MOBİL ROBOTLAR İÇİN GÖMÜLÜ SİSTEM TASARIMI

YÜKSEK LİSANS TEZİ

Hasan Selçuk YALÇIN
(141134113)

Anabilim Dalı: Mekatronik Mühendisliği
Programı: Elektronik Sistemler
Danışman: Doç. Dr. Ömür AYDOĞMUŞ

OCAK - 2019

T.C
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MOBİL ROBOTLAR İÇİN GÖMÜLÜ SİSTEM TASARIMI

YÜKSEK LİSANS TEZİ

Hasan Selçuk YALÇIN

(141134113)

Tezin Enstitüye Verildiği Tarih : 24 Aralık 2018
Tezin Savunulduğu Tarih : 24 Ocak 2019

Tez Danışmanı : Doç. Dr. Ömür AYDOĞMUŞ (F.Ü)

Diğer Jüri Üyeleri : Doç. Dr. Mehmet ÇAVAŞ (F.Ü)

Dr. Öğr. Üyesi Reşat ÇELİKEL (B.Ü)

OCAK-2019

ÖNSÖZ

Bu tezin tasarımında ve gerçekleştirilmesinde çalışmalarımı yönlendirerek her aşamada bilgi ve motivasyon desteğini eksik etmeyen tez danışmanım Sayın Doç. Dr. Ömür AYDOĞMUŞ'a, Sayın Arş. Gör. Mehmet Rıza SARAÇ'a, Sayın Arş. Gör. Mustafa Can BİNGÖL'e ve hayatımın her safhasında bana destek olan aileme teşekkür ederim.

Hasan Selçuk YALÇIN
ELAZIĞ - 2019

İÇİNDEKİLER

Sayfa No

ÖZET

Son zamanlarda otonom ve yarı otonom mobil robotlar üzerine yapılan araştırmalar hız kazanmıştır. Her geçen gün robotlar daha akıllı sistemler haline dönüşmektedir. Robotik uygulamalara otonom özelliğinin eklenmesi ile sistemlerin kullanım esnekliği önemli ölçüde artmaktadır. Bu tez çalışmasında, yarı otonom bir mobil robotun istenilen komutu yerine getirerek arzu edilen noktaya gelmesi amaçlanmıştır. Mobil robotun kendi kendine karar vermesi sayesinde robotun engellerden kaçınarak, nesnelere ve kendine zarar vermeden en kısa rotadan gitmesi ve üzerindeki sensörler ile istenilen bilgilerin alınması hedeflenmiştir. Tasarlanan mobil robot platformu ARM tabanlı bir kontrolör kartı ile kontrol edilmiş ve robot işletim sistemi olarak bilinen (ROS) açık kaynak kodları kullanılmıştır. Bu sayede esnek programlama kabiliyetine sahip bir robot platformu oluşturulmuştur.

Anahtar Kelimeler: Robot İşletim Sistemi, Derinlik Kamerası, Otonom Robot, Mobil Robot, Görüntü İşleme

SUMMARY

DESIGN OF AN EMBEDDED SYSTEM FOR MOBILE ROBOTS

Recently, the popularity of researches on autonomous and semi-autonomous mobile robots are increasing. Day by day, the robots have become more intelligent systems. The flexibility of the systems is significantly increased with the autonomy feature in robotic applications. In this thesis, semi-autonomous mobile robot is aimed to achieve the desired point by performing the desired command. It is aimed to go the shortest path without damaging objects and itself by avoiding obstacles on its operation path thanks to the self-determination feature of the mobile motor. Additionally, Sensors of the robot have provided the desired information. The purposed robot has been controlled by an ARM-based controller card. The open source codes known as Robot Operating System (ROS) have been used. In this way, the robot platform having a flexible programming capability has been obtained in this work.

Keywords: Robot Operating System, Depth Camera, Autonomous Robot, Mobile Robot, Image Processing

ŞEKİLLER LİSTESİ

Sayfa No

KISALTMALAR LİSTESİ

ROS	**:** Robot Operating System (Robot İşletim Sistemi)
RISC	**:** Reduced Instruction Set Computer (İndirgenmiş Komut Takımıyla Hesaplama)
ARM	**:** Advenced RISC Machine (Acorn RISC Machine)
PC	**:** Personal Computer (Kişisel Bilgisayar)
BSD	**:** Berkeley Software Distribution (Berkeley Yazılım Dağıtımı)
PMDC	**:** Permanent Magnet Direct Current (Daimi Mıknatıslı Doğru Akım)

1. GİRİŞ

Bilgisayar Mühendisliği, Elektronik Mühendisliği, Kontrol Mühendisliği ve Makine Mühendisliği dallarının ortak olarak çalışması sonucunda disiplinler arası bir uygulama alanı olan Robotik elde edilmektedir. Robotlar bir algoritma temeline dayanan ve bu algoritma ile yönetilen kontrollü ve yeniden programlanabilen makinelerdir. Günümüz robot teknolojisi, diğer teknolojik gelişmelere paralel olarak bilimsel ve teknolojik olgunluğunu henüz tamamlamamış bir teknolojidir. Robot teknoloji her geçen gün hızlı bir şekilde ilerlemeye devam etmektedir.

Teknolojinin gelişmesi ile birlikte robotların endüstriyel uygulamalarda ve günlük yaşantıdaki yeri gittikçe artmaktadır [1-4]. Bu tür çalışmaların insan sağlığı açısından tehlikeli olabilecek birçok alanda kullanılması önemlidir [5- 7]. Robotlar çalışma şekline göre ya hareketli (mobil) ya da bir çalışma platformuna sabitlenerek çalıştırılmaktadırlar. Mobil robotlar birçok farklı çevre koşullarında çalışabilen robotlardır. Özellikler karada, suda ve havada hareket edebilmeleri için çeşitli istekleri gerçekleştirebilmek için tasarlanmaktadırlar. Mobil robotlar tekerlek/palet veya pervane kullanılarak hareketlerini sağlayabilmektedirler. Mobil robotlar standart bir endüstriyel amaçlı manipülatörden farklı olarak çevresel şartlara bağlı olarak hareket etmeleri gerektiğinden dolayı daha karmaşık bir yapıya ve kontrol algoritmasına ihtiyaç duyarlar. Bunun üstesinden gelebilmek için çevreden birçok bilgi toplamaları gerekmekte ve bu bilgileri işlemeleri gerekmektedir. Çevresel bilgiler kamera, radar, lidar, sonar, lazer, IMU, GPS gibi farklı sensörlerden toplanarak merkezi bir birimde anlamlandırılması gerekmektedir. Çünkü bu sensörlerden gelen bilgiler ham bilgilerdir. Bundan dolayı hızlı işlem yapabilen işlemci alt yapılarına ihtiyaç duymaktadırlar. Mobil robotların asıl amacı insanların gidemedikleri veya gitmelerinin tehlikeli olabileceği ortamlarda uzaktan kontrol edilerek arzu edilen işleri yerine getirmektir. Örneğin insanoğlunun Mars gibi uzak mesafelere gidebilmesi henüz mümkün olmadığı için dünyadan kontrol edilebilen mobil robotlar ile Mars yüzeyinde keşif görevlerini yerine getirebilmeleri için tasarlanabilmektedir. Ayrıca, insanlar için tehlikeli olabilecek her ortam için farklı tasarlanarak uzaktan kontrolleri sağlanabilmektedir. Mobil robotlar kendi başlarına karar verebilecekleri gibi uzaktan insan tarafından da kontrolü sağlanabilir. Otonom veya insan tarafından kontrol edildiğinde sensör desteğine ihtiyaç duyulmaktadırlar.

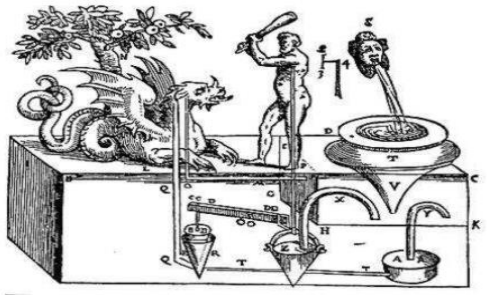
Bu tezde, 4 tekerlekli bir robot platformu için yazılım geliştirilmiştir. Yazılım temel olarak iki kısım olarak geliştirilmiştir. Birinci aşamada; motorları sürmek için ROS ile haberleşebilen bir mikrodenetleyici programı gerçekleştirilmiştir. Geliştirilen mikrodenetleyici programı ROS tarafından gelen komutları anlamlandırarak motorların kapalı çevrim altında kontrol edilmesini sağlamaktadır. İkinci aşamada ise ROS işletiminin çalıştırılması için bir ARM çekirdekli onboard PC kullanılmıştır. İşletim sistemi olarak Linux-16.04.2 (Xenial) tercih edilmiş ve ROS için Kinetic sürümü kullanılmıştır. Geliştirilen yazılımlar gerçek robot platformunda test edilerek arzu edilen sonuçlar elde edilmiştir.

1.1. Robot Tarihi

Robotların tarihinin kökeni antik dünyadadır. Modern konsept, karmaşık mekaniğin kullanımına izin veren Endüstri Devrimi'nin başlaması ve ardından elektriğin ortaya çıkmasıyla gelişmeye başlamıştır. Bu durum, küçük kompakt motorlu makinelere güç vermeyi mümkün kılmıştır. Robotun kelime olarak ilk kullanımı yeni olsa da kavram açısından geçmiş tarihte ilk olarak Homeros tarafından kaleme alınan İlyada eserinde üç ayaklı mekanizmaya sahip hareketli cihazlar olarak bahsedilmiştir.

Aristo tarafından M.Ö. 4.YY ’da yazılan eserde insanlar yerine çalışabilecek mekanizmalardan bahsedilmiştir. Böylece robot ile ilgili düşüncenin olduğu, bilinen ilk eseri Aristo yazmıştır.

M.Ö. 300'lü yıllarda İskenderiye'li Hero, Herkül tarafından bir ejderhanın okla öldürülmesini ifade eden ve Şekil 1.1’de gösterilen otomatı tasarlamıştır [8]. M.Ö. 250'de de başka bir mucit olan İskenderiye'li Ctesibius ise suyla çalışan bir saat mekanizması yapmıştır. Bu icatlar ilk nesil robotlar olarak sayılabilecek otomatlara önayak olmuştur. Bu otomatların birçoğu basit saat zembereği mekanizmasına sahip süs ve oyuncaklardır.



Şekil 1.1 İskenderiye'li Heron'un otomatı [8].

Robot kelimesi Çek dilinde, robota, "zorla çalıştırma" anlamına gelmektedir. “Robot” kelimesi ilk olarak Çekli yazar Karel Čapek tarafından 1920 ‘de yazılan ve 1921 ‘de sahnelenen R.U.R. (Rossumovi Univerzální Roboti - Rossum'un Evrensel Robotları) adlı oyunda kurgusal bir insanı ifade etmek için kullanılmıştır. Ancak kelimenin gerçek mucidi Karel'in kardeşi Josef Čapek'tir. Robotlarla ilgili başka bir kavram olan Elektronik, 1948 ’de İngiltere ’de William Gray Walter tarafından yapılan ilk elektronik özerk robotların ve 1940’ların sonunda John T. Parsons’un Bilgisayar Sayısal Kontrol (CNC) tezgâhlarının ortaya çıkmasıyla teknolojik gelişmenin itici gücü haline gelmiştir. İlk ticari, dijital ve programlanabilir robot, 1954 'te George Devol tarafından yapılmıştır ve Unimate olarak adlandırılmıştır. Bu robot 1961 'de General Motors'a satıldıktan sonra New Jersey Ewing Township'in West Trenton bölümündeki Inland Fisher Kılavuz Fabrikası'ndaki sıcak metal parçalarını döküm makinelerine kaldırmak için kullanılmıştır. Robotlarla ilgilenen bilim dalı “Robotik” olarak adlandırılır ve bu ifadeyi ilk kez Isaac Asimov kullanmıştır. Robotik; Bilgisayar Mühendisliği, Elektrik ve Elektronik Mühendisliği ve Makine Mühendisliği disiplinlerini kapsayan ortak bir çalışma alanıdır. Robotlar farklı işlerde insan odaklı işgücünün yerine geçerek faydalı bir amaç için iş ve değer üreten, yönetimi bir yazılım aracılığı ile gerçekleştirilen, belirli derecelerde iletişim ve karar verme ve iletişim kurma yeteneği olan ve ayrıca biçimsellik açısından canlı organizma gibi davranabilen makinelerdir. Isaac Asimov robotların gelecekte insanların hayatında büyük ölçüde yer alacağını düşündüğü için robot kavramında üç önemli kuram belirtmiştir. Bu kuramlar "Asimov Robot Kanunları" olarak adlandırılmıştır [9].

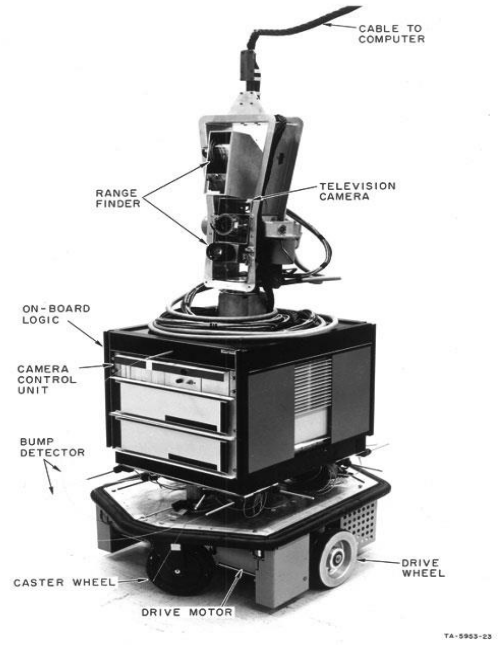
- Bir robot insana zarar veremez ya da bir insanın zarar görmesine müsaade etmez.
- Bir robot birinci kanuna aykırı bir davranış olmadığı sürece insanlar tarafından verilen tüm emirlere uymak zorundadır.
- Bir robot ilk 2 kanuna aykırı bir davranış olmadığı sürece kendi varlığını korur.

Daha sonra ise 0’ncı kuram eklenmiştir,

- Robotlar asla insanlara zarar vermemelidir.

Shakey Robot, kendi eylemleri hakkında düşünebilecek ilk genel amaçlı mobil robottur. Her robot için, daha büyük bir görevi yerine getirme adımında başka robotlara talimat verilmesi gerekmekle birlikte, Shakey komutları analiz edip bunları temel parçalara ayırabilen bir robot olmuştur. Shakey Projesi, doğası gereği robotik araştırma, bilgisayar

görüşü ve doğal dil işleme konularının bir araya gelmesi ile oluşmuştur. Bu nedenle, mantıksal akıl yürütme ve fiziksel eylemi bulunan ilk proje olmuştur. Şekil 1.2'de verilen Shakey otonom mobil robot, Stanford Araştırma Enstitüsü'nün Yapay Zeka Merkezi'nde (şu anda SRI International olarak adlandırılmıştır) geliştirilmiştir [10]. Projenin en dikkat çekici sonuçlarından bazıları A* arama algoritmasını, Hough dönüşümünü ve görünürlük grafik yöntemini içermesidir. Shakey, 1966-1972 yılları arasında Charles Rosen katkıları ile geliştirilmiştir. Katkıda bulunan diğer önemli kişiler ise Alfred Brain, Bertram Raphael, Nils Nilsson, Sven Wahlstrom, Richard Duda, Peter Hart, Richard Fikes, Richard Waldinger, Thomas Garvey, Jay Tenenbaum, Helen Chan Wolf ve Michael Wilber'dir. Bu proje İleri Savunma Araştırma Projeleri Ajansı (DARPA) tarafından desteklenmiştir [11].



Şekil 1.2 Shakey otonom mobil robot [10].

Robotun programlaması öncelikle LISP'de yapıldı. Kullandığı STRIPS planlayıcısı, kullandığı yazılım için ana planlama bileşeni olarak tasarlandı. Otonom mobil araçların geliştirilmesi konusu yaşadığımız son elli yılın en önemli araştırma konuları arasında yer almıştır. Kara mayınlarının temizliği, uzayın keşfi gibi konularda araştırmayı, mekanik ve işlemsel gereksinimlere, algılamaya ve otonom karar verebilme mekanizması için gerekli zekâya yönlendirmiştir.

2. ROBOT ÇEŞİTLERİ

Robotlar kullanım alanlarına göre birçok farklı şekillerde ve boyutlarda üretilirler ve çok farklı fonksiyonlara sahiplerdir. Günümüzde Robotlar fabrika sahasından uzaklaşıp eğlence, oyuncak, kişisel hizmet, tıp ve cerrahi operasyon, askeri ve su altı ve uzay gibi tehlikeli çevrelerde kullanılabilir hale gelmiştir [12].

2.1. Endüstriyel Robotlar

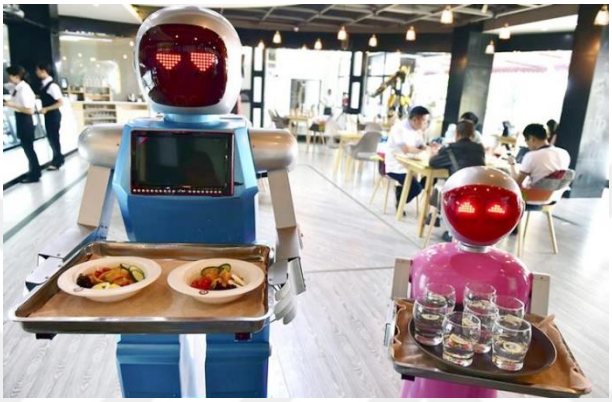
Endüstriyel robotlar otomatik olarak kontrol edilebilen, tekrar programlanabilen, 3 veya daha fazla sayıda eksende programlanabilen çok amaçlı manipülatörlerdir. Zaman, iş gücü ve maddi olarak kazanım sağlamasından dolayı günümüz endüstrisinde kullanımı gittikçe artmıştır [13]. Endüstriyel robotlar genellikle Şekil 2.1'de gösterildiği gibi otomotiv sektöründe kullanılmaktadır [14].

Şekil 2.1 Otomotiv sektöründe kullanılan manipülatörler [14].

2.2. Kişisel Hizmet Robotları

Hizmet Robotları insanlara yardım eden robotlardır. Tipik olarak tozlu, sıkıcı, mesafeli, tehlikeli veya tekrarlayan işlerde kullanılırlar. Onlar tipik olarak otonomdurlar ve bir kontrol sistemi tarafından çalıştırılırlar. Uluslararası Robot Federasyonu bu tür robotlar için geçici bir tanımlama ileri sürmüştür: "Bir hizmet robotu, üretim görevinin dışında, insanların yararına yarı otonom veya tam otonom şekilde hizmet edecek şekilde çalışan bir robot ve donanımdır. Son yıllarda, servis robotlarının sosyal çehresi, bu robotların yerleştirildikleri

bölgede sadece gezinmesinin değil aynı zamanda bu bölgedeki insanlarla iyi bir şekilde iletişim halinde olması ve bu insanlara yararlı hizmetler sunmasının beklenildiğini açıklığa kavuşturmuştur. "İnsan odaklı robotik ve insan" adlı çalışma böyle görevleri yerine getirebilmeyi amaçlar ve bu çalışma hareketi robotikte en ilgi çekici alanlardandır. Genellikle, bir servis robotu, dikkatini insanlara vermelidir ve etrafında insan olması durumunda bu insanların varlığının farkında olmalıdır [15]. Şekil 2.2'de hizmet vermekte olan robotlar gösterilmiştir [16].



Şekil 2.2 Hizmet servis robotları [16].

2.3. Medikal Robotlar

Medikal robotlar cerrahların daha az invazif ve daha kesin metotların kullanıldığı operasyonlara daha etkili bir şekilde giriş yapmalarını sağlar. Bu robotlar bir taraftaki cerrahın eylemlerini kullanarak diğer taraftaki efektörün kontrolünü sağlarlar. Sağlık hizmetlerindeki teknolojik yenilikler, maliyetin büyümesinde önemli faktörlerdendir. Doktorlar ve hastalar, henüz erdemleri ve zayıflıkları tam olarak anlaşılamadan sık sık yeni tedavi yöntemleri benimserler. Bu teknolojik yeniliklerin kullanıldığı tedavi yöntemleri daha eski tedavi yöntemlerine göre daha pahalı olduğundan veya bu yöntemler tedavi edilen hastaların sayısında ve çeşitliliğinde büyümeye sebep olduğundan maliyetin artmasına sebep olabilir. Sağlık hizmetlerinde kullanılan robotlar, rehabilitasyon merkezleri, ameliyathane gibi farklı ortamlarda farklı görevler üzerine çalışmaktadırlar [17]. Telerobotik sistemler, ameliyatları gerçekleştirebilmek adına düzenli olarak kullanılırlar. Bu sayede daha kısa sürede bir iyileşme sürecine ve bazı prosedürlerde daha güvenilir sonuçlara olanak sağlarlar. Robotik rehabilitasyon sistemleri, başarılı bir şekilde fiziksel ve mesleki terapiler ortaya koyarlar. Böylelikle, hastanın ihtiyaçlarına sürekli olarak uyumluluk gösterebilen tedavi

üzerine daha da yoğunlaşmayı mümkün kılar [18]. Günümüzde cerrahi operasyonlarda en çok tercih edilen robot Şekil 2.3'te gösterilen da Vinci ameliyat robotudur [19].

Şekil 2.3 Da Vinci ameliyat robotu [19].

2.4. Askeri Robotlar

Pek çok askeri organizasyon şimdilerde askerlerin yapamayacağı riskli işleri yerine getirmede robotlardan yardım alırlar. Bu robotlar askeri organizasyonlara büyük katkı sağlarlar. Askeri robotlar insansız hava araçları, küçük insansız arazi araçları, çok fonksiyonlu lojistik ve ekipman sağlayıcı robotlar ve silahlandırılmış robotik araçlar gibi birkaç gruba ayrılmıştır [20]. Günümüzde birçok ülke bu teknolojilere çok büyük yatırımlar yapmaktadır. Şekil 2.4'te İngiltere'nin geliştirildiği MAARS (Modular Advanced Armed Robotic System) robotuna ait bir fotoğraf verilmiştir [21]. Bu robotun bataryası 3-12 saat arasında çalışabilmektedir. Saatte 11km hıza çıkabilmekte ve kontrol merkezinden yaklaşık 1 km uzağa gidebilmektedir. Silah sistemi, 120 lb (54 kg) kaldırabilen bir manipülatör kolu ile değiştirilebilir, böylece 155 mm topçu mermisi taşıyabilir ve 300 lb (140 kg) çekebilir.

Şekil 2.4 Modüler hareketli silahlandırılmış robot sistemi (MAARS) [21].

3. OTONOM ROBOTLAR

Bazı robotlar çevresi ile etkileşim kurabilir. Çevreden farklı nitelikteki bilgileri sensörler (duyarga) ile algılayıp bu bilgileri mikro denetleyicisinde çözümledikten sonra harekete geçen robotlara "otonom robotlar" denir. Otonom robotlar, üzerlerinde yüklü olan yazılım sayesinde görüntü, ses, ışık ve sıcaklık gibi faktörlere cevap verebilecek şekilde tasarlanır [22]. Günümüzde bilinen bazı otonom robotlar eklem bacaklı olarak geliştirilmektedir. Zürih Robotik Sistemler Laboratuvarında geliştiren tam otonom robot Şekil 3.1a'da gösterilmiştir [23]. Bu robotla eşdeğer olan bir başka robot ise Boston Dynamics firmasının geliştirdiği Şekil 3.1b'de gösterilen SpotMini adlı eklem bacaklı otonom robottur [24].

(a) (b)

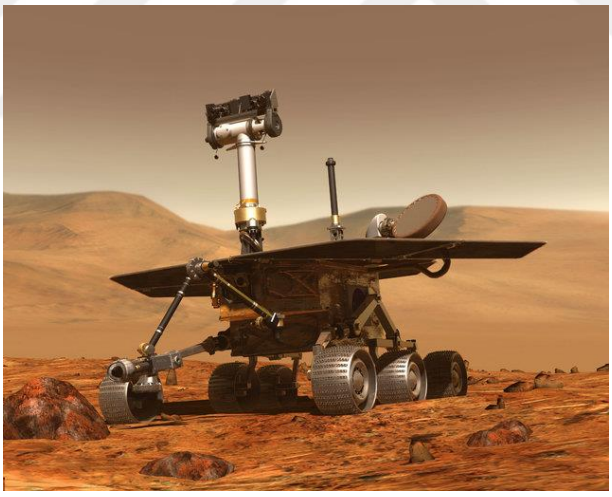
Şekil 3.1 a) Robot ANYmal [23], b) SpotMini [24].

ANYmal zorlu ortamlarda otonom çalışma için tasarlanmış dört ayaklı bir robottur. Uyumluluk düzeyi yüksek ve hassas tork kontrollü aktüatörler tarafından tahrik edilen bir sistemdir. Bu sayede dinamik koşu ve yüksek mobil tırmanma yeteneğine sahiptir. Entegre lazer sensörler ve kameralar sayesinde robot, sürekli olarak harita oluşturmak ve doğru bir şekilde konumlandırmak için çevresini algılayabilir. Bu bilgilere dayanarak, navigasyon yolunu bağımsız olarak planlayabilir ve yürürken ayaklarını dikkatli bir şekilde konumlandırabilir. İlk gerçek ortamdaki uygulaması olarak; petrol ve gaz santrallerinde denetim yapmak için teste edilmiştir. ANYmal'in bataryaları 2 saatten daha fazla çalışabilmektedir. Optik ve termal kameralar, mikrofonlar, gaz algılama sensörleri ve aktif aydınlatma gibi farklı duyusal ekipmanları bünyesinde barındırır. Bu yüklerle robot 30 kg'dan daha az ağırlığa sahiptir. Dolayısıyla tek bir operatör tarafından kolayca taşınabilir [25]. SpotMini, bir ofise veya eve rahatça uyan küçük, dört ayaklı bir robottur. 25 kg ağırlığındadır. 5-eksenli kol ile birlikte 30 kg ağırlığa sahiptir. SpotMini tamamen elektrikli ve yaptığı işe bağlı olarak yaklaşık 90 dakika çalışabilir. SpotMini, Boston Dynamics

firmasının tasarladığı en sessiz robottur. 5 eksenli kolu ve algı sensörlerini kullanarak nesneleri toplama ve işleme yeteneğine de sahiptir. Sensör paketi, stereo kameralar, derinlik kameraları, IMU ve bacaklarda pozisyon / kuvvet sensörleri içerir. Bu sensörler navigasyon ve mobil manipülasyon ile yardımcı olur [26].

3.1. Yarı Otonom Robotlar

Yarı otonom robotlar, bazı görevleri kendi başlarına yapabilen ancak temelde bir operatörün yönlendirmesine ihtiyaç duyan mobil robotlardır [27]. Sensörden gelen verilerin bir araya getirilmesi, çalışma esnasında kullanılabilecek özelliklerin bu verilerden elde edilmesi, temel hareket işlevlerinin gerçekleştirilmesi, acil bir durum veya arıza durumunda sistemin nasıl çalışması gerektiğinin belirlenmesi ve daha önceden standart olarak tasarlanmış kuralların uygulanması yarı otonom robotların yerine getirebileceği görevlerden bazılarıdır [28]. Yarı - otonom robotların başında NASA'nın geliştirmiş olduğu Şekil 3.2'de gösterilen Mars Exploration Rover Mission (MER) Mars keşif robotu yer alır [29].



Şekil 3.2 Mars keşif gezisi robotu [29].

3.2. Tam Otonom Robotlar

Tam otonom robotlar, verilen bir görevin başarı ile tamamlanması için alınacak kararların hepsini kendi başına alan ve yapılacak tüm işleri de yine kendi başına yapabilen robotlardır. Böyle bir senaryoda operatörün sadece yapması gereken robota görevini bildirmek veya tek bir görev için robotu o görev doğrultusunda programlamaktır. [30-33]. Bu robot özellikle fabrika içi parça sevkiyatından kullanılmaktadır. Şekil 3.3'te Skoda otomobil fabrikalarında

kullanılmakta olan Omron firmasının yapmış olduğu tam otonom robot gösterilmiştir [34]. Bu robot kendi başına haritalama ve navigasyon işlemlerini yapabilmektedir. Ayrıca bu robotlar birden fazla kullanılarak birbirleri ile koordineli çalışabilme yeteneğine sahiptir. Maksimum hızları yaklaşık 1.5 m/s'dir. Bu robotun modeline göre maksimum taşıma aralığı 60-130 kg civarındandır. 180 °/s gibi iyi bir hareketlilik yanı sıra ±100mm pozisyon doğruluğu sunar. Açısal doğruluğu ise ±2°'dir. Şekil 3.3'te gösterilen robot sonar sensör kullanarak engellerden kaçabilir. Ayrıca anlık olarak ön kısmında bulunan lazer sensörü sayesinde konumlama ve haritalama yapabilmektedir.

Şekil 3.3 Tam otonom mobil robot Omron [34].

4. ROBOT İŞLETİM SİSTEMİ

ROS, kelime olarak işletim sistemi gibi anlaşılsa da temelde bir işletim sistemi değildir. ROS, robot bileşenlerini bir bilgisayar aracılığı ile kontrol etmemize olanak sağlayan BSD (Berkeley Software Distribution) lisanslı bir yazılım sistemidir. ROS, açık kaynak kodlu bir yazılımdır ve onu kullanabilmek için Linux tabanlı bir işletim sistemi gerekmektedir. Bu yazılımın amacı robot ve programcı arasındaki bağlantıyı sağlamaktır. ROS sahip olduğu standart kütüphanelerin ve desteklediği aygıtların haricinde, geliştiriciler aracılığı ile yazılmış kütüphaneler ve sürücüleri de desteklediği için günden güne kullanım alanını arttırmakta ve robot dünyasındaki standart yerini almaktadır.

4.1. ROS'un Tarihçesi

ROS'un ilk kez geliştirme çalışmaları 2007 yılında Stanford Yapay Zekâ Laboratuvarı'nda başlamıştır. 2008 – 2013 yılları arasında Willow Garage enstitüsünde geliştirme çalışmaları devam etmiştir. Enstitüdeki mühendisler bu süreç zarfında yirmiden fazla kurum ile görüşerek işbirliği içerisine girmişlerdir. Bu kurumlar, ROS projesini geliştirebilmek için, donanım ekleme ve kod örnekleri ile katkıda bulunmaya başlamışlardır. Bundan sonraki süreçte ise robot üreten şirketler ürünlerini ROS'da kullanabilecekleri şekilde geliştirmeye başlamışlardır. 2013 yılının şubat ayında ROS projesinin yönetimini OSR Vakfı devraldı. 2013 yılı Ağustos ayında Suitable Technologies şirketi, Willow Garage enstitüsünü devraldı. Willow Garage tarafından oluşturulan PR2 robotunun destek hakları ise Clearpath Robotics tarafından alındı. Bütün bu değişim ve gelişmelerin akabinde robotlardaki aktüatörler ve sensörler ROS ile kullanılmak üzere tekrardan düzenlenmiştir. Günden güne ROS destekli aygıtların miktarı artmaktadır.

4.2. ROS Kullanılmasının Avantajları

ROS 'un tercih edilme sebebinin birçok cevabı bulunmaktadır. Kısa ve net olarak bir işletim sisteminden daha çok bir arayüz olarak tanımlanabilen bir uygulama olması söylenebilir. ROS'un sağladığı avantajlardan birisi de yazılan kodların her robot için farklı yazılması yerine tek seferde yazılan kodun desteklenen tüm aygıtlarda çalıştırılabilmesidir.

Böylelikle yazılan standart bir algoritma, standart bir robota hızlı bir şekilde yüklenerek hemen çalıştırılabilir durumda olacaktır. ROS ile bilgisayar arasındaki iletişim sayesinde robot üzerindeki sensörlerden alınan veriler bilgisayara ulaşacaktır ve burada işlendikten sonra veri robota tekrar görev olarak gönderilebilecektir. Haberleşme sistemi ROS arayüzündeki konular ve mesajlar aracılığı ile sağlanacaktır. ROS, Gazebo simülasyon ortamında tam uyumlu olarak çalışabilmektedir. Böylelikle Gazebo simülasyon ortamı, projenin prototipi esnasında ROS Package haline getirilen ve geliştirilen veri yapısını ölçmek ve bu ortamda hızlı bir gelişim sağlamak için kullanılabilir.

Diğer avantajları ise şu şekilde maddeler halinde belirtilmiştir.

- ROS'un amacı kodu desteklemek ve böylelikle tekrar kullanılabilirliği arttırmaktır. Bu durum her projede aynı şeyi tekrar üretme probleminin önüne geçer.
- İki işletim sistemi arasında çalışabilir. Örneğin; robot üzerinden oluşturulan ROS Master'a Windows üzerinde kurulu olan MATLAB'tan robota kod gönderilebilir ve robottan geriye veri döndürülebilir. Böylece ROS kullanıcıyı sadece Linux kullanmaya zorlamaz ve kullanıcının diğer işletim sistemlerinin sunduğu kolaylıklardan da yararlanabilmesini sağlar.
- Bağımsız bir programlama dili yapısına sahiptir. Python, java, lisp, C++ ve lua gibi pek çok farklı programlama dilini desteklemektedir ve kullanıcıyı belli bir programlama dilini öğrenmek zorunda bırakmaz.
- Açık kaynak kodlu bir yapıya sahiptir. Topluluk deposunda yer alan güçlü araçları kullanıcı istediği gibi kullanılıp yeniden düzenleyebilir. Bu araçlardan öne çıkanları hata ayıklama, simülatörler ve görselleştirme araçlarıdır.
- Desteklediği robot ve sensör sayısı günden güne artmaktadır.
- Farklı platformlar arası çalışabilir. Oluşturulan düğümler çeşitli programlama dilleri ile yazılabilir. Örneğin; Bir düğüm C++ dilinde yazılırken diğer düğüm python yada java dili ile yazılabilir.
- Modülerdir. Diğer robot uygulamalarının birçoğunda sorun çıktığında bu sorun ana kodu kitleyebilir ve robot uygulamasını durdurabilir. Ancak ROS'ta ise bu durum böyle değildir. Her işlem için ayrı bir düğüm olduğundan dolayı düğümün birinin bozulması halinde diğer düğümler işlevselliğini sürdürür.
- Karmaşıklığı azaltmak ve bu sayede hata ayıklamayı kolaylaştırabilmek adına her bir işlem için ayrı konular yayınlanır.
- Aktif bir topluluğa sahip olduğundan dolayı ROS ile ilgili birçok kitap, doküman, bildiri, kaynak ve makale bulunabilir.

- 2014 yılında çıkan İndigo versiyonu 2019, 2016 yılında çıkan Kinetic Kame versiyonu 2021 yılına kadar desteklenmektedir.
- ROS FastSLAM 2.0 algoritmasını GMapping paketinin altında desteklemektedir.

4.3. Örnek 4 tekerlekli ROS Robot Projeleri

Şekil 4.1'de gösterilen Rosbot CORE2-ROScontroller üzerinde çalışan otonom bir açık kaynak robot platformudur [35]. ROS için öğrenme platformu olarak kullanılmasının yanı sıra çeşitli robotik uygulamalarına temel olarakta kullanılabilir. Örneğin; denetim robotları, özel servis robotları gibi.

Şekil 4.1 Rosbot [35].

Şekil 4.2'de verilen Warthog, karada ve suda hareket etme yeteneğine sahip her arazide gidebilen büyük bir insansız kara aracıdır [36]. Yumuşak topraklar, bitki örtüsü, çamurlar ve dik eğimlere karşı uyum sağlayan dayanıklı yapısı, düşük zemin basıncı ve çekiş lastikleri ile zorlu koşulların üstesinden gelebilir. Yük taşıma plakaları ve erişilebilir güç ve iletişim portları, Warthog'un, madencilik, tarım ve çevre denetimi alanlarında çok çeşitli robotik uygulamaları barındıracak sensörler, manipülatörler ve diğer yükler ile kolaylıkla özelleştirilebilmesini sağlar.

Şekil 4.2 Warthog UGV [36].

Şekil 4.3'de sunulan Hamster dayanıklı bir insansız mikro otonom kara aracıdır [37]. 4 çekişli bir motora sahiptir.

Şekil 4.3 Hamster [37].

Şekil 4.4'de verilen Guardian, Robotnik Otomasyon tarafından yapılmış yüksek hareket kabiliyetine sahip modüler bir robottur [38].

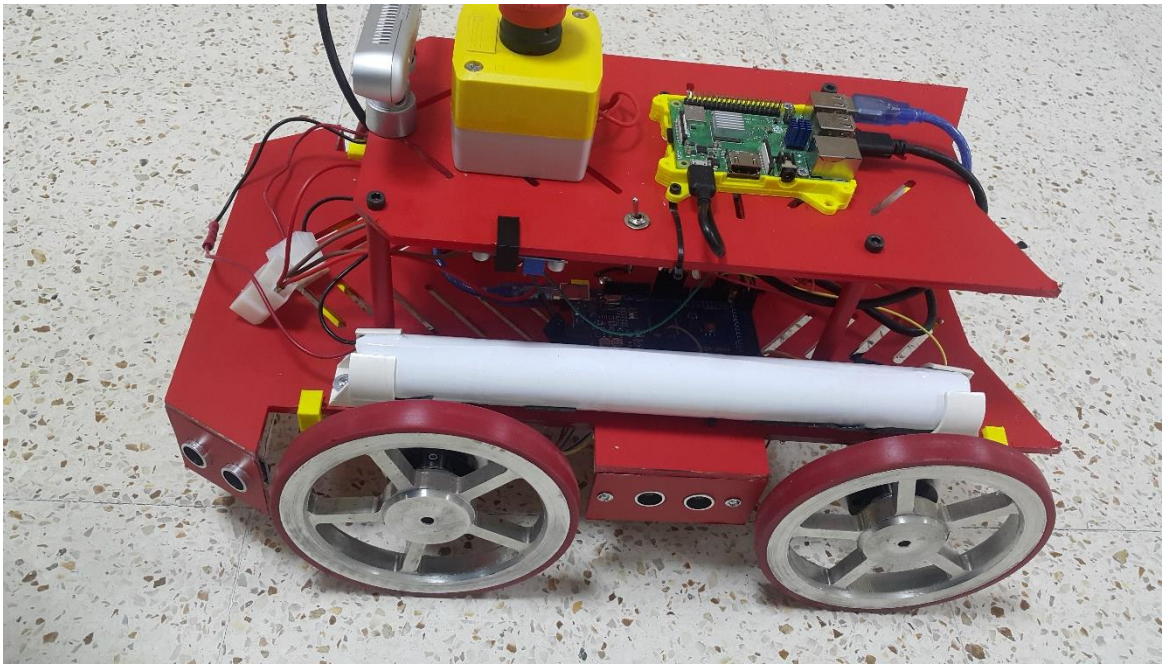
Şekil 4.4 Guardian [38].

Şekil 4.5'de verilen Bulldog (profesyonel Robot), dış mekân operasyonları için tasarlanmış, sağlam ve çok yönlü orta boy bir platformdur [39]. Bu robot her havada ve her arazide merdivenlerden ve eğimli yerlerden tırmanabilir ve engebeli yoldaki ilgili noktaya doğrudan manevra yapabilir. Robot uzaktan kontrol edilebilir, yönlendiren kişiyi takip edebilir veya GPS yol navigasyonunu kullanabilir.

Şekil 4.5 Bulldog [39].

5. OTONOM MOBİL ROBOT PLATFORMUNUN TASARIMI

Otonom bir robotun kendi başına çalışabilmesi için ortam verilerine ihtiyaç duymaktadır. Bu tezde geliştirilen mobil robot platformu yazılım geliştirmek için hazırlanmıştır. Robot Şekil 5.1'de gösterildiği gibi bağımsız 4 tekerlekten oluşmaktadır. Robot motor, kamera, mikrodenetleyici, on-board PC ve bataryadan oluşmaktadır.



Şekil 5.1 Robotun mekanik tasarımı

5.1. Dört Tekerlekli Mobil Robotun Kinematiği

Mobil robot platformunun kinematiği, Şekil 5.2 'de gösterilmektedir. Global koordinat çerçevelerinde araç konfigürasyon vektörü Denklem 5.1'de gösterilmiştir.

$$q = [X \quad Y \quad \theta]^T \in \mathbb{R}^3 \tag{5.1}$$

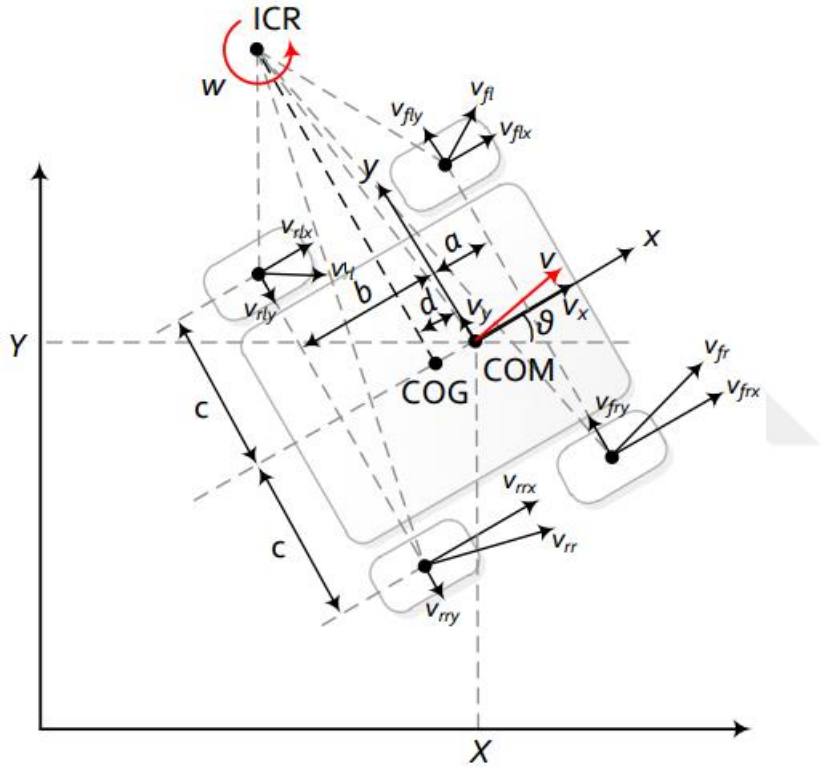
X, Y ve θ, sırasıyla taşıtın konumu ve yönüdür. Aracın kütle merkezi (COM) üzerine bağlanan yerel koordinat çerçevelerinde tanımlanan yerel hızlar ve genel hızlar arasındaki dönüşüm Denklem 5.2'de gösterilmiştir.

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ w \end{bmatrix} \tag{5.2}$$

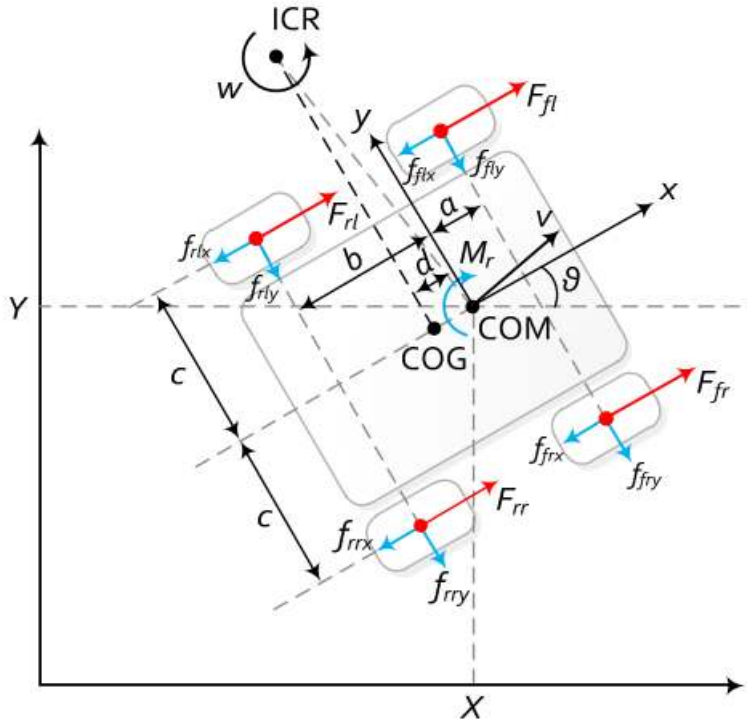
COM d ∈ (0, b) geometri merkezinden (COG) uzaktır. ICR, Şekil 5.2'de gösterildiği gibi, COG ile kesişen eksen üzerinde konumlandırılmıştır. Bu durumda, lateral kızağı sınırlayan non-holonomik işlevsel kısıtlama [40] olarak tanımlanmıştır. Ayrıca Şekil 5.3'de mobil robot platformunun dinamiğine ait parametreler Denklem 5.3 ve Denklem 5.4'te verilmiştir.

$$v_y - dw = 0 \tag{5.3}$$

$$[-\sin\theta \quad \cos\theta \quad -d]\left[\dot{X} \quad \dot{Y} \quad \dot{\theta}\right]^T = A(q)\dot{q} = 0 \tag{5.4}$$



Şekil 5.2 Mobil robot platformunun kinematiği



Şekil 5.3 Mobil robot platformunun dinamiği

Daha sonra Denklem 5.2 tekrar yazılarak S(q)∈R 3x2 matrisi ve η ∈ R2 olarak sırasıyla Denklem 5.5 ve Denklem 5.6'da gösterildiği gibi tanımlanabilir.

$$\dot{q} = S\,(q)\,\eta \tag{5.5}$$

$$S\,(q) = \begin{bmatrix} \cos\theta & -d\ \sin\theta \\ \sin\theta & d\cos\theta \\ 0 & 1 \end{bmatrix}, \eta = \begin{bmatrix} v_x & w \end{bmatrix}^T \tag{5.6}$$

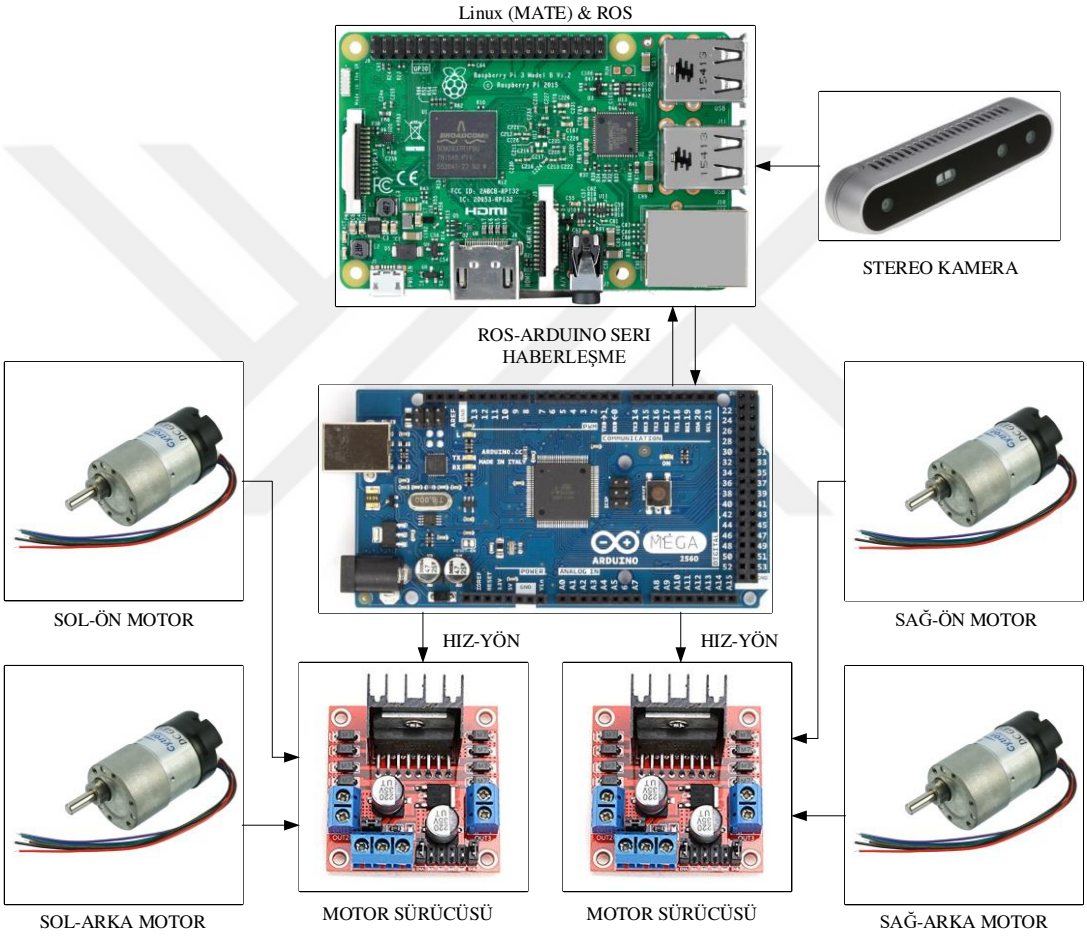
ve S (q) 'nın sütunları her zaman A (q)' nın boş uzayında olduğu için, Denklem 5.7'deki ifade yerine getirilir [40,41].

$$S^T A^T = 0 \tag{5.7}$$

5.2. Sistemin Elektronik Tasarımı

Tahrik sistemi olarak 4 adet encoderli PMDC (Permanent Magnet Direct Current) motor kullanılmıştır. Motorlardan daha fazla moment elde edebilmek için motor ve tekerlek arasında redüktör dişli mekanizması bulunmaktadır. Motorlar L298N sürücüsü kullanılarak kapalı çevrim hız kontrolü sağlanmıştır. L298N DC Step Motor Sürücü 4.8V-46V arasındaki iki motoru sürmek için hazırlanmış bir motor sürücü kartıdır. İki kanallı bu motor sürücü kanal başına 2A akım vermektedir. Motor sürücüleri Arduino Mega mikrodenetleyicisi ile

kontrol edilmiştir. Sistemin genel bağlantı şeması Şekil 5.4'de verilmiştir. Yazılım temel olarak iki kısım olarak geliştirilmiştir. Birinci aşamada; motorları sürmek için ROS ile haberleşebilen bir mikrodenetleyici programı gerçekleştirilmiştir. Geliştirilen mikrodenetleyici programı ROS tarafından gelen komutları anlamlandırarak motorların kapalı çevrim altında kontrol edilmesini sağlamaktadır. İkinci aşamada ise ROS işletiminin çalıştırılması için bir ARM çekirdekli onboard PC kullanılmıştır. İşletim sistemi olarak Linux-16.04.2 (Xenial) tercih edilmiş ve ROS için Kinetic sürümü kullanılmıştır.



Şekil 5.4 Sistemin genel bağlantı şeması

5.3. Robot Yazılımının Gerçekleştirilmesi

Tasarlanan mobil robot endüstriyel açıdan kullanılabilir olmasından dolayı lisans ücreti gerektirmeyen yazılımlar ile gerçekleştirilmiştir. Robot yazılımı ROS üzerine inşa edilmiştir, hem lisans ücreti gerektirmeyecektir hem de çevresini sürekli algılayabilmesi için istenilen kadar sensör takılma imkânı verecektir.

Robotun aktif olarak çevresindeki değişimleri algılayabilmesi için kullanılacak sensörler mesafe sensörü, basınç sensörü, nem sensörü, kamera ve stereo kamera gibi birden fazladır. Mesafe sensörü, basınç sensörü gibi sensörler kamera türü sensörlere karşın daha basit iletişime geçilip oradan alınan bilgiyi işlenebilecek sensörlerdir. Fakat bu sensörlerden alınan bilgiler endüstriyel bir mobil robot için yetersiz kacaktır. Bundan dolayı robot üzerine stereo kamera entegre edilmiştir. Hem bu kamerayı hem de ROS yazılımını çalıştırabilmek için bir adet Raspberrypi-3 kullanılmıştır. Bu mikrobilgisayar görüntü işlemini yöneteceğinden ve aynı zamanda ROS yazılımını çalıştıracağından dolayı yoğun bir işlem serisi gerçekleştirecektir. Bu yoğun işlem periyodunda tekerlerde bulunan motorların kontrolü işlemini ekleyip mikrobilgisayarın performansının düşmesini tercih edilemez olduğundan dolayı motor kontrol işlemi için Arduino-Mega işlemcisi üzerine ROS ile haberleşen program modülü oluşturuldu. Bu modül ROS çekirdeği ile iletişime geçerek motorları istenilen hızda kontrol etme işlemini yapmaktadır.

5.3.1. Raspberrypi Üzerinde ROS Çekirdeğinin Oluşturulması

Kullanılacak mikrobilgisayar olan Raspberrypi'nin ilk olarak kendi işletim sisteminin oluşturulması gerekmektedir. Raspberrypi üzerine ROS'u kolaylıkla çalıştırabileceğimiz işletim sistemlerinden biri olan buntu MATE işletim sisteminin kurulmasına karar verilmiştir. İlk olarak yapılan bu işlemlerin hepsi Linux işletim sistemli bir bilgisayarda gerçekleştirilmiştir. Ubuntu MATE işletim sisteminin sıkıştırılmış kalıp dosyası (k1) ile simgelenen kod satırı ile indirildi.

```
$ wget https://ubuntu-mate.org/raspberry-pi/ubuntu-mate-16.04.2-desktop-armhf raspberry-pi.img.xz
```
(k1)

İndirilen bu kalıp sıkıştırılmış halinden (k2) ile simgelenen kod satırı ile çıkartıldı.

```
$ unxz ubuntu-mate-16.04.2-desktop-armhf-raspberry-pi.img.xz
```
(k2)

Daha sonra kalıp Raspberrypi'nin hafıza kartına (k3) ile simgelenen kod satırı ile yazıldı.

```
$ sudo dd if=/path/to/ubuntu.img of=/dev/sdb bs=8M
```
(k3)

Hazırlanan hafıza kartı mikrobilgisayar üzerine takıldıktan sonra bilgisayar başlatıldı ve gereken ilk ayarlar yapıldı. ROS çekirdeğinin oluşturulması için *http://wiki.ros.org/kinetic/Installation/Ubuntu* sitesinden yararlanarak açılan bilgisayarda

aynı terminal ekranında (k4) ile simgelenenden (k12) ile simgelenene kadar bütün kod satırları sırasıyla çalıştırıldı.

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc)
main" > /etc/apt/sources.list.d/ros-latest.list'
```
(k4)

Bu kod satırı ROS paketlerini işletim sisteminin kaynak dosyasına eklemektedir. Ardından (k5) ile simgelenen kod satırı ile paketlere erişim için gerekli şifre verisi girişi sağlanmaktadır.

```
$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key
421C365BD9FF1F717815A3895523BAEEB01FA116
```
(k5)

Bilgisayarın güncel olup olmadığını kontrol etmek için (k6) ile simgelenen kod çalıştırıldı.

```
$ sudo apt-get update
```
(k6)

Bilgisayar güncellendikten sonra (k7) numaralı kod satırı ile ROS çekirdeği kuruldu.

```
$ sudo apt-get install ros-kinetic-desktop-full
```
(k7)

ROS çalıştırmadan önce ROS bağımlılıklarını (k8) ve (k9) numaralı kod satırları ile derlenip çalıştırıldı.

```
$ sudo rosdep init
```
(k8)

```
$ rosdep update
```
(k9)

Bağımlılıklar halledildikten sonra ROS ortamı kurulumu için (k10) ve (k11) numaralı kod satırı çalıştırıldı.

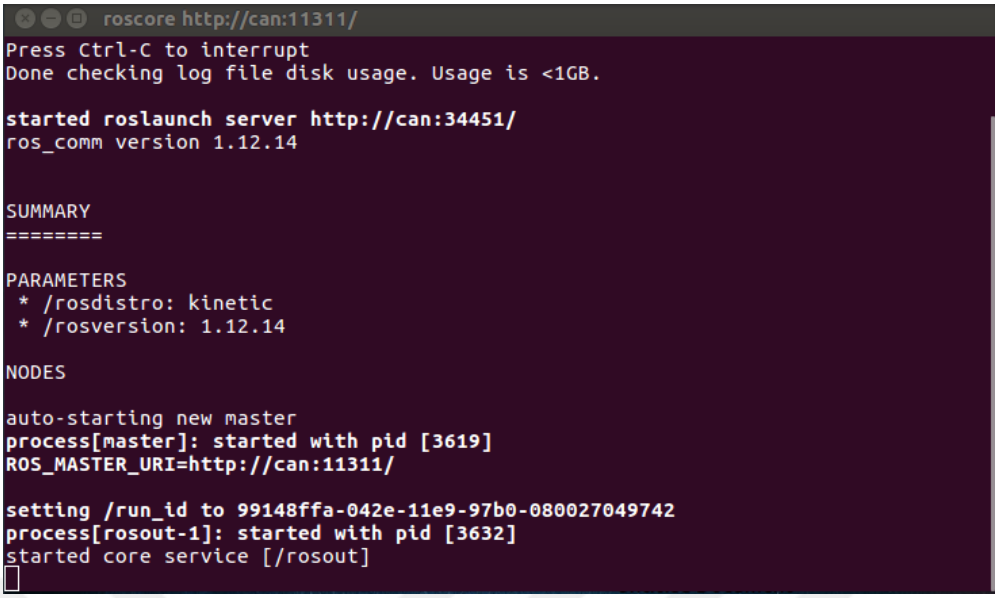
```
$ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
```
(k10)

```
$ source ~/.bashrc
```
(k11)

Son olarak oluşturulacak paketler için gerekli yüklemeleri gerçekleştirecek satır olan (k12) numaralı kod satırı terminalde çalıştırıldı.

```
$ sudo apt install python-rosinstall python-rosinstall-generator python-wstool
build-essential
```
(k12)

Bu işlemler bittikten sonra terminal ekranına *'roscore'* yazılarak yükleme işlemi Şekil 5.5'te gösterildiği üzere test edildi.



Şekil 5.5 ROS açılış ekranı

ROS kurulduktan sonra Arduino'nun geliştirme platformu olan Arduino IDE mikrobilgisayar üzerine yeni bir terminal ekranında (k13) numaralı kod ile yüklendi.

```
$ sudo apt-get install arduino
```
(k13)

Kurulum işlemi tamamladıktan sonra Arduino-ROS iletişimini sağlayacak olan rosserial_arduino paketi (k14) ve (k15) numaralı kod satırları ile yüklendi.

```
$ sudo apt-get install ros-kinetic-rosserial-arduino
```
(k14)

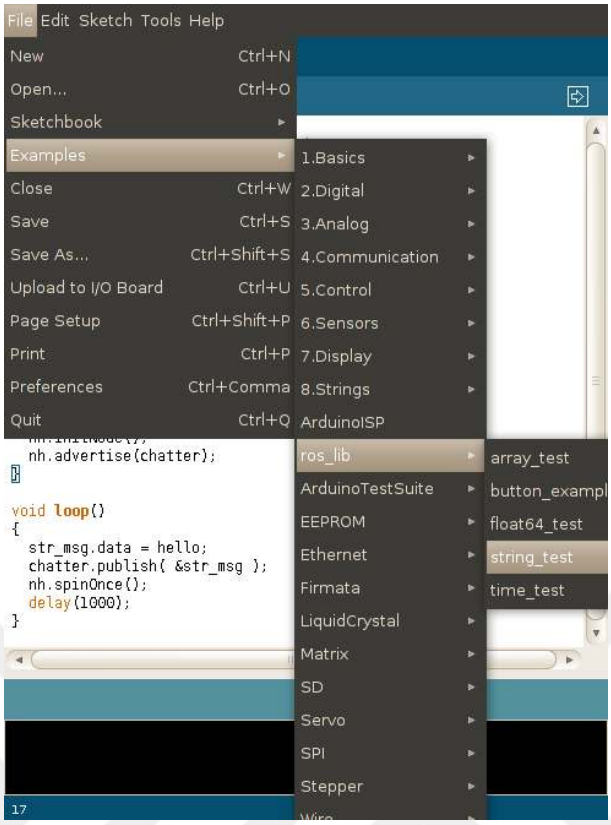
```
$ sudo apt-get install ros-kinetic-rosserial
```
(k15)

Bu işlemden sonra mikrobilgisayar üzerinda bulunan *home* klasöründe bulunan *sketchbook* klasörü içine girildi. Burada bulunan *libraries* klasörü açılarak burada terminal açıldı. Açılan bu terminale (k16) ve (k17) ile simgelenen kod satırları sırası ile çalıştırıldı.

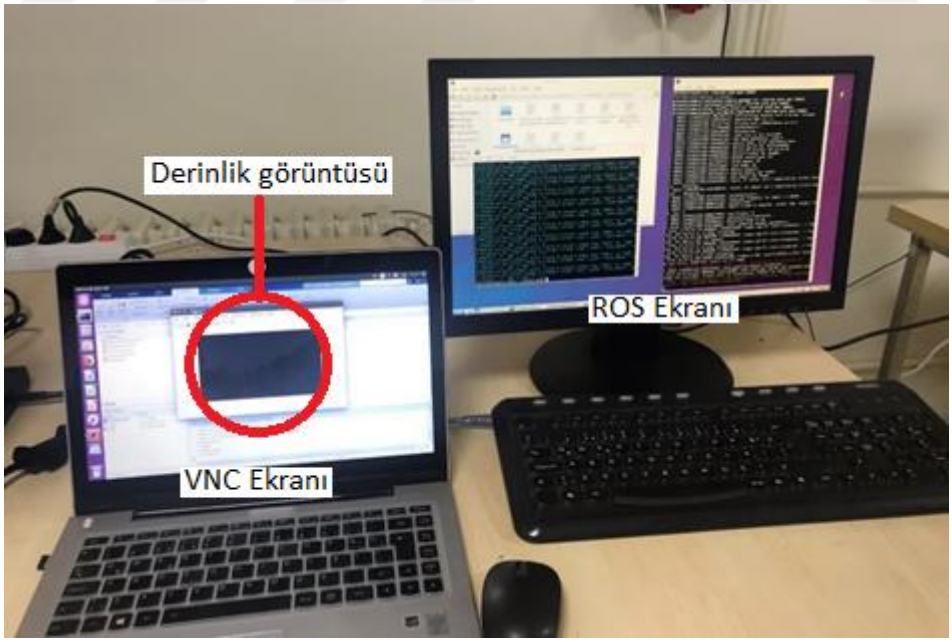
```
$ rm -rf ros_lib
```
(k16)

```
$ rosrun rosserial_arduino make_libraries.py
```
(k17)

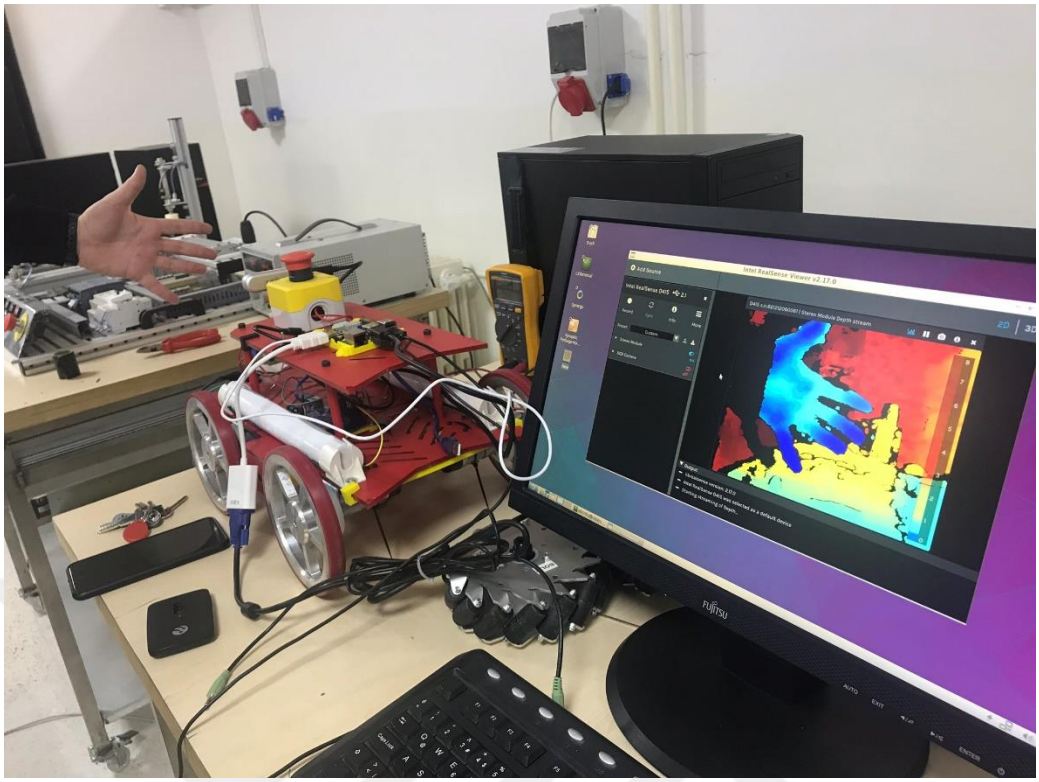
(k16) ile simgelenen satır daha önce oluşturulmuş olabilecek olan *ros_lib* klasörünü silerken (k17) numaralı kod satırı yeni *ros_lib* dizinini eklemektedir. Bu sayede Arduino IDE patformuna Şekil 5.6.'da göründüğü gibi ROS kütüphanesini eklenilmiştir [42]. Şekil 5.7'de Realsense kamerası derinlik algılaması gösterilmiştir. Ayrıca Şekil 5.8'de Realsense kamerasından renkli görüntü sunulmuştur.



Şekil 5.6 ROS kütüphanesi eklenmiş Arduino IDE [42].



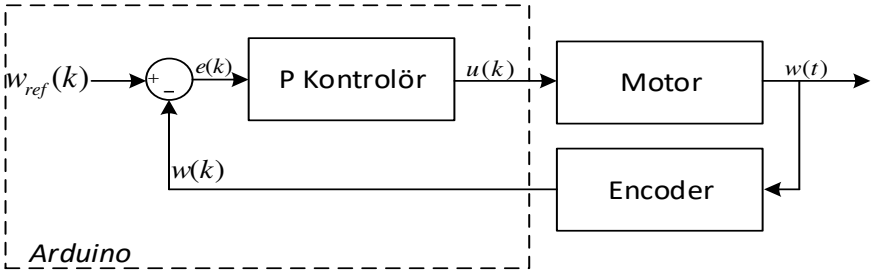
Şekil 5.7 Realsense kamerası derinlik algılaması



Şekil 5.8 Realsense kamerasından renkli görüntü alınması

5.4. Arduino Üzerinde ROS Modülünün Oluşturulması

ROS ve Arduino kod geliştirme platformu olan Arduino IDE kurulduktan sonra mikrobilgisayar üzerindeki işlemler tamamlanmış bulunmaktadır. Motorların kontrolü ve ROS ile iletişimi kuracak olan modül doğrudan Arduino üzerinde yapılmıştır. Oluşturulan bu modül iki kısımdan oluşmaktadır. Birinci kısım motorların kontrolü, ikinci kısım ise ROS ile iletişimin kurulacağı kısımdır. Motor kontrolünde Bölüm 5.2.'de anlatılan motorlar ve motor sürücüler kullanılmaktadır. Motorların kontrolü için uygulanan algoritma Şekil 5.9'da görülmektedir.



Şekil 5.9 Motor hız kontrol blok diyagramı

Bu şekilde $w_{ref}(k)$ referans hızı, $e(k)$ ayrık zamanda hatayı, $u(k)$ ayrık zamanda kontrol sinyalini $w(t)$ sürekli zamanda hız bilgisini ve $w(k)$ ayrık zamanda hız bilgisini simgelemektedir. Kesikli çizgi ile gösterilen kısım Arduino içerisinde gerçekleştirilmiştir. Kontrolör olarak Oransal (P) Kontrolör seçilmiştir. Bunun nedeni işlemcinin motor sürme ve ROS ile iletişim sırasında zaman kavramı standart olmayacaktır ve bundan dolayı türev integral alınması pek mümkün olmadığından dolayıdır. P katsayısı ise yapılan deneyler sonrasında 3 olarak seçilmiştir. Dört motor içinde bu kontrol yapısı kullanılmıştır.

ROS yapısına bakıldığında temelde iki kısımdan oluşmaktadır. Bunlar yayıncılar (publisher) ve aboneler (subscriber) kısımlarıdır. Bu çalışmada ROS çekirdeği yayıncı Arduino ise abone olarak kullanılmıştır. Bunun nedeni, yapılan robot bir mobil robot olduğundan dolayı, ROS çekirdeği ile motor kontrolü arasında hız verisi alışverişi yapılmasıdır. ROS *my_cmd_vel* mesajı ile robotun hızını kontrol etmektedir ve bu durum ROS'un grafik ekranı olan *rqt_graph* görüntüsü ile Şekil 5.10.'da gösterilmiştir. *cmd_vel* mesajı ROS sistemi üzerinde *'geometry_msgs/Twist'* kütüphanesi içerisinde bulunur ve içerisinde üç adet doğrusal hız üç adet ise açısal hız barındırır. Burada kullanılan *my_cmd_vel* mesajı bir *cmd_vel* mesajı türüdür. Hazırlanan Arduino ROS mödülünün Arduino kodu EKLER bölümünde'de açıklamaları ile verilmiştir. Bu program Arduino IDE yardımı ile yüklendikten sonra ROS çekirdeği bu modüle (18) numaralı satırı çalıştırdıktan sonra bağlantı kurar.

```
$ rosrun rosserial_python serial_node.py /dev/ttyUSB0 _baud:=57600          (k18)
```

Şekil 5.10 rqt_graph görüntüsü

Bu şekilde /rostopic deyimi ile başlayan kısım ROS çekirdeğini, /serial_node ile ifade edilen kısımı Arduinoyu ve /my_cmd_vel kısmı ise gönderilen hız mesajını simgelemektedir.

5.5. Intel Realsense D415 Derinlik Kamerası

Bu tez çalışmasında derinlik kamerası olarak Şekil 5.11'de verilen Intel firmasının RealSense D415 modeli kullanılmıştır [43]. Bu kamera Intel tarafından üretilmiş 1920x1080 RGB çözünürlüğüne sahiptir. FOV (Field of View) değeri görüntünün objektiften belirli bir mesafede yatay (veya dikey veya diyagonal) uzunluğudur. FOV diyagonal değeri 77, yatay değeri 42.5 ve dikey değeri 69.4'tür. Minimum derinlik değeri 300 mm'dir. USB 3.0 Type-C arayüzüne sahip bir cihaz olup birçok platform için SDK'ya (Software Developer's Kit) sahiptir. Maksimum RGB fps değeri 60. Maksimum derinlik fps değeri 90'dır. Derinli çözünürlüğü RGB değerinden daha düşüktür. Derinli çözünürlüğü maksimum 1280x720 olarak belirlenmiştir.

Şekil 5.11 Intel realsense D415 derinlik kamerası [43]

Kamera derinliği Stereoskop konsepti ile algılamaktadır. Stereoskopi, bir görüntüdeki derinlik yanılsamasını, binoküler görüntünün stereopsisi vasıtasıyla oluşturmak veya geliştirmek için kullanılan bir tekniktir [44]. Stereoskopi kelimesi Yunancada katı anlamına gelen στερεός (stereos) ve görmek için anlamına gelen σκοπέω (skopeō) kelimelerinin birleşiminden oluşmaktadır [45,46]. Herhangi bir stereoskopik görüntüye stereogram denir. Başlangıçta, stereogram bir stereoskop kullanılarak görüntülenebilen bir çift stereo görüntü anlamına gelmekteydi. Kamera yaklaşık 16cm değerinden 10m'ye kadar derinlik algısı oluşturabilmektedir.

Stereo görme uygulaması sol görüntüleyici, sağ görüntüleyici ve opsiyonel bir infrared projektörden oluşur. Kızılötesi projektör, düşük dokuya sahip sahnelerde derinlik doğruluğunu artırmak için görünür olmayan statik IR desenini yansıtır. Sol ve sağ

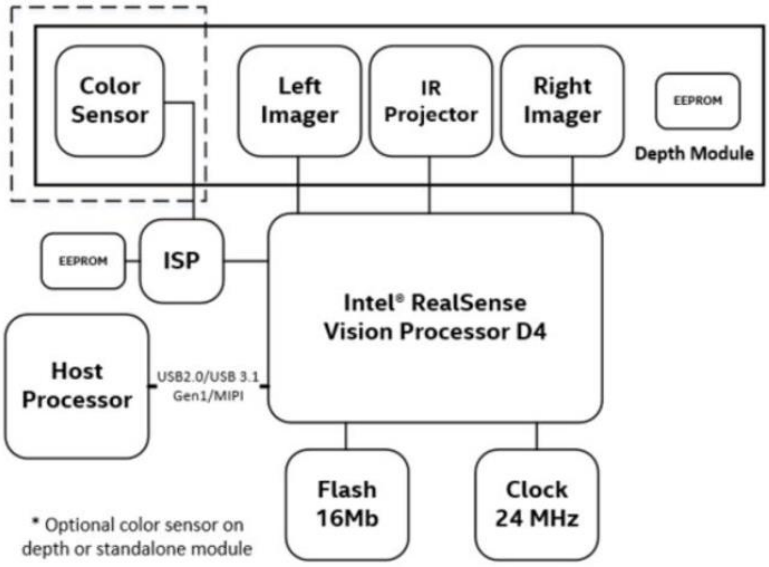
görüntüleyiciler sahneyi yakalar ve görüntü verisini derinlik görüntü işlemcisine gönderir, görüntüdeki her piksel için derinlik değerlerini, soldaki görüntüdeki noktaları sağ görüntüye ve soldaki görüntü ile sağdaki görüntü arasındaki nokta arasında ilişki kurarak hesaplar. Derinlik piksel değerleri bir derinlik çerçevesi oluşturmak için işlenir. Sonraki derinlik çerçeveleri, derinlikli bir video akışı oluşturur. Kameranın çalışma prensibine ait akış Şekil 5.12'de sunulmuştur [47].

Şekil 5.12 Aktif Infrared (IR) stereo görme teknolojisi [47].

Derinlik piksel değeri Şekil 5.13'te görüldüğü gibi, görüntüdeki mutlak aralığın değil, görüntüleyicilerin paralel düzleminden bir ölçümdür [47].

Şekil 5.13 Derinlik ölçümü (Z)'e karşı menzil (R) [47].

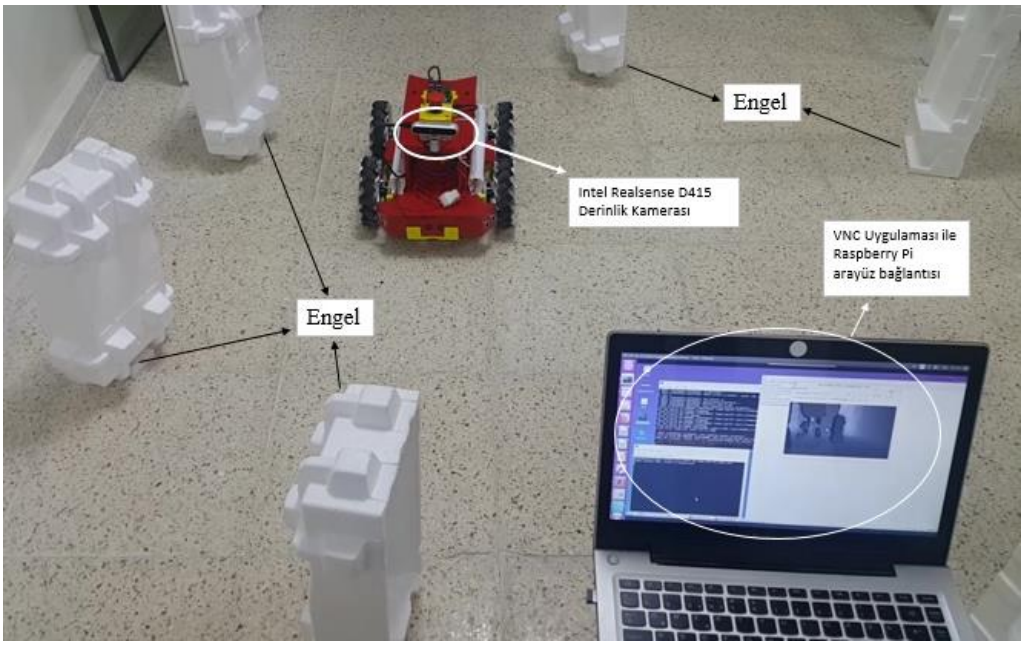
Şekil 5.14'te gösterilen kamera sistemi; D4 görüntü işlemcisi ve derinlik modülü olmak üzere iki ana bileşene sahiptir [47]. Görüntü işlemcisi hem ana işlemci ana kartına hem de ana işlemciye USB 2.0/USB 3.1 veya MIPI bağlantısı olan ayrı bir kart üzerinde bulunur. Derinlik modülü, opsiyonel IR projektör ve RGB renk sensörü ile stereo görüntü için sol ve sağ görüntüleyicileri içerir.



Şekil 5.14 Görüntü işlemcisi D4 kamera sistemi blok diyagramı [47].

5.6. Deneysel Sonuçlar

Bu tez çalışmasında geliştirilen robot ve yazılım Şekil 5.15'te gösterildiği gibi deneysel olarak test edilmiştir. Robot engellerden kaçacak şekilde programlanmıştır. Derinlik kamerasından gelen derinlik matrisi üzerinde görüntü işleme algoritması çalıştırılarak gelen bilgi anlamlı hale getirilmiştir. Daha sonra robot önündeki engelin orta noktasını belirleyerek sağa/sola doğru hareket etmektedir.



Şekil 5.15 Araç deneme çalışması

Robotun önündeki engellerden kaçması kameraya çekilerek Şekil 5.16'da gösterildiği şekilde parçalara bölünmüştür. Robotun anlık pozisyonu ve bilgisayar ekranındaki engelin konumu Şekil 5.16'da görülmektedir. Robot engelleri başarılı bir şekilde geçebilmiştir. Ayrıca robotun aşamayacağı bir engel olursa robot otomatik olarak durmaktadır.

Şekil 5.16 Aracın hareket halindeki görüntüsü

6. SONUÇ

Günümüzde mobil robotlara olan ilgi artmaktadır. Robotik araştırmalarının birçoğu mobil robotlar üzerine yapılmaktadır. Bundan dolayı bu tez çalışmasında mobil robotlar için bir yazılım platformu geliştirilmiştir. Bu sistem; ROS altyapısını kullandığından dolayı C++ ve Python diller ile program geliştirmeye müsait bir yapıya sahiptir. Elde edilen sistem Linux işletim sistemi üzerinde çalışmaktadır. Ayrıca ROS yazılımı sayesinde lisans problemi bulunmaması bu sistemin önemli avantajlarından biridir. Tezde Linux işletim sistemini çalıştırabilen bir ARM çekirdek kartı kullanılmıştır. Bu karta derinlik kamerası USB üzerinden bağlanmıştır. Derinlik kamerasından elde edilen bilgiler ROS işletim sisteminde anlamlandırılmış ve kartta bulunan Wi-Fi modülü üzerinden ana programın çalıştırdığı bilgisayara gönderilmiştir. Ana bilgisayarda temel görüntü işleme algoritmaları kullanılarak robotun önündeki engelleri algılaması başarılı bir şekilde gerçekleştirilmiştir. Belirlenen engellere çarpmaması için robot engelin az olduğu tarafa doğru yönelmesi sağlanmıştır. Ayrıca robotun aşamayacağı bir engel olursa robot otomatik olarak durabilmesi sağlanmıştır. Tezde arzu edilen sonuçlar deneysel çalışmalar ile gösterilmiştir.

7. KAYNAKLAR

[1] **Thrun S. , Beetz M. , Bennewitz M. , Burgard W. , Creemers A.B. , Dellaert F. , Fox D. , Hahnel D. , Rosenberg C. , Roy N. , Schulte J. and Schulz D.** , (2000). Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva. International Journal of Robotics Research, 19(11), 972-999.

[2] **Thrapp R. , Westbrook C. and Subramanian D.** , (2006). Robust localization algorithms for an autonomous campus tour guide, IEEE International Conference on Robotics and Automation. 2, 2065-2071.

[3] **Ben-Tzvi P. , Shingo I. and Goldenberg A.A. ,** (2007). Autonomous stair climbing with reconfigurable tracked mobile robot. IEEE International Workshop on Robotics a nd Sensor Environment. 1-6.

[4] **Wu S. , Cheng M. and Hsu W. ,** (2005). Design and implementation of a prototype vision-guided golf-ball collecting mobile robot. IEEE International Conference on Mechatronics, 611-615.

[5] **Yukawa T. , Nakata N. , Obinata G. and Makino T. ,** (2010). Assistance system for bedridden patients to reduce the burden of nursing care (First report – Development of a multifunctional electric wheelchair, portable bath, lift, and mobile robot with portable toilet). IEEE International Symposium on System Integration, 132-139.

[6] **Bajracharya M. , Maimone M.W. , and Helmick D. ,** (2008). Autonomy for Mars Rovers: Past, Present, and Future. Journal of Computer. 41(12), 44-50.

[7] **Kayani S.A. , Bhatti W.H. , and Jarral K.K. ,** (2009). On design and fabrication of a prototype teleoperated mobile surveillance robot. International Conference on Emerging Technologies, 157- 161.

[8] https://www.mt-oceanography.info/science+society/lectures/illustrations/lecture35/hero.html

[9] **Connor Clinton D. ,** "Sensor Fusion, Navigation and Control of Autonomous Vehicles", A Thesis Presented For The Master of Science Degree, Virginia Polytechnic Institute and State University, July 2000, pp. 17-40

[10] http://www.ai.sri.com/shakey/images.php

[11] **Gage Douglas W. ,** "A Brief History of Unmanned Ground Vehicle (UGV) Development Efforts", Special Issue on Unmanned Ground Vehicles, Unmanned Systems Magazine, Volume 13, Number 3, Summer 1995, pp. 4-7

[12] **M. Atilla**, “Tekerlekli Otonom Robot Yapımı ve Haritalama”, p. 5, June 2007

[13] **H. Castle**, “Made by Robots: Challenging Architecture at a Larger Scale”, p. 80, May 2014

[14] http://www.workNC.com/news/casestudies/audi-toolmaking-cad-cam-success

[15] **N. Bellotto and H. HU**, “Multisensor-Based Human Detection and Tracking for Mobile Service Robots”, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 39.1 (2009): 167-181

[16] https://thespoon.tech/can-tech-completely-automate-the-restaurant-front-of-house/

[17] **G. I. Barbash and S. A. Glied**. "New Technology and Health Care Costs The Case of Robot-Assisted Surgery.", New England Journal of Medicine, 363:701-704, August 2010

[18] **A. M. Okamura, M. J. Mataric and H. I. Christensen.** "Medical and Health-Care Robotics.", IEEE Robotics and Automation Magazine, 17.3:26-27, September 2010

[19] https://www.futurenotes.org/da-vinci-robotu-nedir/

[20] **M. Premkumar** “Unmanned Multi-Functional Robot Using Zigbee Adopter Network For Defense Application”, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume 2, Issue 1, p. 47-55, January 2013

[21] http://www.defenceturk.com/index.php?topic=2373.0

[22] https://www.yapbenzet.com/otonom-robotlar-giris/

[23] https://robots.ieee.org/robots/anymal/

[24] https://www.bostondynamics.com/spot-mini

[25] http://www.rsl.ethz.ch/robots-media/anymal.html

[26] https://www.bostondynamics.com/spot-mini

[27] **Koide S. , Andou H. , Suzuki S. ve Oda M.** , “Semi-Autonomous Teleoperation Control System with Layered Architecture-An Application to Space Robots”, Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS '93, Cilt 3, 1995 – 2001, 1993.

[28] **Kelley R.B.** , “Semi-Autonomous Robotic Manipulation”, IEEE International Conference on Systems, Man, and Cybernetics, 'Computational Cybernetics and Simulation', Cilt 2, 1759-1764, 1997.

[29] http://thescienceexplorer.com/universe/why-we-can-t-investigate-water-mars

[30] **Yılmaz N. , Sağıroğlu Ş. ve Bayrak M.** , "Genel Amaçlı Web Tabanlı Mobil Robot: SUNAR", Gazi Üniversitesi, Mühendislik Mimarlık Fakültesi Dergisi, Cilt 21, No: 4, 745-752, 2006.

[31] **De Almeida A.T. , Khatib O**. , Autonomous Robotic Systems (Lecture Notes in Control And Information Sciences), Springer-Verlag, 1998.

[32] **Yılmaz N. ve Sağıroğlu Ş.** , "Real-Time Line Tracking Based on Web Robot Vision", 2009 Wiley Periodicals, Inc. Computer Applications in Engineering Education, DOI 10.1002/cae.20367, 1-10, 2009.

[33] **Sagıroğlu Ş. ve Yılmaz N.** , "Web-based mobile robot platform for real-time exercises", Expert Systems with Applications, Cilt 36, No 2, 3153- 3166, 2009.

[34] https://www.skoda-storyboard.com/en/press-releases/skoda-auto-uses-fully-autonomous-transport-robot-at-vrchlabi-plant/

[35] https://www.instructables.com/id/ROSbot-Autonomous-Robot-With-LiDAR/

[36] https://www.clearpathrobotics.com/blog/2016/10/clearpath-robotics-launches-amphibious-robot-warthog-ugv/

[37] https://wiki.cogni.io/index.php/Category:Hamster

[38] https://robots.ros.org/guardian/

[39] https://robots.ros.org/bulldog/

[40] **L. Caracciolo, A. De Luca, S. Iannitti**, "TrajectoryTracking Control of a Four-Wheel Differentially DrivenMobile Robot," IEEE Int. Conf. on Robotics andAutomation, Detroit, Michigan, pp. 2632-2638, May 1999.

[41] **K. Kozlowski, D. Pazderski, I. Rudas, J. Tar**, "Modeling and control of a 4-wheel skid-steering mobile robot: Fromtheory to practice," Budapest Polytechnic Jubilee Conference, Science in Engineering, Economics and Education, Budapest, Hungary, September 2004

[42] http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup

[43] https://click.intel.com/intelr-realsensetm-depth-camera-d415.html

[44] "The Kaiser (Emperor) Panorama". June 9, 2012.

[45] Jump up to:a b The Logical Approach to Seeing 3D Pictures. www.vision3d.com by Optometrists Network. Retrieved 2009-08-21

[46] στερεός Tufts.edu, **Henry George Liddell, Robert Scott,** A Greek-English Lexicon, on Perseus Digital Library

[47] https://www.mouser.com/pdfdocs/Intel_D400_Series_Datasheet.pdf

8. EKLER

Arduino Kodu

```
#include <ros.h>     // ros kütüphanesi
#include <geometry_msgs/Twist.h> // cmd_vel için gerekli kütüphane
#include<SoftwareSerial.h>


// SAG MOTORLAR
#define PWM_R1 7   // siyah
#define IN4_R1 22  // beyaz
#define IN3_R1 24  // gri
#define IN2_R2 26  // mor
#define IN1_R2 28  // mavi
#define PWM_R2 6   // yesil

// SOL MOTORLAR
#define PWM_L1 5   // sari
#define IN4_L1 30  // turuncu
#define IN3_L1 31  // kirmizi
#define IN2_L2 32  // kahverengi
#define IN1_L2 33  // siyah
#define PWM_L2 4   // beyaz


//Enkoder kesmeleri
#define ENC_A_R1 2   // INT 0 pin 2
#define ENC_A_R2 3   // INT 1 pin 3
#define ENC_A_L1 19   // INT 4 pin 19
#define ENC_A_L2 18   // INT 5 pin 18

int val = 0;
int err1,err2,err3,err4;
volatile byte encoder_R1;
volatile byte encoder_R2;
volatile byte encoder_L1;
volatile byte encoder_L2;
unsigned int speed_R1,speed_R2,speed_L1,speed_L2;
unsigned long time_R1,time_R2,time_L1,time_L2;
int x,y,teta; //ROS


geometry_msgs::Twist new_vel; //cmd_vel tipinde veri oluşturma
ros::NodeHandle n; //ros nodu oluşturma

void velCallback(const geometry_msgs::Twist& vel) // ros abone veri alma
{
new_vel.linear.x = vel.linear.x;
```

```
   new_vel.linear.y = vel.linear.y;
   new_vel.angular.x = vel.angular.x;
   }

   ros::Subscriber<geometry_msgs::Twist> sub("my_cmd_vel", &velCallback); // ros
abone nesnesi oluşturma

   void setup()
   {
         //Motor PWMi yön ve encoder pinleri tanıtımı
   pinMode(PWM_R1 , OUTPUT);pinMode(IN4_R1 , OUTPUT);pinMode(IN3_R1 ,
OUTPUT);pinMode(IN2_R2 , OUTPUT);pinMode(IN1_R2 ,
OUTPUT);pinMode(PWM_R2 , OUTPUT);

   pinMode(PWM_L1 , OUTPUT);pinMode(IN4_L1 , OUTPUT);pinMode(IN3_L1 ,
OUTPUT);pinMode(IN2_L2 , OUTPUT);pinMode(IN1_L2 ,
OUTPUT);pinMode(PWM_L2 , OUTPUT);


pinMode(ENC_A_R1,INPUT);pinMode(ENC_A_R2,INPUT);pinMode(ENC_A_L1,INP
UT);pinMode(ENC_A_L2,INPUT);


   attachInterrupt(0, count_R1, RISING);
   attachInterrupt(1, count_R2, RISING);
   attachInterrupt(4, count_L1, RISING);
   attachInterrupt(5, count_L2, RISING);

   n.initNode();                    // ros nodu başlatma
   n.subscribe(sub);    // ros aboneliğini aktifleştirme
   }

   void loop()
   {
      x=new_vel.linear.x;
      y=new_vel.linear.y;
      teta=new_vel.angular.x;

                  // encoderleri okuma
   if (encoder_R1 >= 20) {
      speed_R1 = 2000/(millis() - time_R1);
      time_R1 = millis();
      encoder_R1 = 0;
   Serial.println(speed_R1,DEC);
   }
      if (encoder_R2 >= 20) {
      speed_R2 = 2000/(millis() - time_R2);
      time_R2 = millis();
      encoder_R2 = 0;
```

```
    Serial.println(speed_R2,DEC);
  }
    if (encoder_L1 >= 20) {
    speed_L1 = 2000/(millis() - time_L1);
    time_L1 = millis();
    encoder_L1 = 0;
    Serial.println(speed_L1,DEC);
  }
    if (encoder_L2 >= 20) {
    speed_L2 = 2000/(millis() - time_L2);
    time_L2 = millis();
    encoder_L2 = 0;
    Serial.println(speed_L2,DEC);
  }


  /*
  * Seri porttan gelen decimal değer 4 basamaklı en ağırlıklı digit motor seçimi son 3 digit ise motor hızı
  * Motor hızı 1-999 arasında ayarlanacak
  * Duracak motorlara x000 gönderilecek x=motor numarası *
  * */


  // x hareketi
  if (x!=0 && y==0 && teta==0){
  val = x+sign(x)*1000;
  err1 = ((val-sign(val)*1000)/4)-sign(val)*speed_R1;
  if ((val>1000) && (val<2000)){ motor(3*err1,PWM_R1,IN4_R1,IN3_R1);} //sağ ön
  if ((val<-1000) && (val>-2000)){ motor(3*err1,PWM_R1,IN4_R1,IN3_R1);} //sağ ön

  val = x+sign(x)*2000;
  err2 = ((val-sign(val)*2000)/4)-sign(val)*speed_R2;
  if ((val>2000) && (val<3000)){ motor(3*err2,PWM_R2,IN2_R2,IN1_R2);} //sağ arka
  if ((val<-2000) && (val>-3000)){ motor(3*err2,PWM_R2,IN2_R2,IN1_R2);} //sağ arka

  val = x+sign(x)*3000;
  err3 = ((val-sign(val)*3000)/4)-sign(val)*speed_L1;
  if ((val>3000) && (val<4000)){ motor(3*err3,PWM_L1,IN4_L1,IN3_L1);} //sol ön
  if ((val<-3000) && (val>-4000)){ motor(3*err3,PWM_L1,IN4_L1,IN3_L1);} //sol ön

  val = x+sign(x)*4000;
  err4 = ((val-sign(val)*4000)/4)-sign(val)*speed_L2;
  if ((val>4000) && (val<5000)){ motor(3*err4,PWM_L2,IN2_L2,IN1_L2);} //sol arka
  if ((val<-4000) && (val>-5000)){ motor(3*err4,PWM_L2,IN2_L2,IN1_L2);} //sol arka
  }

      // y hareketi
```

```
   else if (x==0 && y!=0 && teta==0){
   val = y+sign(y)*1000;
   err1 = -(((val-sign(val)*1000)/4)-sign(val)*speed_R1);
   if ((val>1000) && (val<2000)){ motor(3*err1,PWM_R1,IN4_R1,IN3_R1);} //sağ ön
   if ((val<-1000) && (val>-2000)){ motor(3*err1,PWM_R1,IN4_R1,IN3_R1);} //sağ ön

   val = y+sign(y)*2000;
   err2 = ((val-sign(val)*2000)/4)-sign(val)*speed_R2;
   if ((val>2000) && (val<3000)){ motor(3*err2,PWM_R2,IN2_R2,IN1_R2);} //sağ arka
   if ((val<-2000) && (val>-3000)){ motor(3*err2,PWM_R2,IN2_R2,IN1_R2);} //sağ
arka

   val = y+sign(y)*3000;
   err3 = ((val-sign(val)*3000)/4)-sign(val)*speed_L1;
   if ((val>3000) && (val<4000)){ motor(3*err3,PWM_L1,IN4_L1,IN3_L1);} //sol ön
   if ((val<-3000) && (val>-4000)){ motor(3*err3,PWM_L1,IN4_L1,IN3_L1);} //sol ön

   val = y+sign(y)*4000;
   err4 = -(((val-sign(val)*4000)/4)-sign(val)*speed_L2);
   if ((val>4000) && (val<5000)){ motor(3*err4,PWM_L2,IN2_L2,IN1_L2);} //sol arka
   if ((val<-4000) && (val>-5000)){ motor(3*err4,PWM_L2,IN2_L2,IN1_L2);} //sol arka
   }


   // teta hareketi
   else if (x==0 && y==0 && teta!=0){
   val = teta+sign(teta)*1000;
   err1 = ((val-sign(val)*1000)/4)-sign(val)*speed_R1;
   if ((val>1000) && (val<2000)){ motor(3*err1,PWM_R1,IN4_R1,IN3_R1);} //sağ ön
   if ((val<-1000) && (val>-2000)){ motor(3*err1,PWM_R1,IN4_R1,IN3_R1);} //sağ ön

   val = teta+sign(teta)*2000;
   err2 = ((val-sign(val)*2000)/4)-sign(val)*speed_R2;
   if ((val>2000) && (val<3000)){ motor(3*err2,PWM_R2,IN2_R2,IN1_R2);} //sağ arka
   if ((val<-2000) && (val>-3000)){ motor(3*err2,PWM_R2,IN2_R2,IN1_R2);} //sağ
arka

   val = teta+sign(teta)*3000;
   err3 = -(((val-sign(val)*3000)/4)-sign(val)*speed_L1);
   if ((val>3000) && (val<4000)){ motor(3*err3,PWM_L1,IN4_L1,IN3_L1);} //sol ön
   if ((val<-3000) && (val>-4000)){ motor(3*err3,PWM_L1,IN4_L1,IN3_L1);} //sol ön

   val = teta+sign(teta)*4000;
   err4 = -(((val-sign(val)*4000)/4)-sign(val)*speed_L2);
   if ((val>4000) && (val<5000)){ motor(3*err4,PWM_L2,IN2_L2,IN1_L2);} //sol arka
   if ((val<-4000) && (val>-5000)){ motor(3*err4,PWM_L2,IN2_L2,IN1_L2);} //sol arka
   }
```

```
  else {
  // durma-stop
  if (val==1000){ motor(0,PWM_R1,IN4_R1,IN3_R1); }   //sağ ön
  if (val==2000){ motor(0,PWM_R2,IN2_R2,IN1_R2);} //sağ arka
  if (val==3000){ motor(0,PWM_L1,IN4_L1,IN3_L1);} //sol ön
  if (val==4000){ motor(0,PWM_L2,IN2_L2,IN1_L2);} //sol arka
  }


n.spinOnce();            // ros döngüsünü tamamlama
delay(1);

  }


  void motor(int speed,int pwm_pin,int c1_pin,int c2_pin){
  analogWrite(pwm_pin,abs(speed));
     if (speed<=0){
     digitalWrite(c1_pin,LOW);
     digitalWrite(c2_pin,HIGH);}
     else
     {digitalWrite(c1_pin,HIGH);
     digitalWrite(c2_pin,LOW);}
                                        }

  void count_R1(){encoder_R1++;}
  void count_R2(){encoder_R2++;}
  void count_L1(){encoder_L1++;}
  void count_L2(){encoder_L2++;}

  int sign(int s){
  if (s>0) {return 1;}
  else if (s<0) {return -1;}
  else return 0;
  }
```

ÖZGEÇMİŞ

Hasan Selçuk YALÇIN

hasanselcukyalcin@gmail.com

1990 Mersin / Tarsus'da doğdu.

2010-2014 Fırat Üniversitesi Teknoloji Fakültesi Mekatronik Mühendisliğinden mezun oldu.

2012-2016 Çift anadal programı kapsamında Fırat Üniversitesi Teknoloji Fakültesi Yazılım Mühendisliğinden mezun oldu.

2016-devam Mersin Büyükşehir Belediyesinde Bilgi İşlem Dairesi Başkanlığında sistem yöneticisi olarak çalışmaya devam etmektedir.