçalı ve fırçasız olarak iki tip DC motor bulunmaktadır ve farklı tip DC motorların sürülmesi için farklı tipte sürücüler bulunmaktadır. Fırçasız DC motorlarda komütasyon (akım yönü değiştirme) motor içindeki fırça tarafından sağlandığı için bu motorları sürmek daha kolaydır. Fırçasız DC motorlarda ise komütasyon elektriksel olarak motor sürücü tarafından yapılmalıdır. Bu nedenle fırçasız DC motor sürücüler biraz daha karmaşıktır.

Fırçalı DC motorları sürmek teoride basittir. Ancak pratikte bazı zorlukları bulunmaktadır. En basit Fırçalı DC motor sürme uygulaması motoru direk olarak enerji kaynağından sürmek, bir potansiyometre ile voltajı ve hızı ve bir anahtar ile yönü kontrol etmektir. Eğer motor gömülü bir sistemin bir parçası olacaksa bir motor sürücü entegre ve bir kontrol mantığına ihtiyacınız olacaktır. Fırçalı DC motor sürücüleri H köprüsü adı verilen devrelerden oluşurlar. H köprüleri DC motorları çift yönlü kontrol edebilmek amacıyla kurulan H şeklinde devrelerdir.

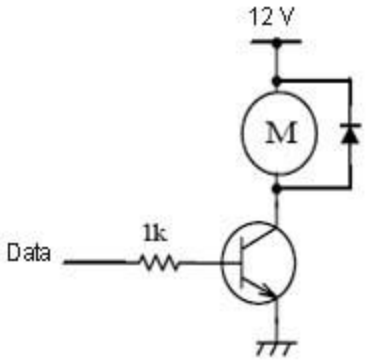
Fırçasız DC motorlarda motor içerisinde akım yönü değiştiren bir fırça bulunmadığından, bu işlem elektronik olarak sürücü tarafından yapılmalıdır. Bu nedenle fırçasız DC motor sürücüleri biraz daha karmaşıktır. Fırçasız DC motor sürücüleri karmaşık bir elektronik kontrol devresinin yanında aynı zamanda rotor pozisyonunu da sürekli takip eden bir sensöre ihtiyaç duyarlar. Fırçalı DC motorların hızları uygulanan voltajla belirlenirken, fırçasız DC motorların anahtarlama frekansına bağlıdır. Motorlar PWM sinyalleri ile sürülmektedir.

- **H Köprüsü Devresi**

H köprüsü devreleri transistörler ya da fetler ile DC motorların direkt elektrik sinyallerinden kontrol edilmesi için hazırlanan devrelerdir.

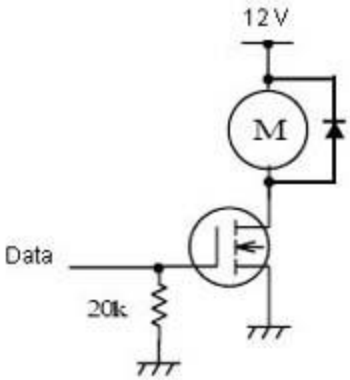
Şekil 4.16 ve Şekil 4.17 deki devrelerde 5 V sinyal verildiğinde motor dönmeye başlar, 0 V sinyal verildiğinde ise motor durur. DC motorların kontrolünde motor yönünün terslenebilmesi gerekmektedir. Transistörler tek kutuplu elemanlar olduğundan motor yönünün terslenebilmesi için iki yada dört transistör ile hazırlanan devreler kullanılır.

Transistör



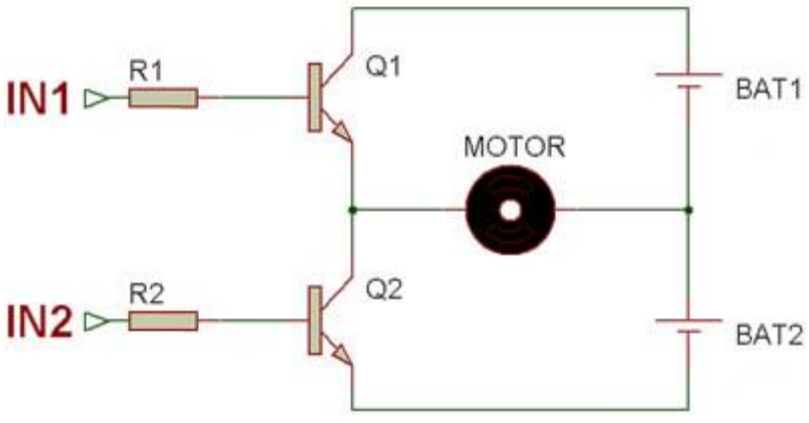
Şekil 4.16: Transistör ile DC Motor Sürüşü

Fet



Şekil 4.17: Fet ile DC Motor Sürüşü

Şekil 4.18 de ise iki transistör kullanılarak hazırlanan yarım köprü motor kontrol devrelerinde kontrol voltajının kutbu kullanılan transistörün tipine bağlıdır. NPN transistör ya da P kanal Fet kullanılıyor ise bu elemanlar pozitif voltaj uygulandığında iletime geçer. PNP transistörler ise negatif voltaj uygulandığında iletkendir.

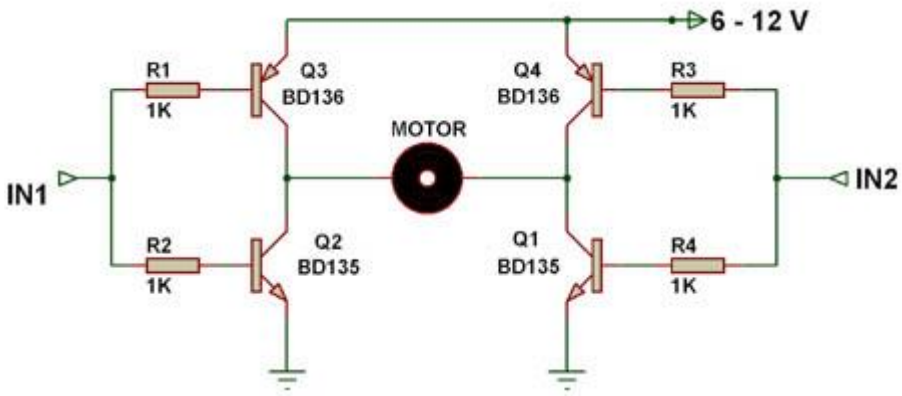


Şekil 4.18: İki Transistörlü Yarım Köprü Devresi

Şekil 4.18 de ise iki transistör kullanılarak hazırlanan yarım köprü motor kontrol devrelerinde kontrol voltajının kutbu kullanılan transistörün tipine bağlıdır. NPN transistör ya da P kanal Fet kullanılıyor ise bu elemanlar pozitif voltaj uygulandığında iletime geçer. PNP transistörler ise negatif voltaj uygulandığında iletkendir.

Yarım köprü devresi için yukarıda verilen şemada; birinci girişe sinyal verildiğinde motor ileriye doğru döner, ikinci girişe sinyal verildiğinde motor geriye doğru döner, her iki girişe de sinyal verilmediğinde motor durur. İki girişe aynı anda sinyal vermek bataryalarda kısa devreye neden olacağı için yasaklı bir koşuldur.

Yarım köprü devrelerinin dezavantajı dual ya da simetrik bir güç sağlayıcının kullanılmasının gerekmesi ve devrenin karmaşıklaşmasıdır. Bu nedenle dört transistörlü H köprüsü devrelerinin kullanılması daha uygundur. Şekil 4.19 da Dört Transistörlü H köprüsü devresi verilmiştir.



Şekil 4.19: Dört Transistörlü H Köprüsü Devresi

H köprüsü devresinde iki sinyal kaynağı kullanılmaktadır. Bu uygulamada dört olasılık bulunmaktadır.

1) Birinci girişe sinyal verildiğinde motor ileriye doğru döner,
2) İkinci girişe sinyal verildiğinde motor geriye doğru döner,
3) Her iki girişe de sinyal verilmediğinde motor durur,
4) Her iki girişe aynı anda sinyal vermek yasaklı koşuldur.

Yarım köprüde ya da H köprüsünde kullanılacak olan transistörler kontrol edilecek olan akıma bağlıdır. Bu akım motorlara göre kararlaştırılır ve birkaç miliamperden 2 amperin üstündeki akım değerlerine kadar değişebilir.

4.4 Lidar

4.4.1 Lidar Çalışma Prensibi

"LİDAR" temel olarak ölçme işlemi için lazer veya ışık kullanan bir uzaktan algılama yöntemidir. 3 boyutlu lazer tarama tekniğiyle çalışan Lidar 1960'lı yıllarda havadan deniz altıların tespiti için geliştirilmiş olup, 1970'li yıllarda kullanılmaya başlanmıştır. Daha sonraki yıllarda ise hem havadan hem de karadan olmak üzere Lidar tekniğini kullanan sensörlerin kullanım alanı ve çeşitlerinde artış meydana gelmiştir. Lidar Teknolojisi günümüzde mimarlık, arkeoloji, şehir planlaması, petrol ve doğal gaz aramaları, haritalandırma, orman ve su araştırmaları alanlarında kullanılmaktadır. Öte yandan günümüz teknolojisinin trendlerinden olan otonom yani sürücüsüz araçlarda yine bu yöntemi kullanırken anlık olarak veri üretimi ve analizi yaparak bir nevi lazer gözü ile kontrollü bir şekilde hareket edebilirler.

Açılımı "Light Detection and Ranging" olan LİDAR sistemi özetle bir ışık sinyal gönderdiği ve sinyalin geri dönmesini beklediği anlamına gelir. Yayılan ışığın sensöre ne zamanda geri döndüğü belirlenerek konumlandırma yapar. Bu konumlandırmalarını milyonlarca noktadan oluşan nokta bulutları haline dönüştürülür.

- Otonom araçlarda LIDAR'IN genel görevi

-Orta-uzak mesafeli engellerin saptanması
-Görsel algıya dayalı sistemlerin uzaklık hesaplarında stereo kamera ile işbirliği

- Bizim Projemizdeki Görevi

-Kısa vadede
    -Yol üstündeki durağan engelleri saptamak
-Orta vadede
    -Hareketli nesneleri saptamak
-Uzun vadede
    -Derinlik bilgisi sağlamak

4.5 Lipo Pil

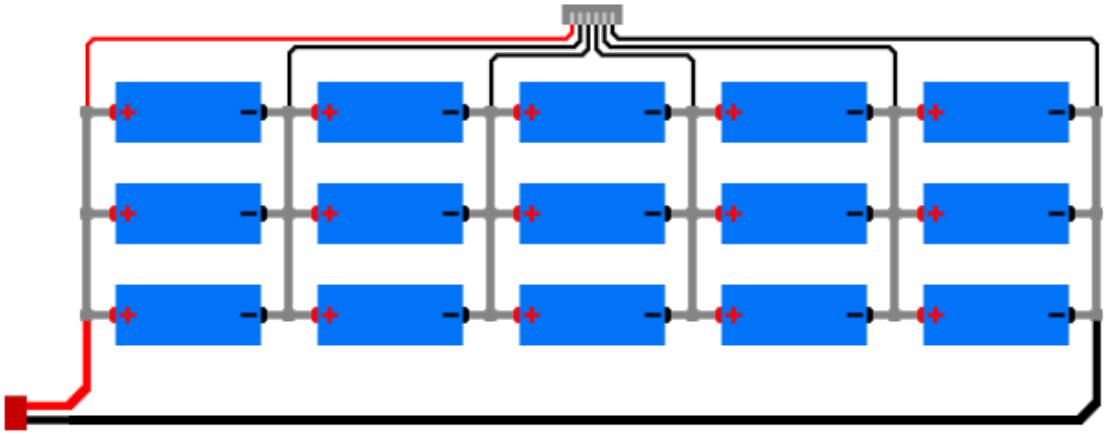
Lityum-iyon polimer pil olarak adlandırılmaktadır. Lityum-iyon pilin daha gelişmiş bir tasarımıdır ve aynı karakteristikleri taşımaktadır. Bu tür pillerde hafıza etkisi yoktur ve kullanılmadıkları zamanlardaki enerji kayıpları yavaştır. Uygunsuz kullanılmaları halinde tehlikeli olabilirler. Eğer gerekli önlemler alınmaz ise diğer pil türlerine göre ömürleri daha kısa olabilir.

Kullanım Alanları:

Lityum-iyon pil çeşitleri dizüstü bilgisayarlar, cep telefonlar, müzik çalarlar ve daha bir çok taşınabilir dijital cihazlarda kullanılmaktadırlar. Lipo piller küçük boyutlarına rağmen yüksek enerji gereksinimlerini karşılayabildikleri için RC kontrollü elektrikli modellerde kullanılmaktadırlar. Ayrıca 2013 yılında batarya sorunu yüzünden uçağın bir bölümünde çıkan yangın sonucu uçuşları bir süreliğine yasaklanan Boeing 787 tipi yolcu uçaklarında dahi bu tarz piller kullanılmaktadır.

4.5.1 Lipo Pillerin Teknik Özellikleri

Lipo piller hücrelerden oluşmaktadırlar. Tek bir hücrenin boş hali 3V dolu hali ise 4.2V olmalıdır. Lipo pillerin hücreleri seri(S) veya paralel(P) bağlı olabilir. Bu piller bağlantı şekline göre adlandırılırlar. Örneğin; 2 hücre seri bağlı ise 3S, 3 hücre seri bağlı ise 3S, 2 hücre paralel bağlı ise 2P, 3 hücre seri bağlı ve bunlara paralel bağlı bir takım daha var ise 3S2P olarak adlandırılırlar. Peki nedir bu S,P ? Seri bağlı demek voltaj değerimizin arttığı anlamındadır. Paralel bağlı da aynı voltaj değerinde verilen akımın arttığını gösterir. Şekil 4.20 de Lipo Pil için örnek bir bağlantı şeması verilmiştir.



Şekil 4.20: 5S3P Lipo Pil Bağlantı şeması

Örneğin; aynı özelliklere sahip hücreleri düşünürsek:
3S => 3 x 4.2V = 12.6 V

4S=> 4x 4.2V = 16.8V

3S2P => 3 x 4.2V = 12.6 V (Paralel bağlı olduğu için 3S ye göre 2 kat daha fazla akım verir.)

mAh Nedir?

mAh değeri pilin kapasitesini göstermektedir (miliamper/saat). 2000 mAh kapasiteli bir pilden 1000 miliamper çekilirse 2 saatte pil tamamen deşarj olur. 2000 miliamper çekilirse 1 saatte tamamen deşarj olacaktır. mAh değeri arttıkça kapasite artacaktır ve aracınızı kullanma süreniz uzayacaktır.

"C" değeri Nedir?

"C" değeri pilin deşarj hızını temsil eder. 10C değerine sahip bir pil kapasitesinin 10 katı kadar hızda deşarj edilebilir. 20C değerindeki pil 20 katı kadar, 30C değerindeki pil 30 katı kadar … bu şekilde devam eder. Örnekle açıklamak gerekirse, 2000mAh kapasiteli 10C bir pilden sürekli olarak en fazla 20 amper çekilebilir. 5000 mAh 25C bir pilden sürekli olarak en fazla 125 amper çekilebilir. Genelde 25-30C değerindeki piller işimizi görmektedir fakat imkan varsa daha yüksek C değerine sahip pilleri tercih etmekte fayda vardır.

4.5.2 Lipo Pil Nasıl Şarj Edilir?

Yukarıda voltajlarla alakalı kısa bir bilgilendirme yaptık, tekrarlarsak lipo hücresi 4.2V' a ulaştığında tamamen dolmuş, 3.7V' a ulaştığında tamamen boşalmış olacaktır. Lipo pil şarj etmek için bilgisayarlı şarj cihazı kullanmanızı tavsiye ederiz. Turnigy Accucel-6, Imax B6 yaygın kullanılan şarj aletleridir. Her pil kendi özelliğine göre şarj edilme koşullarına sahiptir. Örnek olarak 3S 5000 mah bir pili şarj edersek, Lipo balance modunda 3S seçip kapasitenin %80′ i kadar yani 4.0 Amper ile şarj etmeliyiz. Bahsettiğimiz şarj aletleri otomatik şarjı kesme özelliğine sahiptirler. Aynı pil için konuşursak şarj edilen mAh 5000 değerine ulaştığında şarjı kesecektir. Ancak pil tamamen boş olmadığında Voltajı kontrol ederek 3S pilin maksimum olması gereken voltajı (4.2V*3) 12.6 Voltta şarjı kesecektir.

Lipo Balance modunda şarj etmenin avantajı her hücreye ayrı şarj işlemi uygulaması. Bazı durumlarda hücreler birbirinden farklı voltaj değerlerine sahip olabiliyor. Örneğin 2S bir pilde 1.hücre 3.9V 2. hücre 4.15V değerlerinde olabiliyor, eğer normal modda şarj edersek 2.hücre 4.2 olduğunda 1. hücre henüz tamamlanmamış olacak, daha da kötüsü ,1.hücreyi 4.2

Volta getirmeye çalışırken 2. hücre kapasitenin üzerine çıkacak ve hücreye zarar verecek. Bu yüzden balans modunda şarj ederek her hücreyi gerektiği kadar şarj ediyoruz.

Özellikleri 2200mAh 3S1P 20C bir pilin üzerinden giderek teorik olarak anlatırsak;

- Bu pil sabit 2.2 amper akım çeken bir sisteme bağlanırsa yaklaşık olarak 1 saat kullanılabilir. Yani başlangıçtaki akım değeri olan 2.2A, 1 saatte durmaksızın verebileceği akım değeridir.
- 2.2A x 20 C = 44A , bu pil en fazla 44A akım verebilmektedir. Buradaki C değeri ne kadar büyük ise pilin kapasitesi yani verebileceği akım değeri o kadar büyüktür.
- Bu pil 1 saat (60 dakika) boyunca durmadan 2.2A akım verebiliyorsa 44A akımı ne kadar süre verebilir. Bu sorunun cevabını basit bir oran-orantı yaparak buluyoruz. (60 dk x 2.2A)/44A=3 dk boyunca durmadan 44A verebilir.
- Bu pil, lipo piller için üretilmiş bir şarj aleti kullanılarak 1C değeri ile yani 2.2A ile 1 satte şarj edilir.

4.5.3 Lipo Pillerin Kullanımı

Lipo piller için oldukça tehlikeli olduğu her zaman benzer konularda dile getirilmektedir. Fakat lipo piller çoğu zaman doğru kullanılmadığı için tehlike oluşturmaktadırlar.

Lipo pillerin kullanımında dikkat edilmesi gerekenler:

- Lipo piller kesinlikle kapasite değerlerinin üzerinde kullanılmamalıdır. Örneğin bir quadrotor da motor, esc hatta tüm elektronik sistemin ihtiyacı olan akım değeri ve havada ne kadar süre kalınmak istendiği hesaplanıp buna uygun bir motor seçilmelidir. Lipo pillerden verebilecekleri max akımdan daha fazlası istendiğinde yangın ve benzeri olaylar ile karşılaşılabilir.
- Lipo pillerin uç kabloları KESİNLİKLE kısa devre (birbirine değmemeli) edilmemelidir.
- Lipo pilleri onlar için özel olarak üretilmiş balanslı şarj aletlerini kullanarak şarj etmeliyiz. Ayrıca pilin 1C değerindeki akım ile şarj etmeliyiz. Örneğin 2200mah 3S 20C bir pil için 1C=2.2A dir.
- Lipo piller kullanılırken hücre başına gerilim değeri 3V un altına düşerse hücremiz ölebilir ve bir daha kullanılamaz. RC modellerde programlı ESC ler bunu engellemektedir. Fakat diğer projelerinizde kesinlikle hücre başına gerilimi kontrol edebileceğiz bir sistem kullanmalısınız.

- Hücre başına gerilimin 3V un altına düşmesi, sıcaklık vb. gibi durumlarda pillerde şişme meydana gelir. Piyasada modelcilikle uğraşıp kullanan birçok insan olmasına rağmen şişmiş piller kullanılmamalıdır.
- Lipo pillerde hücre başına gerilim 4.2V dan yüksek olmamalıdır. Zaten lipo piller için üretilmiş bir şarj cihazı kullanıyorsanız, cihazınız buna izin vermeyecektir.
- Pil seçimi kullanılan modele ve sisteme uygun olmalıdır. Örneğin; elektrikli modellerde daha fazla kullanım için C değeri daha yüksek piller tercih edilmelidir. Ama bu da beraberinde ağırlık sorununu getirmektedir.
- Lipo pilleri her zaman max performansta kullanmayın. Her zaman %20 lik bir kullanım boşluğu sağlıklı kullanım açısından önemlidir.

4.5.4 Lipo Pillerin Saklama Koşulları:

- Lipo piller hücre başına yarı gerilimle (3.6V-3.75V) saklanmalıdır.
- Lipo piller serin kuru ve ışıktan uzak bir ortamda saklanmalıdır.
- Lipo pilleri saklamak için özel saklama kapları da mevcuttur.

Şekil 4.21 de çeşitli boyutlarda lipo pil görselleri verilmiştir.

Şekil 4.21: Çeşitli Boyutlarda Lipo Pil Örnekleri

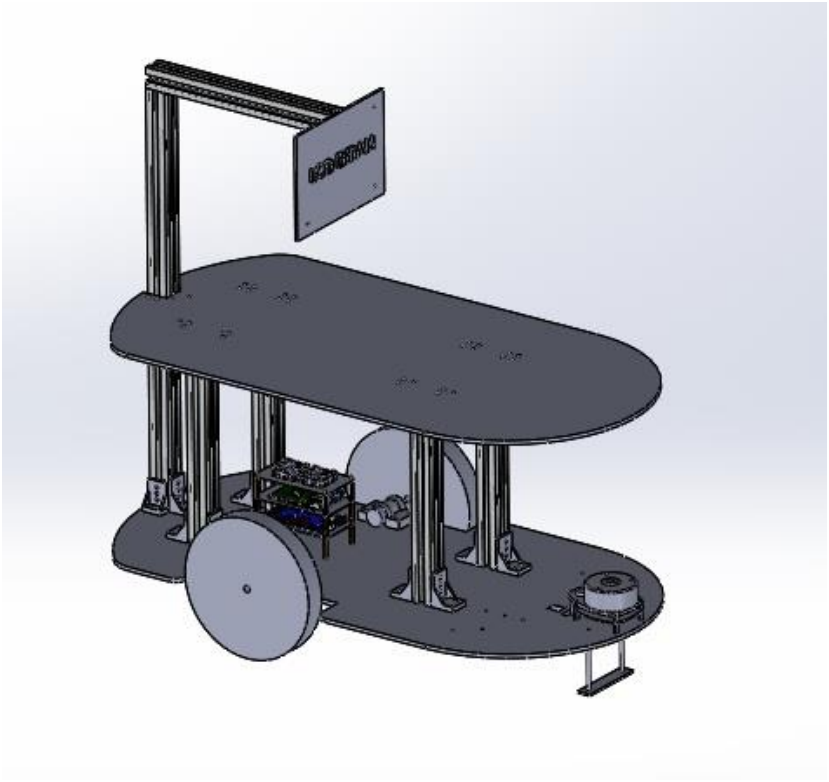
5. GERÇEKLEŞTİRİLEN SİSTEMİN YAPISI VE YÖNTEMLER

5.1 Sistemin Mekanik Tasarımı

Projenin amacı doğrultusunda robotun tasarımı Solidworks programı ile gerçekleştirilmiştir. Gerçekleştrilen tasarım sonucunda şeffaf pleksi levhalar kullanılarak robotun ana gövdesi oluşturulmuştur. Oluşturulan bu gövdenin sağlamlığı sigma profiller ve 3 boyutlu yazıcıdan basımı yapılan parçalar ile sağlanmıştır.

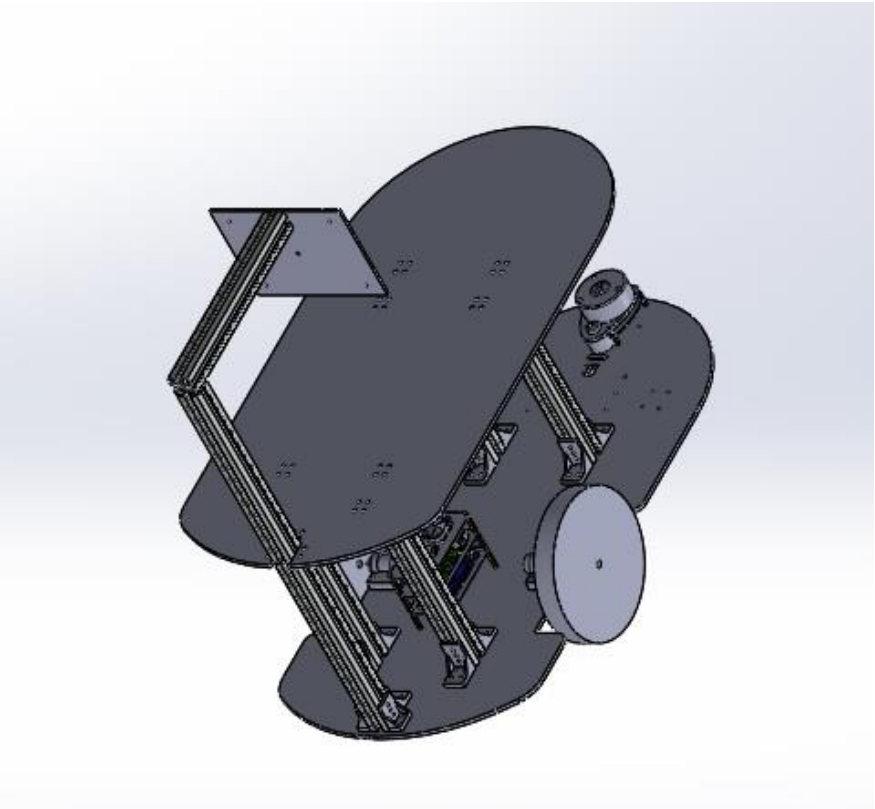
Şekil 5.1 de görüldüğü gibi robot üzerinde kontrol kartlarının yerleşimi 2 teker arasında oluşturulan platforma yapılacaktır. Motorların ana gövdeye sabitlenmesi ise kestirilen metal motor tutucular tarafından sağlanacaktır. Robota görevler verilmesi amacıyla kullanılacak olan LCD ekran ise robotun arka kısmından yukarıya doğru çıkarılan sigma profilin uç kısmına sabitlenecektir. Kullanılacak olan lidar sensörünün yerleşimi ise robotun uç kısmında oluşturulan platforma yapılacaktır. Aşağıdaki resimlerde robotun farklı açılardan görünümleri verilmiştir.

Robotun tasarımının ön çaprazdan görünüşü Şekil 5.1 de verilmiştir.



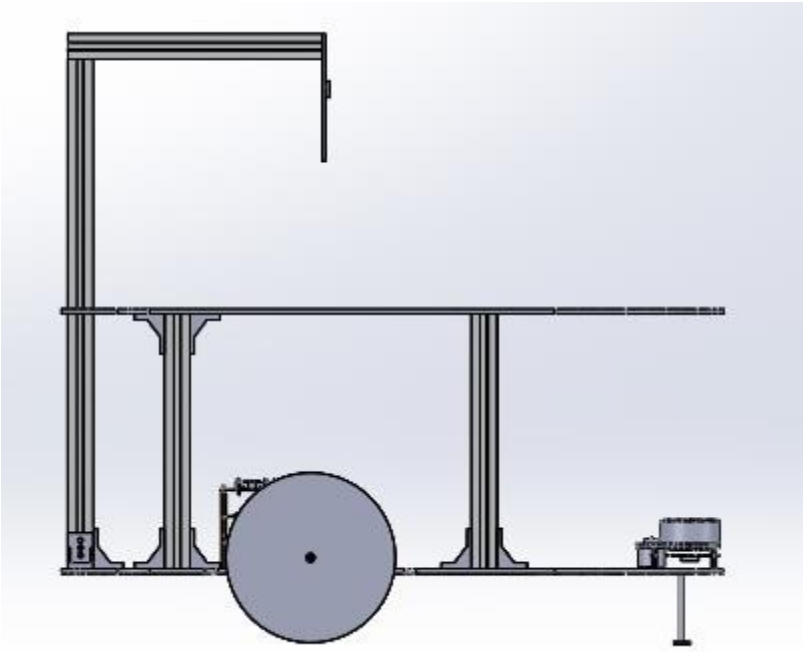
Şekil 5.1: Robotun Ön Çapraz Görüntüsü

Robotun tasarımının arka çaprazdan görünüşü Şekil 5.2 de verilmiştir.



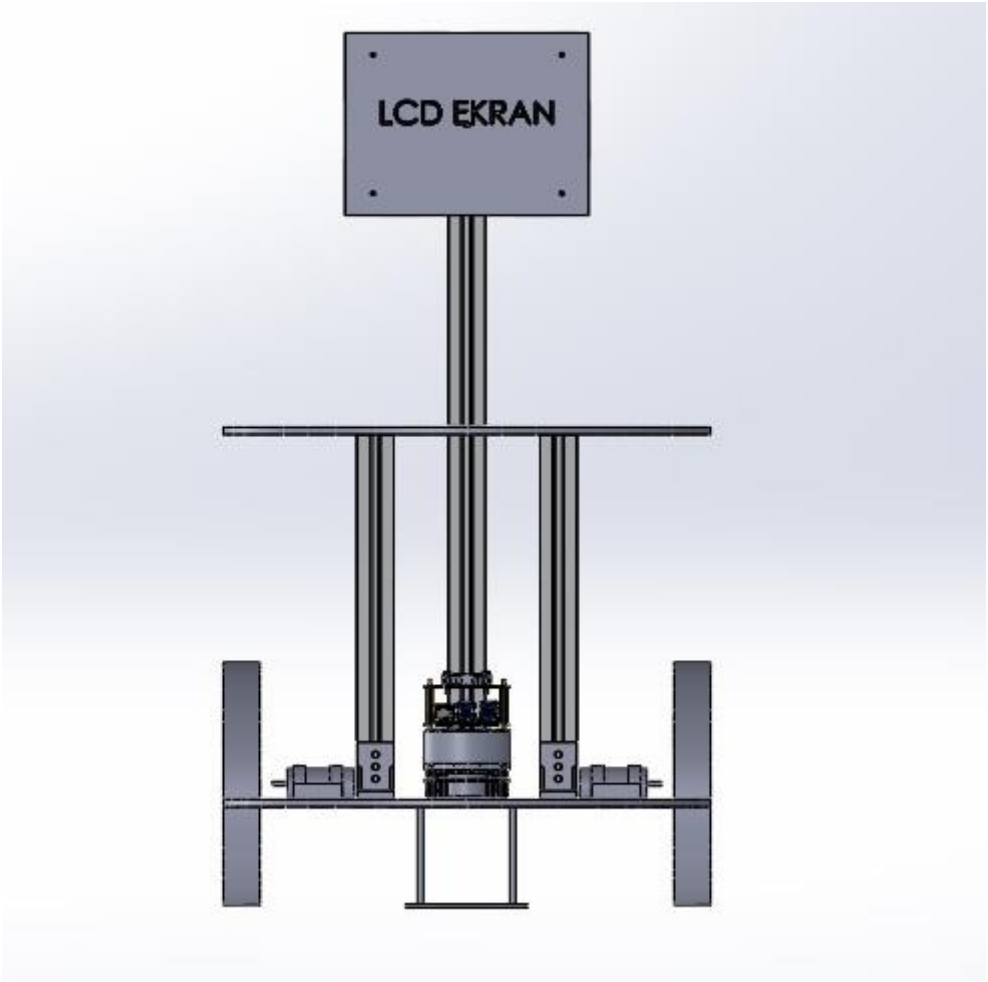
Şekil 5.2: Robotun arka çaprazdan görünümü

Robotun tasarımının yandan görünüşü Şekil 3.5'te verilmiştir.



Şekil 5.3: Robotun yandan görünüşü

Robotun tasarımının önden görünüşü Şekil 5.4 de verilmiştir.



Şekil 5.4: Robotun önden görünüşü

5.2 YÖNTEMLER

5.2.1 SLAM Nedir?

SLAM problemi, bilinmeyen bir ortamda nereye konulduğu bilinmeyen bir mobil robotun artımlı olarak tutarlı bir harita inşa edebiliyor mu ve aynı anda bu haritadaki konumunun belirleyebiliyor mu sorusunu sorar [11].

SLAM probleminin çözümü robot topluluğu için son on yılda dikkate değer bir başarıdır. SLAM formüle edilmiş ve teorik olarak değişik formlarda çözülmüştür.

Farklı SLAM çözümleri vardır. Bunlar:

- İçeri Robotlar
- Dışarı Robotlar
- Denizaltı Robotlar

· Hava Robotları

Sistemlerinde ayrı ayrı SLAM çözümleri vardır.

5.2.2 SLAM Paradigmaları

SLAM çözümünde kullanılan 3 ana paradigma vardır. İlki Extended Kalman Filter (EKF) SLAM, tarihçe olarak en eskisi olsa da hesaplama özelliklerinde olan kısıtlamalardan dolayı popülaritesini yitirmiştir. İkincisi ise grafiksel sunum tabanlıdır. Başarılı bir şekilde seyrek doğrusal olmayan optimizasyon metotlarına uygulanabiliyor ve bu sayede tüm SLAM problemlerine çözüm için ana paradigma olmuştur. Sonuncu metot ise parametrik olmayan istatiksel filtreleme tekniği olan Particle Filter'dir. Çevrimiçi SLAM için popüler bir metottur. SLAM'ın veri ilişkilendirmesi sorununa yeni bir çözüm sağlar.

5.2.2.1 Extended Kalman Filters (Genişletilmiş Kalman Filtresi)

Bu anlatımda Extended Kalman Filter (Genişletilmiş Kalman Filtresi) GKF kısaltması ile anlatılacaktır.

SLAM problemi için birçok çözüm mevcuttur bunlardan birisi de Genişletilmiş Kalman Filtresi çözümüdür. Bu çözümün diğerlerinden farkı robotun lokalizasyonunu yaparken aynı zamanda oryantasyon bilgisini de bulabilmesidir.

SLAM'ın ilk ve en etkili paradigmasıdır. Bu algoritma, özellik tabanlı çevre sunumunu objelerin uygun parametre uzayında etkili bir şekilde nokta olarak temsil edildiği metrik bir sunum yapar.

Kalman filtresi ile karşılaştırıldığında Kalman Filtresi sadece doğrusal sistemler için çalışır. Ama Genişletilmiş Kalman Filtresi doğrusal olmayan sistemlerle de çalıştığı için tercih edilir.

Genişletilmiş Kalman Filtresi (GKF), Kalman Filtresi'nin doğrusal olmayan fonksiyonlarda da kullanılmasını sağlayan bir metottur, robotun hareket ve algı fonksiyonları doğrusal olmadığından robotik problemlerinde de kullanılır. Kalman filtresi formülleri içerisinde sadece küçük bir değişiklik yapılarak Genişletilmiş Kalman filtresi(GKF) ortaya atılmıştır.

GKF'yi örnek üzerinden anlatmak gerekirse:

Bir aracı düşünelim t zaman uzayında hareket ediyor olsun. İlerlerken gözlem ve düzeltme yaparak ilerler. Bu sırada belirsizlik ve işaretlerin konumunu öngörür. Belirsizliklerdeki değişmeler buralarda gözlemlenebilir. Başlangıçta gözlem yapar ve ilerler. Sonraları hem gözlem hem de düzeltme yaparak gitmelidir. İlerledikçe araç önceki adımlardaki işaretleri ve bulunduğu konumu öngörür. Ve düzeltme için algılayıcı gözlemlerine ihtiyaç duyar. Bu şekilde ilerledikçe kendi konumunu ve işaretlerin konum belirsizliklerini çözümler.

Durum matrisi adlı bir sütun matrisinde GKF'nin bütün durum değişkenlerinin parametreleri tutulur. Bu parametreler SLAM probleminde robot konumu ve yer işaretçilerinin konumu olmaktadır. Bu matriste başlangıçta robotun konumu ile ilgili, sonra yer işaretçileriyle ilgili parametreler tutulur. Bu algoritmada hareket güncellemesi ve algı güncellemesi bulunur. Hareket güncellemesinde kinematik kullanılarak hesaplanan robotun konumunu ifade eden Gauss dağılımının ortalama ve kovaryans değerleri güncellenir. Algıda ise en son görülen yer işaretçisinin önceden görülmüş olan yer işaretçilerinden biri olup olmadığı kontrol edilir. Buradaki sonuca göre (önceki bir işaretçi ya da yeni bir işaretçi) gözlemin güvenirliliğine bağlı olarak yer işaretçisi ve robot konumunda düzeltmeler yapılır, ilk defa görülüyorlar ise durum matrisine eklenir.

Hareket Güncellemesi: Robotun konum inancına odometri verisi yardımıyla uygulanan güncellemedir.

Algı Güncellemesi: Robotun inancıyla algılayıcı verileri yardımıyla uygulanan güncellemedir.

GKF 'nin güçlü tarafı robotun konumlarından ve işaretçi nesnelerin yerinden, robotun kendi konumunun koşullu olasılığını gerçek zamanlı olarak çok iyi bir şekilde tahmin edebilmesidir. Bunu ancak gaussian metodları içinde barından yöntemler yapabilir.

GKF'nin gerçek zamanlı olarak robotun konumlarından ve işaretçi nesnelerin yerinden kendi konumunu n koşullu olasılığını yüksek derecede tahmin edebilmesidir. Bunu yaparken de Gaussian Metotları yardımıyla yapar.

GKF'nin eksilerinden bahsetmek gerekirse;

· Yaklaşık Gauss fonksiyonu hesaplanması beraberinde belirsizliklerden dolayı karmaşıklık getirir.

· İşlemlerin artması. (Jacobian Matrislerinin hesaplanması)

· Lineerleştirme (Doğrusallaştırma) gerçek inanç değerlerinden sapar.

· Robotun hareket sisteminin çok iyi yapılmış olması gerekir. Algılayıcı gürültüleri (tekerlek kayması, takılma, ayna vs.) çok iyi bilinmesi gerekir.

5.2.2.2. Particle Filter (Parçacık Filtresi)

İkinci pardigma olan Parçacık metotları Parçacık Filtresi tabanlıdır. Bu filtre son yıllarda popüler olmaya başlamıştır.

Sonlu sayıda dağılmış olan parametrelerin inanç değerlerine yakınlaşması ve Bayes filtrelerinin parametrik olmayan (nonparametric) alternatif gerçekleştirimidir. GKF ile karşılaştırıldığında dağılımların Gauss dağılımı olması zorunluluğu yoktur. PF'de durumların önceki dağılımlarına parçacık adı verilir. Genellikle parçacık sayısı 1000 gibi büyük rakamlar olur.

Parçacık Filtesi algoritması Genişletilmiş Kalman Filtresi ile benzer şekilde inanç değerlerini yaklaşık olarak hesaplar. Bu hesaplamalardaki zamansal gecikmeler kaynaklı zamanı yakınsama hataları oluşur.

Parçacık Filtresi'nin temeli bir değişkenin tüm olasılık dağılım fonksiyonunun örnekleme tabanlı bir temsilini yapmasıdır. Bu örnekleme süreçlere bağlıdır, değişkenin değerleri süreçlerden çıkan sonuçlara göre değişkenin durumu değişir. Bu değişkenin birden fazla parçacık kopyası bulunur. Ve bu kopyaların her birinin kendisiyle ilişkilendirilmiş bir ağırlığı bulunur. Bu kopyalarının her birinin ağırlığı alınarak ana değişkenin değeri hakkında bir tahmin yapılır.

Parçacık Filtresi iki adımda gelişen özyinelemeli bir algoritmadır. Bu adımlar tahmin ve güncelleme adımlarıdır.

Tahmin Adımı: Robotun her konum değiştirmesinde gürültünün rastgele bir değer seçilerek kullanılan modele göre değişkenin değiştirilmesidir.

Güncelleme Adımı: Tahmin adımından sonra çalışır. Algılayıcıdan alınan son bilgiler ışığında her parçacığın ağırlığının tekrar hesaplanmasıdır.

Tahmin adımında parçacıklar için rastgele gürültü değerleri ekler model. Güncelleme adımı parçacık ağırlıklarını güncelleyebilmek için sensörlerden gelen verilerden yardım alır,

bunun amacı hareket halindeki robotun olasılık dağılım fonksiyonunun tam olarak tanımlanmasıdır.

Robotun konum tahmini için 3 farklı seçenek mevcuttur. Bunlardan ilki ağırlıklı ortalama, ikincisi en iyi parçacık ve üçüncüsü ise en iyi parçacığın etrafında bir çerçeve içinde ağırlıklı ortalama kullanan gürbüz ortalamadır. En iyi parçacık yönteminin dezavantajı olarak ayrıklaştırma hatalarını ve çok modelli dağılımlarda iyi sonuç verememesini sayabiliriz. En iyi yöntem olarak gürbüz ortalama sayılabilir ama bu da fazlasıyla işlem yükü getirecektir.

Parçacık Filtresi'ndeki önemli problemlerden birisi de birkaç yinelemeden sonra parçacıkların tükenmesidir. Parçacıkların konum tahminlerinin sapma yapıp robotun gerçek konumundan çok farklı konumlarda tahmin yapmaları ve ağırlıklarının robotun konumunu bulunmasına katkılarının küçülmesi de ayrı bir problemdir. Küçük bir not olarak FastSLAM algoritması parçacık filtresi kullanır.

5.2.2 3. Graf Tabanlı Optimizasyon Teknikleri

Sonuncu paradigma olan Graph-Based Optimization Tecniques adı altında incelenen bu paradigma SLAM problemini doğrusal olmayan seyrek optimizasyon ile çözer. SLAM'ın grafiksel temsilinden alınan sezgileri çizerler.

Graf tabanlı SLAM çözümünde temel sezgiler akıcıdır. Yer işaretleri ve robotun pozisyonu grafın üzerindeki birer düğümdür. Her yer işareti çifti birbirine yayla bağlıdır. Bunlar odometrinin okunması ve iletilmesiyle temsil edilirler. Robotun hissetmesiyle (sensörlerden) diğer yer işaretleri ve konum bilgileri de zamanla oluşur. İletmenin verimine göre robot haritayı ve yolu yumuşatır.

Grafiksel paradigma veri ilişkilendirme problemini ele alabilmek için kolayca genişletilebilir.

Bir diğer avantajı ise yüksek boyutlu haritaları EKF(GKF) SLAM'ından daha iyi boyutlandırabiliyor olmasıdır. GKF kovaryans matrisindeki anahtar limiti sorunu da Grafiksel paradigmada yoktur. Güncelleme zamanı ise sabittir ve gerekli bellek miktarı sabittir(bazı varsayımların altında).

Graf tabanlı paradigma ile yapılmış olan en geniş bazı SLAM haritaları oluşturulmuştur fakat genelde çevrimdışı biçimde.

Problemi temsilen graf kullanılır. Grafikteki her düğüm haritalama sırasında robotun bir pozuna karşılık gelir. İki düğüm arasındaki her kenar, aralarındaki uzamsal kısıtlamaya karşılık gelir. Graf tabanlı SLAM temelde; grafiği oluşturup, kısıtlamalar tarafından getirilen hatayı en aza indirecek bir düğüm yapılandırması bulmak üzerinedir.

Her düğüm robotun pozisyonuna ve lazer ölçümüne karşılık gelir. Her köşe arası düğümler ise düğümler arasındaki mekânsal kısıtları temsil eder. Hedef kısıtlamalar tarafından getirilen hatayı en aza indirgeyen düğümlerin bir biçimini bulmaktır.

5.3 Haritalandırma

Robotların otonom bir yapıda olabilmesi için 3 yetiye sahip olması gerekir. Bunlar yürüyebilme, konumlama ve haritalama fonksiyonlarıdır. Bu konu başlığı altında temel fonksiyonlardan birisi olan haritalama problemi üzerinde durulacaktır.

Robotik haritalama için iki bilgi kaynağı vardır. Bunlar allothetic ve idiothetic bilgilerdir. Idiothetic bilgi kaynağı bilgiyi odometriden alır. Robot hareketi hakkında iç bilgiyi destekler. Ve bu bilgiyle dead reckoning yaklaşımı robotun pozisyonunu bulabilir. Allothetic bilgi ise algı ve sensör kaynaklıdır. Çevre hakkındaki dış bilgiyi sağlar. Robot için bilgi lazer, sonar, sensör veya görüntü kaynaklı olabilir.

Robotik haritalamadaki bir problem de ölçüm gürültüsü adıyla bilinir. Bu sorun eğer gürültü istatiksel olarak bağımsız olan çeşitli ölçümler içindeyse kolayca çözülebilir.

Haritalama çeşitli LIDAR, Lazer gibi sensörler yardımıyla yapılabildiği gibi görüntü sistemleriyle de yapılabilir. Bu düşük çözünürlüklü bir LIDAR'dan daha yüksek çözünürlük sunar. Kamera yardımıyla haritalama imaj işleme algoritmaları ile ilişkilidir ve bu da daha yüksek işlem gücü gerektirir.

Haritalar topografiksel ve metrik olmak üzere ikiye ayrılır.

Metrik haritalarda çevre koordinatlı obje gruplarının iki boyutlu uzayda metrik sistemle oluşturulmuş olarak tanıtılır. Mimari taslağına çok benzer. Birçok robotun kullanabileceği haritalardır. Topografiksel haritalara göre oluşturulması daha kolaydır.

Topografiksel haritalarda çevre özel yerlerin ve bu yerlerin arasındaki ilişki olarak temsil edilir. Topografiksel haritalar çeşitli yerler arasındaki bağlantıyı tanımlar ve metrik harita oluşturulurken otomatik olarak çıkartılabilirler. Geometrik çözünürlüğü arttırır ve bu

bilgiler planlama, navigasyon gibi görevler için destekleyici olabilir. Büyük ölçekli ortamlar için kullanılır.

Haritalama için bulunan çözümler arasında gmapping ve Hector SLAM metotlarını inceleyeceğiz. Yaygın olarak kullanılan çözüm gmapping algoritmasıdır.

5.3.1 Hector SLAM Algoritması ile Haritalama

Hector SLAM metodu ışınların bitiş noktalarının hizalanmasının optimize edilmesi üzerinedir. Bilgisayar görüşündeki işten esinlenen Gauss-Newton yaklaşımı temel fikirdir. Lucas ve Kanade (1981) "An iterative image registration technique with an application to stereo vision" adlı araştırmasında DARPA Image Understanding Workshop içerisinde yayınlanmıştır. Bu yaklaşım ile ışınların bitiş noktaları arasında veri ilişkisi aramaya veya kapsamlı pozisyon aramasına ihtiyaç kalmamıştır.

Bu metotta sadece z bitiş noktasındaki filtreleme tabanlı koordinat kullanılmış. Böylece amaçlanan tarama düzleminin eşiğindeki sadece bitiş noktaları tarama eşleştirme işleminde kullanılmıştır. Burada algoritma kaynaklı olmayan sıkıntılar doğabilir, örneğin pencereden dışarı çıkan ışınlar haritaya dâhil edilebilir.

5.3.2 Gmapping Algoritması ile Haritalama

Bu haritalama metodu bilinen en popüler ve açıkça bilinen SLAM problemi algoritmasıdır. Giorgia Grisetti, Wolfram Burgard ve Cyrill Stachniss tarafından geliştirilmiştir. Robotun pozisyonlarını takip etmek için Rao-Blackwellized parçacık filtresini kullanır. Hâlihazırda oluşturulmuş olan harita parçaları ve sensör verileri temellidir.

Her bir parçacık = robotun geçmiş pozisyon örneği + harita üzerinde verilen önceki pozisyon örnek geçmişi; tüm parçacıklar bağımsız olarak kendi haritasını tutar. Gmapping olasılıksal dağılım modeli robotun son gözlemlerini de hesaba katarak bir yayılım oluşturmaktadır. Bu sayede etkin bir şekilde haritalama yapılarak belirsizlikler de aynı zamanda ortadan kalkar

Rao-Blackwellized önem yeniden örnekleme filtresini örnekler. Bunu sensör gözlemleri ve odometri verisi mevcut oldukça artımlı haritalama işlemi için yapar. Aracın yörüngesi ve haritayı temsil eden örnekler setini günceller. Süreç dört adımdan oluşur.

1) Örnekleme: Yeni nesil parçacık { ()}, önerme dağılımı π'den örnekleme yoluyla -1 () neslinden elde edilir. Genellikle, olasılık odometri hareket modeli teklif dağılımı olarak kullanılır.

2) Önem Ağırlığı: Her parçaya, önem örnekleme ilkesine göre bireysel bir önem ağırlığı () atanır.

3) Yeniden Örnekleme: Parçacıklar, önem ağırlığına oranla orantılı olarak çizilir. Sürekli bir dağılıma yaklaşmak için yalnızca sonlu sayıda parçacık kullanıldığından bu adım gereklidir. Ayrıca, yeniden örnekleme, hedef dağılımın öneriden farklı olduğu durumlarda bir parçacık filtresi uygulamamızı sağlar. Yeniden örneklemeden sonra tüm parçacıkların ağırlığı aynıdır.

4) Harita Tahmini: Her bir parçacık için o parçacığın yörüngesi ve gözlemleri temelli karşılık gelen harita tahmini hesaplanır.

Odometri verileri olmadan gmapping yapılamaz.

Rao-Blackwellized filtresinin gmapping algoritması üzerinde yaptığı geliştirmeler sonucu karmaşıklıklar azaltılmış ve daha doğru sonuçlar sağlanmıştır. Yeniden örnekleme kısmında yapılan değişiklikler sonucu bütün parçacıklar değerlerini kaybetmiyor sadece düşük değerli parçacıklar kayboluyor ve yerine daha yüksek olasılıklı parçacıklardan türeyen bir parçacık konuluyor.

6.SONUÇLAR

6.1 Sonuçların Tartışılması

SolidWorks üzerinde 3 boyutlu tasarımı gerçekleştirilen robotun, Robot Operating System (ROS) üzerinde modellenerek simülasyonu sağlanmıştır. Robotun ve çalışacağı ortamın simülasyonu gazebo ve rviz üzerinde xacro dili kullanılarak oluşturulmuştur. Gerçek zamanlı çalışan bir sistemin Robot Operating System üzerinde modellenmesi ve robotun kontrol algoritmalarının ROS paketleri üzerinden gerçekleştirilmesi geliştirme sürecini hızlandırmaktadır.

6.2 Özsonuçlar ve Öneriler

Gerçek zamanlı yapılan testlerde ve simülasyon ortamında yapılan testlerde aracın tekerlerinden alınan odometry verisinin tek başına yeterli olmadığı, robotun hassas dönüşlerinde odometry verisinde kaymaların olduğu gözlemlenmiştir. Odometry verisi için motor encoder verilerinin sanında imu sensörünün kullanılması dönüşlerde daha hassas odometry verisinin oluşturulması açısından fayda sağlamaktadır. Gerçek zamanlı olarak hectormapping, gmapping metotları test edilmiş ve çıkarılan haritaların başarısı incelendiğinde hectormapping metodu ile çıkarılan haritaların daha kaliteli olduğu tespit edilmiştir.

KAYNAKLAR

1. http://www.avenof.com/
2. http://www.ros.org/
3. http://wiki.ros.org/
4. https://opencv.org/
5. http://mesutpiskin.com/blog
6. https://www.solidworks.com/
7. http://market.samm.com
8. https://www.arduino.cc
9. http://www.robotiksistem.com
10. http://elektrikloji.blogspot.com.tr
11. http://yapbenzet.kocaeli.edu.tr

EK-1 ARDUİNO KODLARI

```
#include <ros.h>
#include <std_msgs/String.h>

ros::NodeHandle  nh;

std_msgs::String str_msg1;
ros::Publisher chatter1("tekerhiz", &str_msg1);

std_msgs::String str_msg2;
ros::Publisher chatter2("tekerencoderbilgileri", &str_msg2);


String istenenhiz="";
void messagehiz( const std_msgs::String& toggle_msg){
  istenenhiz=String(toggle_msg.data);

}

ros::Subscriber<std_msgs::String> sub1("istenenhiz", messagehiz );

String istenenyon="";
void messageyon( const std_msgs::String& toggle_msg){
  istenenyon=String(toggle_msg.data);
}

ros::Subscriber<std_msgs::String> sub2("istenenyon", messageyon );


int tursol = 0;//Kesme geldiği an
long zamansol=0;//Süre
long oncekizamansol=0;//Süre
```

```
float surefarkisol=0;//Süre farki
float rmpsol=0;//hesaplanan Rpm
float hizsol=0;//hesaplanan hiz


int gelen=0;//
int i =0;

int analoghiz=0;

int msol1=7;
int msol2=6;

int msag1=5;
int msag2=4;

int enbsol=10;//
int enbsag=9;


float zamansagyoneksi=0;
int ilerisag=0;
int gerisag=0;

float zamansolyoneksi=0;
int ilerisol=0;
int gerisol=0;

String readString;

////////Sag taraf
int tursag=0;//sag tekerin turu
long zamansag=0;
```

```
long oncekizamansag=0;
float rpmsag=0;
float surefarkisag=0;
float hizsag=0;

//Tekerlerin Gorevleri
int solgorev=0;
int saggorev=0;

int sayacsol=0;
int sayacsag=0;
float hizsoltakili=0;
float hizsagtakili=0;

int tursag2=0;
long zamansag2=0;
long oncekizamansag2=0;

float rmpsag2=0;
float hizsag2=0;
float surefarkisag2=0;

int tursol2=0;
long zamansol2=0;
long oncekizamansol2=0;

float rmpsol2=0;
float hizsol2=0;
float surefarkisol2=0;



float asilhizsol=0;
float asilhizsag=0;
```

```
String gonderilecekhizlar="";
int bakim=0;

float solortalamahiz=0;
float sagortalamahiz=0;

unsigned long toplamsolencoder=0;
unsigned long toplamsagencoder=0;

float donusmiktari;

void setup()
{

//Serial.begin(9600);**********************************************************************
**********************
    pinMode(13, OUTPUT);
    nh.initNode();
    nh.advertise(chatter1);
     nh.advertise(chatter2);
    nh.subscribe(sub1);
   nh.subscribe(sub2);
  attachInterrupt(0,rpmolcsol, RISING);
   attachInterrupt(1,rpmolcsol2, RISING);
    attachInterrupt(2,rpmolcsag, RISING);
   attachInterrupt(3,rpmolcsag2, RISING);


  pinMode(msol1, OUTPUT);
  pinMode(msol2,OUTPUT);

  pinMode(msag1, OUTPUT);
  pinMode(msag2,OUTPUT);
```

```
  pinMode(enbsol, OUTPUT);
  pinMode(enbsag, OUTPUT);

}

void loop()
{

  /*
//////////////////////////////////////////////////////////////////////////////////////////////////
********
  while (Serial.available())
  {
    if(i==0)
    {
      readString="";
    }
    delay(3);
    char c = Serial.read();
    readString += c;
    i++;
  }

*//////////////////////////////////////////////////////////////////////////////////////////////////
**************
  //analoghiz=readString.toInt();



//istenenhiz=readString;**************************************************************************
********************************

   parcala(istenenhiz);
```

```
  istenenhizsol(solgorev);
  istenenhizsag(saggorev);


//float asilhizsol=0;
//float asilhizsag=0;

asilhizsol=(hizsol+hizsol2)/2;
asilhizsag=(hizsag+hizsag2)/2;

if (millis()-zamansag>2)
{
  asilhizsag=0;
}
if (millis()-zamansol>2)
{

  asilhizsol=0;
}

solortalamahiz+=asilhizsol;
sagortalamahiz=+asilhizsag;

if(bakim==100)
{
  bakim=0;
  ///m2018
/*
String deneme=istenenyon.substring(0);
if(deneme.toInt()==1)
{
toplamsolencoder+=solencoder;
}
if(deneme.toInt()==2)
```

```
{
toplamsolencoder-=solencoder;
}

//////////////////////
deneme=istenenyon.substring(2);
if(deneme.toInt()==1)
{
toplamsagencoder+=sagencoder;
}
if(deneme.toInt()==2)
{
toplamsagencoder-=sagencoder;
}*/

String yonsag="+";
if(ilerisag>gerisag)
{
  yonsag="+";
  ilerisag=0;
  gerisag=0;

}
else if (ilerisag<gerisag)
{
  yonsag="-";
  ilerisag=0;
  gerisag=0;
}
String yonsol="+";
if(ilerisol>gerisol)
{
  yonsol="+";
  ilerisol=0;
```

```
  gerisol=0;
}
else if (ilerisol<gerisol)
{
  yonsol="-";
  ilerisol=0;
  gerisol=0;
}

gonderilecekhizlar=String(yonsol)+""+String(int(solortalamahiz))+"x"+String(yonsag)+""+St
ring(int(solortalamahiz));
char cpass[gonderilecekhizlar.length()+1];
gonderilecekhizlar.toCharArray(cpass,gonderilecekhizlar.length()+1);
 str_msg1.data = gonderilecekhizlar.c_str();

  //str_msg.data="+10x+5";
  chatter1.publish( &str_msg1 );

if(istenenyon=="+x+")
{
digitalWrite(13,HIGH);
}

String
tekerencoder=String(toplamsolencoder)+"x"+String(toplamsagencoder)+"x"+String(yonsol)+
"x"+String(yonsag);
char cpass2[tekerencoder.length()+1];
tekerencoder.toCharArray(cpass2,tekerencoder.length()+1);
 str_msg2.data = tekerencoder.c_str();

  //str_msg.data="+10x+5";
  chatter2.publish( &str_msg2 );
```

```
  nh.spinOnce()
  solortalamahiz=0;
  sagortalamahiz=0;
}
bakim++;
  //delay(100); Daha sonra delaya bak

  //Serial.println(String(hizsol)+"-"+String(hizsag));
  ////////////////////////////Sol
  /*Serial.print("analog hiz = ");
  Serial.print(analoghiz);
  Serial.print("\t");
  Serial.print("rpm = ");
  Serial.print(rmpsol);
  Serial.print("\t");
  Serial.print("m/s = ");
  Serial.println(hizsol);*/

  ////////////////////////////Sag
  /*Serial.print("analog hiz = ");
  Serial.print(analoghiz);
  Serial.print("\t");
  Serial.print("rpm = ");
  Serial.print(rpmsag);
  Serial.print("\t");
  Serial.print("m/s = ");
  Serial.println(hizsag);*/
  ////////////////

  i=0;
}

int index1=0;
int index2=0;
```

```
int index3=0;
void parcala(String sagsolgorevler)
{
   String str1, str2;

   index1 = sagsolgorevler.indexOf("a");
   index2 = sagsolgorevler.indexOf("b");
   index3 = sagsolgorevler.indexOf("c");

 solgorev=0.0;
   saggorev=0.0;


   if(index1!=-1 && index2!=-1 && index3!=-1){

   str1 = sagsolgorevler.substring(index1 + 1, index2);
   str2 = sagsolgorevler.substring(index2 + 1, index3);
    solgorev = str1.toInt();
   saggorev = str2.toInt();
  }


   index1=-1;
index2=-1;
index3=-1;
  }

void istenenhizsol(int istenenhiz)
{

 if(istenenhiz<=0)
 {
   digitalWrite(msol1, LOW);
   digitalWrite(msol2, HIGH);
```

```
   }
   else
   { digitalWrite(msol1, HIGH);
     digitalWrite(msol2, LOW);
    }
  istenenhiz=abs(istenenhiz);
  analoghiz=soltekerhiz(istenenhiz);
  analogWrite(enbsol, analoghiz);
  }

void istenenhizsag(int istenenhiz)
{

  if(istenenhiz<=0)
  {
    digitalWrite(msag1, LOW);
    digitalWrite(msag2, HIGH);
   }
   else
   { digitalWrite(msag1, HIGH);
     digitalWrite(msag2, LOW);
   }

  istenenhiz=abs(istenenhiz);

  analoghiz=sagtekerhiz(istenenhiz);

  analogWrite(enbsag,analoghiz);
}



int pwmsol=0;
int soltekerhiz(float istenen)
{
```

```
 if (istenen==0)
 {
  pwmsol=0;
 }
 if(istenen>hizsol*100)
 {
   pwmsol+=2;
   if(pwmsol>145)
   {
    pwmsol=pwmsol-2;
   }
 }
  else if(istenen<hizsol*100)
  {
   pwmsol-=2;
    if(pwmsol<0)
    {
     pwmsol=0;
     }
  }
  if (istenen==0)
 {
  return pwmsol;
 }
 else{
 return pwmsol+110;
 }
}


int pwmsag=0;
int sagtekerhiz(float istenen)
{
```

```
 if (istenen==0)
 {
  pwmsag=0;
 }
 if(istenen>hizsag*100)
 {
   pwmsag+=2;
   if(pwmsag>145)
   {
    pwmsag=pwmsag-2;
   }
 }
  else if(istenen<hizsag*100)
  {
   pwmsag-=2;
    if(pwmsag<0)
    {
     pwmsag=0;
     }
  }
  if (istenen==0)
 {
  return pwmsag;
 }
 else{
 return pwmsag+110;
 }
}



void rpmolcsag()
{
```

```
 toplamsagencoder++;

  tursag++;
  zamansagyoneksi=zamansag;
  zamansag=millis();

 if(abs(zamansag-zamansag2)<abs(zamansag2-zamansagyoneksi))
  {
    ilerisag++;

  }
  else if(abs(zamansag-zamansag2)>abs(zamansag2-zamansagyoneksi))
  {

    gerisag++;
  }

  if (tursag==119)
  {

    tursag=0;

    ////////////////////////////
    surefarkisag=zamansag-oncekizamansag;   //119 pals suresi
    oncekizamansag=zamansag;
    ////////////////////////////
    rpmsag=6000/surefarkisag;  //rpm
    hizsag=53.38/surefarkisag;    //hiz

  }

}
```

```
void rpmolcsag2()
{
  tursag2++;
  zamansag2=millis();
  if (tursag2==119)
  {
    tursag2=0;

    //////////////////////////
    surefarkisag2=zamansag2-oncekizamansag2;   //119 pals suresi
    oncekizamansag2=zamansag2;
    //////////////////////////
    rmpsag2=6000/surefarkisag2;  //rpm
    hizsag2=53.38/surefarkisag2;    //hiz

  }

}

void rpmolcsol()
{
  toplamsolencoder++;

  tursol++;
  zamansolyoneksi=zamansol;
  zamansol=millis();

if(abs(zamansol-zamansol2)>abs(zamansol2-zamansolyoneksi))
  {
    ilerisol++;
  }
  else if(abs(zamansol-zamansol2)<abs(zamansol2-zamansolyoneksi))
```

```
  {
   gerisol++;
  }

  if (tursol==119)
  {

   tursol=0;
   ////////////////////////////
   surefarkisol=zamansol-oncekizamansol;   //119 pals suresi
   oncekizamansol=zamansol;
   ////////////////////////////
   rmpsol=6000/surefarkisol;  //rpm
   hizsol=53.38/surefarkisol;    //hiz
  }
}

void rpmolcsol2()
{
  tursol2++;
  zamansol2=millis();

  if (tursol2==119)
  {
   tursol2=0;

   ////////////////////////////
   surefarkisol2=zamansol2-oncekizamansol2;   //119 pals suresi
   oncekizamansol2=zamansol2;
   ////////////////////////////
   rmpsol2=6000/surefarkisol2;  //rpm
   hizsol2=53.38/surefarkisol2;    //hiz

  }
```

EK-2 HECTOR MAPPING

```
<?xml version="1.0"?>
<launch>
  <!--node pkg="tf" type="static_transform_publisher" name="link1_broadcaster" args="0 0 0 0 0 0 base_link laser 100" /-->
  <!--node pkg="tf" type="static_transform_publisher" name="link1_broadcaster" args="0 0 0 0 0 0 odom base_link 100" /-->
  <node pkg="tf" type="static_transform_publisher" name="map_odom_broadcaster1"
      args="0.36 0 0.1 3.14 0 0 base_link laser 100"/>
  <node pkg="hector_mapping" type="hector_mapping" name="hector_height_mapping" output="screen">

    <param name="scan_topic" value="scan" />
    <param name="base_frame" value="base_link" />
    <param name="odom_frame" value="odom" />

    <param name="output_timing" value="false"/>
    <param name="advertise_map_service" value="true"/>
    <param name="use_tf_scan_transformation" value="true"/>
    <param name="use_tf_pose_start_estimate" value="false"/>
    <param name="pub_map_odom_transform" value="true"/>
    <param name="map_with_known_poses" value="false"/>
    <param name="map_pub_period" value="0.5"/>
    <param name="update_factor_free" value="0.45"/>
    <param name="map_update_distance_thresh" value="0.02"/>
    <param name="map_update_angle_thresh" value="0.1"/>
    <param name="map_resolution" value="0.05"/>
    <param name="map_size" value="1024"/>
    <param name="map_start_x" value="0.5"/>
    <param name="map_start_y" value="0.5"/>
  </node>
</launch>
```

EK-3 GMAPPING

```
<?xml version="1.0"?>
<launch>
  <master auto="start"/>
  <param name="/use_sim_time" value="true"/>

  <!--node pkg="tf" type="static_transform_publisher" name="link1_broadcaster" args="0 0 0 0 0 0 base_link laser 100" />
  <node pkg="tf" type="static_transform_publisher" name="link2_broadcaster" args="0 0 0 0 0 0 odom base_link 100" />
  <node pkg="tf" type="static_transform_publisher" name="link3_broadcaster" args="0 0 0 0 0 0 map odom 100" /-->

  <!--- Run gmapping -->
  <node pkg="gmapping" name="slam_gmapping" type="slam_gmapping" output="screen">

    <remap from="scan" to="scan"/>
    <param name="base_frame" value="base_link" />

    <param name="map_update_interval" value="5.0"/>
    <param name="maxUrange" value="8.0"/>
    <param name="sigma" value="0.05"/>
    <param name="kernelSize" value="1"/>
    <param name="lstep" value="0.05"/>
    <param name="astep" value="0.05"/>
    <param name="iterations" value="5"/>     <param name="lsigma" value="0.075"/>
    <param name="ogain" value="3.0"/>
    <param name="lskip" value="0"/>     <param name="minimumScore" value="50"/>
    <param name="srr" value="0.1"/>
    <param name="srt" value="0.2"/>
    <param name="str" value="0.1"/>
    <param name="stt" value="0.2"/>
```

```
    <param name="linearUpdate" value="0.3"/>
    <param name="angularUpdate" value="0.4"/>
    <param name="temporalUpdate" value="3.0"/>
    <param name="resampleThreshold" value="0.5"/>
    <param name="particles" value="30"/>
    <param name="delta" value="0.01"/>
    <param name="xmin" value="-20"/>
    <param name="xmax" value="20"/>
    <param name="ymin" value="-20"/>
    <param name="ymax" value="20"/>
    <param name="llsamplerange" value="0.01"/>
    <param name="llsamplestep" value="0.01"/>
    <param name="lasamplerange" value="0.005"/>
    <param name="lasamplestep" value="0.005"/>
  </node>
</launch>
```

EK-4 ROBOT MODELİ

```
<?xml version="1.0"?>
<robot name="nevbot" xmlns:xacro="http://ros.org/wiki/xacro">

  <xacro:include filename="$(find nevbot_description)/urdf/model.gazebo"/>

  <xacro:property name="boy" value="0.8" />
  <xacro:property name="en" value="0.35" />
  <xacro:property name="yukseklik" value="0.1" />
  <xacro:property name="pi" value="3.1415" />
  <xacro:property name="teker_yaricap" value="0.1" />
  <xacro:property name="teker_en" value="0.05" />
  <xacro:property name="yaricap" value="0.07" />
    <xacro:property name="kolonx" value="0.03"/>
  <xacro:property name="kolony" value="0.03"/>
  <xacro:property name="kolonz" value="0.33"/>
  <material name="blue">
    <color rgba="0 0 0.8 1"/>
  </material>

  <material name="black">
    <color rgba="0 0 0 1"/>
  </material>

  <material name="white">
    <color rgba="1 1 1 1"/>
  </material>

  <!-- Fiziksel degerler blogunu makro olarak tanimladik -->
  <xacro:macro name="default_inertial" params="mass">
    <inertial>
      <mass value="${mass}" />
```

```
    <inertia ixx="0.4" ixy="0.0" ixz="0.0" iyy="0.4" iyz="0.0" izz="0.2" />
  </inertial>
 </xacro:macro>

 <link name="base_link">
  <visual>
    <geometry>
      <box size="${boy} ${en} ${yukseklik}"/>
    </geometry>
    <material name="blue"/>
  </visual>
 <collision>
  <geometry>
    <box size="${boy} ${en} ${yukseklik}"/>
  </geometry>
 </collision>
 <xacro:default_inertial mass="40"/>
 </link>



<xacro:macro name="onkolon" params="prefix reflect refrect2">
  <link name="${prefix}_onkolon">
       <visual>
    <geometry>
      <box size="${kolonx} ${kolony} ${kolonz}"/>
    </geometry>
      <origin xyz="0 0 0" rpy=" 0 0 0"/>
    <material name="blue"/>
      </visual>

  <collision>
    <geometry>
      <box size="${kolonx} ${kolony} ${kolonz}"/>
```

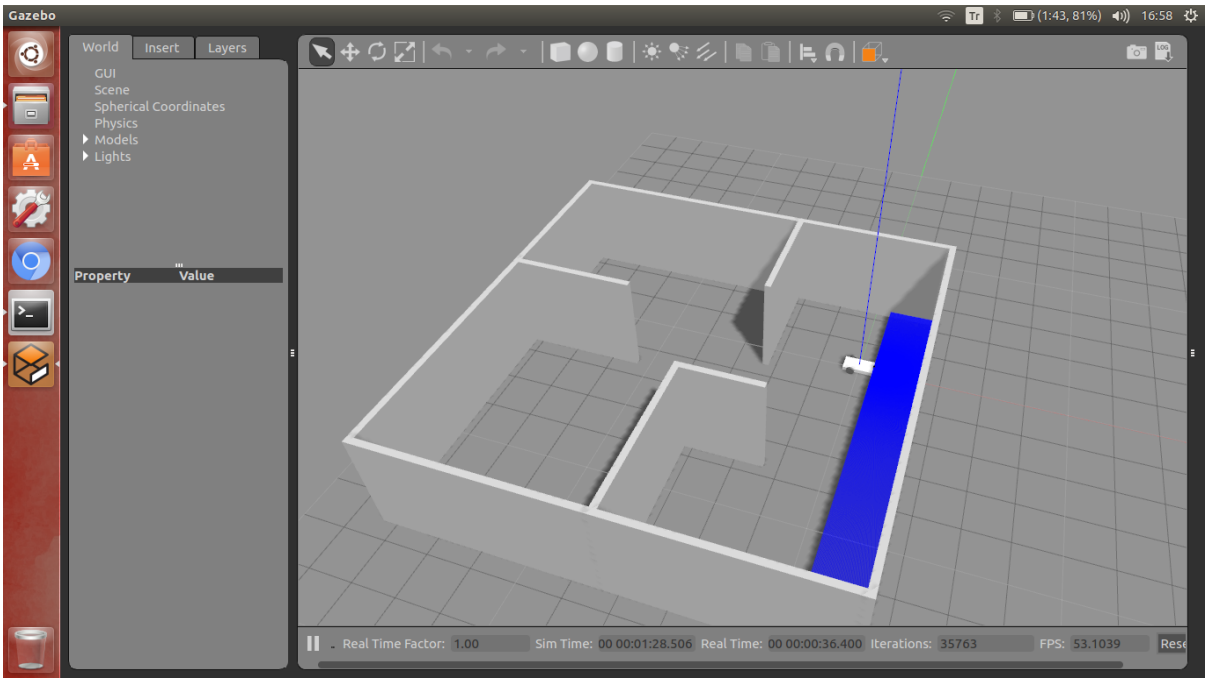
```
      </geometry>
          <origin xyz="0 0 0" rpy=" 0 0 0"/>
    </collision>


<xacro:default_inertial mass="1"/>


    </link>
      <gazebo reference="${prefix}_onkolon">
    <material>Gazebo/Blue</material>
      </gazebo>
    <joint name="base_to_${prefix}_onkolon" type="fixed">
      <parent link="base_link"/>
      <child link="${prefix}_onkolon"/>
       <origin xyz="${-0.18*reflect} ${-0.08*refrect2} 0.2" rpy=" 0 0 0"/>
    </joint>
</xacro:macro>

<xacro:onkolon prefix="onsag" reflect="1" refrect2="1"  />
<xacro:onkolon prefix="onsol" reflect="-1" refrect2="1"  />
<xacro:onkolon prefix="arkasag" reflect="1" refrect2="-1"  />
<xacro:onkolon prefix="arkasol" reflect="-1" refrect2="-1"  />

  <!-- Tekerler icin makro blogu tanimladik -->
  <xacro:macro name="wheel" params="prefix reflect">
    <link name="${prefix}_wheel">
      <visual>
        <origin xyz="0 0 0" rpy="${pi/2} ${pi/2} 0" />
        <geometry>
          <cylinder radius="${teker_yaricap}" length="${teker_en}"/>
        </geometry>
        <material name="black"/>
      </visual>
```

```
    <collision>
      <origin xyz="0 0 0" rpy="${pi/2} ${pi/2} 0" />
      <geometry>
        <cylinder radius="${teker_yaricap}" length="${teker_en}"/>
      </geometry>
    </collision>
    <xacro:default_inertial mass="10"/>
  </link>
  <joint name="base_to_${prefix}_wheel" type="continuous">
    <axis xyz="0 1 0" rpy="0 0 0" />
    <parent link="base_link"/>
    <child link="${prefix}_wheel"/>
    <origin xyz="-0.2 ${0.18*reflect} 0" rpy="0 0 0"/>
  </joint>
  <gazebo reference="${prefix}_wheel">
    <mu1 value="0.5"/>
    <mu2 value="0.5"/>
    <kp value="10000000.0" />
    <kd value="1.0" />
    <material>Gazebo/Grey</material>
  </gazebo>
</xacro:macro>

<xacro:wheel prefix="right" reflect="-1"/>
<xacro:wheel prefix="left" reflect="1"/>

<!-- Sarhos tekerler icin makro blogu tanimladik -->
<xacro:macro name="front_wheel" params="prefix reflect">
  <link name="front_${prefix}_wheel">
    <visual>
      <origin xyz="0 0 0" rpy="${pi/2} ${pi/2} 0" />
      <geometry>
        <sphere radius="${yaricap}"/>
      </geometry>
```

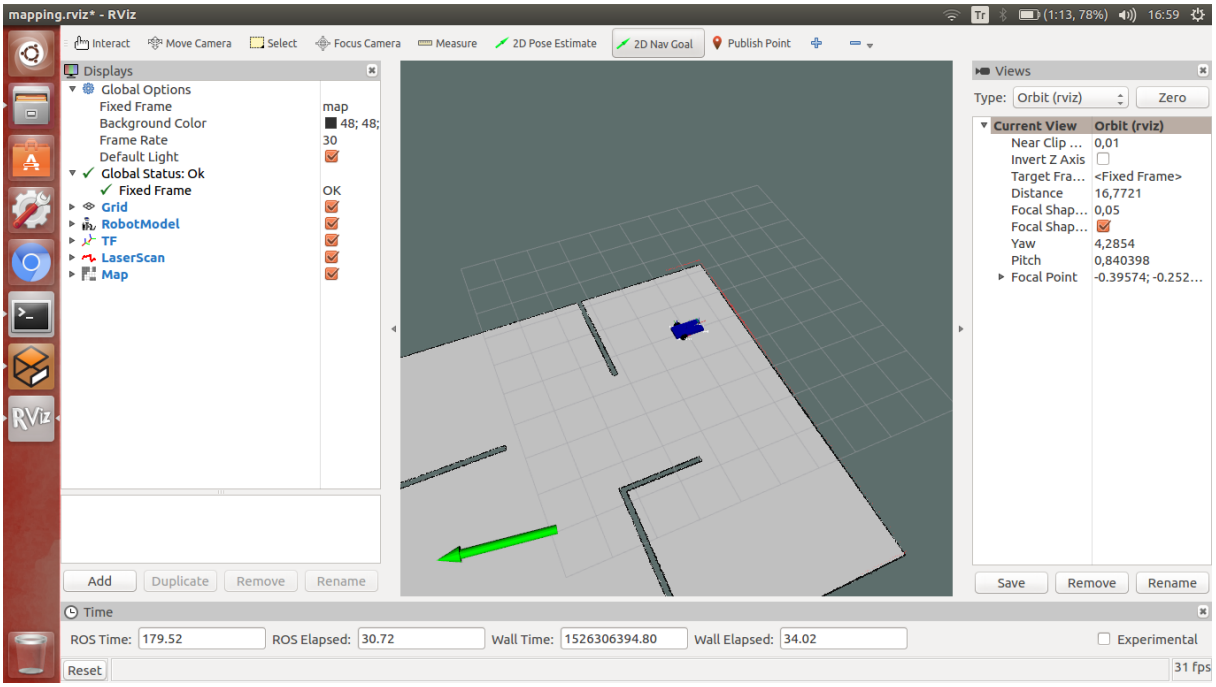
```
      <material name="white"/>
    </visual>
    <collision>
      <origin xyz="0 0 0" rpy="${pi/2} ${pi/2} 0" />
      <geometry>
        <sphere radius="${yaricap}"/>
      </geometry>
    </collision>
    <xacro:default_inertial mass="5"/>
  </link>
  <joint name="base_to_front_${prefix}_wheel" type="continuous">
    <axis xyz="0 1 0" rpy="0 0 0" />
    <parent link="base_link"/>
    <child link="front_${prefix}_wheel"/>
    <origin xyz="0.2 ${0.1*reflect} -0.03" />
  </joint>
  <gazebo reference="front_${prefix}_wheel">
    <mu1 value="0.0"/>
    <mu2 value="0.0"/>
    <kp value="10000000.0" />
    <kd value="1.0" />
    <material>Gazebo/Grey</material>
  </gazebo>
</xacro:macro>

<xacro:front_wheel prefix="right" reflect="-1"/>
<xacro:front_wheel prefix="left" reflect="1"/>

<!-- Hokuyo Laser -->
<link name="laser">
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
  <box size="0.1 0.1 0.1"/>
```

```
    </geometry>
   </collision>

   <visual>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
     <mesh filename="package://nevbot_description/meshes/hokuyo.dae"/>
    </geometry>
   </visual>
   <inertial>
    <mass value="1e-5" />
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <inertia ixx="1e-6" ixy="0" ixz="0" iyy="1e-6" iyz="0" izz="1e-6" />
   </inertial>
  </link>

  <joint name="laser_joint" type="fixed">
   <parent link="base_link"/>
   <child link="laser"/>
   <axis xyz="0 0 0" />
   <origin xyz="0.35 0 0.085" rpy="0 0 0"/>
  </joint>

  <gazebo>
   <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
    <robotNamespace>/</robotNamespace>
   </plugin>
  </gazebo>
</robot>
```

EK-5 ROBOTUN SIMULASYON RESİMLERİ



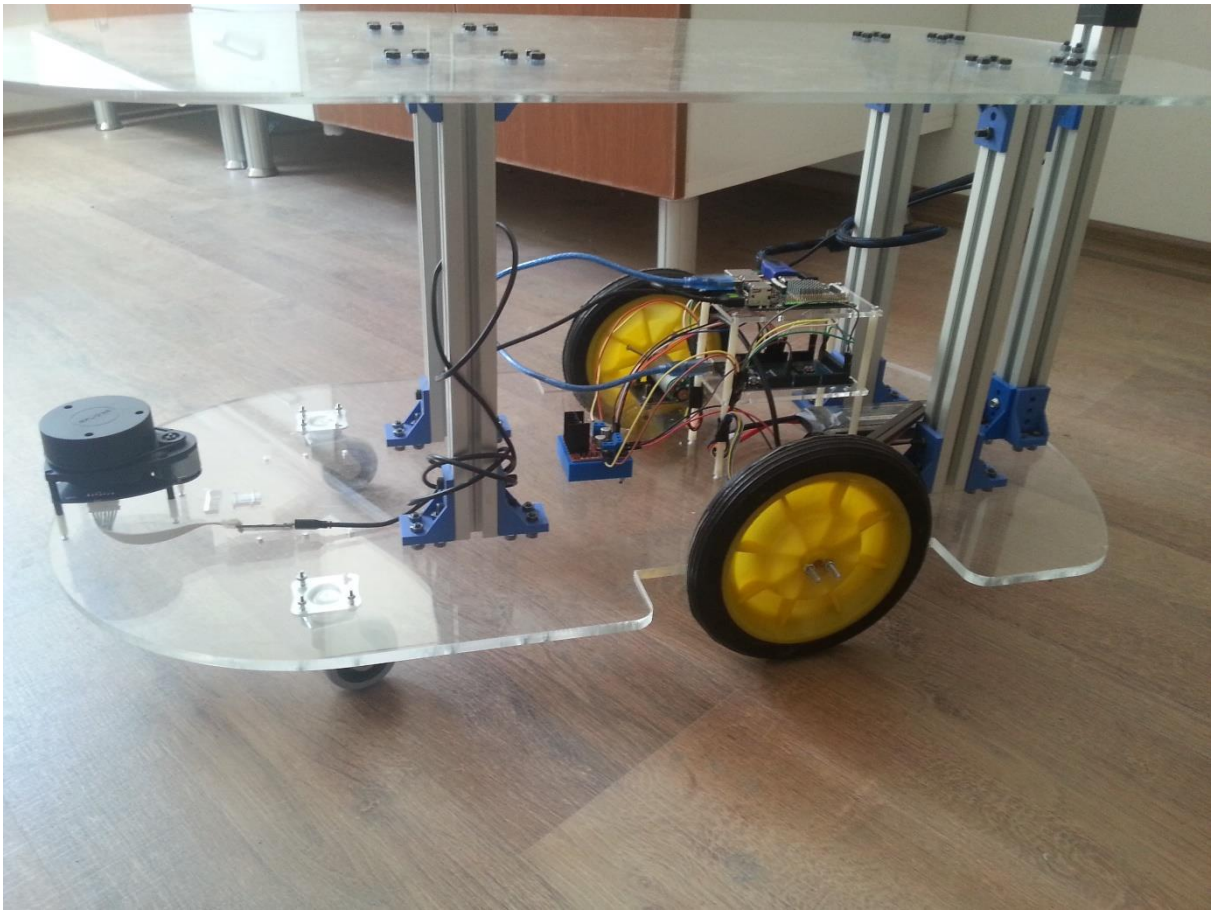
ŞEKİL EK-1: Robotun gazebo modeli



ŞEKİL EK-2: Robotun rviz modeli

EK-6 ROBOTUN GERÇEK RESİMLERİ

ŞEKİL EK-3: Robotun üstten görünümü



Şekil EK-4: Robotun yandan görünümü