ROBOTİK UYGULAMALRDA YAZILIM GELİŞTİRME AMAÇLI SİMULATÖR KULLANIMI

LİSANS TEZİ

Ahmet Yasin CİVAN

Danışman

Yrd.Doç.Dr Serkan ÇAŞKA

AFYON KOCATEPE ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ

LİSANS TEZİ

ROBOT SİMÜLATÖRLERİ VE V-REP'DE MODELLENMİŞ DONANIM ÇEVRİMLİ MOBİL ROBOT SİMÜLATÖR ÇALIŞMASI

Ahmet Yasin CİVAN

Danışman
Yrd.Doç.Dr Serkan ÇAŞKA

MEKATRONİK MÜHENDİSLİĞİ

06.2017

TEZ ONAY SAYFASI

Ahmet Yasin CİVAN tarafından hazırlanan “Robot Simülatörleri Ve V-Rep'de Modellenmiş Donanım Çevrimli Mobil Robot Simülatör Çalışması” adlı tez çalışması lisansüstü eğitim ve öğretim yönetmeliğinin ilgili maddeleri uyarınca 23/06/2017 tarihinde aşağıdaki jüri tarafından **oy birliği** / **oy çokluğu** ile Afyon Kocatepe Üniversitesi Fen Bilimleri Enstitüsü Fakültesi Mekatronik Mühendisliği bölümünde **BİTİRME TEZİ** olarak kabul edilmiştir.

		İmza
Danışman	**:** Yrd.Doç.Dr. Serkan Çaşka	…….
Üye	**:** Yrd. Doç.Dr. Barış GÖKÇE	…….
Üye	**:**Yrd. Doç.Dr Guray SONUGÜR	**...**…….

BİLİMSEL ETİK BİLDİRİM SAYFASI
Afyon Kocatepe Üniversitesi

Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- Görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- Başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- Atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

22/06/2017

Ahmet Yasin CİVAN

ÖZET
Lisans Tezi

ROBOT SİMÜLATÖRLERİ VE V-REP'DE MODELLENMİŞ DONANIM ÇEVRİMLİ MOBİL ROBOT SİMÜLATÖR ÇALIŞMASI

Ahmet Yasin CİVAN

Afyon Kocatepe Üniversitesi

Teknoloji Faküktesi

Mekatronik Mühendisliği Bölümü

Danışman: Yrd.Doç.Dr. Serkan ÇAŞKA

Bu araştırmada, Robotların bilgisayar ortamında modellenip algoritma ve ara yüz geliştirme işlemlerinin robot gerçek hayatta olamadan yapabilme imkânı araştırılmış ve uygulamaya sokulmuştur. Aynı zamanda robotun simülasyon içerisinde karşılaştığı etkilere hem simülasyon hem de gerçek hayatta tepkisi ölçülmesi amacı ile bir donanım çevrimli simülatör tasarlanmıştır. Örneğin; Robot simülasyonda bir engel ile karşılaştığı zaman gerçek hayattaki robot bir engel ile karşılaşmış gibi manevra yapıp donanım testi gözlemlenmiştir.

2016, vi + 46(*) sayfa

Anahtar Kelimeler: Robot Simülasyonu, Robot yazılım geliştirme ortamı, Donanım çevrimli simülasyon, Emilatör, V-REP, Mobil Robot

ABSTRACT
Thesis

MOBILE ROBOT SIMULATOR STUDY OF ROBOT SIMULATORS AND MODELED HARDWARE IN V REP

Ahmet Yasin CİVAN

Afyon Kocatepe University

Faculty of Technology

Department of Mechatronic Engineering

Supervisor: Yrd.Doç.Dr. Serkan ÇAŞKA

In this research, the robot was modeled in the computer environment and the possibility of algorithm and interface development without using the robot in real life was researched and put into practice. At the same time, the simulator designed a hardware simulator with the aim of measuring both the simulation and the response in real life to the effects that the robot has in the simulation. For example; When the robot encountered an obstacle in the simulator, the robot maneuvered as if it were encountering a real obstacle and the hardware test was observed.

2017, vi + 46(*) pages

Keywords: Robot Simulation, Robot Software Development Environment, Hardware Cycle Simulation, Emulator, V-REP, Mobile Robot

TEŞEKKÜR

Öncelikle aile ve Abdullah Tahir CİVAN'a, lisan eğitimim boyunca üniversite hocalarıma ve her zaman bir abi hassasiyeti ile yaklaşan çok değerli hocam Yrd.Doç.Dr Serkan ÇAŞKA'ya en içten dileklerim ile teşekkür ederim.

Ahmet Yasin CİVAN
AFYONKARAHİSAR, 2017

İçindekiler

ÖZET i
ABSTRACT ii
TEŞEKKÜR iii
SİMGELER ve KISALTMALAR DİZİNİ vi
ŞEKİLLER DİZİNİ vii
1. GİRİŞ 1
1.1. Konu 1
1.2 Amaç 2
2. LİTERATÜR BİLGİLERİ 2
2.1 Simülasyon Platformları 2
2.1.1 ANYKODE MARILOU 2
2.1.2 COGMATION ROBOTSİM 3
2.1.3 MICROSOFT ROBOTICS DEVELOPER STUDIO 4
2.1.4 V-REP PRO 4
2.1.5 MATLAB ROBOTICS TOOLBOX 5
2.1.6 STDR SIMULATOR 6
2.1.7 SIMBAD 3D ROBOT SIMULATOR 6
2.1.8 MINIBLOQ ve ARDUİNO 6
2.1.9 GAZEBO 7
2.2 Literatür Değerlendirmesi 7
3. MATERYAL ve METOT 10
3.1 V-REP Simülatöründe Robot Modelinin Oluşturulması 10
3.1.1 LUA Programlama Dili 14
3.2 Robotun Ara Yüz Programının Oluşturulması 15
3.3 Robotun Fiziksel Ortamdaki Gerçek Mekanik Modeli 17
3.4 Robotun Elektronik Bileşenleri 17
3.4.1 Mikrodenetleyici STM32F051R8 17
3.4.2 Step Motor Sürücü 18
3.4.3 Step Motorlar 19
3.4.3.1 Step Motor Kullanım Alanları 21
3.4.3.2 Step Motorların Avantajları 22
3.4.3.3 Step Motorların Dezavantajları 23
3.4.4 Deney Boardı 23
3.4.5 Atlama Kablosu 24

3.5 Robotun Haberleşme Protokolleri 24
4. BULGULAR 25
4.1 Robot Algoritma Testleri 26
4.1.1 V-REP Ortamında Geliştirilen P Kontrol yazılımı 26
4.2 Simülasyonda Ara Yüz Geliştirip Gerçek Robotta Çalıştırmak 29
4.2.1. Robotun Robot Kontrol Ara yüzü İle Çalışması İçin V-REP e yazılan LUA Yazılımı 29
4.2.2. Seri Port Kontrol Ara Yüzü Yazılımı 31
4.3 Donanımı Simülasyon İle Bütünleştirmek 36
4.3.1. Donanım Çevrimli Simülasyon İçin Gömülü Sistem Programı 36
4.3.2. V-REP LUA Yazılımı 40
4.4 Robot Simülasyonların Mühendislik Eğitiminde Kullanılması 42
5. TARTIŞMA ve SONUÇ 42
Başvurular 43
ÖZGEÇMİŞ 46

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler

WMR	Wheeled Mobile Robot
4WD	Four Wheel Drive
DDV	Differentially Driven Vehicle
SSMR	Skid-Steered Mobile Robot
CAD	Computer Aided Design
4TT	Dört Tekerlekten Tahrik Edilen
2TT	İki Tekerlekten Tahrik Edilen
YKYMR	Yanal Kayma ile Yönlendirilen Mobil Robot
ADM	Ani Dönme Merkezi
KM	Kütle Merkezi
GM	Geometrik Merkez
TMR	Tekerlekli Mobil Robot
DSMR	Diferansiyel Sürüşlü Mobil Robot
ATMR	Araba Tipi Mobil Robot
BDT	Bilgisayar Destekli Tasarım
3B	Üç Boyutlu
SAE	Otomotiv Mühendisleri Topluluğu
CAN	Kontrolcü Alan Ağı
LIDAR	Lazerli Mesafe ve Nesne Algılama
EZKH	Eş Zamanlı Konum Belirleme ve Haritalama
GKF	Genişletilmiş Kalman Filtresi
MCU	Mikro Kontrolcü Birimi
CPU	Merkezi İşlemci Birimi
GPU	Grafik İşlemci Birimi
FPU	Kayan Nokta Birimi
DSP	Sayısal Sinyal İşlemcisi
IMU	Eylemsizlik Ölçme Birimi
SLA	Bakımsız Kurşun Asit
LED	Işık Yayan Diyot
CTM	Hesaplanmış Tork Yöntemi
SMC	Kayma Kipli Kontrol
VFO	Vektör Alanı Yönlendirmesi
GPS	Küresel Konumlandırma Sistemi

ŞEKİLLER DİZİNİ

Şekil 3.1 Mobil Robot V-REP Modeli 11

Şekil 3.2 Robotun V-REP Uzayındaki Görünümü 11

Şekil 3.3 Robot Simülasyonunda Robotun Basit Tanımı 12

Şekil 3.4 Robotun Mekanik Ve Sensör Elemanlarının Bağlantıları 12

Şekil 3.5 Atalet Matrisi Ve Kütle Merkezi 13

Şekil 3.6 Rijit Gövde Dinamik Özellikleri 13

Şekil 3.8 LUA dili V-REP Robot Kontrol Algoritması Örnek Kodları 14

Şekil 3.9 Seri Port bağlantı Ayarları Bölümü 15

Şekil 3.10 Seri Port Gelen Veri Ekranı 16

Şekil 3.11 Robot Kontrol Ara Yüzü 16

Şekil 3.12 Seri Port Robot Kontrol AYC Ara yüzü 17

Şekil 3.13 ARM Mikrodenetleyici Kart Görseli 18

Şekil 3.14 Step Motor Sürücü 19

Şekil 3.15 Step Motor Kullanım Şematiği 19

Şekil 3.16 Step Motor Teknik Resmi 20

Şekil 3.17 Step Motor Devir - Tork Diagramı 21

Şekil 3.18 Step Motor Görünüm 21

Şekil 3.19 Örnek Deney Bordu Kullanımı-1 24

Şekil 3.20 Örnek Deney Bordu Kullanımı-2 24

Şekil 3.21 USB TTL dönüştürücü 25

Şekil 0.1 Robot Simülasyonun C# dan hazırlanmış Robota Bağlanması 29

Grafik Listesi

Grafik 1 : Mobil Robot Simülatörlerinin toplam Akademik Yayınlarda Kullanılma sayıları [5]..... 9

Grafik 2 : Mobil Robot Simülatörlerinin Kıyaslanması [5] 9

1. GİRİŞ

1.1. Konu

Mobil robotlar ilk olarak ikinci dünya savaşı sırasında askeri olarak kullanılmaya başlanmıştır. Sonrasında belirli işleri yapmak için geliştirilen mobil robotlar, günümüzde farklı görevleri yerine getirmek üzere karşımıza çıkmaktadır [1, 2]. Robotlar istenilen görevleri yerine getirmek için tasarlanmış mekanik, elektronik ve yazılım proseslerini kesişiminden meydana gelmiş bir sistemdir.

Robotların çalıştığı ortam hava, kara ve su olabilir. Geliştirilen bazı robotlar ise bu ortamların her birinde çalışabilir. Haraketli robotlar gezgin veya mobil robotlar olarak isimlendirilmektedir. Mobil robot yapı itibari ile mekanik bir şasi, hareket etmesi için eyleyici, çevresini algılayan algılayıcılar ve tüm bu donanımı kontrol eden denetleyici elemandan oluşur. Kullanılacağı iş için tasarlanan mekanik şasi üzerine tüm donanımlar eklenir. Hareketi sağlayan eyleyici olarak çoğunlukla doğru akım elektrik motoru kullanılır. Robotun otonom olarak hareket etmesi için bir karar mekanizmasına gerek vardır. Bu işlem bir denetleyici eleman tarafından sağlanır. Denetleyici olarak, mikrodenetleyici entegreler, bilgisayarlar ve özel tasarlanan bütünleşik devreler kullanılabilir. Bu kontrolörler mobil robot üzerinde bulunan motorları kontrol ederek hareketi sağlayabilir. Hareketin verimli ve düzgün olabilmesi için ortamdan bazı sinyaller alınması gerekmektedir. Robotun yönünün, konumunun, etrafındaki nesnelerin algılanması için çeşitli algılayıcılar geliştirilmiştir. Robotun yerine getireceği işleme göre seçilen algılayıcılar şasi üzerine monte edilir. Robot üzerinde bulunan denetleyici ise verilen görev çerçevesinde algılayıcılardan aldığı bilgileri yorumlayarak motorları hareket ettirir. Yukarıda sayılan donanımlar bir mobil robotun otonom çalışabilmesi için gereklidir. [3]

Robot ve teknolojileri, mekanik, elektronik ve yazlım konularında en ileri seviyelerde çalışmaların yapıldığı bir teknolojidir. Bu çalışmaların geliştirilmesi ve iyileştirilmesi çok fazla zaman ve maliyet almaktadır. Robotik çalışmanın gerçek ortamda çalıştırıp gözlemlemeden önce bir bilgisayar ortamında robotun mekanik modelini tanımlayıp yine bu ortamda algoritma testleri ve kinematik özellikleri gözlemlenebilir.

Bu tez çalışmasında robotların bilgisayar ortamında verimli bir şekilde benzetimi yapılıp yapılamayacağı ve bu benzetim üzerinden donanımı çalıştırarak donanımdan gelen cevaplara bağılı kalarak simülasyondaki robotun kontrolü ve aynı zamanda

simülasyon içerisindeki ortamda robotun maruz kaldığı durumlarda da robotun cevap olarak verdiği etkilerin gözlenmesi üzerine bir çalışma yapılmıştır.

Hazırlamış olduğumuz robotik sistemin çok farklı ortamlarda test edilmesi gerekebilir bu ortamların her biri için bir platform hazırlamak bir test düzeneği hazırlamak çok maliyetli ve zamandan da ciddi bir kayba neden olacaktır. Bu yüzden robotun donanımını bilgisayar ortamında oluşturulmuş mekanik ve elektronik donanımlarının bire bir modeli ile eşleyerek robotun bilgisayar ortamında aldığı kararlar ile donanımın cevabı gözlenebilir.

Tüm bunların yanında robotun bilgisayar benzetimi üzerinden robotlar için ara yüzü de geliştirilebilmektedir. Bu gerçek hayatta robotun elimizde olmadan yazılım algoritmaları test edilir. Ve sisteme ekleme çıkartmalar yapılır. Böylece sisteme erken müdahale edilmiş olur. Robotlar verilmiş görevleri yerine getirme işlemi esnasında ara yüz de robotun durumu gözlemlenebilmektedir.

1.2 Amaç

Bu çalışmada robotların bilgisayar modelleri üzerinde algoritma ve ara yüz yazılımlarını pratik bir şekilde geliştirmek. Geliştirilen model ve simülasyon doğrultusunda bilgisayar da ki robot simülasyonu ile robotun donanımını iletişime geçirerek simülasyonda karşılaştığı senaryo ile donanımın tepkisini ölçme olacaktır. Şuan da kullanılan robot simülasyonlarının araştırılması ve bu simülasyonlar hakkında bilgi toplama işlemi yapılacaktır.

2. LİTERATÜR BİLGİLERİ

2.1 Simülasyon Platformları

2.1.1 ANYKODE MARILOU

1999 yılında çalışmalarına başlayan Fransız şirket mobil robot ve çevre simülasyonunu yapan Marilou ismindeki yazılımın son halini 2010 yılında satışa sunulmuştur. Programda üç boyutlu çevre tasarımı yapılabilmekte ve çalışma esnasında gerçek hayata yakın (yerçekimi, moment, ağırlık vb.) tepkiler verebilmektedir. Öğrenciler için kısıtlı yazılım ve süre için ücretsiz olarak sunulan programda, ticari olarak Pro, Project ve Edu olmak üzere üç adet lisanslama seçeneği sunulmaktadır [4]

Yazılım geliştirme ve simülasyon aşamalarını; ilk olarak robot tasarımı yapılması, sonra çevrenin üç boyutlu(3D) olarak yapılandırılması ve en son olarak robot kontrol yazılımı

geliştirilerek simülasyon yapılması olarak sınıflandırabilir. Programda mobil robot tasarımı esnekçe yapılabilmektedir. 3D çizim ortamında mobil robotun, motor konumları, gövde yapısı belirlenebilmektedir. Programda, mobil robot ile çevre arasında veri akışını sağlamak için birçok algılayıcı bulunmaktadır. Tasarlanan bir mobil robota jiroskop, GPS, Lidar gibi algılayıcılar ekleyerek robot kontrol yazılımı geliştirilebilir. Yazılım geliştirme aşamalarında ise Marilou çeşitlilik sunmaktadır. .NET tabanlı yazılımlardan C, C++, C++ CLI, C#, VB # ve J # gibi dilleri desteklemektedir. Bu mobil robot tasarımcısı için birçok kolaylık sağlamaktadır. Ayrıca programın, MATLAB, Bonavision iRSP gibi yazılımlarla da geliştirilmesi farklı yazılım ortamlarda, farklı algoritmaların denenebilmesine imkan sağlayacaktır. Kontrol yazılımlarının geliştirilmesinin ardından derlenen program ile robotu 3D ortamda davranışları gözlemlenebilmektedir. Program gerçek dünyadan fiziksel verileri gönderme ve alma gibi bir özellik de sunmaktadır. Bu sayede gerçek zamanlı çalışmalar yapılabilir. [5]

Çok yönlü tekerleklerin kinematik analizinin yapıldığı, [6], çok robotlu ve çok kullanıcılı ortamda dağıtık fonksiyonların analizinin yapıldığı [7], mobil robotlarda bulanık durum haritalamasının yapıldığı [8] bu ve bunun gibi birçok çalışmada Marilou Robotik Simülatörü kullanılmaktadır

2.1.2 COGMATION ROBOTSİM

LEGO grubunun desteklediği Canada'da 2005 yılından itibaren faaliyet gösteren Cogmation Robotics firmasının ürettiği yazılımdır. LEGO® MINDSTORMS denetleyici ve fiziksel parçaların bulunduğu yazılımda geliştirilen projelerin, gerçek ürünler üzerinde denenebilme imkanı bulunmaktadır. LEGO ürünlerinin yanı sıra iROBOT, Nao insansı robotunun da 3D simülasyon ve yazılım geliştirme işlemleri kolaylıkla yapılabilmektedir. Marilou yazılımda olduğu gibi çevre fizik kanunlarına uygun olarak oluşturulabilmektedir. Bu yazılımda kullanılan robotlar gerçek robottur. Aynı firmanın robotBuilder yazılımı ise robot tasarımı için geliştirilmiştir. Ancak yapılan tasarımlar LEGO veya standart parçaların birleştirilmesi ile sağlanmaktadır. Bu simülasyonu yapılan robotun, gerçekte de kullanabilmesine imkan sağlamaktadır. C++ API kullanılarak geliştirilen altyapı ile robotların programlanması herhangi bir programlama dili kullanılarak yapılabilmektedir. Geliştirilen NI LabVIEW Kütüphanesi

ile doğrudan LabVIEW üzerinden kontrol imkanı sunulmaktadır. Mekanik ve elektronik bilgisinin çok fazla bilinmese de projelerin geliştirmesindeki kolaylık ve yazılımın basit oluşu sebebi ile robotSim orta öğretim öğrencileri tarafından çok kullanılmaktadır [9]. LEGO NXT veya EV3 denetleyicileri kullanılarak yapılan akademik araştırmalarda da robotSim yazılımı kullanılarak çeşitli simülasyonlar yapılmaktadır. [10] [11]

2.1.3 MICROSOFT ROBOTICS DEVELOPER STUDIO

Microsoft tarafından 2006 yılından itibaren geliştirilen Windows tabanlı mobil robot kontrol ve simülasyon programıdır. En son 2012 yılında Robotics Developer Studio 4 versiyonu çıkmıştır. NET tabanlı geliştirilen uygulama çoğunlukla blok programlama şeklinde geliştirilmektedir. Visual Programming Language adı verilen bu dil önceden tanımlanmış çeşitli blokların art arda bağlanması ile çalışmaktadır. Bilinen programla dillerinin aksine, dil robot kontrolü ve çevre oluşturmada bazı sınırlar getirmektedir. Bu platformda daha önceden tanımlanmış gerçek olarak da üretilen iRobot, LEGO NXT gibi mobil robotlar mevcuttur. Algılayıcı kütüphanesi diğer yazılımlara göre daha az ve sınırlıdır. Fakat birçok uygulama için gerekli olan mesafe sensörü, ip kamera gibi önemli cihazlar mevcuttur. 3D çevre yapılandırması, önceden tanımlanmış modeller kullanarak sağlanmaktadır. Microsoft tarafında geliştirilmesi yavaş ilerleyen bu yazılım ücretsiz olarak sunulmaktadır. [12]

X MAI ve arkadaşlarının yapmış oldu yörünge planlama simülasyonu [13], robotlarda model bir sürüş yaklaşımı [14], labirent çözümü [15] gibi birçok mobil robot uygulamasında Microsoft Robotics Developer Studio kullanılmaktadır. Ayrıca yazılım ile robot kol simülasyonu ve kontrolü de yapılabilmektedir. [16]

2.1.4 V-REP PRO

İlk sürümünü 2010 yılında çıkartan İsviçre firması Coppeliarobotics'in yapmış olduğu güncelleştirmeler ile 2014 yılında V-REP 3.1.1. sürümünü çıkarmıştır. Birçok faklı işletim sistemi üzerinde çalışabilen program yapı itibari Marilou yazılımına benzemektedir. 3D çevre tasarımının basitçe yapılabildiği programda zengin çevre araçları, robot ve robot algılayıcıları ile büyük bir kullanım kolaylığı sunmaktadır. V-REP gerçek fiziki ortamı, ODE, BULLET ve VORTEX kütüphanelerini kullanarak sağlamaktadır. Kontrolör yazılımı geliştirilmek için sunulan C/C++, Python, Java, Lua,

Matlab, Octave ve Urbi dilleri farklı kullanıcılar için adaptasyon süreçlerini kısaltır. Kontrol yazılımı geliştirilmesi 7 farklı dil için dışarıda yazılarak veya programın kendi kod parçaları değiştirilerek yapılabilmektedir. Akademik çalışmalarda geliştirilen yapay zeka algoritmalarını denemek için Pyton, Matlab gibi yazılımlarının kullanımı, robot programlama ve simülasyon için büyük kolaylık sağlamaktadır. Program içerisinde fiziksel robot mevcuttur. Bunlardan bazıları E-puck, hexapot, Nao, line follower vb. [17]

V-REP mobil robotları simülasyonunun yanı sıra robot kol simülasyonu da yapabilmektedir. Kinematik ve ters kinematik denklerimi gibi robot kol problemlerinin çözümü kolaylıkla sağlanmaktadır. Yine programda tanımlı olarak Kuka, Jabo ve ABB robot kolları hazır olarak sunulmaktadır. [5].

Akademik olarak, Nero-PID kontrollü yürüyüş kontrolünün yapıldığı çalışma [18], adım uyumlu endüstriyel programlama sisteminin geliştirildiği çalışma [19], parçacık sürü optimizasyon yöntemi ile insansı robot geliştirildiği bu ve bunun gibi bir çok çalışmada V-REP robot simülatörü kullanılmıştır. [20]

2.1.5 MATLAB ROBOTICS TOOLBOX

Açık kaynak kodlu bir araç kutusudur. Geliştirilen kodlar ücretsiz ve geliştirilebilir şekilde sunulmasına karşın Robotics Toolbox'ın çalıştırılabilmesi için MATLAB programının yüklü olması gerekmektedir. Robotics toolbox Peter Corke tarafından 2011 yılında geliştirilmeye başlanmış ve 2014 yılına kadar 9 güncelleme çıkmıştır. Matlab ortamında geliştirilen kodlardan bazıları; döndürme matrisleri, kinematik, ters kinematik, jakobiyen matrisleri, mobil robotlar için quadrator, Bug, D* algoritma fonksiyonlarıdır. Bu fonksiyonların hazır olarak sunulması ve MATLAB programının altyapısı ile robot programlanabilir ve sonuçlar analiz edilebilir. 3D çevre ve simülasyonun sunulmadığı programda sonuçlar, grafikler ve tablolar şeklinde alınmaktadır. Farklı kontrol algoritmalarının da rahatlıkla kullanılabildiği Matlab ortamında yazılacak robot algoritmalarının denenmesinde büyük kolaylık sağlayabilecek bir araç kutusudur. Robotics Toolbox kullanımı için ücretsiz olarak elektronik kitap sunulmaktadır. Robot kol simülasyonu, robotlar için kontrolör tasarımı gibi birçok akademik çalışma bu araç kutusu kullanılarak yapılmıştır [21, 22, 23]

2.1.6 STDR SIMULATOR

Açık kaynak kodlu Linux ortamında geliştirilen ücretsiz bir yazılımdır. Yazılım iki boyutlu(2D) olarak bir veya birden fazla mobil robotun çalışma simülasyonunu yapabilmektedir. Grafiksel bir çevreninin sunulmadığı yazılımda labirent, duvar gibi nesne algılama, birden fazla robot simülasyonu gibi denemeler yapılabilmektedir. STDR, ROS Kütüphanesi ile çalışmaktadır. Yazılım derlenmesi için yine ROS tabanında bulunan derleyiciler kullanılmaktadır. Basit bir grafik arayüzü bulunan programda labirent oluşturma robot ekleme işlemleri yapılabilmektedir. Robot kontrolü için iyi bir programlama bilgisi gerekmektedir. Linux ortamında çalışan kullanıcıların basit bir arayüzde ile bir veya birden fazla robot haberleşmesi, kontrolü, birlikte çalışma gibi işlemleri yapmasına imkan sağlamaktadır [24]

2.1.7 SIMBAD 3D ROBOT SIMULATOR

Eğitim ve akademik çalışmaların denebilmesi amacı ile geliştirmiş Java tabanlı 3D robot simülatörüdür [25]. Gerçek bir 3D çevre sunmamasına karşın, Java platformunda geliştirilen kod kütüphanelerinin kullanılması ile yapay zeka tabanlı otonom robotların programlamasında büyük kolaylık sağlamaktadır. Program geliştirmesi Java ortamında robot kontrolü ve algılayıcıların önceden tanımlanmış kod parçalarını kullanması ile sağlanmaktadır. Bu sebeple yazılım geliştirmek için iyi bir programlama bilgisi gerekmektedir. Simbad 3d Robot Simulator sunduğu değiştirilebilir basit simülasyon arayüzü ile robot simülasyonu yapmayı sağlamaktadır. Ücretsiz olarak sunulan bu yazılım için Java3D yazılımının yüklenmesi gerekmektedir. Yazılım için geliştirilen web sitesinde programla ile ilgili örnek kodlara ve dokümantasyona ulaşılabilmektedir. [26]

Mobil algılama simülasyonun yapıldığı [27], otonom robot yol planlanmasının yapıldığı [28] bu ve benzer çalışmalarda bu yazılım kullanılmıştır.

2.1.8 MINIBLOQ ve ARDUİNO

miniBloq tam anlamı ile bir robot simülatörü değildir. Program Multiplo™, Arduino™ platformlarını grafiksel programlanması için geliştirilen ücretsiz bir arabirimdir [29]. Kod yazmakta zorlananlar için geliştirilmiş olan blok programlama arayüzüne sahiptir. miniBloq ile Arduino Uno, Mega, Pi-Bot, Red-Bot gibi mobil robotların kontrol

yazılımları grafiksel olarak geliştirilebilir ve fiziksel cihazlara yüklenebilir. Yazılım programlanmaya yeni başlayanlar için oldukça eğiticidir [30, 31].

Arduino platformu son yılların en hızlı gelişen platformlarındandır. Ücretsiz, açık kaynak kodlu kolay geliştirilebilen yazılım altyapısı ve ekonomik donanım seçenekleri ile denemeler yapmaya imkan sağlamaktadır. Hızlı gelişen yazlım kütüphaneleri ve donanım araçları ile bir mobil robotun kontrolü ve donanımı oluşturulabilmektedir. Matlab, Labview, .NET gibi ortamlarda geliştirilen kütüphaneler ile geliştirilen yazılımların denenmesi ve fiziksel simülasyonu kolaylıkla yapılabilmektedir [32].

2.1.9 GAZEBO

Gazebo 2002 yılında Güney Kaliforniya üniversitesinde geliştirilmeye başlanmıştır. Yüksek performanslı fiziksel işlem mekanizması bulunan bu yazılım ODE, Bullet, Simbody ve DART gibi kütüphaneleri içerir. Sadece Linux ortamında çalışan yazılım robotların derlenmesi için C++ dilini kullanmaktadır.

Kontrol yazılımının geliştirmesi yazılım Linux ortamında derlenmesi ile yapılmaktadır. Bu sebeple dil(yazılım) bilgisinin iyi seviyede bilinmesi gerekmektedir. Gazebo platformunda robot simülasyonu ve 3D gerçekçi çevre oluşturulması oldukça basittir. Çoğunlukla bilgisayar mühendislerinin tercih ettiği yazılım oldukça esnek ve ileri seviye robot simülasyonlarının yapılmasına olanak sağlamaktadır. Gazebo tarafından sunulan hazır fiziksel modellerin yanı sıra sunulan hazır robotlarda bulunmaktadır. Bunlardan bazıları iRobot, Pioneer, Kinect v.b. Program mobil robot programlamanın yanı sıra robot kol simülasyonuna da izin vermektedir. Açık kaynak kodlu ve kod geliştirmeye açık olan bu program birçok akademik çalışmada kullanılmıştır [33]. Bunlardan bazıları montaj hattı için robot asistan kontrolünün yapıldığı çalışma [34], insansı robotların internet üzerinden denetlendiği çalışma [35] ve benzerleri gibidir.

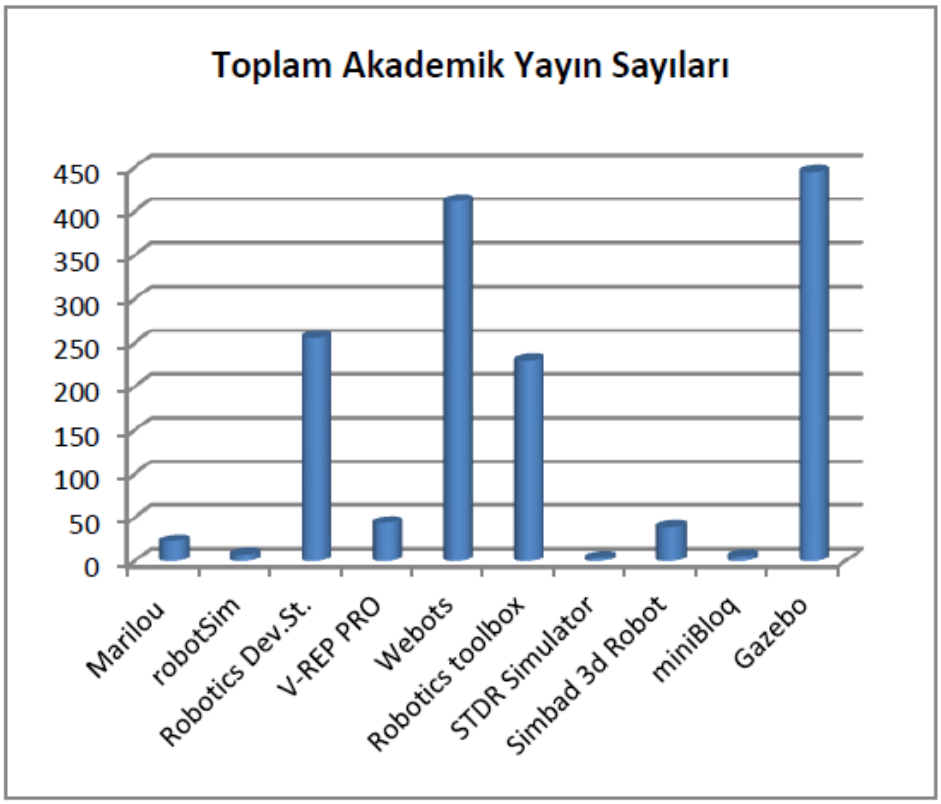
2.2 Literatür Değerlendirmesi

Mobil robot simülasyon platformlarının incelendiği bu çalışmada 5'i ticari olmak üzere 10 adet program ele alınmıştır. Programların kullanım özellikleri, lisans seçenekleri, kullanım yerleri, çalıştığı işletim sistemleri gibi parametreler ilgili bölümlerde detaylı olarak incelenmiştir. Grafik 1'de programlar kullanılarak yapılan toplam akademik yayın sayısı gösterilmektedir. Kullanım kolaylığı gibi bilgiler ise programlar bilgisayar

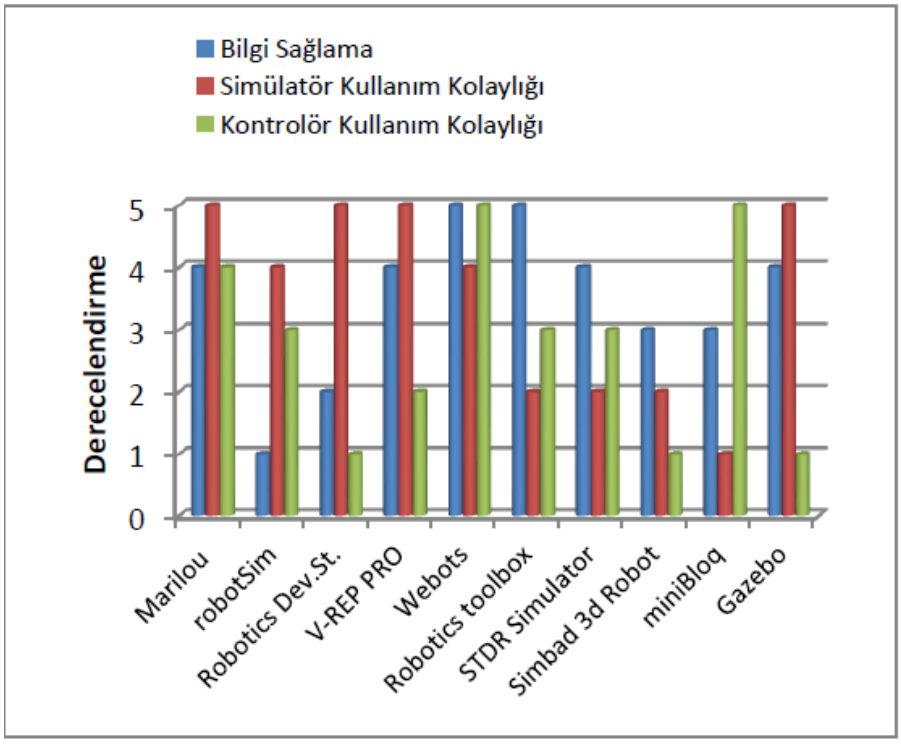
ortamında çalıştırılarak denenmiş ve internet üzerinden program yorumları incelenerek verilmiştir. Grafik 2'de programların birbirlerini göre avantajları gösterilmektedir. Yapılan bu araştırmada Linux ortamında çalışanların çoğunlukla Gazebo Simülatörünü kullandığını görmekteyiz. Windows ortamında ise Microsoft firmasının ürettiği Devoloper Studio yaygın olarak kullanılmaktadır. [5]

Devoloper Studio'nun desteği ve geliştirilmesinin yavaş oluşu programın günümüzdeki akademik yayınlarda kullanımını azaltmıştır. Tüm platformlarda çalışan ve akademik çalışmalarda kullanım kolaylığı ile tercih edilen Webots yazılımının kullanımının yaygın olduğu görülmektedir. Aynı şekilde 3D çevre sunmamasına karşın Matlab altyapısı ile esnek bir programlama sunan Robotics Toolbox for Matlab yayın olarak kullanılmaktadır. İncelenen bu platformlarda, programların yayınlandığı zamanın yeni olması ve geliştirme çalışmaların devam etmesi, ilerleyen günlerde diğer programların da akademik çalışmalarda kullanabileceğini göstermektedir. Yazılımın, üretildiği ülkelerdeki yayınlanan akademik çalışmalarda, o yazılımın kullanım sayısının yüksek olduğu da incelemelerde rastlanmıştır [5].

Mobil robotlar üzerinde çalışacak akademisyenler için bu çalışmada sunulan platformlar ile bilgisayar ortamında modelleme ve simülasyon yapmak, maddi kazanç ve zamandan tasarruf sağlayacaktır. Araştırmada sunulan bilgiler ile bu konuda çalışacak kişiler yeteneklerine yatkın simülatörler seçerlerse çalışmalarda yazılımla uğraşmaktan çok sonuca yönelik verimli çalışmalar yapılabileceklerdir [5].



Grafik 1 : Mobil Robot Simülatörlerinin toplam Akademik Yayınlarda Kullanılma sayıları **[5]**



Grafik 2 : Mobil Robot Simülatörlerinin Kıyaslanması **[5]**

3. MATERYAL ve METOT

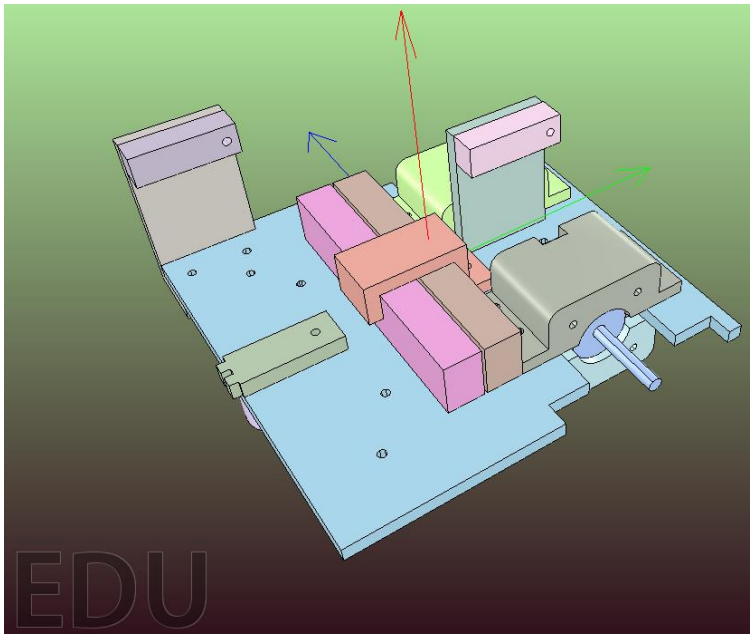
3.1 V-REP Simülatöründe Robot Modelinin Oluşturulması

Robotların bilgisayar ortamında modellenmesi konusunda 10 adet simülatör programı incelenmiştir, kullanım kolaylığı ve işlevselliği bakımından V-REP programı seçilmiştir. V-REP robot simülatör ortamında robotun yazılımı LUA dilinde hazırlanarak robotun seri port üzerinden gerçek robot ile haberleşme işlemi çok büyük bir kolaylık ile gerçekleşmiştir. Bu sebeplerden dolayı V-REP programı uygun görülerek mobil robotumuz bu platformda modellenmiş ve kinematik özellikleri gözlemlenmiştir.

İlk sürümünü 2010 yılında çıkartan İsviçre firması Coppeliarobotics'in yapmış olduğu güncelleştirmeler ile 2014 yılında V-REP 3.1.1. sürümünü çıkarmıştır. Birçok faklı işletim sistemi üzerinde çalışabilen program yapı itibari Marilou yazılımına benzemektedir. 3D çevre tasarımının basitçe yapılabildiği programda zengin çevre araçları, robot ve robot algılayıcıları ile büyük bir kullanım kolaylığı sunmaktadır. V-REP gerçek fiziki ortamı, ODE, BULLET ve VORTEX kütüphanelerini kullanarak sağlamaktadır. Kontrolör yazılımı geliştirilmek için sunulan C/C++, Python, Java, Lua, Matlab, Octave ve Urbi dilleri farklı kullanıcılar için adaptasyon süreçlerini kısaltır. Kontrol yazılımı geliştirilmesi 7 farklı dil için dışarıda yazılarak veya programın kendi kod parçaları değiştirilerek yapılabilmektedir. Akademik çalışmalarda geliştirilen yapay zeka algoritmalarını denemek için Pyton, Matlab gibi yazılımlarının kullanımı, robot programlama ve simülasyon için büyük kolaylık sağlamaktadır. Program içerisinde fiziksel robot mevcuttur. Bunlardan bazıları E-puck, hexapot, Nao, line follower vb. [17]

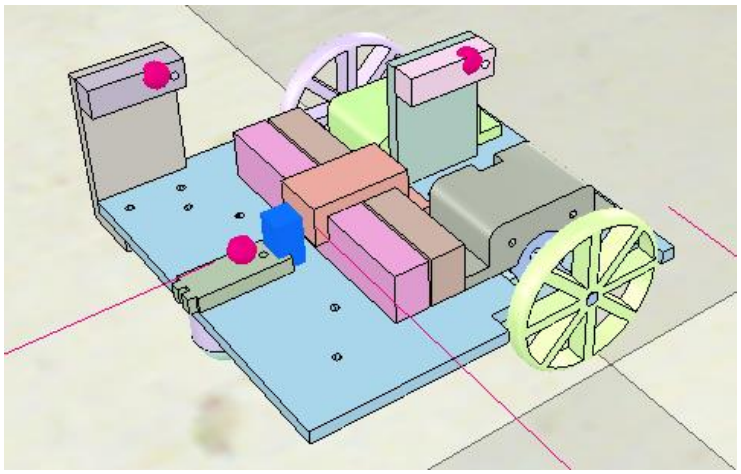
V-REP mobil robotları simülasyonunun yanı sıra robot kol simülasyonu da yapabilmektedir. Kinematik ve ters kinematik denklerimi gibi robot kol problemlerinin çözümü kolaylıkla sağlanmaktadır. Yine programda tanımlı olarak Kuka, Jabo ve ABB robot kolları hazır olarak sunulmaktadır. [5].

Bir CAD programında 3B tasarımını yapmış olduğumuz robot modelini .obj formatında bilgisayarımıza kayıt edip V-REP programına çağırıyoruz. Bu obje Şekil 3.1' de gösterildiği gibi programın içerisinde gelmektedir.



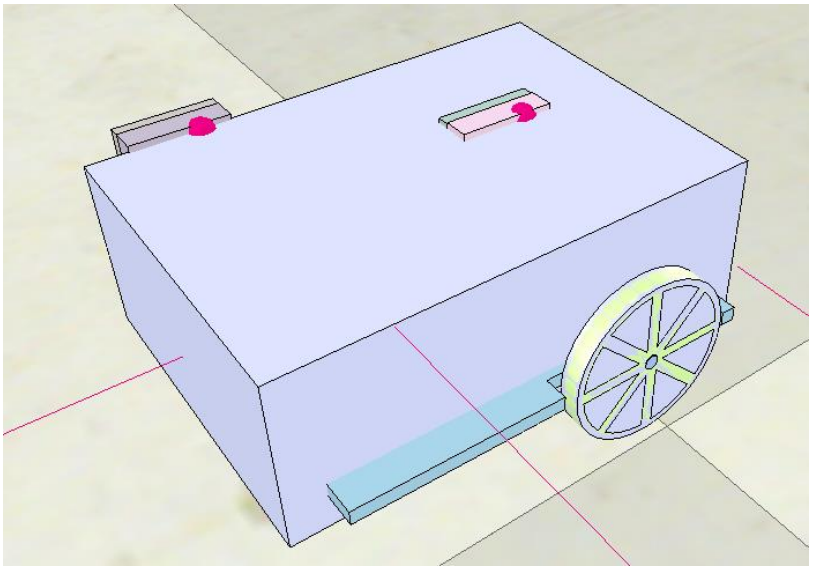
Şekil 3.1 Mobil Robot V-REP Modeli

Bu şekli itibari ile program ile sadece görsel bir değer olmaktan fazlasını ifade etmemektedir. Şekil 3.2' de robotun V-REP uzayında gösterimi



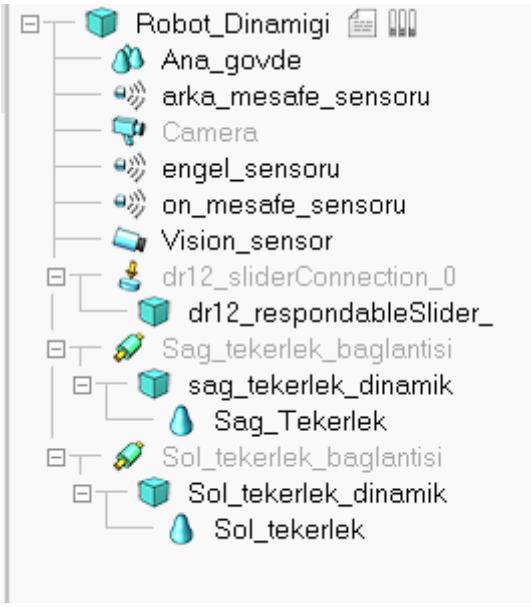
Şekil 3.2 Robotun V-REP Uzayındaki Görünümü

V-REP'e eklenen robotun kinematik bir anlam kazanabilmesi için program içerisinde ağırlığı olan bir küp oluşturuyoruz ve bu küpü robot objesi ile ilişkilendirerek robotun simülasyon içerisinde kinematik bir anlam kazanmasını sağlıyoruz. Şekil 3-3' bu durum ekran görüntüsü ile açıklanmaktadır.



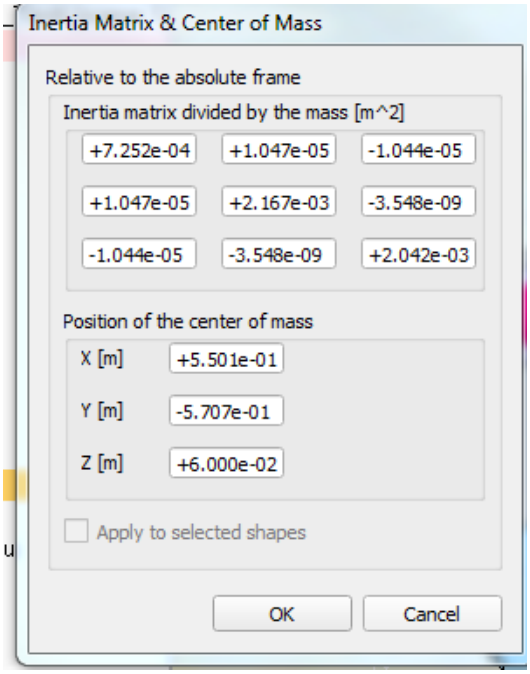
Şekil 3.3 Robot Simülasyonunda Robotun Basit Tanımı

Robotun simülasyondaki ağaç bağlantıları Şekil 3.4'de gösterilmektedir. Bu ağaç bağlantısında robotun sensör tanımları ve dönel kuvvet olan motorların da bağlantıları gösterilmektedir.



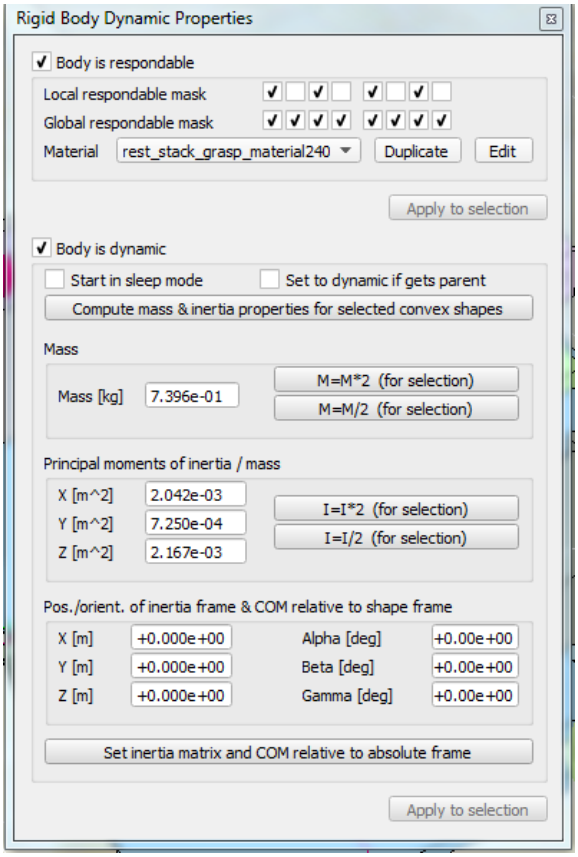
Şekil 3.4 Robotun Mekanik Ve Sensör Elemanlarının Bağlantıları

Robotun dinamik ayarları Şekil 3.5'de verilmiştir. Bu değerler olabildiğince gerçeğe yakın verilmiştir. Hassas ölçümler yapılmamıştır sadece algoritma ve donanım çevrimi için gözlemlemek için ayarlar yapılmıştır.



Şekil 3.5 Atalet Matrisi Ve Kütle Merkezi

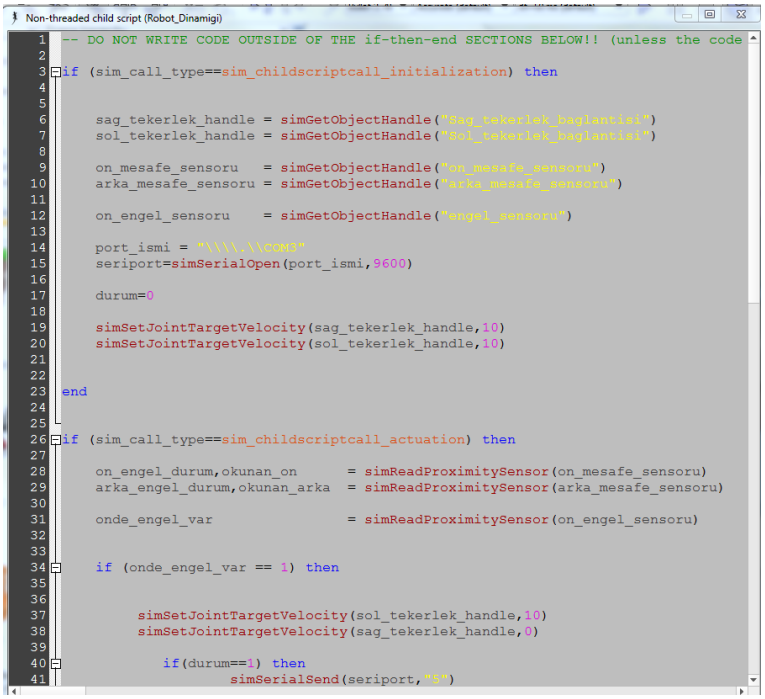
Rijit katı cisim dinamik ayarları Şekil 3.6'da verilmiştir.



Şekil 3.6 Rijit Gövde Dinamik Özellikleri

Robotu mekanik olarak programa tanıttıktan sonra yazılım aşamasında robotun

haraketli parçalarının isimleri tanımlanıyor böylece robot simülasyonda yazılım kararı ile hareket etme yeteneği kazanacaktır. Şekil 3.7' de bulunan robotun yazılımdan örnek bir ekran görüntüsü verilmiştir.



Şekil 3.8 LUA dili V-REP Robot Kontrol Algoritması Örnek Kodları

3.1.1 LUA Programlama Dili

Lua, ağırlıklı olarak gömülü sistemler ve istemciler için tasarlanmış hafif paralel bir programlama dilidir. Lua, ANSI C'de yazılmış olduğu için çapraz platform destekli bir dildir ve nispeten basit bir C API'sine sahiptir.

Lua başlangıçta 1993 yılında özelleştirmeye olan artan talebi karşılamakta yazılım uygulamalarını genişletmek için bir dil olarak tasarlandı. Çoğu prosedürel programlama dillerinin temel olanaklarını sağlamış ancak daha karmaşık veya alana özgü özellikler dahil edilmemiştir; bunun yerine, programcının bu özellikleri uygulamasına imkân tanıyan, dilin genişletilmesi için mekanizmalar içermekteydi. Lua, genel gömülebilir bir uzantı dili olarak tasarlandığından, Lua tasarımcıları, hızını, taşınabilirliğini, genişletilebilirliğini ve geliştirilmesinde kullanım kolaylığını geliştirmeye odaklandı. [36]

V-REP programında da kolay ve esnek bir yapıya sahip olduğu için kullanılmıştır. Bu yüzden hızlı bir şekilde istenilen robot algoritması oluşturduk.

LUA programlama dilinde bir çeşit gömülü sistem programı benzerliği ile kolay bir şekilde istenilen sisteme de entegre edilebilir.

3.2 Robotun Ara Yüz Programının Oluşturulması

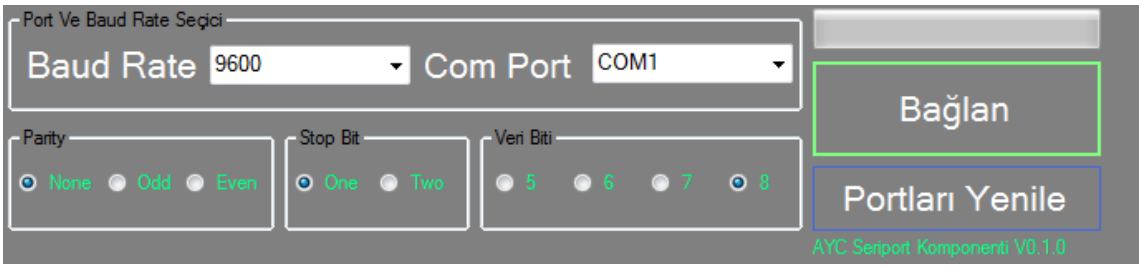
Robotun kontrolü ve sensörlerin algıladığı değerlerin bilgisayar ekranında görüntülenmesi içi bir programa ihtiyaç duyulmaktadır. Bu program robota seri port üzerinden bağlanacak ve robotun hareketlerini düz, sağa, sola, dur olmak üzere 4 farklı komut göndererek robot kullanıcı tarafından bilgisayar üzerinden kontrol edilmesine yarayacaktır.

Bu ara yüz programı visual studio'dan C# programlama dili ile hazırlanmıştır. Son derece basit bir programdır. Öncelikler seriport üzerinden Robotun bağlı olduğu port seçilir ve program ile bağlantı kurması sağlanır.

Seri port programında C# kullanılmasının en temel sebeplerinden birisi bu programın nesne tabanlı bir programlamayı destekliyor olmasıdır. Aynı zamanda bu arayüz OOP standartlarına dayalı olarak yazılmıştır.

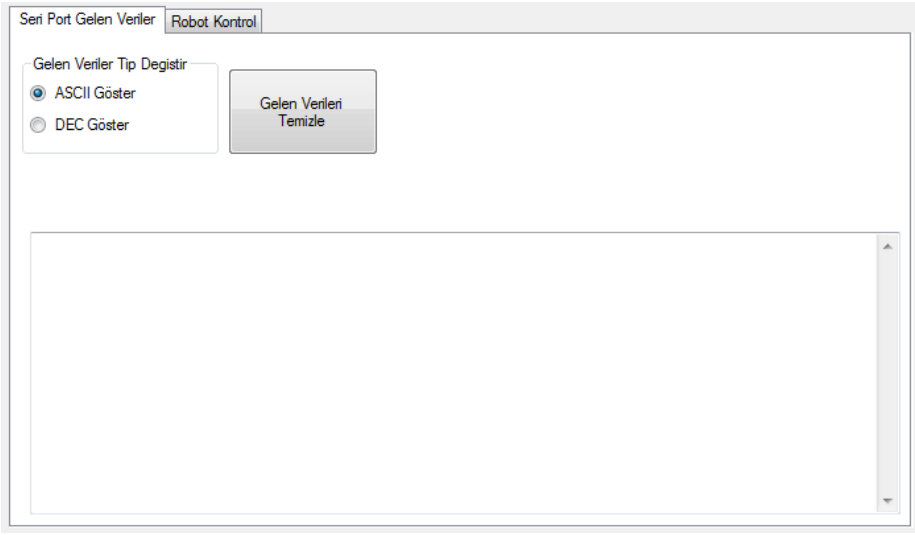
Program toplam 3 aşamadan oluşur, birinci aşamada programın seri porta bağlanma işlemi, ikinci aşama robottan veri okumam ve son aşamada robota veri gönderme.

Seri port bağlanma görseli, bu bölümde seri porta bağlantı gerçekleştirilir ve iletişim için gerekli ayarlar yapılır. Şekil 3.8 tez bünyesinde yazdığımız seri port bağlantı ara yüzünün ayarlar bölümü görülmektedir.



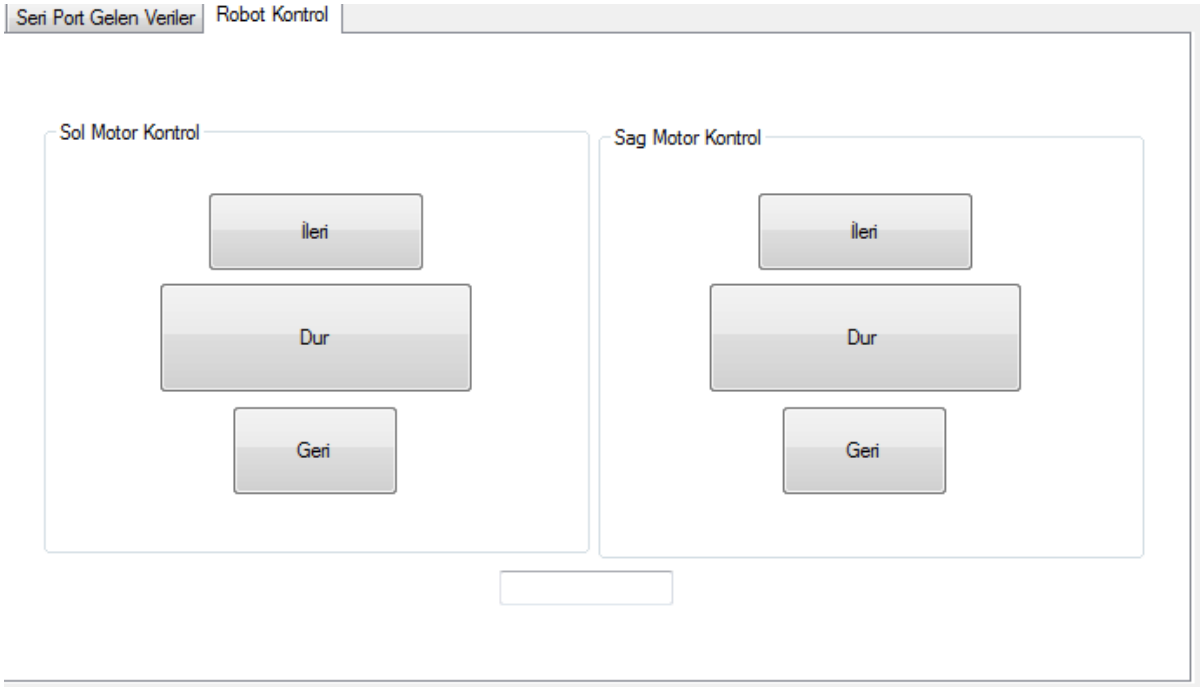
Şekil 3.9 Seri Port bağlantı Ayarları Bölümü

Seri port üzerinden gelen verilerin yazdırılması ve robotun o an ki durumu hakkında bilgi sahibi olmak için seri port iletişim ara yüzümüze bir de gelen veri sekmesi eklenmiştir. Robot üzerinden gelen bilgiler bu ekranda ardı ardına sıralanmaktadır. Robotun sensör bilgileri burada görülmektedir.(Şekil 3.9)



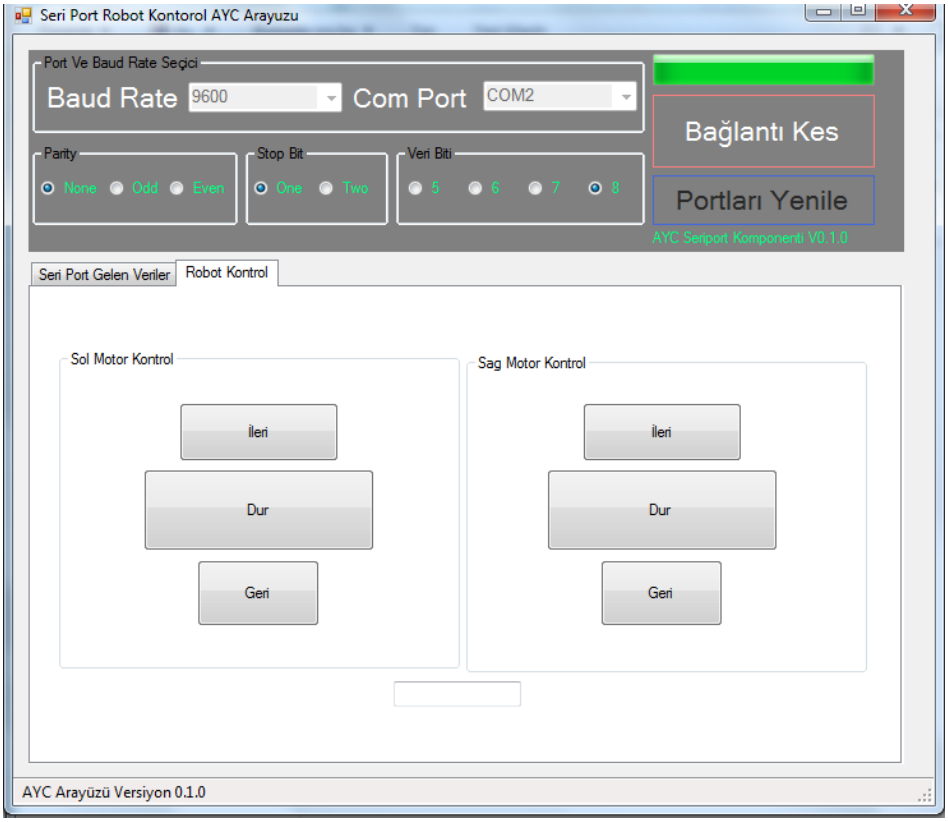
Şekil 3.10 Seri Port Gelen Veri Ekranı

Robot kontrol bölümünde ise robotu butonlar yardımı ile sağ, sol, ileri ve dur işlemleri yapılmaktadır. Bu basit ara yüz çok daha karmaşıklarının hazırlanabileceğini göstermek amacı ile yapılmıştır. Şekil 3.10' bu robot kontrol ara yüzü görülmektedir. Bu ara yüzde bulunan butonlar ile robota bir bilgi paketi gönderilmektedir, robotta seri port üzerinden gelen bu bilgi paketini çözerek istenilen vazifeyi gerçekleştirmektedir.



Şekil 3.11 Robot Kontrol Ara Yüzü

Seri port ara yüzünün genel görüntüsü Şekil 3.11'de ki gibidir. Bu ara yüzde her bir motor için yarı ayrı kontrol butonları kullanılmaktadır.



Şekil 3.12 Seri Port Robot Kontrol AYC Ara yüzü

3.3 Robotun Fiziksel Ortamdaki Gerçek Mekanik Modeli

Robot 3B yazıcıdan hızlı ve kullanışlı bir şekilde çıkarıldı bu çalışmada robot masanın üzerinde tekerlekleri havada duracak şekilde test edilmekte. Bu yüzden mekanik tasarımlarda hassasiyet ve yük ayarları çok hassas bir şekilde yapılmadı.

3.4 Robotun Elektronik Bileşenleri

3.4.1 Mikrodenetleyici STM32F051R8

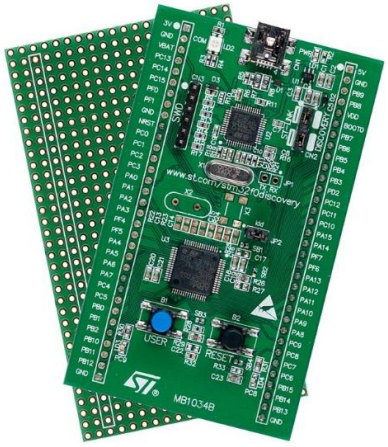
Fiziksel ortamda ki gerçek robotta mikrodenetleyici olarak STM32F051R8 discovery kit kullanılmıştır. Bu ARM mimarisine sahip mikrodenetleyeci çok güçlü bir cihazdır. Aynı zamanda çok da hızlıdır. Projemizde STM32f051R8 mikrodenetleyicisinin bilgisayar ile haberleşmesi için UART birimini ve step motorları sürmek için PWM kanalları kullanılmaktadır.

STM32F0 DİSCOVERY düşük maliyetli olup hızlı ve kolay kullanılabilen bir geliştirme kitidir

Üzerinde STM32F051R8T6 işlemci barındırır.STM32 F0 seri 32-bit ARM® Cortex™ mikroişlemcisi, ST-LİNK/V2 gömülü debug aracı, Led, push buton ve kolay kullanım

için prototip bir kart içermektedir. STM32F051R8T6 mikrodenetleyici kartı 64 KB Flash, 8 KB RAM bulunmaktadır. Kartın çalışma frekansı 48MHz dir. 6 Mbit/s USART, 18 Mbit/s SPI, 1 Mbit/s I²C, 5x 16-bit PWM, 1x 32-bit PWM, 12 MHz I/O özelliklerine sahiptir.

Projede kullanılan ARM discovery kartı Şekil 3.13'de gösterilmiştir.



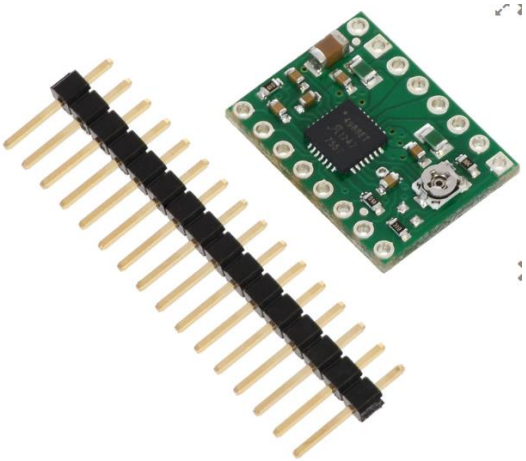
Şekil 3.13 ARM Mikrodenetleyici Kart Görseli

3.4.2 Step Motor Sürücü

Robotik sistemimizde kararlı çalışması ve kontrol kolaylığı sebebi ile step motor tercih edilmiştir. Step motor sürücü fotoğrafı Şekil 3.14'de verilmiştir.

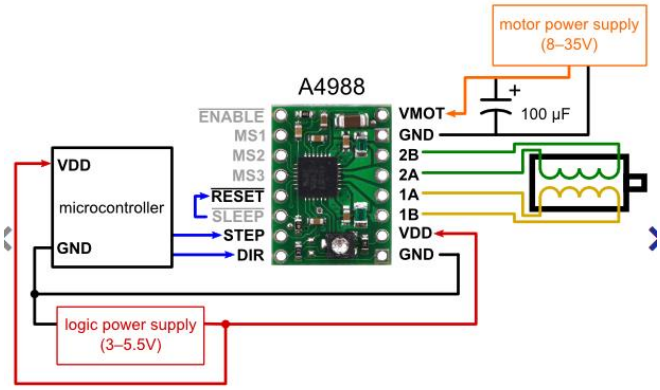
Voltaj regülatörlü A4988 step motor sürücü kartı Allegro'lar için kullanımı kolay mikrostep çift kutuplu bir step motor sürücüsüdür. [37]

Kartın üzerinde,ayrı bir locik ve motor temini için ihtiyacı ortadan kaldırmak için iki ayrı voltaj regülatörü (5 V ve 3,3 V) bulunmaktadır. Sürücü mevcut akım limitlemeyi gösterir ve beş farklı Mikro step çözünürlükleri sınırlı özellikleri ayarlanabilir. 8 - 35 V arası çalışmakta ve her bir bobin için 2 A akım vermektedir. Bir Bipolar Step motor sürebilmektedir. [37]



Şekil 3.14 Step Motor Sürücü

Step motor kullanım şematiği Şekil 3.15'de verilmiştir. Aşağıdaki şematik izlenerek devre deney bordu üzerine kurulmuştur.



Şekil 3.15 Step Motor Kullanım Şematiği

3.4.3 Step Motorlar

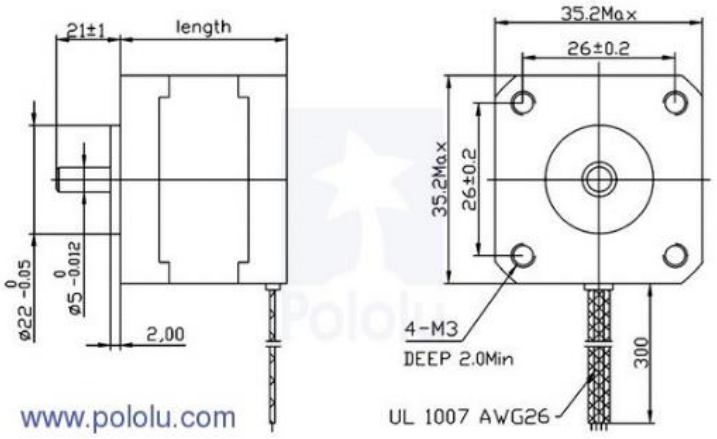
Mobil robot test sistemimizde kararlı çalışması ve kullanım kolaylığı açısından step motor tercih edilmiştir. Bu step motorlar Kolay kullanımlı NEMA serisi bipolar step motorlar ile konum kontrolü uygulamalarınızı gerçekleştirebilir, motor devrini step motor sürücü kartlar ile ayarlayabilir

Step motorlar (adım motorları), giriĞlerine uygulanan darbe dizilerine karĞılık (digital veya sayısal giriĞ), rotorunda analog dönme hareketi yapabilen elektromagnetik elemanlardır. Bu özellikleri nedeniyle "dijital makina" olarak da tanınan step motorlar, dijital sistemlerle çok rahat bir Ğekilde kullanılabilirler. Step motorlar, geleneksel elektrik motorlarından farklı olarak, hareketini adım-adım yapan motorlardır . Adından

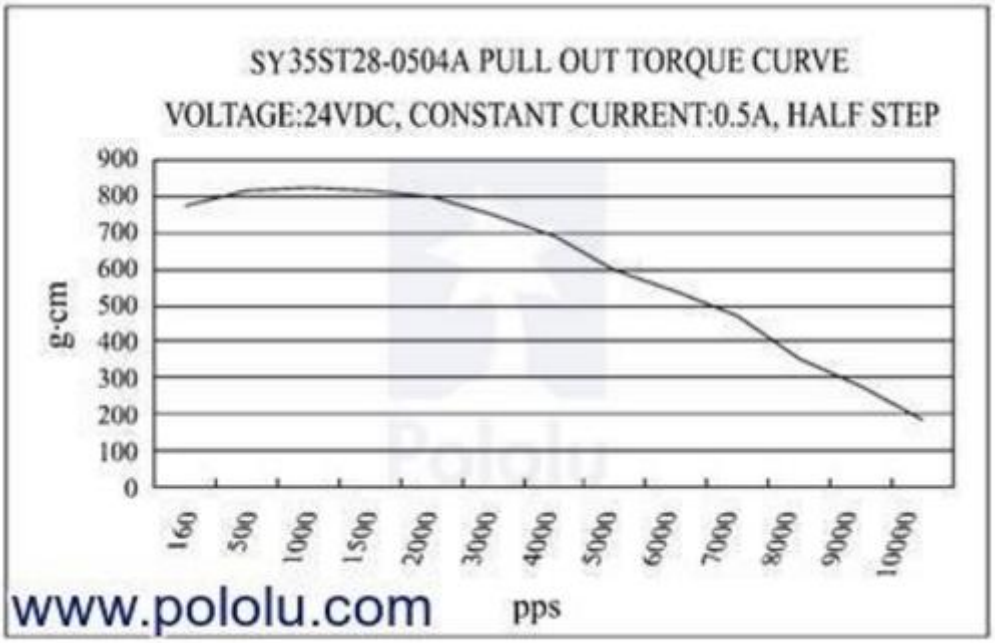
da anlaĢılacağı üzere belirli adımlarla hareket ederek rotorun açısal konumunu değiĢtirirler. Adım açıları motorun sargılarına uygun sinyaller gönderilerek kontrol edilir. Motorun uyartılmasıyla birlikte, rotorun yapacağı hareketin ne kadar olacağı, motorun adım açısına bağlıdır. Motorun yapısına bağlı olarak, adım açısı 90, 45, 18, 7,5, 1,8… derece veya daha farklı açılarda olabilir. Motora uygulanan sinyallerin frekansı değiĢtirilerek motorun hızı da kontrol edilebilir. Aynı zamanda bu sinyallerin sırası değiĢtirilirse step motorlarının dönüĢ yönü, saat ibresi yönünde (CW) veya saat ibresinin tersi yönünde (CCW) olabilir. [38]

Tablo 3.1 Step Motor Özellikleri

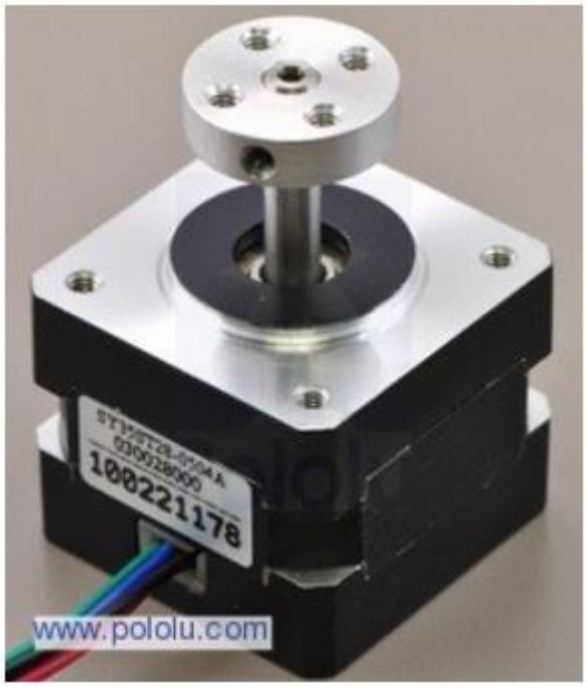
Teknik Özellikler:
Çalışma Voltajı: 10V
Faz Başına Çektiği Akım: 500mA @10V
Faz Direnç Değeri: 20 Ohm
Faz Endüktans Değeri: 13.5 mH
Adım Açısı: 1.8°
Tur Başına Adım Sayısı: 200
Tutunma Torku: 1 kg-cm
Ölçüler: 35x35x28mm (NEMA 14)
Motor Mil Kalınlığı: 5mm
Kablo Uzunluğu: 25cm
Ağırlık: 140g



Şekil 3.16 Step Motor Teknik Resmi



Şekil 3.17 Step Motor Devir - Tork Diagramı



Şekil 3.18 Step Motor Görünüm

3.4.3.1 Step Motor Kullanım Alanları

1944-1957 yılları arasındaki dönemde kapalı çevrim servo sistemleri ile yaygın bir kullanım alanı bulan step motorları, katı hal elektroniğindeki gelişmeler sayesinde 1960'lı yıllarda konum kontrol sistemlerinde kullanılmaya başlanmıştır. 1944 yılından beri var olan step motorları ancak 1960'lı yıllarda yüksek seviyeli doğru akımları

anahtarlayabilen transistörlerin üretilmesiyle beraber ticari anlamda 50 kullanılmaya başlanmıştır. Step motorlarının ilk çalışma ilkeleri 1930‟lu yıllarda açıklanmıştır. Sonraları İngiliz ordusu tarafından kullanılmış ve daha sonra ABD ordusu tarafından ikinci dünya savaşında gemilere uyarlanmıştır. 1970‟li yıllarda başlayan dijital elektronik ve mikroişlemci teknolojisindeki gelişmelerle birlikte step motorlarının kullanımı giderek cazipleşmekte ve tüm dünyada bu motorların üretim ve uygulamalarıyla ilgili geliştirme çalışmaları yapılmaktadır. Günümüzde step motorları endüstride birçok kontrol sistemlerinde, hassas konum kontrolü yapmak amacıyla kullanılmaktadır. Adım motorlarının birçok özel uygulama için kullanım alanları vardır. Çünkü step motorları, gerçekte sayısal formdaki giriş bilgisini mekanik harekete dönüştürür. En çok yazıcılar (printer), çiziciler (plotter), disket sürücüler (floppy driver), harddisk sürücüler (harddisk driver), kart okuyucular… vb gibi bilgisayar çevre cihazlarında bu elemanlardan yararlanılmaktadır. Ayrıca sayısal kontrol sistemlerinde, CNC tezgahlarda, proses kontrol sistemlerinde, robot teknolojisinde (milimetrik hareketlerin kontrolünde) ve uzay endüstrisine ait bir çok sistemde step motorları tahrik elemanı olarak yer almaktadır. [38]

3.4.3.2 Step Motorların Avantajları

Step motorların bu kadar yaygın olarak kullanılmasının sebebi, birçok avantaja sahip olmasıdır. Bu avantajlar aşağıdaki gibi sıralanabilir.

• Step motorlarla, makinaya zarar vermeden defalarca durma ve kalkma işlemleri gerçekleştirilebilir. (Ani durma ve ters yöne dönme isteklerine çok hızlı bir şekilde cevap verebilirler.)

• Step motorlar dijital girişlerle çalışabilir, bu nedenle geri beslemeye ihtiyaç duyulmaksızın açık çevrim olarak çalıştırılabilmesi sağlanabilir. Yani açık çevrim çalıştırılan bir step motoru ile hız, ivme ve konum kontrolü daha basit ve daha az maliyetle gerçekleştirilebilir. Böylelikle kararsızlık problemlerinin de önüne geçilmiş olur.

• Mikroişlemciler ve bilgisayarlı sistemlerle çalıştırılmak için uygundurlar. Çünkü step motorlar digital giriş işaretiyle çalışabilirler.

• Step motorlar, giriş işaretlerinin frekansına bağlı olarak çok geniş bir hız aralığında sürülebilirler.

• Step motorların dönüş yönü, devir sayısı, dönüş hızı gibi değerler mikroişlemci veya bilgisayar yardımı ile kontrol edilebildiğinden, bu motorların hız, konum ve dönüş yönü bilgileri sürekli olarak bilinmektedir. Bu nedenle bu motorlar hassas konum kontrolü gerektiren yerlerde rahatlıkla kullanılabilir. • Mekanik yapısı basit olduğundan bakım gerektirmezler.

• Aşırı yüklenmeden hasar görmezler, oldukça dayanıklıdırlar.

• Her yeni adımla artan (kümülatif) konum hataları yoktur.

• Yağlanma ve kirlenme problemleri yoktur. [38]

3.4.3.3 Step Motorların Dezavantajları

Step motorların bütün bu avantajlarına karşılık bazı dezavantajları da aşağıdaki gibi sıralanabilir.

• Elde edilebilecek çıkış gücü ve momenti sınırlıdır.

• Klasik sürücülerle kullanıldıklarında verimleri düşüktür.

• Sabit adım açılarıyla çalıştığından rotordan alınan hareket sürekli değil darbelidir.

• Adım cevapları nispeten büyük asım ve salınımlıdır.

• Yüksek eylemsizlik yüklerde yetenekleri sınırlıdır.

• Sürtünme kaynaklı yükler, hata kümülatif olmasa dahi açık çevrim çalışmada konum hatası meydana getirebilirler.

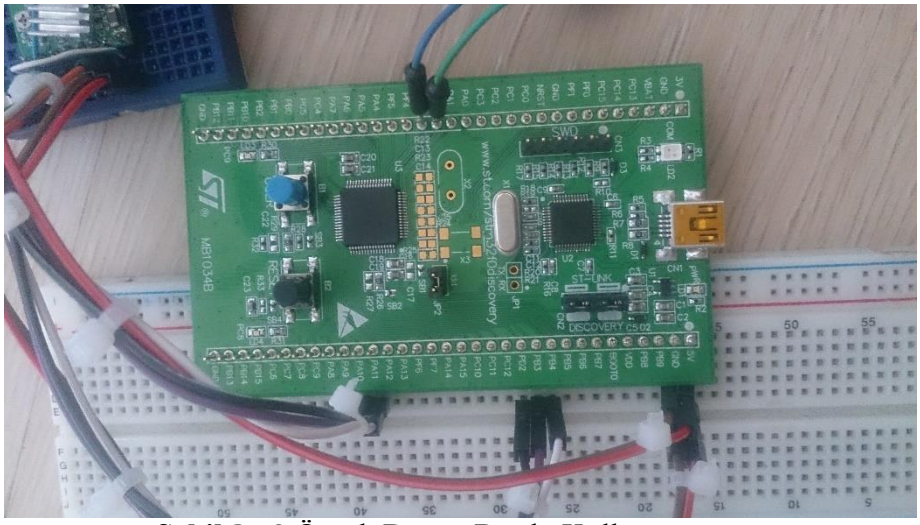
• İyi kontrol edilmezse rezonans meydana gelebilir.

• Oldukça yüksek hızlarda çalıştırmak pek kolay değildir.

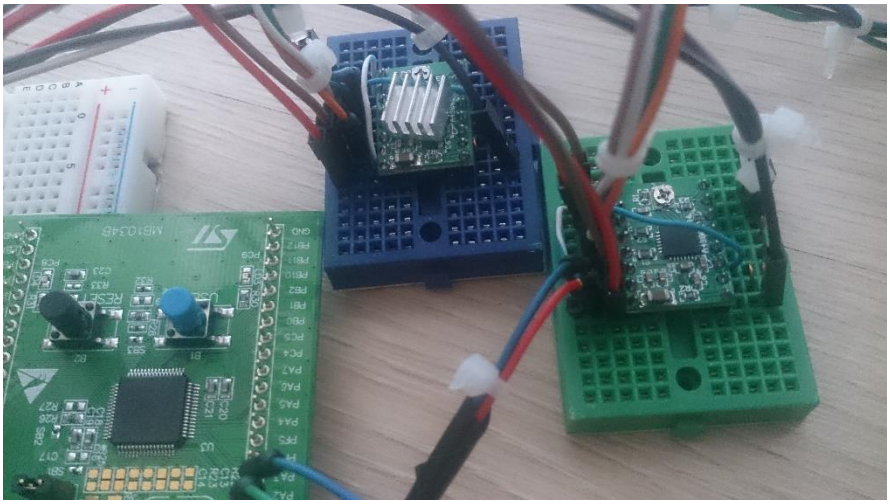
3.4.4 Deney Boardı

Breadboard delikli beyaz bir plakadır, bu plaka üzerinde lehim işlemi yapmadan elektronik devre elemanlarının çalıştırma mümkün olmaktadır. Bu işlem hem çok hızlı hem de geliştirmeye açık bir platform elde etmiş oluyoruz. Yani istediğimiz zaman deney bordu muza hızlı bir şekilde devre elamanı ekleyebiliyoruz ve çıkartabiliyoruz.

Projemizde örnek deney bordu kullanımı Şekil 3.19'de ve Şekil 3.20'da gösterilmektedir.



Şekil 3.19 Örnek Deney Bordu Kullanımı-1



Şekil 3.20 Örnek Deney Bordu Kullanımı-2

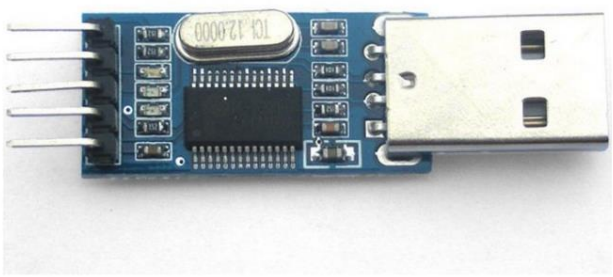
3.4.5 Atlama Kablosu

Deney bordu üzerinde devre elemanlarının ve mikrodenetleyici arasında bilgi ve sinyalleri basit bir şekilde kullanım amaçlı geliştirilmiş iki ucu erkek kablolardır.

3.5 Robotun Haberleşme Protokolleri

Bilgisayar USB'yi TTL iletişim metoduna çeviren bir çevirici ile mikrodeneteleyiciye bağlanmaktadır. Şekil 3.21'da bu çevirici görülmektedir. Çevirici sayesinde bilgisayar USB portu üzerinden kolaylıkla seri port iletişimi gerçekleştirebilmektedir.

Seri veri iletimi, bir veri içindeki bitlerin, aynı hat üzerinden art arda gönderilmesidir. Bitlerden ilk önce gönderilen EKB (En Küçük Değerlikli Bit), en son gönderilen EBB (En Büyük Değerlikli Bit)'dir. Her bit belli bir zaman aralığında gönderilir. Eğer bit aralığı 1 ms ise kelime içindeki her bit için voltaj basamağı 1 ms görünecektir. Dolayısı ile 7 bitlik bir ascıı kodu 7 ms'de gönderilir. Verici ve alıcı eşzamanlı olarak çalışabildikleri gibi eşzamansız olarak ta çalışabilirler. Seri veri iletiminde verinin başlangıç noktasını belirtmek için "başlama" biti, veri aktarma işlemi sırasında oluşabilecek bozulmaları ortaya çıkarabilmek için veri bitlerinin hemen ardından "eşlik" biti ve verinin bitiş noktasını belirtmek için "bitiş" biti kullanılır



Şekil 3.21 USB TTL dönüştürücü

4. BULGULAR

Bu çalışmada elde edilen bilgiler hem mühendislik eğitimi hem de karmaşık yapıdaki robot sistemlerin üretilmesinde son derece önemli olduğu anlaşılmıştır. Bu sistem maliyetli oluşturabilecek birçok projeyi ürün aşamasına geçmeden önce yazılım problemleri öngörebilmeyi ve buna bağlı olarak geliştirmeler yapmayı sağlayacaktır.

Aynı zamanda hedeflenen bir diğer konu ise donanım çevirimi meselesidir. Donanım çevrimi sayesinde çok büyük Mekatronik sistemlerin donanımlarını tehlikeye atmadan yazılım ortamında oluşturulan senaryolarda donanımın nasıl davranacağını, tepki vereceğini gözlemlemiş olacağız.

Elimizde mekanik, elektronik bir birleşen olmadan, mekanik ve elektroniği simülasyonlarda modelleyip her ortamda ve her yerde yazılım geliştirme işlemi maliyetsiz bir şekilde gerçekleştirilebilmektedir.

4.1 Robot Algoritma Testleri

Bu araştırmada, bir robotik projeyi mekanik ve elektronik bileşenlerini gerçek hayatta ortaya koymadan yazılım testi yapma imkânı sağlanmaktadır. Böylece robotik gibi içerisinde yazılımda bulunduğu karmaşık bir sistemin neredeyse maliyetsiz bir şekilde algoritma ve yazılım testlerinin yapılacağı gözlemlenmiştir.

Bu çalışmamın en büyük avantajlarından biri robotik maliyetli robotik sistemimizi hiçbir riske atmadan en ufak bir yazılım değişikliğini bile bilgisayar üzerindeki modelde gerçekleştire biliyor olmamız.

Robotların yazılımlarını bilgisayar ortamında birçok farklı test senaryosu yaparak test etme olanağı sağlandı. Böylece robotların test imkânı daha geniş yelpazeye yayılmış oldu. V-REP simülasyon uzayında robotlar istenilen genişlikte alanlar test imkanları sağlanmış oldu.

Robot simülasyon algoritma testleri V-REP robot simülasyon programı içerisinde bulunan LUA dili ile geliştirildi. Ve örnek bir P kontrol uygulaması yapılıp simülasyonda algoritması geliştirildi.

4.1.1 V-REP Ortamında Geliştirilen P Kontrol yazılımı

Katsayılar ve kontrol yazılımı tamamen simülasyon üzerinde geliştirilmiştir. Program seri portun COM1 ine bağlanmaktadır.

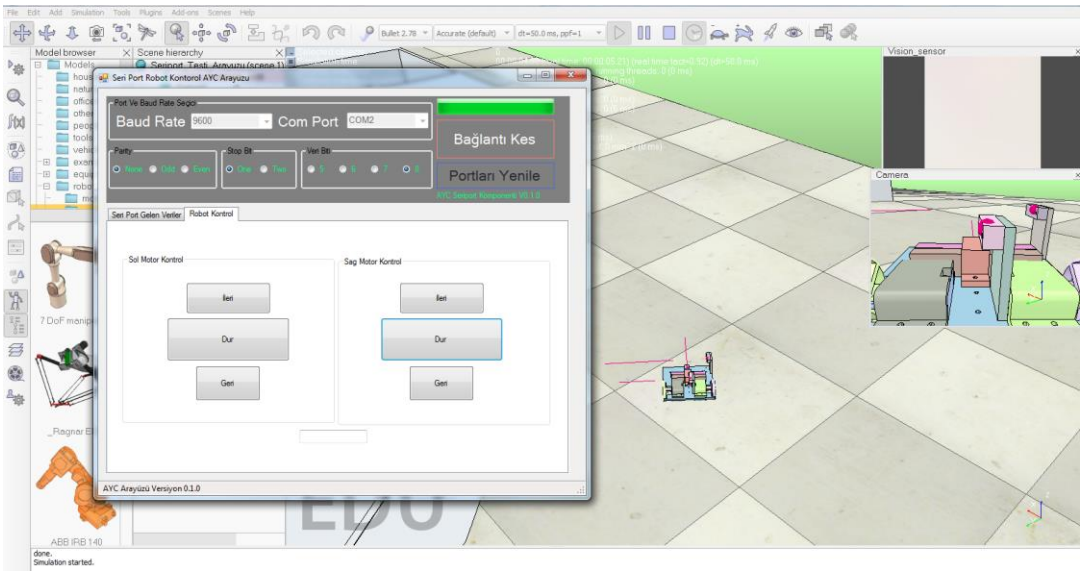
```
if (sim_call_type==sim_childscriptcall_initialization) then

    leftJointHandle         =simGetObjectHandle("dr12_leftJoint_")
    rightJointHandle        =simGetObjectHandle("dr12_rightJoint_")

    sol_tarafa_bakan_sensor = simGetObjectHandle("sol_sensor")

    simSetJointTargetVelocity(leftJointHandle, 1)
    simSetJointTargetVelocity(rightJointHandle,1)

   portNumber="\\\\.\\COM1"

-- for serial port

   seriport=simSerialOpen(portNumber,9600)
    backwardModeUntilTime=0
gecmis_hata_farki=0
hata=0
hata_gecmis=0
```

```
hata_toplam=0
hata_fark=0
p=0
end

if (sim_call_type==sim_childscriptcall_cleanup) then

end

if (sim_call_type==sim_childscriptcall_actuation) then



        simulationTime=simGetSimulationTime()

        hiz = simGetJointTargetPosition(leftJointHandle)

        sol_mesafe,dist = simReadProximitySensor(sol_tarafa_bakan_sensor)

        if(sol_mesafe == 1) then
         hata = 200 - math.floor(dist*1000)

          p = 0.04*hata


          t = p

         simSerialSend(seriport,"hata = "..hata)
         simSerialSend(seriport,"\t\n")
        if (hata<0) then


           simSetJointTargetVelocity(leftJointHandle,6-math.abs(t))
           simSetJointTargetVelocity(rightJointHandle,6)
        end

        if (hata > 0) then
           simSetJointTargetVelocity(leftJointHandle,6+math.abs(t))
           simSetJointTargetVelocity(rightJointHandle,6)
        end

        if (hata==0) then
           simSetJointTargetVelocity(leftJointHandle,7)
           simSetJointTargetVelocity(rightJointHandle,7)
        end
```

```
        --  simSetJointTargetVelocity(leftJointHandle, 0)
         -- simSetJointTargetVelocity(rightJointHandle,0)

           end

        --   simSetJointTargetVelocity(leftJointHandle, 3)
         --  simSetJointTargetVelocity(rightJointHandle,3)

    countReadChar=simSerialCheck(seriport)

     if (countReadChar>0) then
          data = simSerialRead(seriport,10,false,'',0)

       if(data==' ') then
          simSetJointTargetVelocity(leftJointHandle, 0)
          simSetJointTargetVelocity(rightJointHandle,0)

       end

       if(data=='w') then
          simSetJointTargetVelocity(leftJointHandle, 3)
          simSetJointTargetVelocity(rightJointHandle,3)

       end

       if(data=='a') then
          simSetJointTargetVelocity(leftJointHandle, -2)
          simSetJointTargetVelocity(rightJointHandle,0)

       end

       if(data=='d') then
          simSetJointTargetVelocity(leftJointHandle, 0)
          simSetJointTargetVelocity(rightJointHandle,-2)

       end



          simSerialSend(seriport,data)

          print(countReadChar)
          print(readDD)
       end


end
```

4.2 Simülasyonda Ara Yüz Geliştirip Gerçek Robotta Çalıştırmak

Bu çalışmada ara yüz geliştirip gerçek dünyada bulunan robota ara yüzü bağladığımız zamanda da aynı simülasyonda ki verimlilikte çalışıp çalışmadığını gözlemledik. Ve simülasyonda geliştirilen ara yüzü gerçek robota seri port üzerinden bağlayarak gerçek robotu da o ara yüz üzerinden çalıştırdık.



Şekil 0.1 Robot Simülasyonun C# dan hazırlanmış Robota Bağlanması

Simülasyon üzerinde ki robot oluşturulan sanal seri port COM1 e bağlanıyor, hazırlamış olduğumuz robot kontrol ara yüzü ise COM 2 ye bağlanıyor. Böylece bilgisayar üzerinden iki programda köprülenerek ikisi arasında iletişim gerçekleşir. COM 1 üzerinden gönderilen bilgi COM 2 iletilir. COM 2 üzerinden gönderilen bilgi COM 1 e iletilir.

4.2.1. Robotun Robot Kontrol Ara yüzü İle Çalışması İçin V-REP e yazılan LUA Yazılımı

Programda seriport açılıyor ve gelen veriler doğrultusunda robotun tekerlekleri hareket etmektedir. Bunu belirten LUA kodu aşağıda ki gibidir.

```
if (sim_call_type==sim_childscriptcall_initialization) then


    sag_tekerlek_handle = simGetObjectHandle("Sag_tekerlek_baglantisi")
    sol_tekerlek_handle = simGetObjectHandle("Sol_tekerlek_baglantisi")
```

```
    on_mesafe_sensoru   = simGetObjectHandle("on_mesafe_sensoru")
    arka_mesafe_sensoru = simGetObjectHandle("arka_mesafe_sensoru")

    on_engel_sensoru    = simGetObjectHandle("engel_sensoru")

    port_ismi = "\\\\.\\COM1"
    seriport=simSerialOpen(port_ismi,9600)

    durum=0

    simSetJointTargetVelocity(sag_tekerlek_handle,1)
    simSetJointTargetVelocity(sol_tekerlek_handle,1)


end


if (sim_call_type==sim_childscriptcall_actuation) then

    on_engel_durum,okunan_on      = simReadProximitySensor(on_mesafe_sensoru)
    arka_engel_durum,okunan_arka  = simReadProximitySensor(arka_mesafe_sensoru)
    onde_engel_var                = simReadProximitySensor(on_engel_sensoru)

 countReadChar=simSerialCheck(seriport)


if (countReadChar>0) then

        data = simSerialRead(seriport,10,false,'',0)

       if(data=='1') then

        simSetJointTargetVelocity(sag_tekerlek_handle,3)

       end

       if(data=='2') then
        simSetJointTargetVelocity(sol_tekerlek_handle, 3)


       end

       if(data=='3') then

        simSetJointTargetVelocity(sag_tekerlek_handle,-3)

       end
```

```
        if(data=='4') then
            simSetJointTargetVelocity(sol_tekerlek_handle, -3)


        end


        if(data=='5') then
            simSetJointTargetVelocity(sag_tekerlek_handle, 0)

        end

        if(data=='6') then
            simSetJointTargetVelocity(sol_tekerlek_handle, 0)


        end




    end

end



if (sim_call_type==sim_childscriptcall_sensing) then

        -- Put your main SENSING code here

end


if (sim_call_type==sim_childscriptcall_cleanup) then
simSerialSend(seriport,"5")
simSerialSend(seriport,"6")
        -- Put some restoration code here

end
```

4.2.2. Seri Port Kontrol Ara Yüzü Yazılımı

Seri port üzerindeki butonlara basarak bir rakam gönderilir bunlar lua dili tarafından çözülerek robot kontrol edilmiş olur. C# komutları aşağıdaki gibidir.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Robot_Test_Ara_Yüzü
{

   public partial class Form1 : Form
   {
      /*-------------------- Delegate Fonksiyonlar --------------------*/

      delegate void delegate_textbox_yaz(string text, TextBox textbox);

      public void delegate_gonder_textbox(string text, TextBox texBox)
      {
         if (texBox.InvokeRequired)
         {
            this.Invoke(new delegate_textbox_yaz(delegate_gonder_textbox), new
object[] { text, texBox });
         }
         else
         {
            texBox.Text += text;
         }

      }


/**********************************************************************/
      Seri_Port_Okunan_Veri_Makinasi makinayaVeriGonder;
      Versiyon versiyon_bilgisi;

      public Form1()
      {
         InitializeComponent();

         //Porta veri gonderildigi anda olusturulan event, bu eventi kullanıma açmış
olduk, izin verilen olaydan haberdar olmak için.
```

```
        seri_port_baglanti_kontrolu1.Seri_Port_Okunan_Var_EventHandler += new
WindowsFormsControlLibrary1.seri_port_baglanti_kontrolu.seriportokunanverievenha
ndler(eventvar);
        makinayaVeriGonder = new Seri_Port_Okunan_Veri_Makinasi();
        versiyon_bilgisi = new Versiyon();
        versiyon_bilgisi_StatusLabel.Text = versiyon_bilgisi.versiyon;
    }

    /*-------------------- Okunan Verilerin Gosterilmesi --------------------*/

    void eventvar()
    {

        Console.WriteLine("Okunan Veri Paket Boyutu : " +
seri_port_baglanti_kontrolu1.okunan_buffer.Length);

        if (ascii_ac_RadioButton.Checked)
        {

delegate_gonder_textbox(System.Text.Encoding.ASCII.GetString(seri_port_baglanti_k
ontrolu1.okunan_buffer) + " ", gelen_veriler_yaz_text_box);
        }
        else if (dec_goster_RadioButton.Checked)
        {
            for (int i = 0; i < seri_port_baglanti_kontrolu1.okunan_buffer.Length; i++)
            {

                delegate_gonder_textbox(seri_port_baglanti_kontrolu1.okunan_buffer[i] +
"", gelen_veriler_yaz_text_box);

makinayaVeriGonder.verki_cozen_makina(seri_port_baglanti_kontrolu1.okunan_buffer
[i]);

            }

        }

    }


/*********************************************************************/



    private void gelen_veriler_temizle_Click(object sender, EventArgs e)
    {
        gelen_veriler_yaz_text_box.Clear();
    }
```

```
        private void iler_button_Click_1(object sender, EventArgs e)
        {
            byte[] gonder = { 0x32 };
            seri_port_baglanti_kontrolu1.seri_port_veri_gonder(gonder);

        }

        private void sag_don_button_Click_1(object sender, EventArgs e)
        {
            byte[] gonder = { 0x64 };
            seri_port_baglanti_kontrolu1.seri_port_veri_gonder(gonder);

        }

        private void sol_button_Click_1(object sender, EventArgs e)
        {
            byte[] gonder = { 0x34 };
            seri_port_baglanti_kontrolu1.seri_port_veri_gonder(gonder);

        }

        private void button1_Click_1(object sender, EventArgs e)
        {
            byte[] gonder = { 0x36 };
            seri_port_baglanti_kontrolu1.seri_port_veri_gonder(gonder);

        }

        private void robot_kontrol_PreviewKeyDown(object sender,
PreviewKeyDownEventArgs e)
        {


        }

        private void textBox1_PreviewKeyDown(object sender,
PreviewKeyDownEventArgs e)
        {
            if (e.KeyCode == Keys.W)
            {
                byte[] gonder = { 0x31 , 0x32 };
                seri_port_baglanti_kontrolu1.seri_port_veri_gonder(gonder);

            }
            if (e.KeyCode == Keys.S)
            {
```

```
            byte[] gonder = { 0x33, 0x34 };
            seri_port_baglanti_kontrolu1.seri_port_veri_gonder(gonder);

        }

        if (e.KeyCode == Keys.D)
        {
            byte[] gonder = { 0x32 , 0x35 };
            seri_port_baglanti_kontrolu1.seri_port_veri_gonder(gonder);

        }

        if (e.KeyCode == Keys.A)
        {
            byte[] gonder = { 0x36 , 0x31 };
            seri_port_baglanti_kontrolu1.seri_port_veri_gonder(gonder);

        }

        if (e.KeyCode == Keys.Space)
        {
            byte[] gonder = { 0x36 , 0x35 };
            seri_port_baglanti_kontrolu1.seri_port_veri_gonder(gonder);
        }
    }

    private void sag_motor_dur_button_Click(object sender, EventArgs e)
    {
        byte[] gonder = { 0x35 };
        seri_port_baglanti_kontrolu1.seri_port_veri_gonder(gonder);
    }

    private void sag_motor_geri_button_Click(object sender, EventArgs e)
    {
        byte[] gonder = { 0x33 };
        seri_port_baglanti_kontrolu1.seri_port_veri_gonder(gonder);
    }

    private void button2_Click(object sender, EventArgs e)
    {
        byte[] gonder = { 0x31 };
        seri_port_baglanti_kontrolu1.seri_port_veri_gonder(gonder);
    }
  }
}
```

4.3 Donanımı Simülasyon İle Bütünleştirmek

Robotu serip port üzerinden bilgisayar üzerindeki robot ile tümleşik bir şekilde çalışması gerekiyordu bu da hem gömülü sisteme hem de bilgisayar üzerinde çalışan V-REP e gerekli komutların yazılması anlamına geldi.
Simülasyonda çalışan robotu ve gerçek hayatta çalışan robotu bir birine seri port üzerinden bağlıyoruz. Simülasyonda çalışan robot bir engel ile karşılaştığı zaman seri port üzerinden gerçek robota bir bilgi gönderiyor. Böylece gerçek zamanlı olarak donanım çevrimli robot simülasyonu çalışmış oluyor.
Gömülü sistem programı mikroC programı ile yazıldı.

4.3.1. Donanım Çevrimli Simülasyon İçin Gömülü Sistem Programı

Program seri port üzerinden gelen bilgileri dinleyerek robotun tekerleklerini hareket ettirecek yazılım parçacıklarını çalıştırıyor. Bu program aşağıda verilmiştir.

```
 #define       sag_step_motor_yon_pini   GPIOA_ODR.B10
 #define       sol_step_motor_yon_pini   GPIOB_ODR.B3

 #define       sag_step_motor_aktif_pini  GPIOA_ODR.B9
 #define       sol_step_motor_aktif_pini  GPIOD_ODR.B2



#define                                          sol_step_motor_PWM_DURDUR
PWM_TIM3_STOP(_PWM_CHANNEL1)
#define                                         sag_step_motor_PWM_DURDUR
PWM_TIM1_STOP(_PWM_CHANNEL4)

#define sol_step_motor_PWM_CALISTIR    PWM_TIM3_Start(_PWM_CHANNEL1,
&_GPIO_MODULE_TIM3_CH1_PB4);
#define sag_step_motor_PWM_CALISTIR    PWM_TIM1_Start(_PWM_CHANNEL4,
&_GPIO_MODULE_TIM1_CH4_PA11);

 int const  motor_hiz_frekansi = 700;
 int const  motoru_ac     = 0;
 int const  motoru_kapat  = 1;
 int const  motor_ileri   = 1;
 int const  motor_geri    = 0;

  /*** SAG MOTOR KONTROL ICIN KULLANILACAK KONTROL
FONKSIYONU **/
   void sag_motor_adimla(int frekans)
   {
    if(frekans == 0)
    {
        sag_step_motor_aktif_pini = motoru_kapat ;
```

```
     sag_step_motor_PWM_DURDUR;
 }
  else if ( frekans < 0 )
 {
 sag_step_motor_yon_pini   = motor_geri;
 sag_step_motor_aktif_pini = motoru_ac ;
 PWM_TIM3_Init(abs(frekans)); // Pwm Fewkansı belirleniyor
 sag_step_motor_PWM_CALISTIR;

 }

 else if ( frekans > 0 )
  {
   sag_step_motor_yon_pini = motor_ileri;
   sag_step_motor_aktif_pini = motoru_ac ;
   PWM_TIM3_Init(frekans); // Pwm Fewkansı belirleniyor
   sag_step_motor_PWM_CALISTIR;
  }

 }  //void sag_motor_adimla(int bekleme)

/*****************************************************************/


 /*** SOL STEP MOTOR KONTROL FONKSIYONU **/

 void sol_motor_adimla(int frekans)
{
   if(frekans == 0)
 {
     sol_step_motor_aktif_pini = motoru_kapat ;
     sol_step_motor_PWM_DURDUR;
 }
  else if ( frekans < 0 )
 {
 sol_step_motor_yon_pini = motor_geri;
 sol_step_motor_aktif_pini = motoru_ac ;
 PWM_TIM1_Init(abs(frekans)); // Pwm Fewkansı belirleniyor
 sol_step_motor_PWM_CALISTIR;

 }

 else if ( frekans > 0 )
  {
   sol_step_motor_yon_pini = motor_ileri;
   sol_step_motor_aktif_pini = motoru_ac ;
   PWM_TIM1_Init(frekans); // Pwm Fewkansı belirleniyor
   sol_step_motor_PWM_CALISTIR;
```

```
      }

    }// void sol_motor_adimla(int bekleme)

  /*****************************************************************/



  unsigned  okunan;
  char output[]="";
   char topla;
  int sayac=0;

void main()
 {
  GPIO_Digital_Output(&GPIOC_ODR, _GPIO_PINMASK_9) ;
  GPIO_Digital_Output(&GPIOA_ODR, _GPIO_PINMASK_10 | _GPIO_PINMASK_9 ) ; // A10 Yon pini olarak ayarlandi , A9 Aktif pini olarak ayarlandi


  GPIO_Digital_Output(&GPIOB_ODR, _GPIO_PINMASK_3); // Sol step motor yon pini icin cıkıs olarak ayarlandi
  GPIO_Digital_Output(&GPIOD_ODR, _GPIO_PINMASK_2); // sol step motor aktif pini icin cıkıs olarak ayarlandi


  GPIO_Digital_Input (&GPIOA_IDR, _GPIO_PINMASK_0) ;



          //SAG MOTOR ICIN PWM DEGERLERI
          PWM_TIM1_Init(500); // Pwm Fewkansı belirleniyor
          PWM_TIM1_Set_Duty(30000, _PWM_NON_INVERTED, _PWM_CHANNEL4);  // PWM duty saykılını belirliyoruz
          PWM_TIM1_Start(_PWM_CHANNEL4, &_GPIO_MODULE_TIM1_CH4_PA11); // pwmm kanalını açıyoruz, bu parametrelere help kısmından bakılabilir

          //SOL MOTOR ICIN PWM DEGERLERI
          PWM_TIM3_Init(500); // Pwm Fewkansı belirleniyor
          PWM_TIM3_Set_Duty(30000, _PWM_NON_INVERTED, _PWM_CHANNEL1);  // PWM duty saykılını belirliyoruz
          PWM_TIM3_Start(_PWM_CHANNEL1, &_GPIO_MODULE_TIM3_CH1_PB4); // pwmm kanalını açıyoruz, bu parametrelere help kısmından bakılabilir

         sag_motor_adimla(0);
```

```
        sol_motor_adimla(0);
```

UART2_Init(9600); // 9600 hız protokolünde seri port iletişimi sağlanacağını gösteriyor

Delay_ms(10); // Ayarlar yapılırken 100ms bekleme

UART_Set_Active(&UART2_Read, &UART2_Write, &UART2_Data_Ready, &UART2_Tx_Idle); // SeriPort iletişimi aktive ediyor. Okuma,yazma gibi özellikleri açar.

```
        UART2_Write_Text("Acildi");

  do
 {

     if (UART2_data_ready()){ // Eğer seri portdan okunan veri varsa.
```

okunan = Uart2_read(); // seri portdan okunan veriyi "okunan" değişkenine aktarır ve orada tutar.

```
          if (okunan=='1') {
              sag_motor_adimla(motor_hiz_frekansi);
              okunan = 0;
          }

          if (okunan=='2') {
              sol_motor_adimla(motor_hiz_frekansi);
              okunan = 0;
          }
          if (okunan=='3') {
              sag_motor_adimla(-motor_hiz_frekansi);
              okunan = 0;
          }

          if (okunan=='4') {
              sol_motor_adimla(-motor_hiz_frekansi);
              okunan = 0;
          }
          if (okunan=='5') {
              sag_motor_adimla(0);
              okunan = 0;
          }
           if (okunan=='6') {
              sol_motor_adimla(0);
              okunan = 0;
          }

     }
```

```
  }  while (1) ;


}
```

4.3.2. V-REP LUA Yazılımı

Robot bir engel ile karşılaştığı zaman donanıma bilgi gönderiyor. Yazılım aşağıdadır.

```
if (sim_call_type==sim_childscriptcall_initialization) then


   sag_tekerlek_handle = simGetObjectHandle("Sag_tekerlek_baglantisi")
   sol_tekerlek_handle = simGetObjectHandle("Sol_tekerlek_baglantisi")

   on_mesafe_sensoru   = simGetObjectHandle("on_mesafe_sensoru")
   arka_mesafe_sensoru = simGetObjectHandle("arka_mesafe_sensoru")

   on_engel_sensoru    = simGetObjectHandle("engel_sensoru")

   port_ismi = "\\\\.\\COM3"
   seriport=simSerialOpen(port_ismi,9600)

   durum=0

   simSetJointTargetVelocity(sag_tekerlek_handle,10)
   simSetJointTargetVelocity(sol_tekerlek_handle,10)


end


if (sim_call_type==sim_childscriptcall_actuation) then

   on_engel_durum,okunan_on      = simReadProximitySensor(on_mesafe_sensoru)
   arka_engel_durum,okunan_arka  = simReadProximitySensor(arka_mesafe_sensoru)

   onde_engel_var                = simReadProximitySensor(on_engel_sensoru)
```

```
        if (onde_engel_var == 1) then


      simSetJointTargetVelocity(sol_tekerlek_handle,10)
      simSetJointTargetVelocity(sag_tekerlek_handle,0)

        if(durum==1) then
              simSerialSend(seriport,"5")
              simSerialSend(seriport,"2")
              durum = 0
        end



end

else

            if(durum==0) then

               simSerialSend(seriport,"1")
               simSerialSend(seriport,"2")
               durum = 1

            end


            simSetJointTargetVelocity(sol_tekerlek_handle,2)
            simSetJointTargetVelocity(sag_tekerlek_handle,2)


 end

arka_engel_durum  = simReadProximitySensor(arka_mesafe_sensoru)

 if(arka_engel_durum==1)then

   simSerialSend(seriport,"2")
   simSerialSend(seriport,"5")

      simSetJointTargetVelocity(sol_tekerlek_handle,10)
      simSetJointTargetVelocity(sag_tekerlek_handle,0)
 durum=0

 end
```

```
if (sim_call_type==sim_childscriptcall_sensing) then

        -- Put your main SENSING code here

end


if (sim_call_type==sim_childscriptcall_cleanup) then
simSerialSend(seriport,"5")
simSerialSend(seriport,"6")
        -- Put some restoration code here

end
```

4.4 Robot Simülasyonların Mühendislik Eğitiminde Kullanılması

Mekatronik alanında yazlım geliştirir ken PID ve ON-OF kontrolü modellenmiş robot üzerinden LUA dili ile yazılmış program sayesinde çalıştı. Böylece gerçek bir robot olmadan simülasyon üzerinde bu ili konu hakkında gözlem yapma imkanı bulmuş olduk.

5. TARTIŞMA ve SONUÇ

Bu araştırmada, Robotların bilgisayar ortamında modellenip algoritma ve ara yüz geliştirme işlemlerinin robot gerçek hayatta olamadan yapabilme imkânı araştırılmış ve uygulamaya sokulmuştur. Aynı zamanda robotun simülasyon içerisinde karşılaştığı etkilere hem simülasyon hem de gerçek hayatta tepkisi ölçülmesi amacı ile bir donanım çevrimli simülatör tasarlanmıştır. Örneğin; Robot simülasyonda bir engel ile karşılaştığı zaman gerçek hayattaki robot bir engel ile karşılaşmış gibi manevra yapıp donanım testi gözlemlenmiştir.

Bu çalışmada robot simülasyonların önemi çok ciddi bir şekilde vurgulanmış ve anlaşılmıştır. Robot simülasyonun donanım çevrimli olarak çalıştırılması donanımı test edebilme imkanı kazandırmıştır.

Aynı zamanda donanım ile gerçek zamanlı olarak orta bir çalışma gerçekleştirilmiştir. Böylece oluşturulan robotun tüm alanlarda yazlımı test edilip gözlemlenmiş oldu.

Başvurular

[1] W.G., «mitation of Life.,» 1950.

[2] P. a. M. R. Lima, «Instituto Superior Técnico/Instituto de Sistemas e Robótica,» *Mobile Robotics,* 2002.

[3] A. GÜLLÜ, «LABİRENTLERDE YAPAY ZEKA TABANLI YÖN BULMA ALGORİTMALARI KULLANAN BİR GEZGİN ROBOT GELİŞTİRİLMESİ,» *DOKTORA TEZİ,* 2017.

[4] anyKode, «http://anykode.com/,» Marilou website, 06 06 2017. [Çevrimiçi]. Available: http://anykode.com/. [Erişildi: 22 06 2017].

[5] M. O. A. H. K. M. A. Aydın GÜLLÜ, «Mobil Robotların Deneysel Simülasyonunda Kullanılan Platformların İncelenmesi,» *TOK 2014 Bildiri Kitabı,* 2014.

[6] D., «Modelado y simulación de un robot móvil autónomo OMNI-direccional.,» *García Sillas,* 2014.

[7] J. Kim, «Implementation of N: 1/1: N work distribution function for tele-operation under multi-user and multi-robot environments. in Ubiquitous Robots and Ambient Intelligence (URAI), 2013 10th International Conference on.,» *IEEE,* 2013.

[8] Á. e. a. Ballagi, «Fuzzy Situational Maps: A new approach in mobile robot cooperation. in Intelligent Engineering Systems (INES), 2013 IEEE 17th International Conference on,» *IEEE,* 2013.

[9] r. Pro, «Cogmation Robotics,» anykode, 01 06 2014. [Çevrimiçi]. Available: http://anykode.com/i. [Erişildi: 22 06 2017].

[10] J. M. S. a. M. R. Roßmann, «Simulation-Driven Engineering für mobile Systeme. Virtuelle Welten als Entwicklungsplattform für mobile Robotik-Von der Planetenexploration bis in den Wald.,» *at-Automatisierungstechnik Methoden und Anwendungen der Steuerungs-, Regelungs-und Informationstechnik,* no. 61(4), pp. 278-289, 2013.

[11] M. e. a. Vallance, «Applied Information Science Researchin a Virtual World Simulation to Support Robot-Mediated InteractionFollowing the Fukushima Nuclear Disaster.,» *Communications,* cilt 3(5), pp. 222-232, 2013.

[12] «Microsoft Robotics Developer Studio Web Page,» Available, 2014. [Çevrimiçi]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=29081. [Erişildi: 2017].

[13] X. e. a. MAI, «Simulation of path planning based on Microsoft robotics developer studio,» *Journal of Computer Applications,* cilt S1, 2013.

[14] C. e. a. A. Pons, «model-driving approach to constructing robotic systems,» *Journal of Computer Science & Technology,* cilt 14, 2014.

[15] Q. e. a. Zhao, «Using head poses to control a virtual robot walking in a virtualmaze,» *Optics Communications,,* no. 295, pp. 84-91, 2013.

[16] T. e. a. Choi, «Software platform for the Industrial Dual-arm Robot, in Robot Intelligence Technology and Applications,» *Springer,* pp. 911-920, 2013.

[17] «V-REP Pro,» V-Rep, 2010. [Çevrimiçi]. Available: http://www.coppeliarobotics.com/. [Erişildi: 22 Haziran 2017].

[18] H. T. H. a. N. N. Anh, «Novel Robust Walking for Biped Robot Using Adaptive Neural PID Controller.,» 2014.

[19] Xiao, W., J. Huan, and S. Dong,, «A STEP-compliant Industrial Robot Data Model for robot off-line programming systems.,» *Robotics and Computer-Integrated Manufacturing,,* cilt 30, no. 2, pp. 114-123, 2014.

[20] Kuo, P.-H., et al., «Development of Humanoid Robot Simulator for Gait Learning by Using Particle Swarm Optimization. in Systems, Man, and Cybernetics (SMC),,» *IEEE International Conference on.,* 2013.

[21] Xu, W., S.C. Yuan, and Y.H. Cao, «Research on Yinma Palletizing Robot Motion Feasibility Based on MATLAB-Robotics Toolbox,» *Applied Mechanics and Materials,* no. 543, pp. 1296-1300, 2014.

[22] Sepúlveda, R., et al., «Design of Fuzzy Controllers for a Hexapod Robot, in Recent Advances on Hybrid Approaches for Designing Intelligent Systems,» *Springer,* pp. 709-721, 2014.

[23] «Matlab Robotics toolbox,» 2014. [Çevrimiçi]. Available: http://petercorke.com/Robotics_Toolbox.html. [Erişildi: 22 Haziran 2017].

[24] «S.T.D.R. Simulator,» S.T.D.R., 2014. [Çevrimiçi]. Available: http://stdr-simulator-ros-pkg.github.io/. [Erişildi: 22 Haziran 2017].

[25] L. a. N. B. Hugues, «Simbad: an autonomous robot simulation package for education and research,» *From Animals to Animats,* no. 9, pp. 831-842, 2006.

[26] «Simbad,» Simbad, [Çevrimiçi]. Available: http://simbad.sourceforge.net/. [Erişildi: 22 Haziran 2017].

[27] Purohit, A. and P. Zhang, «Controlled-mobile sensing simulator for indoor fire monitoring,» *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International. IEEE.,* 2011.

[28] C. Crous, «Autonomous robot path planning.,» *University of Stellenbosch,* 2009.

[29] «MiniBloq Web Site,» MiniBloq, 2014. [Çevrimiçi]. Available: http://blog.minibloq.org/. [Erişildi: 22 Haziran 2017].

[30] Junior, L.A., et al., A, «Low-Cost and Simple Arduino-Based Educational Robotics Kit».

[31] Geth, F., et al., «Development of an open-source smart energy house for K-12 education,» *n Power and Energy Society General Meeting (PES) IEEE,* 2013.

[32] R. e. a. Grover, «competition-based approach for undergraduate mechatronics education using the arduino platform,» *Interdisciplinary Engineering Design Education Conference (IEDEC) IEEE,* 2014 .

[33] «Gazebo Simulator Web Site,» Gazebo Sim, [Çevrimiçi]. Available: http://gazebosim.org/. [Erişildi: 22 Haziran 2017].

[34] Unhelkar, V.V., et al, «Towards control and sensing for an autonomous mobile robotic assistant navigating assembly lines,» *International Conference on Robotics and Automation (ICRA),* 2014.

[35] F. A. O.-H. a. M. F. Suárez-Ruiz, «Internet-Based Supervisory Teleoperation of a Virtual Humanoid Robot,» *ROBOT2013: First Iberian Robotics Conference,* 2014.

[36] Lua, «WikiPedia,» [Çevrimiçi]. Available: https://tr.wikipedia.org/wiki/Lua_(programlama_dili). [Erişildi: 23 Haziran 2017].

[37] «robitshop.com,» robitshop, 2017. [Çevrimiçi]. Available: http://www.robitshop.com/a4988-step-motor-surucu-stepper-motor-driver-carrier. [Erişildi: 22 Haziran 2017].

[38] M. YILMAZ, «GÜNEŞ PANELİ İLE BESLENEN STEP MOTORUN PIC MİKRODENETLEYİCİ İLE KONTROLÜ,» *FIRAT Üniversitesi Fen bilimleri Enstitüsü,* 2012.

ÖZGEÇMİŞ

Adı Soyadı : Ahmet Yasin CİVAN

Doğum Yeri ve Tarihi : 01.03.1993

Yabancı Dili : İngilizce

İletişim (Telefon/e-posta) : civan.ahmetyasin@gmail.com

Eğitim Durumu (Kurum ve Yıl)

Lise : Konak Çınarlı Mesleki ve Teknik Anadolu Lisesi, BİLİŞİM TEKNOLOJİLERİ ALANI, Teknik Lise (2008-2012)

Lisans : Afyon Kocatepe Üniversitesi, Mekatronik Mühendisliği Bölümü, (2013-2017)