**KOCAELİ ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

**MEKATRONİK MÜHENDİSLİĞİ**
**ANABİLİM DALI**

**YÜKSEK LİSANS TEZİ**

# ALTI BACAKLI BİR ÖRÜMCEK ROBOTUN KİNEMATİK TABANLI HAREKET VE DENGE KONTROLÜ

**ENES VARDAR**

**KOCAELİ 2023**

**KOCAELİ ÜNİVERSİTESİ**

**FEN BİLİMLERİ ENSTİTÜSÜ**

**MEKATRONİK MÜHENDİSLİĞİ**

**ANABİLİM DALI**

**YÜKSEK LİSANS TEZİ**

# ALTI BACAKLI BİR ÖRÜMCEK ROBOTUN KİNEMATİK TABANLI HAREKET VE DENGE KONTROLÜ

**ENES VARDAR**

| | | |
|---|---|---|
| **Prof.Dr. Hüseyin Metin ERTUNÇ** | | |
| **Danışman,** | **Kocaeli Üniv.** | **.....** |
| **Doç.Dr. Selçuk KİZİR** | | |
| **Jüri Üyesi,** | **Kocaeli Üniv.** | **.....** |
| **Dr.Öğr. Üyesi Kenan IŞIK** | | |
| **Jüri Üyesi,** | **Karabük Üniv.** | **.....** |

**Tezin Savunulduğu Tarih: 19.09.2023**

## ETİK BEYAN VE ARAŞTIRMA FONU DESTEĞİ

Kocaeli Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım bu tez/proje çalışmasında,

- Bu tezin/projenin bana ait, özgün bir çalışma olduğunu
- Çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı,
- Bu çalışma kapsamında elde edilen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi,
- Bu çalışmanın Kocaeli Üniversitesi'nin abone olduğu intihal yazılım programı kullanılarak Fen Bilimleri Enstitüsü'nün belirlemiş olduğu ölçütlere uygun olduğunu,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Tezin/Projenin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez/proje çalışması olarak sunmadığımı,

beyan ederim.

[X] Bu tez/proje çalışmasının herhangi bir aşaması hiçbir kurum/kuruluş tarafından maddi/alt yapı desteği ile desteklenmemiştir.

[ ] Bu tez/proje çalışması kapsamında üretilen veri ve bilgiler ..... tarafında ……..... no'lu proje kapsamında kapsamında maddi/alt yapı desteği alınarak gerçekleştirilmiştir.

Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçları kabul ettiğimi bildiririm.

……...

Enes VARDAR

## YAYIMLAMA VE FİKRİ MÜLKİYET HAKLARI

Fen Bilimleri Enstitüsü tarafından onaylanan lisansüstü tezimin/projemin tamamını veya herhangi bir kısmını, basılı ve elektronik formatta arşivleme ve aşağıda belirtilen koşullarla kullanıma açma izninin Kocaeli Üniversitesi'ne verdiğimi beyan ederim. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet hakları bende kalacak, tezimin/projemin tamamının ya da bir bölümünün gelecekteki makale, kitap, tebliğ, lisans, patent gibi çalışmalarda kullanımı, danışmanımın isim hakkı saklı kalmak koşuluyla ve her iki tarafın bilgisi dâhilinde bana ait olacaktır. Tezin/projenin kendi özgün çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin/projenin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanılması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

Yükseköğretim kurulu tarafından yayınlanan "**Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge**" kapsamında tezim aşağıda belirtilen koşullar haricinde YÖK Ulusal Tez Merkezi/ Kocaeli Üniversitesi Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

☐ Enstitü yönetim kurulu kararı ile tezimin/projemin erişime açılması mezuniyet tarihinden itibaren 2 yıl ertelenmiştir.

☐ Enstitü yönetim kurulu gerekçeli kararı ile tezimin/projemin erişime açılması mezuniyet tarihinden itibaren 6 ay ertelenmiştir.

☒ Tezim/projem ile ilgili gizlilik kararı verilmemiştir.

……...

Enes VARDAR

## ÖNSÖZ VE TEŞEKKÜR

Tez kapsamında yapmış olduğum projenin geliştirilmesi sırasında destek olan hocam Prof. Dr. Hüseyin Metin ERTUNÇ, yüksek lisans öğrenimim ve tez çalışmam sırasında yardımcı olmuş ve her türlü desteği sağlamıştır. Hocama, göstermiş olduğu ilgi ve bütün emekleri için çok teşekkür ederim.

Daima yanımda olan aileme, hayat arkadaşıma öğrenimim ve tez çalışmam süresince gösterdikleri sabır ve hoşgörüden dolayı sonsuz sevgilerimi sunarım.

Mayıs – 2023 Enes VARDAR

## İÇİNDEKİLER

## ŞEKİLLER DİZİNİ

## TABLOLAR DİZİNİ

## SİMGELER VE KISALTMALAR DİZİNİ

### Kısaltmalar

2D : Two Dimensions (İki Boyutlu)
3D : Three Dimensions (Üç Boyutlu)
3DS : Three Dimensions File (Üç Boyutlu Dosya)
ACK : Acknowledgment (Onay)
D-H : Denavit-Hartenberg
FBX : Filmbox
GUI : Graphical User Interface (Grafiksel Kullanıcı Arayüzü)
HTTP : Hyper-Text Transfer Protocol (Hiper-Metin Transfer Protokolü)
HEX : Hexadecimal
MQTT : Message Queuing Telemetry Transport
NYP : Nesne Yönelimli Programlama
OBJ : Object
PC : Personal Computer (Bilgisayar)
ST : STMicroelectronics
UART : Universal Asynchronous Receiver/Transmit (Evrensel Asenkron Veri/Alıcı )
VR : Virtual Reality (Sanal Gerçeklik)

**ALTI BACAKLI BİR ÖRÜMCEK ROBOTUN KİNEMATİK TABANLI HAREKET VE DENGE KONTROLÜ**

## ÖZET

Bu çalışmada, her bir bacağında 3 eksen bulunan 6 bacaklı toplam 18 eksenden oluşan bir örümcek robotun; kinematik modeli çıkartılarak, robotun yürümesi, yön değiştirmesi ve denge kontrolü gibi algoritmalar için teorik bir temel sağlanmıştır. Bir örümcek robotun hareket yörüngelerinde kullanılan; her bir bacaktaki eklemlerin birçok durum için sahip olması gerektiği konum derecelerinin bulunduğu tablo yerine bir model oluşturarak sistemin daha özgün ve parametrik çalışması gerçekleştirilmiştir. İlk olarak örümcek robotun kinematik modeli D-H yöntemine dayalı olarak oluşturulmuştur. Çıkarılan kinematik denklemler ile oluşturulan yürüme, dönme, koordinat sistemindeki hareket algoritmaları bir simülasyon ortamında test edilerek hareketlerin simülasyonu gerçekleştirilmiştir. Simülasyon ortamında oluşturulan algoritmalarda herhangi hazır bir matematik kütüphanesi kullanılmadan gerçekleştirilerek, algoritmaların bir programlama diline bağlı kalmadan geliştirilmesi sağlanmıştır. Simülasyon ortamında doğrulanarak test edilen yazılım, gerçek bir örümcek robota aktarılarak sistem başarıyla çalıştırılmıştır. Örümcek robotun simülasyon ortamında gözlenen hareketleri gerçekçi bir şekilde yaptığı gözlemlenmiştir.

**Anahtar Kelimeler:** Altı Ayaklı Robot, Kinematik, Örümcek Robot, Simülasyon, Unity.

**KINEMATICS BASED MOVEMENT AND BALANCE CONTROL OF A SIX LEGGED SPIDER ROBOT**

## ABSTRACT

In this study, the kinematics of a hexapod spider robot consisting of a total of 18 axes with 6 legs with 3 axes on each leg were modeled and a theoretical basis was provided for algorithms such as the ability of the robot to walk, change direction and control the coordinate system in its body. A more original and parametric study of the system was carried out by creating a model instead of a table containing the degrees of position that the joints in each leg should have for many situations, which are used in the movement trajectories of a spider robot. First, the kinematic model of hexapod robot was created based on D-H method. Then, the algorithms of walking, rotation, movement in the coordinate system created by the kinematic equations extracted were tested in a simulation environment and the simulation of movements was visualized. By performing the algorithms created in the simulation environment without using any mathematical library, it has been ensured that the algorithms can be tested in different environments without depending on any programming language. As shown in the simulation in the experimental results, it has been observed that the robot performs the desired movements realistically. The software, which was verified and tested in the simulation environment, was transferred to a real spider robot and the system was successfully operated.

**Keywords:** Hexpod, Kinematics, Spider Robot, Simulation, Unity.

# 1. GİRİŞ

Böceklerde ve eklembacaklılarda uzuvların yapısını ve hareket kontrolünü taklit etmek için oluşturulan çok ayaklı yürüyen robotlar üzerinde onlarca yıldır araştırmalar yapılmaktadır. Birden çok bacağa sahip robotlar arasında örümcek robot, çok çeşitli görevler için en çok kullanılan robotlardan biridir (Urvaev ve diğ., 2022). Örümcek robotların diğer çok bacaklı yürüyen robot türlerine göre birçok avantajı vardır. Hareket halindeyken kolayca dengelerini yakalayabilir ve koruyabilirler; farklı nitelikteki düzensiz yüzeylere uyum sağlama yeteneğine sahiptirler; bacak sayıları fazladır (bir uzuv kaybetseler bile görevlerine devam etmelerini sağlar); çok yönlüdürler ve çevresel koşullardan tekerlekli robotlara göre daha az etkilenirler (Tedeschi ve diğ., 2014).

Robot teknolojisinin gelişmesiyle birlikte robot uygulaması endüstriyel alanla sınırlı kalmamış ve giderek hizmet, tıbbi tedavi ve temizlik gibi daha fazla alana taşınmıştır. Örümcek robotların olası uygulama alanları arasında, volkanik keşif, kurtarma prosedürleri, kara mayınlarının tespiti, denizaltı operasyonları (deniz tabanı) yanı sıra numune toplama, yaşam arama, dünya dışı keşiflerde tanıma misyonları vardır. Bu görevlerin çoğu tehlikelidir ve genellikle insan çalışmalarına elverişli olmayan zorlu ortamlar eşlik eder (Roth, 2019).

Bir örümcek robot hareket ederken; her bir eklem birçok farklı derecelerde bulunabilir. Yapılan bazı uygulamalarda bacakların tüm durumları için her bir eklemin bulunması gereken dereceler, çıkartılır ve robot bu tablo üzerinden hareket eder (Alulema ve diğ., 2017). Bu tarz uygulamalarda robotun uzuvları değiştiğinde bu tabloların tekrar oluşturulması gerekir. Söz konusu tabloların oluşturulması ise oldukça zahmetli bir iştir. Ayrıca robot sadece bu tablo üzerinden hareket ettirildiğinde tablonun boyutu kadar hareket çözünürlüğüne sahip olur ve hareket kabiliyeti ise tablo ile sınırlı olur.

Bu tez çalışmasında ise, altı bacaklı bir örümcek robotun ileri ve ters kinematik denklemleri çıkarılıp örümceğin modeli simülasyon ortamında test edilmiş ve gerçek bir sistem üzerinde uygulanmıştır. Çıkartılan denklemler kullanılarak her bir bacağın yere temas eden noktası; geliştirilen hareket algoritmaları ile kontrol edilerek örümcek robotun gerçekçi bir şekilde yürümesi, kendi etrafında dönmesi sağlanmıştır.

Tez kapsamında yapılan çalışmalar aşağıda bölümler halinde açıklanmaktadır.

Bölüm 2'de, tez kapsamında geliştirilen örümcek robotun tanımına, literatürde yapılan benzer çalışmalara, mekanik özelliklerine, montajına ve üretimine yer verilmiştir. Robotun mimari diyagramı, kullanıcı durumları, yazılım senaryoları ve gereksinimleri gibi başlıklardan bahsedilmiştir.

Bölüm 3'te, bir örümcek robotun ileri ve ters kinematiğin temel tanımlamalarından bahsedilerek, bir örümcek robotun bacağının kinematik çözümleri yapılmıştır.

Bölüm 4'te, çıkartılan ileri ve ters kinematik denklemleri kullanılarak bir örümcek robotun yürümesi, dönmesi vb. hareketleri yapabilmesi için tez kapsamında geliştirilen algoritmalar açıklanmıştır.

Bölüm 5'te, Unity 3D oyun motorunun genel özelliklerinden bahsedilmiştir. Ayrıca tez kapsamında geliştirilen örümcek robotun Unity 3D oyun motoruna nasıl aktarıldığı ve geliştirme motorunun sahip olduğu ebeveyn-çocuk ilişkisi anlatılmıştır.

Bölüm 6'te, nesne yönelimli programlamadan (NYP), faydalarından, bu tez kapsamında oluşturulan nesnelerden ve bunların nasıl kullanıldığından bahsedilmiştir. Oluşturulan nesneler verilen Ek dosyalarda gösterilerek bu Ek dosyaların birbiriyle ilişkisi anlatılmıştır. NYP dışında; parametrik programlamadan ve tez içerisinde parametrik programlamanın nasıl kullanıldığını açıklayan Ek dosyalar ile gerekli açıklamalar yapılmıştır. Uzak kontrol yazılımı ve oluşturulan mesaj paketleri açıklanmıştır. Bir elektronik sistemin farklı networklerde bulunan arayüzler ile kontrol edilmesine olanak sağlayan MQTT yapısı ve bu yapının ücretsiz veya lisanslı kullanılmasını sağlayan ticari bir platform olan HiveMQ sunucusu anlatılmıştır.

Bölüm 6'da, çalışma kapsamında yapılan yazılımlar ile elde edilen sonuçlardan bahsedilmiştir. Çalışmanın sanal bir ortamda geliştirilip, gerçek bir sisteme aktarılmasının sonuçları ve faydaları anlatılmıştır.

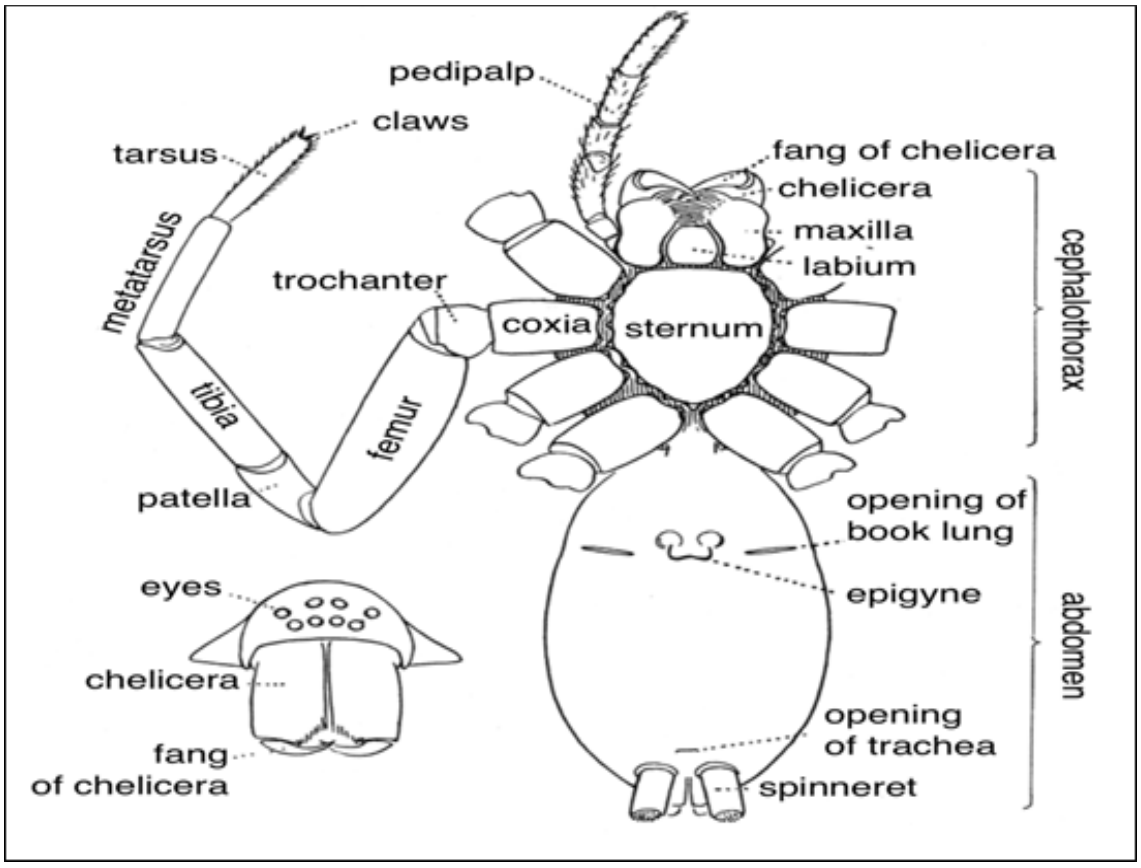
Çalışmada asıl amaçlardan biri de yazılım geliştirilmesinin yapılacağı gerçekçi bir sanal ortam kurarak burada geliştirilen algoritmaların gerçek bir sisteme aktarılmasıdır. Gerçek bir sisteme aktarılacak yazılım üzerinden yapılacak minimum değişiklikle örümcek

robotun gerçekleştirilmesi hedeflenmektedir. Ayrıca robotun herhangi bir bilgisayardan, herhangi bir konumdan kontrol edilmesi istendiğinde; ileride kamera vb sensörler takılması halinde uzaktan kontrol sırasında robotun görüş alanı izlenerek kontrol edilmesine teorik bir temel oluşturulması amaçlanmıştır.

## 2. GENEL BİLGİLER

Bu bölümde robotun tanımından, literatürde yapılan benzer çalışmalardan, mekanik tasarımdan, yazılım gereksinimlerinden, mimari diyagramından ve kullanıcı ile robot arasındaki ilişkiden söz edilmiştir. Tasarlanan örümcek robotun açısal olarak hareket kabiliyetleri mekanik tasarım bölümünde anlatılmıştır. Robotun üretilmesinde kullanılan parçalar ise malzeme ve yöntem bölümünde gösterilmiştir. Yazılım geliştirme öncesinde oluşturulan yazılım gereksinimleri açıklanmıştır. Bu gereksinimler doğrultusunda kullanıcının robot üzerindeki kabiliyetleri kullanıcı durumları bölümünde anlatılmıştır. Mimari diyagram üzerinde robotun sahip olduğu bileşenler gösterilmiştir. Kullanıcı ile robot arasında gerçekleşebilecek senaryolar, bu bölümünde gösterilmiştir.

Örümceklerin bacak yapılarına ilişkin genel olarak kabul edilen isimlendirmeler olan *coxia*, *femur* ve *tibia* gibi terimler Şekil 2.1'de gösterilmiştir (Jocqué, 2001). Bu terimler, örümcek anatomisinin temel yapı taşlarını ifade eder ve örümceklerin bacaklarının farklı bölümlerini tanımlamak için kullanılmıştır.




Şekil 2.1. Örümcek Anatomisi (URL-1)

*Coxia*, örümceğin bacağının vücuda bağlandığı ilk segmenti temsil eder. *Femur,* ikinci bacak segmentidir ve genellikle örümceğin hareket kabiliyetini sağlayan kaslar ve eklemlerle birleşmiştir. *Tibia, femur* segmentinden sonra gelen üçüncü bacak segmentidir ve örümceğin bacağının orta bölümünü oluşturur. Bu isimlendirmeler, örümceklerin bacaklarının anatomi ve morfolojisini ifade ederken yaygın olarak kullanılan standart terimlerdir. Bu nedenden dolayı örümcek bacaklarının anatomik terimleri; bir örümcek robotun tasarımında ve geliştirilmesinde kullanılarak doğal örümceklerin benzer hareketleri gerçekleştirebilme amacını taşır.

*Coxia* benzeri bir bağlantı parçası, örümcek robotunun bacağını gövdesine bağlar ve dönme hareketine izin verir. *Femur* segmenti, örümceğin bacağının ikinci segmentidir ve örümceklerin kasları ve eklemleri ile birleştiği kritik bir bölümdür. Örümcek robotlarda *femur* benzeri bir yapı, bacak hareketini kontrol etmek için kullanılır. Bu segment, robotun bacağının uzunluğunu ve hareket kabiliyetini sağlar. *Tibia*, örümceğin bacağının üçüncü segmentidir ve örümceklerin bacaklarının orta bölümünü oluşturur. Örümcek robotlarda, *tibia* benzeri bir bacak segmenti, genellikle *femur* ile birleşir ve bacağın uzunluğunu artırarak hareketin stabilitesini ve etkinliğini artırır.

Örümceklerin bacaklarının anatomik terimleri, örümcek robotlarının tasarımında temel bir referans noktası sağlar ve bu robotların doğal hareket yeteneklerini taklit etmeyi amaçlayan birçok araştırma ve geliştirme çalışmasının temelini oluşturur. Bu bağlamda, örümcek robotların tasarımı ve hareket mekanizmaları konusunda birçok önemli literatür çalışması bulunmaktadır. Ayrıca literatürdeki bazı çalışmalarda, deney setinde testler yapılmadan önce gerçekçi bir sanal ortamda denemelerin yapılması algoritmaların geliştirilmesi, gerçek dünyada yapılacak birçok deneyin daha hızlı test edilebilmesini için Unity 3D oyun motoru kullanılmıştır. Literatürde bulunan bazı çalışmalar aşağıdaki gibidir.

Sun ve arkadaşları (2022), altı bacaklı bir örümcek robotun yürüme ve dönme gibi yeteneklerini deneysel olarak gerçekleştirmişlerdir. Çalışmada robotun hareket algoritmalarının gerçekleştirilebilmesi için ileri ve ters kinematik denklemleri, kinematik denklemlerin çıkartılabilmesi için D-H parametrelerini kullanmışlardır. Robotun ayak uçlarındaki hareket yörüngelerini, kosinüs, sinüs ve düz çizgi fonksiyonları ile planlayarak, bu yörüngelerin uygulanmasıyla ilgili hareketleri gerçekleştirmişlerdir.

Erkol (2015), çok eklemli robotların gerçek zamanlı kinematik kontrolü üzerine bir tez çalışması gerçekleştirmiştir. Çalışmada kinematik hesapların getirdiği işlem yüküne odaklanılmış ve bunun için FPGA tabanlı bir donanım geliştirmiştir. Geliştirilen bu donanım ile kinematik hesaplar yapılarak altı bacaklı örümcek robotun hareket kontrolü gerçekleştirmiştir. Kinematik hesapların oluşturulabilmesi için D-H parametrelerini kullanmışlardır. Robot üzerindeki her bir ayak tabanı, ilgili bacağın gövdeye bağlandığı koordinat sistemine göre referans alınarak yörüngeler oluşturulmuş ve robot hareket ettirilmiştir.

Xu ve arkadaşları (2019), duvara tırmanma yeteneğine sahip altı bacaklı bir robotun kinematiği ve kararlılığı üzerine çalışmalarda bulunmuşlardır. Bir örümcek robotun bacak hareketlerinin vücut ile ilişkisini, matematiksel denklemlerle açıklamışlardır. Bu kinematik modellemeyi, robotun belirli bir konumda nasıl hareket edeceğini ve bir duvara tırmanma görevini nasıl yerine getireceğini açıklamak için kullanmışlardır.

Alulema ve arkadaşları (2017), bir örümcek robotun eğimli yüzeylerde dengede kalma problemini incelemişlerdir. Eğimli bir yüzeyde robotun dengede kalma problemini çözmek için PI ve bulanık mantık kontrol yöntemlerini kullanarak tasarladıkları deneysel sete gerçek zamanlı olarak uygulamışlardır.

Ekelund (2018), bir örümcek robotun yürümesi ve dengede durması için bir tez çalışması gerçekleştirmiştir. Sistemin dengede durabilmesi için PID kontrolü kullanmıştır. Robotun yürüme ve dengede durması için gerçekleştirmiş olduğu algoritmalarını hem simülasyon ortamında test etmiş hem de gerçek bir sistemde deneysel sonuçlarını incelemiştir.

Thilderkvist ve Svensson (2015), bir örümcek robotun kontrolü için model tabanlı tasarım kullanarak tez çalışması gerçekleştirmiştir. Örümcek robotun hareket, yürüme, dönme ve vücut yüksekliğini değiştirmek için geliştirilmeler yapmışlardır. Aynı zamanda tez çalışmasında başarılı bir dengeleme modu içermektedir. Yapmış oldukları deneysel sonuçlarda robotun bulunduğu zemin açısı değişirken ana gövdeyi düz tutabilmişlerdir.

Yamağan (2013), yapmış olduğu tez çalışmasında altı bacaklı bir örümcek robotun, kinematik analizini ve gerçek zamanlı bir sisteme uygulanabilmesi için MATLAB ortamında simülasyon gerçekleştirmiştir. Gerçekleştirilen bu çalışmanın, gerçek

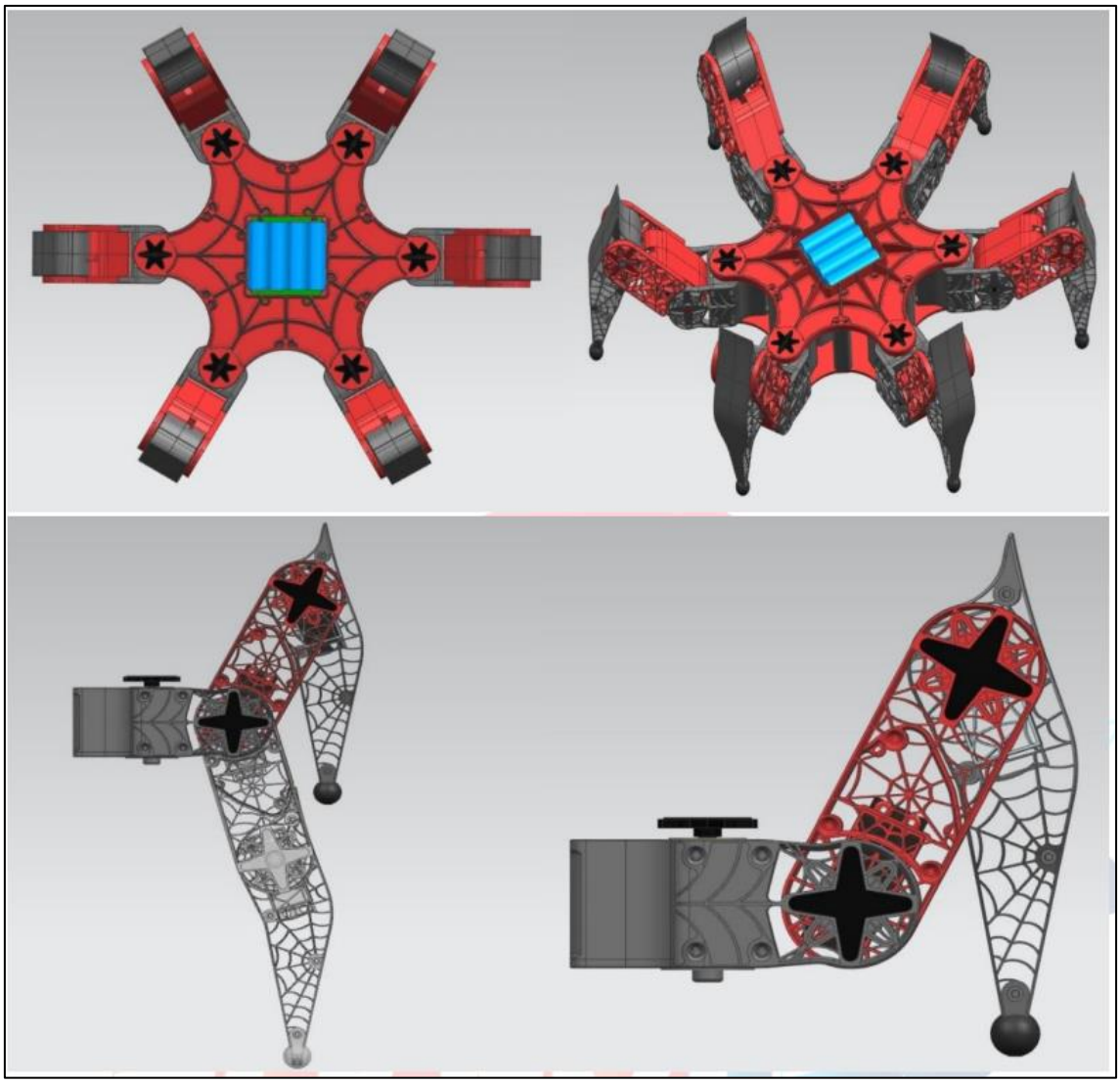
uygulamaların donanımsal olarak benzetimi ve hızlı prototipleme uygulamaları için bir ön çalışma olması amaçlanmıştır. Tasarlanan sistemin ileri/ters kinematiğinin çözümlenmesinde yaygın olarak kullanılan D-H yaklaşımından yararlanılmış, ayrıca robot dinamiğinin çözümünde kullanılan Lagrange-Euler denklemine de değinilmiştir.

Arshad ve arkadaşları (2018), tarafından oluşturulan robot kinematik öğrenme sistemi, Unity 3D motorunu temel alarak geliştirmişlerdir. Çalışmada, robotik öğrenmenin önemine ve sınıfta öğretilen materyallerin öğrenme sonuçlarını iyileştirmek için kullanılabilecek bir sanal öğrenme platformunun avantajlarına ve geliştirilmesine değinmişlerdir. Çalışmada öğrencilerin robotlarla ilgili konularda teorik bilgileri öğrenmelerine ve aynı zamanda robotların kinematik hareketlerini modellemelerine olanak tanıyan bir yazılım aracı geliştirmişlerdir.

Khoswanto ve arkadaşları (2021), Unity 3D oyun motorunu kullanarak bir üç eksenli bir manipülatörün ileri ve ters kinematikleri açıklamışlardır. Çalışmada bir manipülatörün kinematik modeli ve kinematik çözümüne odaklanmışlardır. Bu makalede, manipülatörün konum ve yönelimini hesaplamak için D-H parametreleri ve manipülatörün ters kinematik çözümü için Jacobian matrisi kullanmışlardır.

### 2.1. Mekanik Tasarım

Çok ayaklı robotlar sınıflandırmasında yer alan örümcek robot, hareket etmek için altı ayağı olan bir robot olarak tanımlanmaktadır. Her bir bacak 3 serbestlik derecesine sahiptir. Robot üzerinde bulunan 6 bacaktan ve her bacaktaki 3 eklem ile toplam 18 eklemden oluşmaktadır. Bu sayede gelişmiş hareket kabiliyetine sahiptir. Üzerinde bulunan bacaklar sayesinde istenildiğinde yüksekliğini ayarlanabilmekte ve yer düzlemine olan açısını değiştirebilmektedir. Gövdenin bulunduğu farklı eğimlerde ve yüksekliklerde robot hareket edebilmektedir. Tasarlanan robotun gövde kısmı 70 mm yükseklik ve 290 mm genişliğe sahip simetrik bir yapıdadır. Aracın her biri üç eklem ve üç servo motora sahip altı adet hareketli bacaktan oluşmaktadır. Bu bacaklar hareket aktarıcı, ara eklem ve ayak parçasının servo motorlar yardımıyla hareket etmesiyle yürüme işlemini yapmaktadır. Bacaklar ara eklemin 180°'lik hareketi, ayak kısmının ise 135°'lik hareketi ile gövdenin yerden 15 mm ile 190 mm arasındaki yüksekliklerde hareket edebilecek kabiliyette tasarlanmıştır.

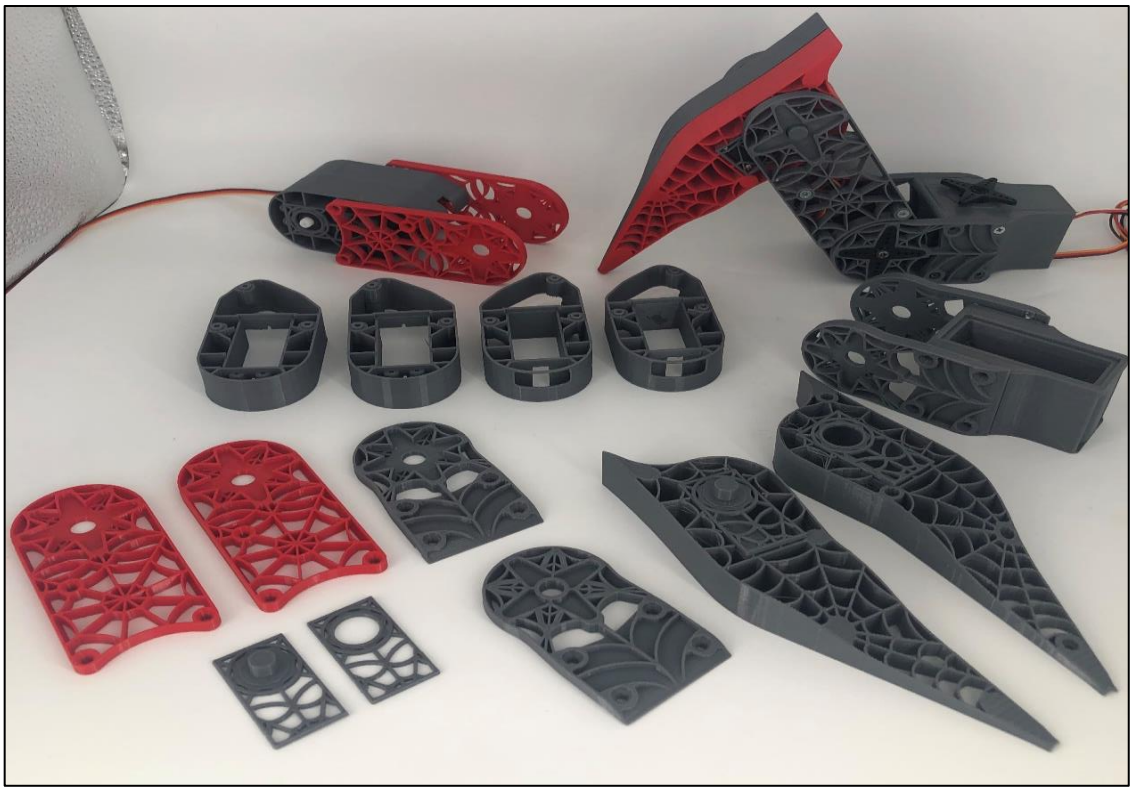


Şekil 2.2. Örümcek Robot

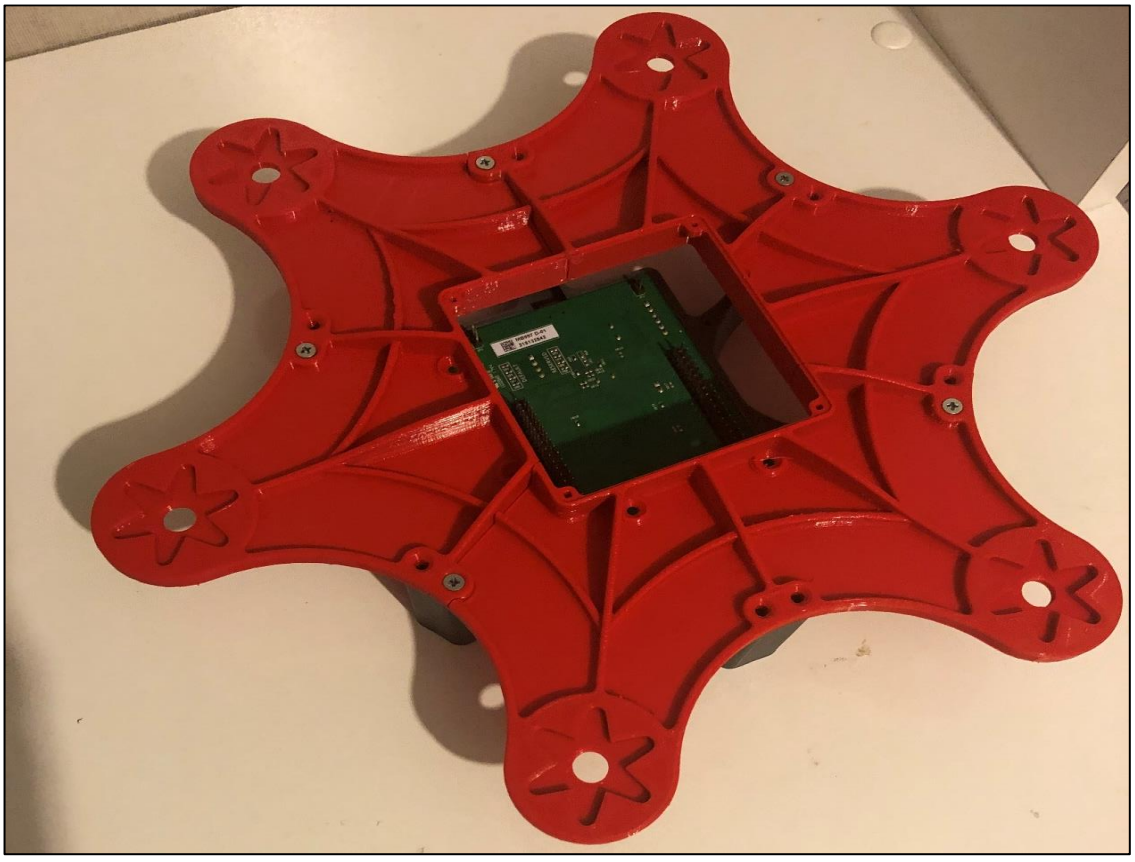
Şekil 2.2'de tez için tasarlanan örümcek robot gösterilmektedir. Şekilde bir bacağın açısal olarak hareket kabiliyeti, bacak sayısı ve her bir bacağın gövdeye monte edildiği yerler gösterilmektedir. Robot örümcek ağını andıran güçlendirme kirişleri ile estetik görünüm, hafiflik ve sağlam olacak şekilde tasarlanmıştır. Üzerinde devre kartı, pil yuvası ve altı adet bacak için 60° aralıklı bağlantı noktaları bulunmaktadır.

## 2.2. Malzeme ve Yöntem

Robotun kontrolü için STM32F407VG isimli ST firmasına ait bir geliştirme kartı ve STM32 ile bir web uygulamasından haberleşebilmek için ESP32-WROOM-32D geliştirme kartı kullanılmıştır. Robot üzerinde bulunan eklemlerin kontrolü 18 adet MG995 metal dişli servo motor ile sağlanmıştır.

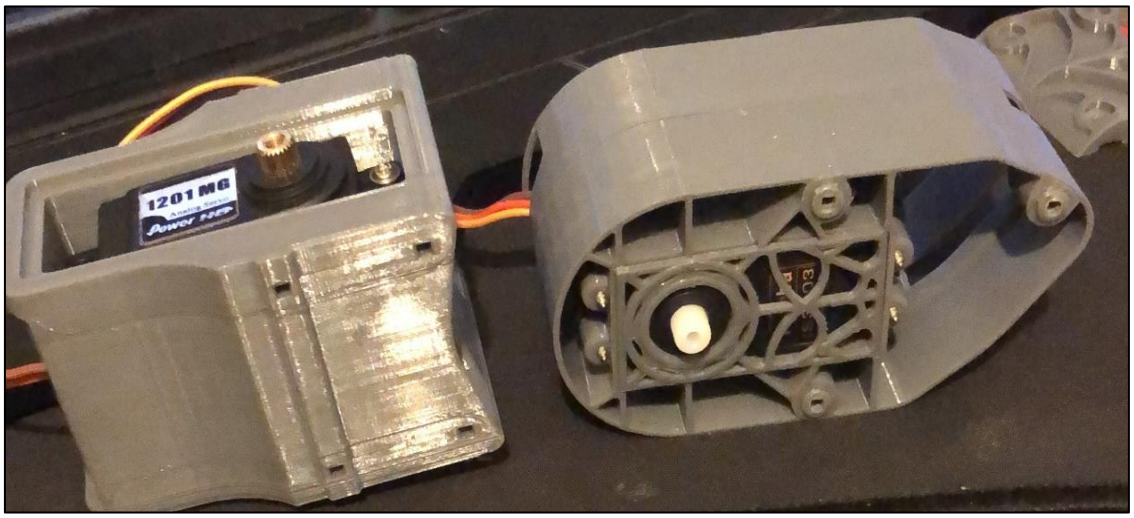


Şekil 2.3. 3D yazıcı çıktısı



Şekil 2.4. Gövde 3D yazıcı çıktısı

Örümcek robot Şekil 2.3'de gösterildiği gibi bir 3D yazıcı kullanılarak üretilmiştir. 3D yazıcı baskı malzemesi olarak PLA filament kullanılmıştır. Bu servo motorlar Şekil 2.5'de gösterildiği gibi eklemlere, mikroişlemciler ise Şekil 2.4'de gösterildiği gibi gövdeye monte edilmiştir.




Şekil 2.5. Eklemler 3D yazıcı çıktısı

Her bir bacak Şekil 2.4'de gösterilen gövdeye monte edilmiştir. Eklemler ise Şekil 2.5'de gösterildiği gibi üretilmiştir.

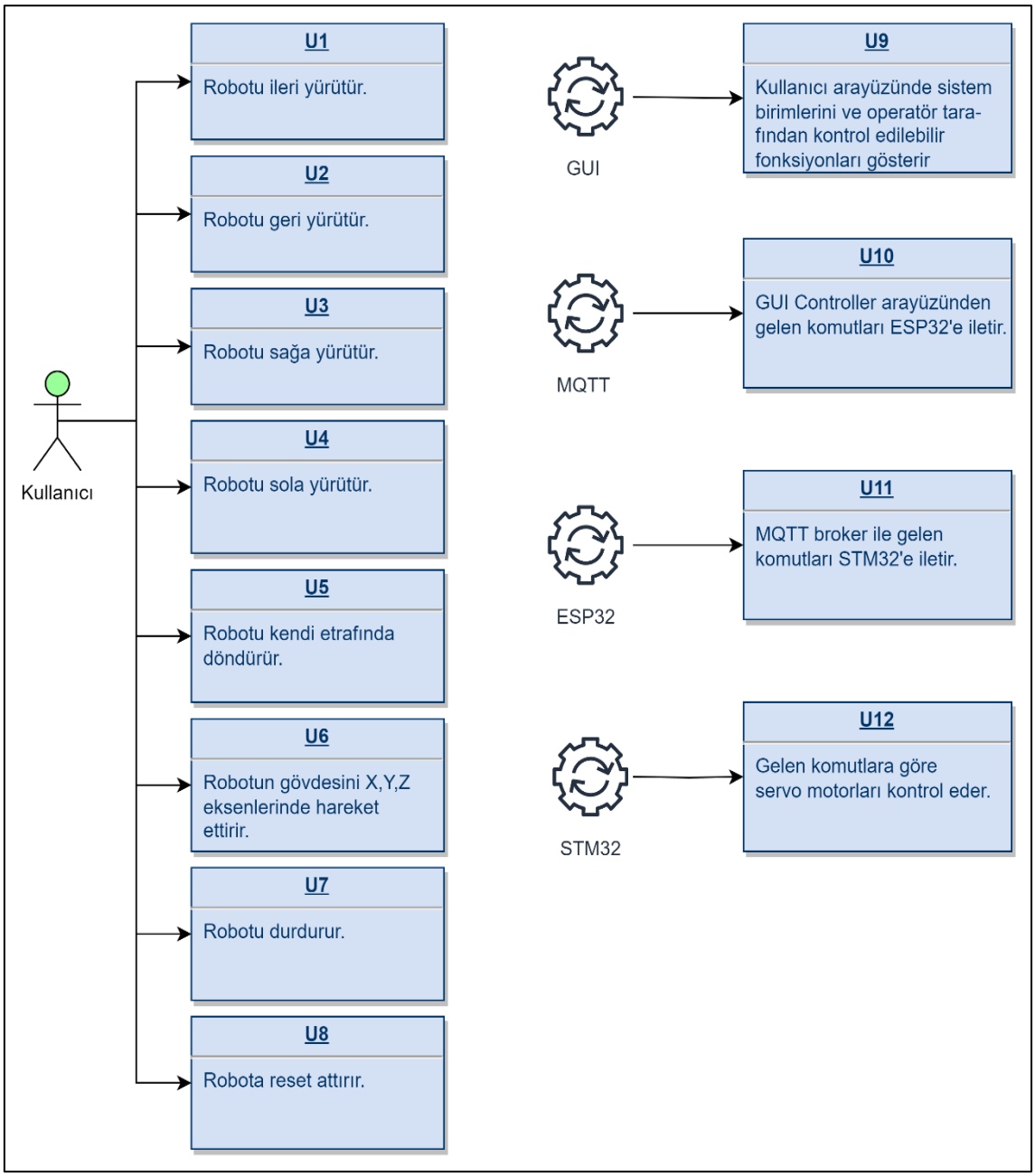
## 2.3. Gereksinimler

Robotun ve yazılımın kabiliyetlerinin belirlenmesi için yazılım geliştirme aşamasından önce aşağıdaki gereksinim maddeleri oluşturulmuştur.

R1. Kullanıcı robotu bir arayüz üzerinden ileri yönde hareket ettirebilmeli.
R2. Kullanıcı robotu bir arayüz üzerinden geri yönde hareket ettirebilmeli.
R3. Kullanıcı robotu bir arayüz üzerinden sağ yönde hareket ettirebilmeli.
R4. Kullanıcı robotu bir arayüz üzerinden sol yönde hareket ettirebilmeli.
R5. Kullanıcı robotu bir arayüz üzerinden olduğu yönde döndürebilmeli.
R6. Kullanıcı robot hareket etmeden sadece gövdesini kontrol edebilmeli.
R7. Kullanıcı gerektiğinde robota reset atabilmeli.
R8. Kullanıcı; arayüz ve robotun farklı ağlarda ve sabit ip olmadan herhangi bir bilgisayardan robotu kontrol edebilmeli.
R9. Kullanıcı gerektiğinde robotu durdurabilmeli.

## 2.4. Kullanıcı Durumları

Tez kapsamında gerçekleştirilen örümcek robot için çıkartılan gereksinimler doğrultusunda Şekil 2.6'daki kullanıcı durumları oluşturulmuştur. Şekil 2.6'da gösterildiği üzere kullanıcı robotu sağa, sola, ileri ve geri hareket ettirebilir. Ayrıca robotun gövdesini kontrol edebilir. Bu işlemlerin yapılabilmesi için gerekli olan birimler Şekil 2.6'da gösterildiği gibidir.




Şekil 2.6. Sistem kullanım durumu

### 2.4.1. GUI Kontrol Birimi

Bu birim içerisinde React ve NodeJS uygulamalarının bulunduğu birimdir. React kullanıcı arayüzü oluşturmaya yarayan açık kaynak kodlu bir Javascript kütüphanesidir. Kullanıcı bu arayüz üzerinde çeşitli işlemler yaptığında bu işlemler NodeJS uygulamasında gönderilir. NodeJS ise bu komutları bağlı olduğu bir MQTT birimine göndererek komutların robota gönderilmesini sağlar.

### 2.4.2. MQTT Birimi

Bu birim farkı network ağlarına bağlı olan, sabit IP adresine sahip olmayan cihazların haberleşmesi için kullanılır. Bu tezde GUI Kontrol birimi ile ESP32'nin haberleştirilmesi için kullanılmıştır. Bu sayede kullanıcının web arayüzünde gerçekleştirmiş olduğu komutlar ESP32'e iletilmiştir.

### 2.4.3. ESP32 Birimi

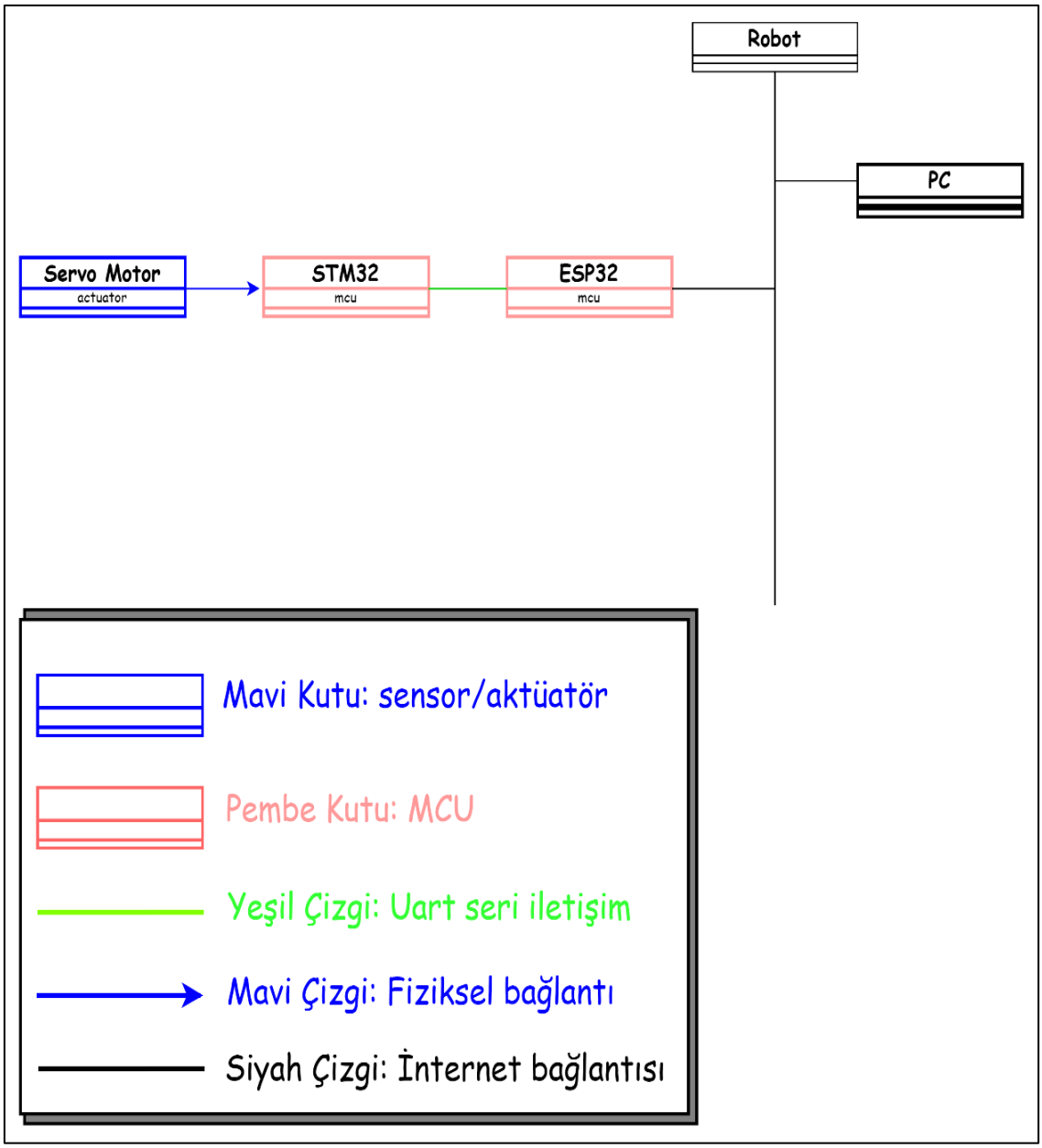
Bu birim kendisine MQTT üzerinden gelen mesajları, STM32'nin anlayabileceği mesaj yapısına çevirerek STM32'e gönderen birimdir. Bu birim sayesinde STM32'e herhangi bir networke bağlı olan bir bilgisayardan komutlar gönderilebilmektedir.

### 2.4.4. STM32 Birimi

Bu birim ESP32'den kendisine gelen komutlar doğrultusunda çeşitli kinematik hesaplar yapılması ve robotun kontrolü için oluşturulmuştur. Kullanıcının yaptığı işlemin komut olarak gönderildiği son birim bu birimdir. Diğer her bir birim kullanıcının yaptığı işlemi bu birime aktarmak için geliştirilmiştir.

## 2.5. Mimari Diyagram

Sistem içerisinden bir robotun kinematik algoritmaların koştuğu hareketin kontrolü için geliştirilen kodun gömülü olduğu bir STM32 isimli ARM tabanlı mikroişlemci kullanılmıştır. STM32, ESP32'den gelen komutlar doğrultusunda çeşitli kinematik hesaplar yaparak kendisine fiziksel olarak bağlı servo motorları Sinyal Genişlik Modülasyonu (Pulse Width Modulation) (PWM) ile sürerek robota hareketler yaptırılmıştır.




Şekil 2.7. Mimari Diyagram

Sistemin genel mimarisi Şekil 2.7'de gösterilmektedir. Genel mimaride gösterildiği gibi STM32'e ESP32 üzerinden gelen mesajlar bir network ağ yapısı üzerinden gelmektedir. Bu nedenle Wi-Fi bağlantı özelliklerine sahip ESP32 kullanılmıştır. ESP32 160-240 MHz hız bantları arasında çalışan bir işlemcidir. İçerisinde analog, dijital ve birçok haberleşme gibi çevre birimleri vardır (Oner, 2021). STM32 ise 168 MHz ile çalışan ESP32'e göre daha çok pin bulundurmaktadır (Pakdel, 2020). Bu nedenle 18 servo motorun sürülmesinde STM32 kullanılmıştır. Genel mimaride gösterildiği gibi ESP32 ile STM32 arasında haberleşme UART protokolü ile gerçekleştirilmiştir.

## 2.6. Senaryolar

Oluşturulan sistem kullanım durumu doğrultusunda ortaya çıkan senaryolar sırası ile bu bölümde anlatılmaktadır.

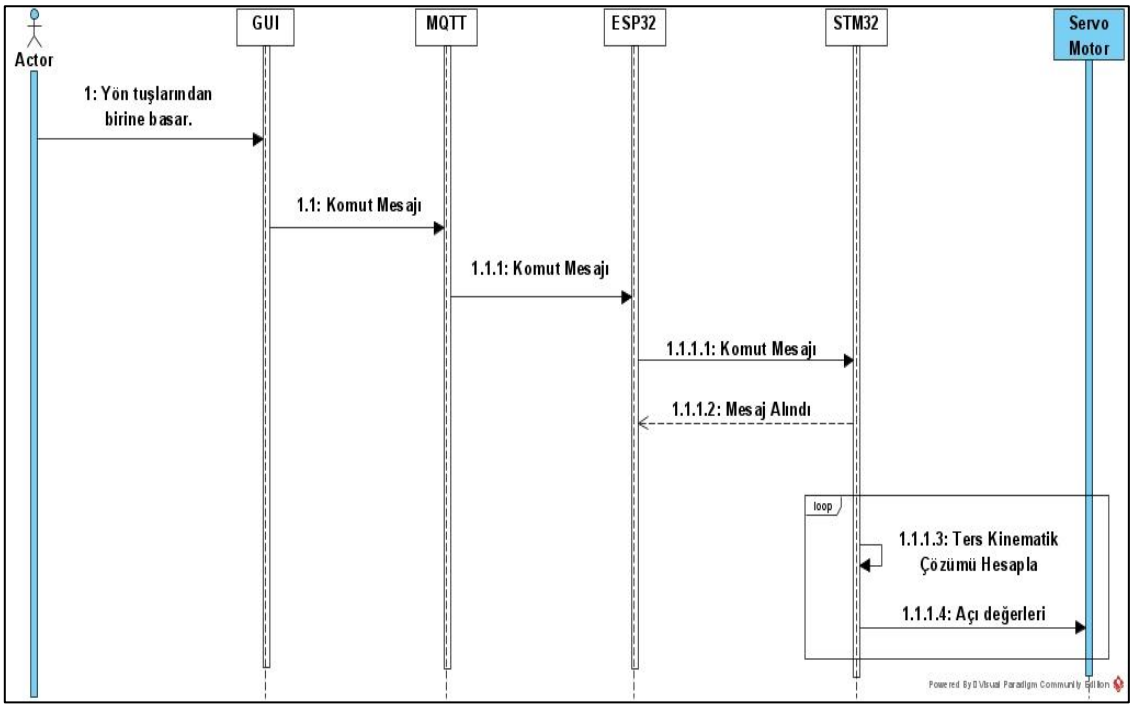
### 2.6.1. Kullanıcı Durumu (1-5)

Durum: Robota ileri, geri, sağ, sol yönlere veya kendi etrafında döndürme hareketi yaptırır.

Senaryo: Kullanıcı yön tuşlarına veya döndürme butonuna basar.

Ön koşullar:

- Robotta güç vardır.
- Kullanıcı arayüzüne girilmiştir.
- Herhangi bir tuşa basılı değildir.




Şekil 2.8. Senaryo 1-2-3-4-5

Sonraki koşullar:

- Robot basılı yönde hareket etmektedir.
- İlgili yöndeki tuş basılı konumdadır.

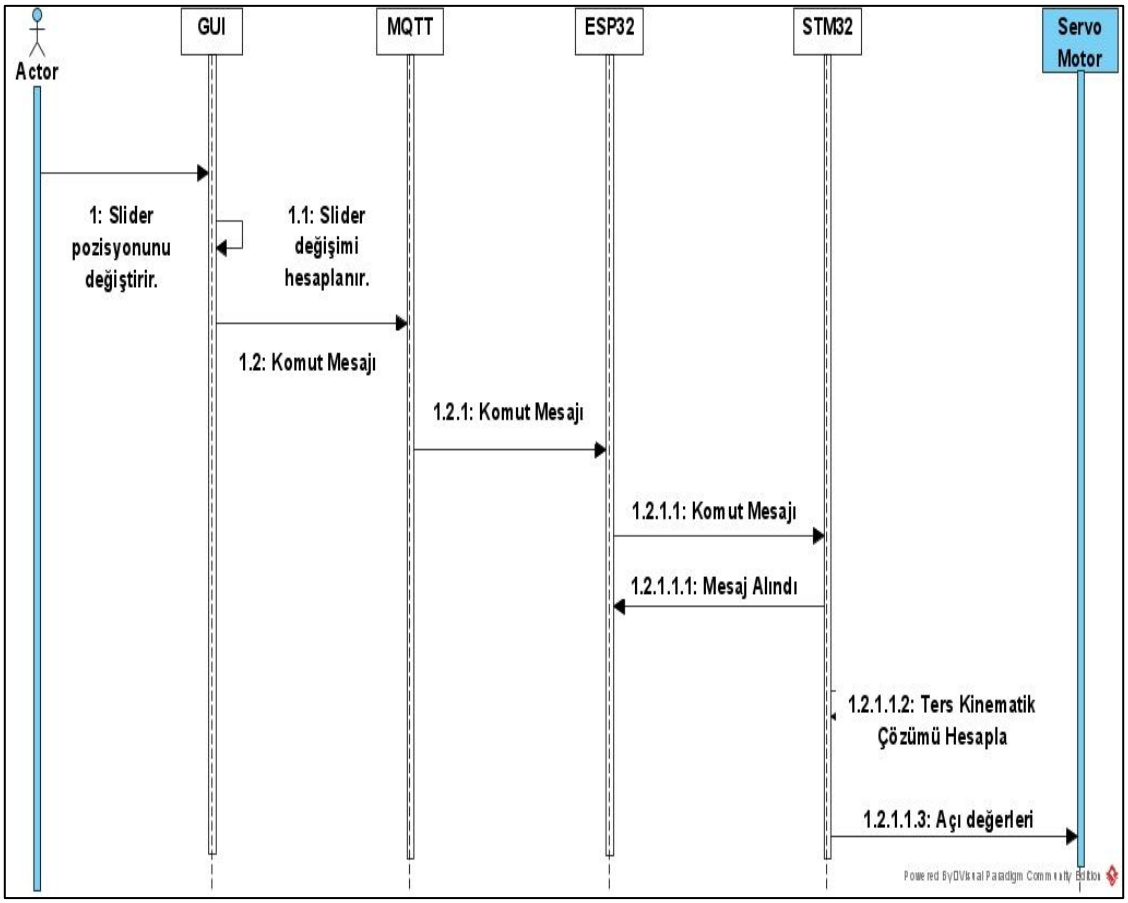
### 2.6.2. Kullanıcı Durumu-6

Durum: Sadece robotun gövdesini hareket ettirir.

Senaryo: Kullanıcı slider ile gövdeyi kontrol eder.

Ön koşullar:

- Robotta güç vardır.
- Kullanıcı arayüzüne girilmiştir.




Şekil 2.9. Senaryo 6

Sonraki koşullar:

- Robot gövdesi hareket eder.

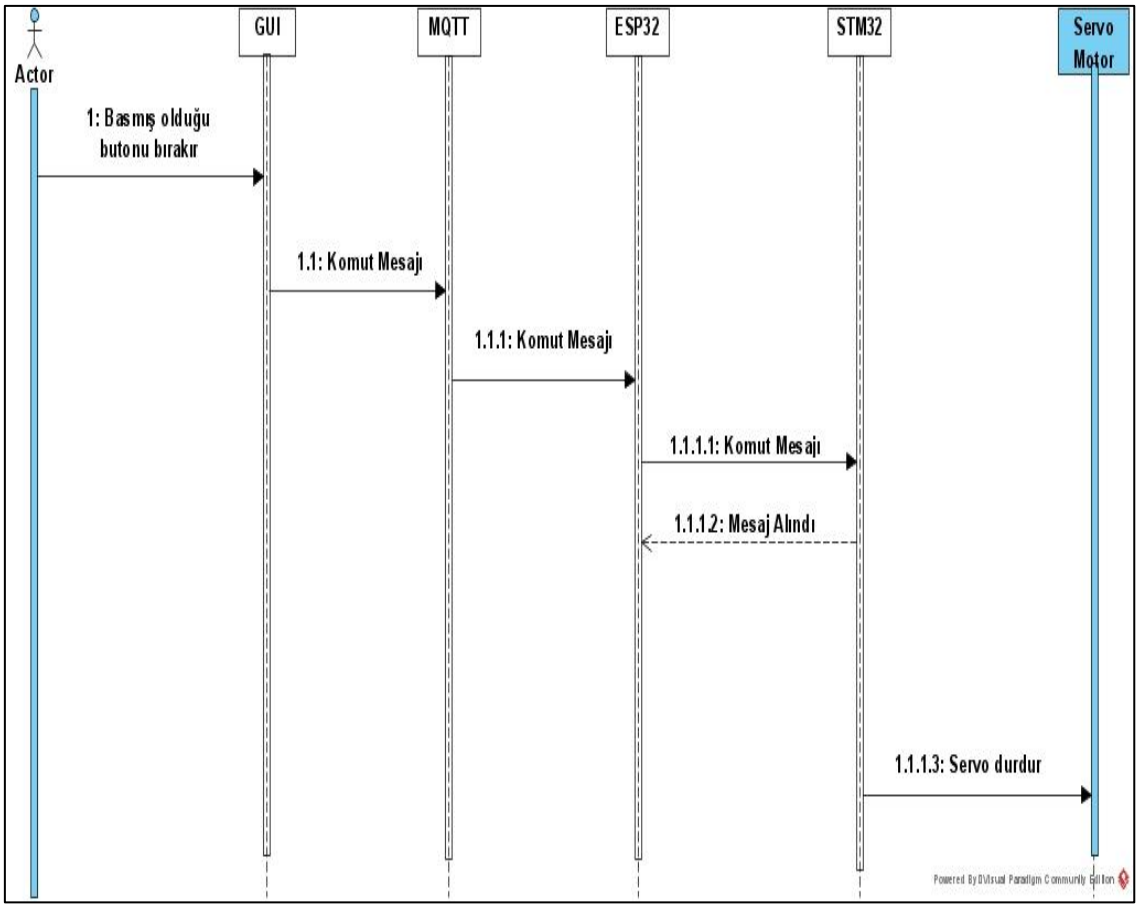
### 2.6.3. Kullanıcı Durumu-7

Durum: Robotu durdurur.

Senaryo: Kullanıcı basmış olduğu yön tuşlarından birini bırakır.

Ön koşullar:

- Robotta güç vardır.
- Yön tuşlarından birine basılıdır.
- Robot hareket etmektedir.
- Kullanıcı arayüzüne girilmiştir.




Şekil 2.10. Senaryo 7

Sonraki koşullar:

- Robot durmuştur.
- Yön tuşları bırakılmıştır.

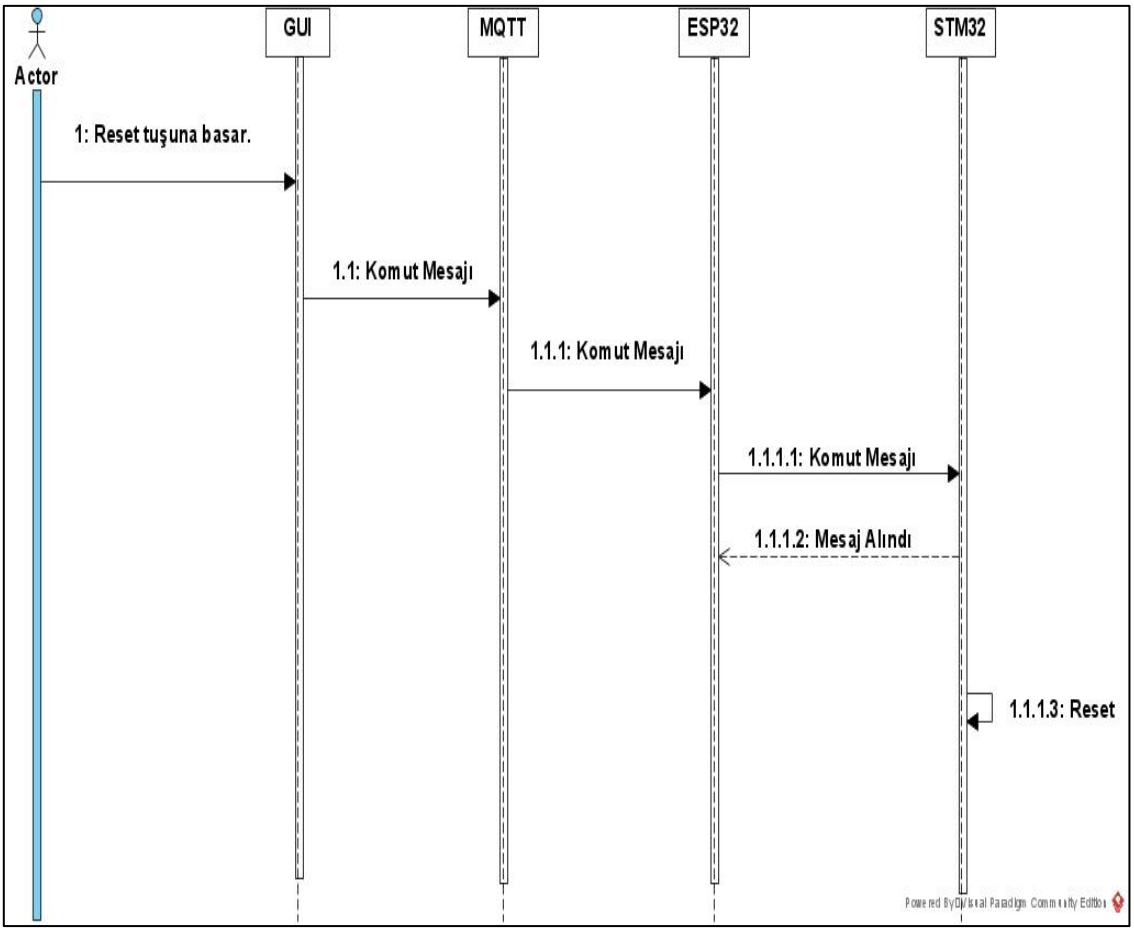
### 2.6.4. Kullanıcı Durumu-8

Durum: Robota reset atılır.

Senaryo: Kullanıcı reset tuşuna basar.

Ön koşullar:

- Robotta güç vardır.
- Kullanıcı arayüzüne girilmiştir.




Şekil 2.11. Senaryo 8

Sonraki koşullar:

- İşlemciye doğrudan reset atılır.
- Sistem başlangıç pozisyonuna döner.

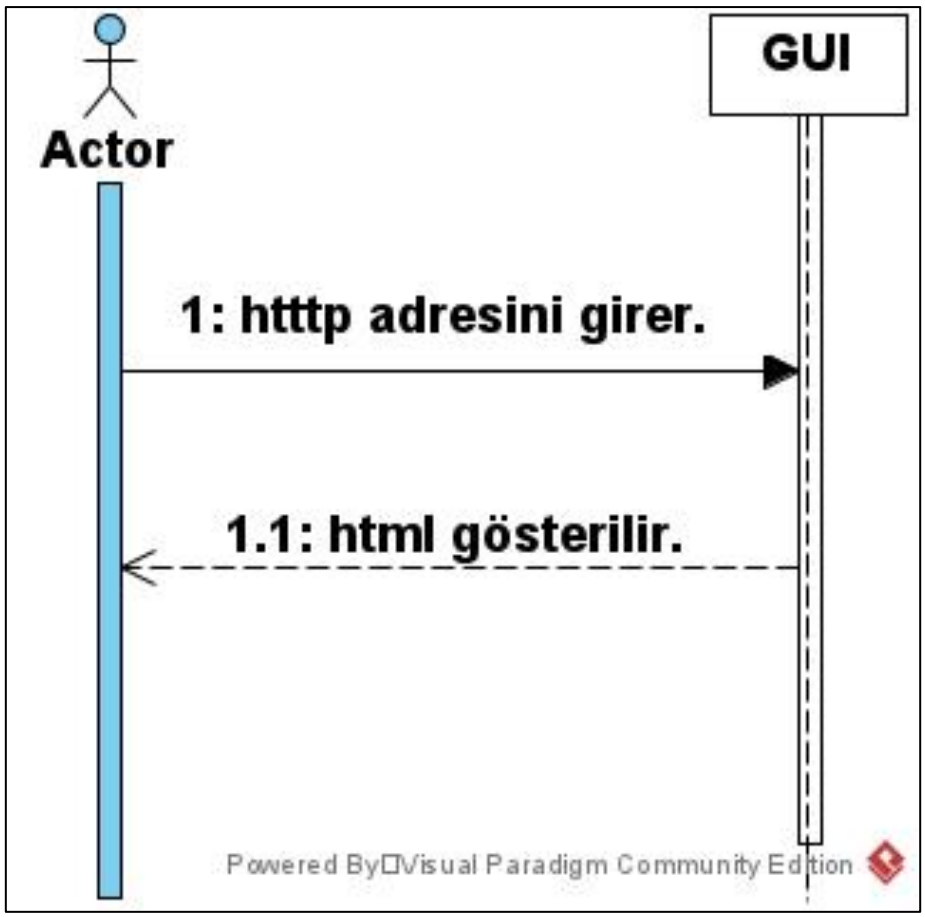
### 2.6.5. Kullanıcı Durumu-9

Durum: Kullanıcı arayüzünde sistem birimlerini ve operatör tarafından kontrol edilebilir fonksiyonları gösterir.

Senaryo: GUI kontroller birimi html dosyasını gösterir.

Ön koşullar:

- Robotta güç vardır.
- Kullanıcı arayüzüne girilmemiştir.




Şekil 2.12. Senaryo 9

Sonraki koşullar:

- Kullanıcı arayüzü gösterilir.

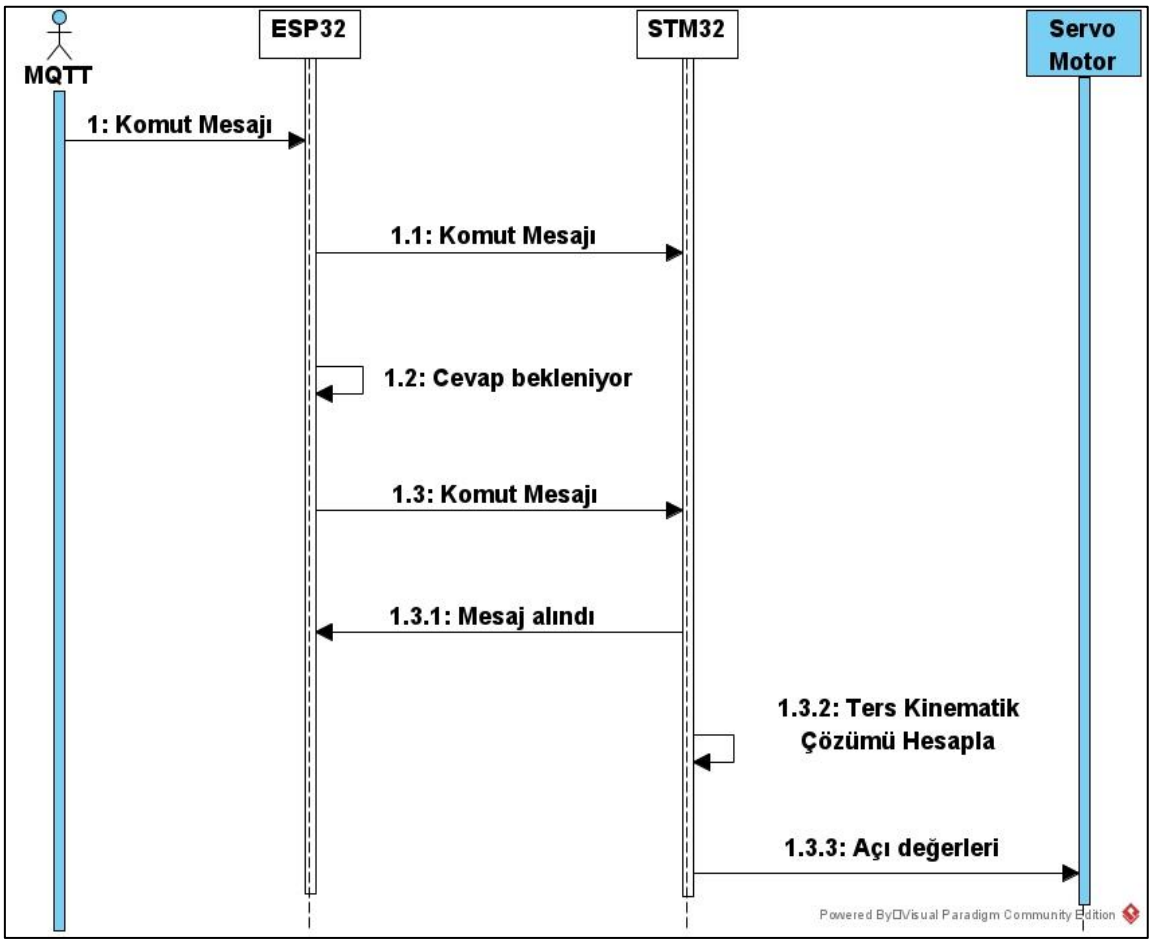
### 2.6.6. Kullanıcı Durumu-10

Durum: MQTT üzerine gelen mesaj ESP32'e aktarılır.

Senaryo: Kullanıcı herhangi bir işlem yapmıştır. ESP32'e bir mesaj gelmiştir. Fakat çeşitli nedenlerden dolayı STM32 mesajı doğru alamamış ve okey komutu göndermemiştir. Bu durumda ESP32 belirli bir süre bekler ve mesajı tekrar gönderir.

Ön koşullar:

- Robotta güç vardır.



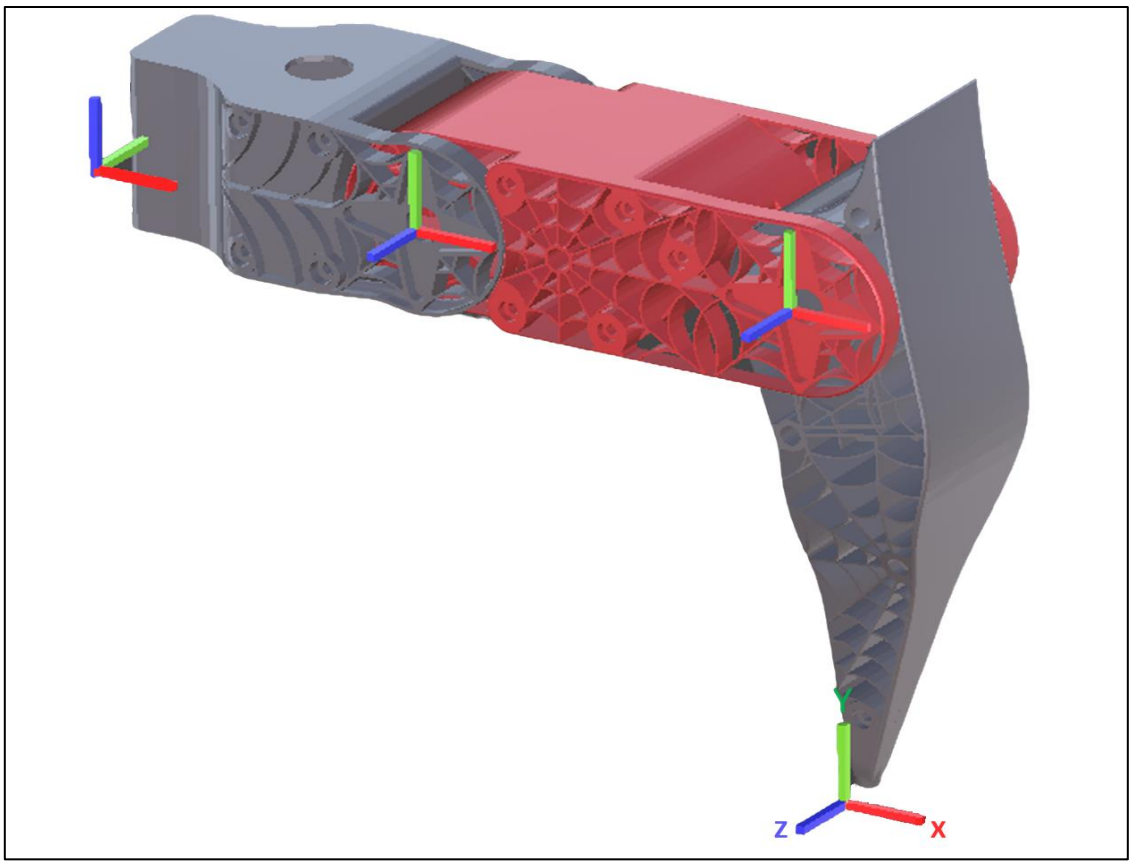

Şekil 2.13. Senaryo 10

Sonraki koşullar:

- İlgili komut doğrusunda ilgili işlem yaptırılır.

## 3. KİNEMATİK

Bu bölümde tez kapsamında geliştirilen bir örümcek robotun ileri kinematik ve geri kinematik çözümleri anlatılmaktadır. Robotun yürümesi ve benzeri çeşitli hareketlerinde kullanılan algoritmalar bu bölümde çıkartılan temel kinematik denklemler kullanılarak geliştirilmiştir.

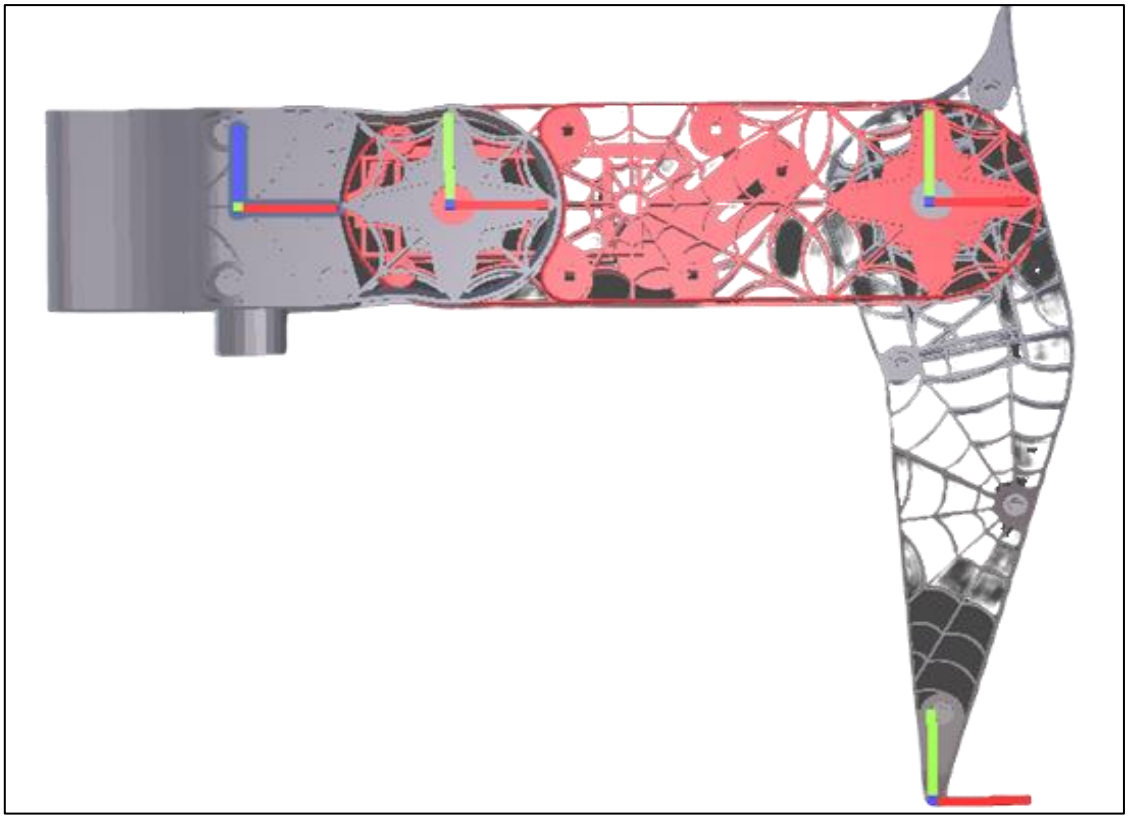



Şekil 3.1. Bir bacak üzerinde bulunan eksenler

Her bir eklemdeki serbestlik derecelerinin dönüş yönleri Şekil 3.1'de gösterildiği gibidir. Bacakların sahip olduğu ara eklemlerin hepsi $Z$ ekseni etrafında dönebilmektedir. Robotun hareket ettirilmesinde kullanılan hayali koordinat sistemleri bulunmaktadır. Bunlardan biri gövdenin ortasında diğerleri ise her bir bacağın gövdeye olan bağlantı noktasında bulunmaktadır. Ayrıca her bir bacağın ayak tabanında hayali noktalar bulunmaktadır. Her bir bacaktaki eklemlerde birer adet hayali koordinat sitemi bulunmaktadır. Bu hayali noktaların pozisyonları, kendi bacağını robotun gövdesine bağlandığı yerdeki koordinat sistemine göre çözüm yaparak bulunur. Bu hayali noktalar istenilen pozisyonlara getirilerek robota hareketler yaptırılmıştır.

### 3.1. İleri Kinematik

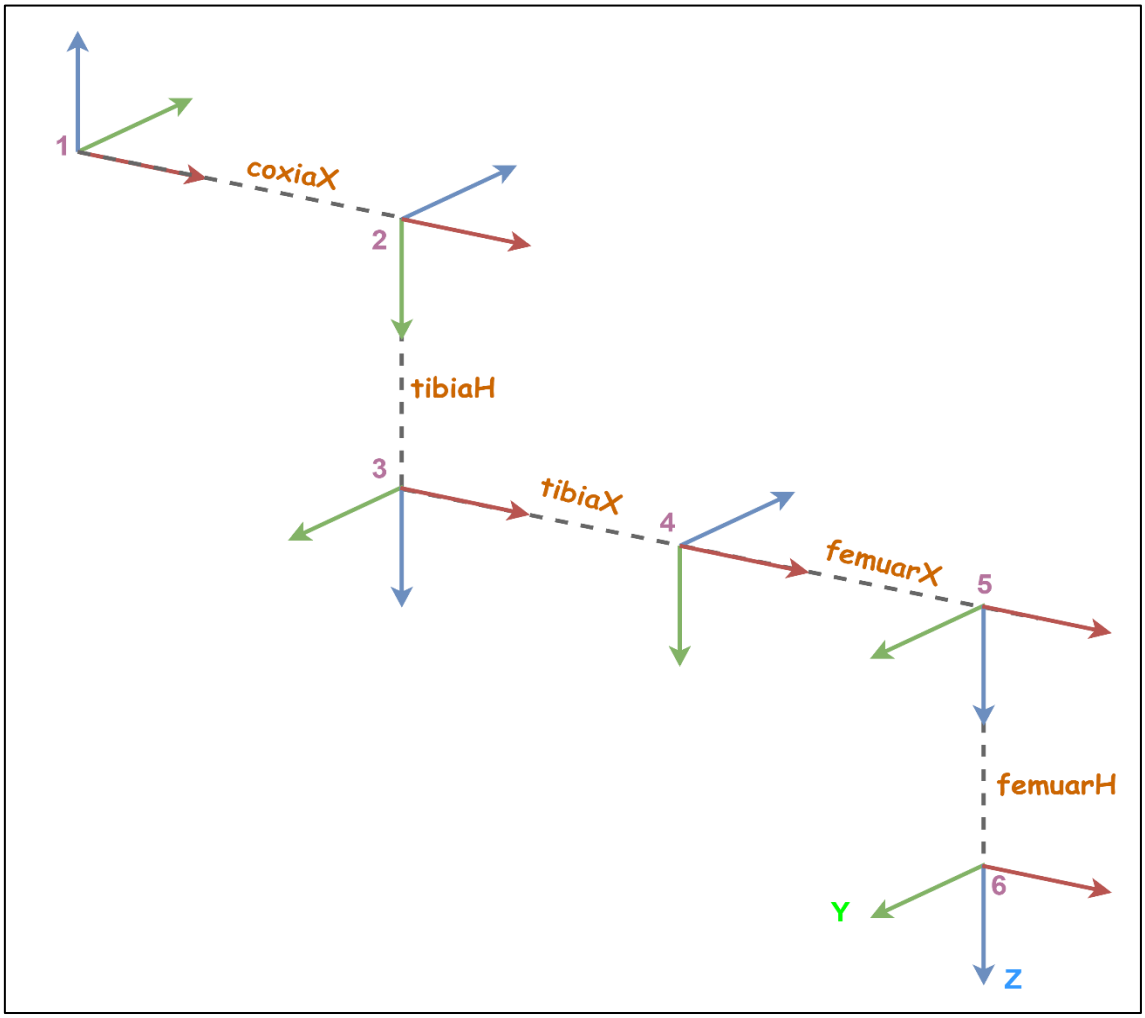
İncelenen robotun kinematiğini elde etmek için, D-H algoritmasını kullanmak ve bunu örümcek robotun bir bacağına uygulamak gerekir (Bingül ve Küçük, 2015). Bu robot, her bir bacakta üç serbestlik derecesine sahip altı özdeş bacaktan oluşan simetrik bir yapıdan oluşur. Şekil 3.1'de gösterildiği gibi bir bacak 3 ana eksenden ve bacak tabanının pozisyonun tutulduğu eksenden oluşmaktadır. Koordinat sisteminde mavi olan eksen *Z* eksenini, yeşil olan eksen *Y* eksenini, kırmızı olan eksen ise *X* eksenini göstermektedir. Şekil 3.1'de gösterildiği gibi ilk 3 eksen *Z* ekseni etrafında dönme kabiliyetine sahiptir. 4. eksen ise ayak tabanının pozisyonunu tutmaktadır. Bu koordinat sistemi ayak tabanının pozisyon bilgisini tutmak için konulduğundan herhangi bir dönme yetisine sahip değildir.



Şekil 3.2. Bir bacağın ayak tabanındaki Y ekseninden görünüşü

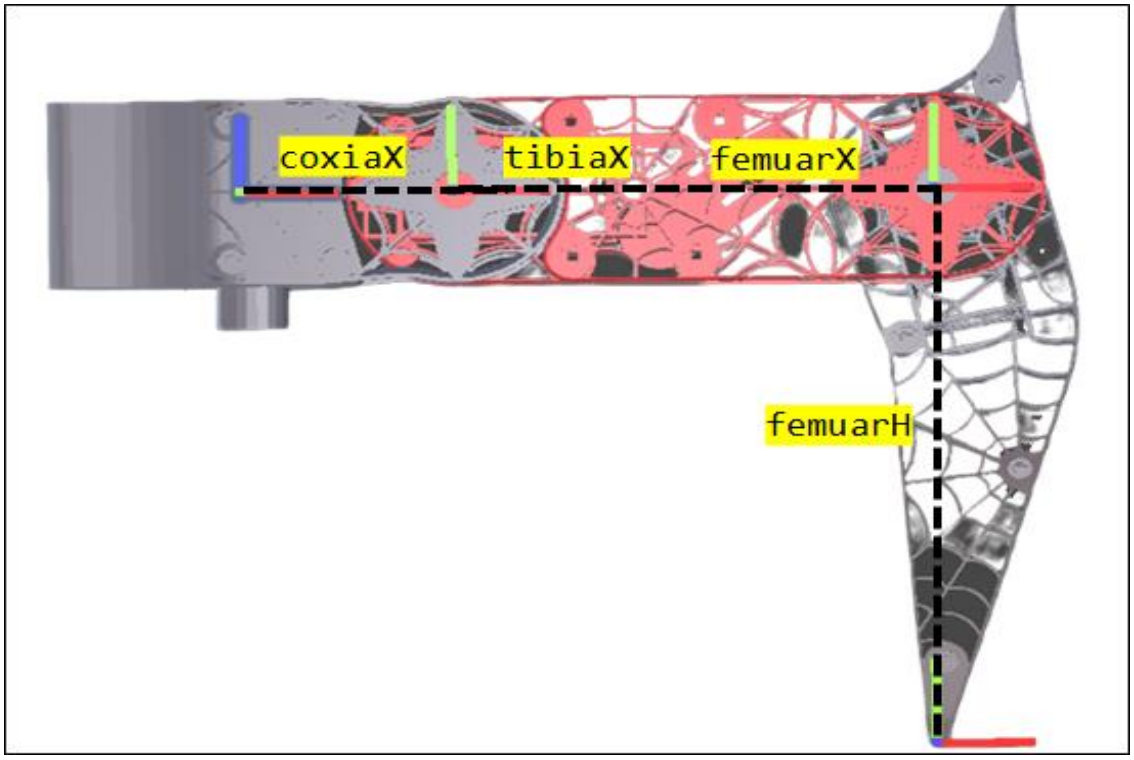
Şekil 3.2'te bir bacağın ayak tabanındaki koordinat sistemindeki *Y* ekseninden görünüşü verilmektedir. Bacaktaki her bir eksen sahip oldukları *Z* eksenleri etrafında dönebilmektedir. 2. eksen, 1.eksenin *X* ekseninde 90 derece döndürüldüğü bir koordinat sistemine sahiptir. 3. eksen, 2. eksenin sadece ötelenmiş halidir. İlk 3 eksen aynı *X* eksenindedir.

4. Eksen sanal bir eksen olup bir dönme yetisine sahip değildir. 1 eklem sayesinde bacağın dönüş hareketleri, 2. eksen ve 3. eksenler sayesinde bacağın yüksekliği değiştirilebilmektedir.




Şekil 3.3. Genel koordinat sistem yapısı

Şekil 3.3'de ileri kinematik denkleminde kullanılacak olan ana yapı gösterilmektedir. Şekilde gösterildiği gibi ana yapı Şekil 3.2'deki yapıdan farklıdır. Bunun nedeni sistemin parametrik olmasının hedeflenmesidir. İstenildiği zamanlarda, başka yapıdaki örümcek robotlara da uygulanabilir olmasıdır. Ana yapıdaki koordinat sistemlerinde 1, 2 ve 4. koordinat sistemleri gerçek eksenler olup sadece bu eksenler dönme kabiliyetine sahiptir. Şekilde gösterildiği gibi robotun mekanik yapısına göre $coxia_X$, $tibia_H$, $tibia_X$, $femuar_X$ve $femuar_H$ uzunluk değerleri parametre olarak girilerek örümcek robot önemli ölçüde tanımlanmıştır.




Şekil 3.4. Bacak üzerindeki uzunluk değerleri

Bu çalışmada kullanılan robot bacağında Şekil 3.4'de gözüktüğü gibi $tibia_H$ değeri sıfır ve $tibia_X$ ile $femuar_X$ değerleri yan yanadır. Bu eklemlere D-H yönteminin uygulanmasıyla elde edilen parametreler Tablo 3.1'de gösterilmektedir.

Tablo 3.1. D-H değişkenleri

| $i$ | $a_i$ | $a_{i-1}$ | $d_i$ | $\theta_i$ |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | $\theta_1$ |
| 2 | 90 | $coxia_X$ | 0 | $\theta_2$ |
| 3 | 90 | 0 | $tibia_H$ | $0$ |
| 4 | -90 | $tibia_X$ | 0 | $\theta_3$ |
| 5 | 90 | $femuar_X$ | 0 | 0 |
| 6 | 0 | 0 | $femuar_H$ | 0 |

D-H değişkenlerinin belirlenmesiyle her bir eklem için aşağıdaki genel eklem dönüşüm matrisi kullanılır (Bingül ve Küçük, 2015). Bu neden ile robotun ileri kinematik denklemlerinin elde edilmesinde Denklem (3.1) kullanılmıştır.

$$ {}^{i-1}_{i}T = \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & a_{i-1} \\ sin\theta_i cos \propto_{i-1} & cos\theta_i cos \propto_{i-1} & -sin \propto_{i-1} & -sin \propto_{i-1} d_i \\ sin\theta_i sin \propto_{i-1} & cos\theta_i sin \propto_{i-1} & cos \propto_{i-1} & cos \propto_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.1) $$

D-H yöntemi ve tablodaki parametre değerleri ile her bir eksendeki dönüşüm matrisi aşağıda verilen denklemlerdeki gibi bulunmuştur.

$$ {}^{0}_{1}T = \begin{bmatrix} cos\theta_1 & -sin\theta_1 & 0 & 0 \\ sin\theta_1 & cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2) $$

$$ {}^{1}_{2}T = \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 & coxia_X \\ 0 & 0 & 1 & 0 \\ -sin\theta_2 & -cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3) $$

$$ {}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & tibia_H \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4) $$

$$ {}^{3}_{4}T = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 & tibia_X \\ 0 & 0 & -1 & 0 \\ sin\theta_3 & cos\theta_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.5) $$

$$ {}^{4}_{5}T = \begin{bmatrix} 1 & 0 & 0 & femuar_X \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6} $$

$$ {}^{5}_{6}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & femuar_H \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7} $$

Denklem (3.2) - (3.7) ile dönüşüm matrislerinin çarpılmasıyla ana çerçeveden araç çerçeveye doğru ${}^{0}_{6}T$ ileri robot kinematiği Denklem (3.8) kullanılarak çıkartılmıştır.

$$ {}^{0}_{6}T = {}^{0}_{1}T\ {}^{1}_{2}T\ {}^{2}_{3}T\ {}^{3}_{4}T\ {}^{4}_{5}T\ {}^{5}_{6}T \tag{3.8} $$

Aşağıdaki $a$, $b$, $c$ ve $d$ kısaltmaları ile ayak tabanı pozisyonu (3.13), (3.14) ve (3.15) elde edilmiştir.

$$ a = femuar_X cos\theta_{2+3} \tag{3.9} $$

$$ b = femuar_H sin\theta_{2+3} \tag{3.10} $$

$$ c = tibia_X cos\theta_2 \tag{3.11} $$

$$ d = tibia_H sin\theta_2 \tag{3.12} $$

$$ p_x = cos\theta_1(coxia_X + a - b + c - d) \tag{3.13} $$

$$ p_y = sin\theta_1(coxia_X + a - b + c - d) \tag{3.14} $$

$$ p_z = -a - b - tibia_H cos\theta_2 - tibia_X - sin\theta_2 \tag{3.15} $$

### 3.2. Ters Kinematik

Ters kinematik, kartezyen sisteminde bacağın istenilen konuma getirilmesi için gerekli olan açıların koordinatlar cinsinden belirlenmesi işlemidir (Bingül ve Küçük, 2015). Doğrudan kinematik tarafından ortaya konan problemden farklı olarak, denklemleri elde etme prosedürü robotun konfigürasyonuna büyük ölçüde bağlıdır. Bu durum, söz konusu işlemi karmaşık bir prosedür haline getirir; çünkü bu denklemleri sistematik olarak elde etmek çok zordur. Bu durumda ters kinematik, bacağın şekline dayalı geometrik değerlendirmelerle elde edilir (Bingül ve Küçük, 2015). Bu nedenle sistemin ters kinematik denklemlerini çıkartmak için ileri kinematik denklemleri ile Denklik (3.16) elde edilmiştir.

$$ {}^{0}_{1}T^{-1}\ {}^{0}_{6}T\ =\ {}^{1}_{2}T\ {}^{2}_{3}T\ {}^{3}_{4}T\ {}^{4}_{5}T\ {}^{5}_{6}T \tag{3.16} $$

Denklem (3.16) çarpımın sol tarafına $S_L$, çarpımın sağ tarafına ise $S_R$ olarak adlandırılarak $S_L$ Denklem (3.17) ve $S_R$ Denklem (3.18) bulunmuştur.

$$ \begin{bmatrix} \dots & \dots & \dots & p_x cos\theta_1 cos\theta_2 - p_z sin\theta_2 - cos\theta_2 coxia_X + p_y cos\theta_2 sin\theta_1 \\ \dots & \dots & \dots & sin\theta_2 coxia_X - p_z cos\theta_2 - p_x cos\theta_1 sin\theta_2 - p_y sin\theta_1 sin\theta_2 \\ \dots & \dots & \dots & p_y cos\theta_1 - p_x sin\theta_1 \\ \dots & \dots & \dots & 1 \end{bmatrix} \tag{3.17} $$

$$ \begin{bmatrix} \dots & \dots & \dots & tibia_X + femuar_X cos\theta_3 - femuar_H sin\theta_3 \\ \dots & \dots & \dots & tibia_H + femuar_H cos\theta_3 - femuar_X sin\theta_3 \\ \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & 1 \end{bmatrix} \tag{3.18} $$

Denklem (3.17) ve (3.18) karşılıklı eşitlenerek Denklem (3.21), (3.24) ve (3.25) bulunmuştur.

$$ S_{L1} = p_x cos\theta_1 cos\theta_2 - p_z sin\theta_2 - cos\theta_2 coxia_X + p_y cos\theta_2 sin\theta_1 \tag{3.19} $$

$$ S_{R1} = tibia_X + femuar_X cos\theta_3 - femuar_H sin\theta_3 \tag{3.20} $$

$$S_{L1} = S_{R1} \tag{3.21}$$

$$S_{L2} = sin\theta_2 coxia_X - p_z cos\theta_2 - p_x cos\theta_1 sin\theta_2 - p_y sin\theta_1 sin\theta_2 \tag{3.22}$$

$$S_{R2} = tibia_H + femuar_H cos\theta_3 - femuar_X sin\theta_3 \tag{3.23}$$

$$S_{L2} = S_{R2} \tag{3.24}$$

$$0 = p_y cos\theta_1 - p_x sin\theta_1 \tag{3.25}$$

Denklem (3.25) kullanılarak $\theta_1$ açı değeri; Denklem (3.26)'deki gibi bulunmuştur.

$$\theta_1 = tan\frac{p_y}{p_x} \tag{3.26}$$

Denklem (3.19) ve (3.20); $a$ ile sadeleştirilerek Denklem (3.28) ve (3.29) elde edilmiştir.

$$a = p_x cos\theta_1 - coxia_X + p_y sin\theta_1 \tag{3.27}$$

$$S_{L1} = cos\theta_2 a - p_z sin\theta_2 \tag{3.28}$$

$$S_{L2} = -sin\theta_2 a - p_z cos\theta_2 \tag{3.29}$$

$S_{L1} = S_{R1}$, $S_{L2} = S_{R2}$ eşitliklerinin kareleri alınıp, toplanarak aşağıdaki $a$, $b$, $c$ ve $d$ kısaltmaları ile Denklem (3.32) elde edilmiştir.

$$b = (2tibia_H a + 2p_z tibia_X) \tag{3.30}$$

$$c = (-2tibia_X a + 2p_z tibia_H) \tag{3.31}$$

$$d = sin\theta_2 b + cos\theta_2 c \tag{3.32}$$

Denklem (3.32) kullanılarak $\theta_2$ açı değeri; Denklem (3.33)'deki gibi bulunmuştur.

$$\theta_2 = atan2(b, c) + atan2((b^2 + c^2 - d^2)^{1/2}, d) \quad (3.33)$$

$S_{L2} = S_{R2}$ eşitliği aşağıdaki $a$, $b$ ve $c$ kısaltmaları ile çözülerek $\theta_3$; Denklem (3.37) deki gibi bulunmuştur.

$$a = femuar_X \quad (3.34)$$

$$b = femuar_H \quad (3.35)$$

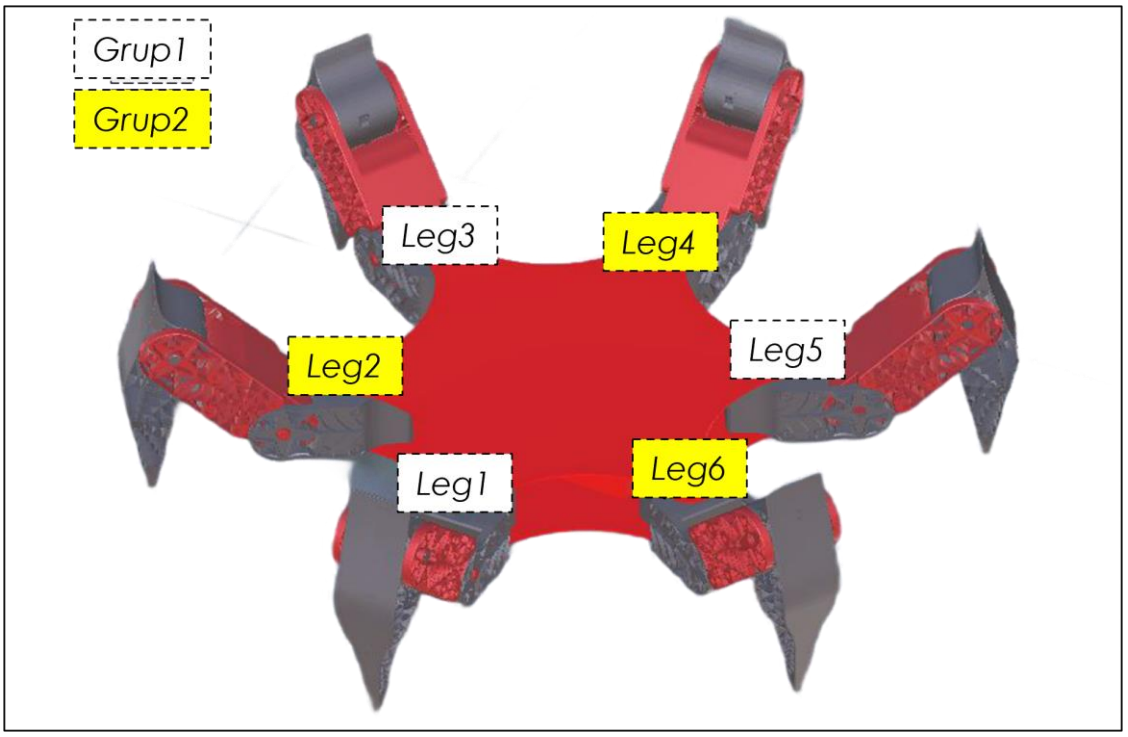
$$c = sin\theta_2 coxia_X - p_z cos\theta_2 - p_x cos\theta_1 sin\theta_2 - p_y sin\theta_1 sin\theta_2 - tibia_H \quad (3.36)$$

$$\theta_3 = atan2(a, b) + atan2((a^2 + b^2 - c^2)^{1/2}, c) \quad (3.37)$$

Bu bölümde farklı tasarımlara sahip örümcek robotlar için geçerli olabilecek bir kinematik model seçilerek çözümler gerçekleştirilmiştir. Sonuç olarak bir örümcek robotun bacağının ileri kinematik denklemleri bulunmuştur. Bu ileri kinematik denklemlerin ters kinematik çözümleri yapılarak eklemlerin açılarının hesaplanabilmesi için gerekli denklemler elde edilmiştir.

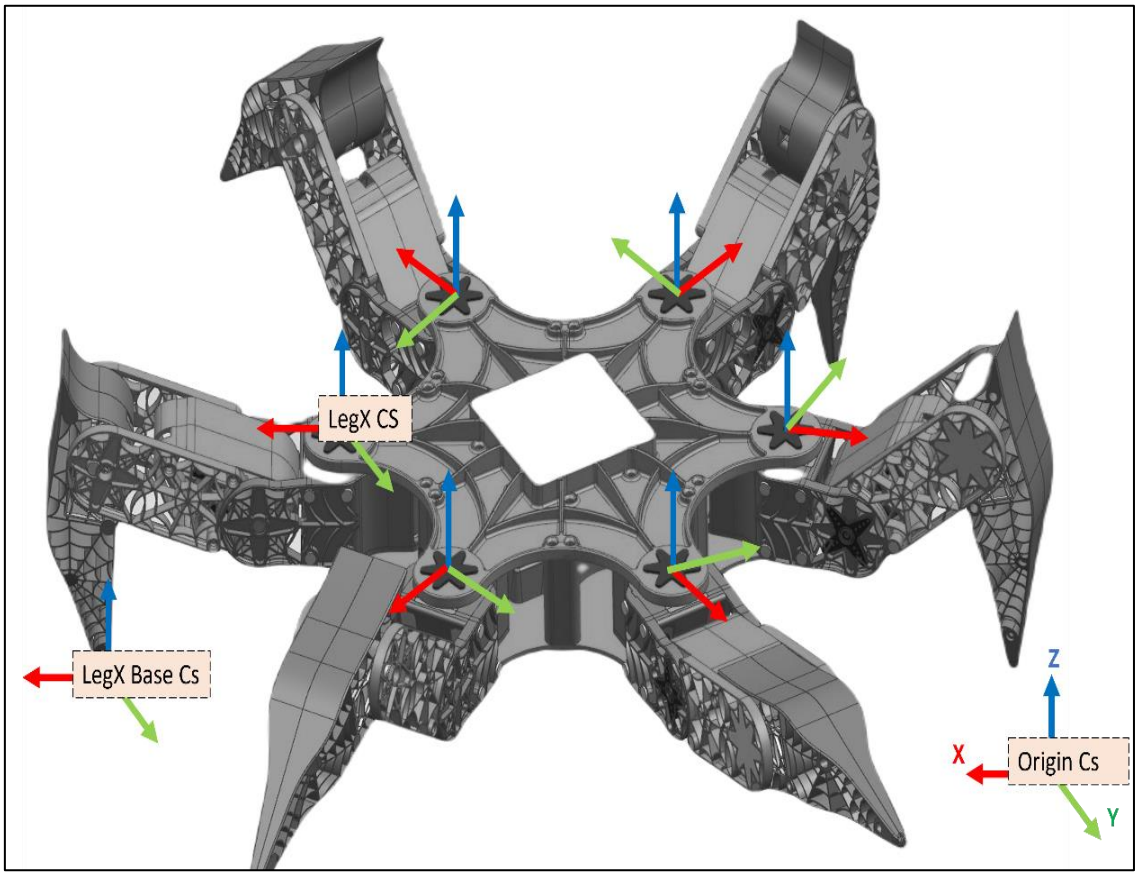
# 4. ALGORİTMALAR

Bu bölümde tez kapsamında geliştirilen örümcek robotun yürüme, dönme, dengede durma, yön stabilizasyonu gibi hareketleri için geliştirilen algoritmalar açıklanmaktadır. Çalışmada gövde üzerinde bulunan koordinat sistemleri hareket ettirilerek, her bir bacağın gövdeyle olan ilişkisi kurulmuş ve robotun gövde hareketinin doğal bir şekilde ortaya çıkması sağlanmıştır. Ayrıca bacağın gövdeyle ilişkisinin kurulması sayesinde, gövde üzerinde bulunan koordinat sisteminin rotasyonu değiştirildiğinde bu rotasyona uyumlu bir şekilde yürüme hareketi yapılması sağlanmıştır. Ayrıca bu ilişki sayesinde, robota; eğimli yüzeylerde dengeyi koruma ve yön stabilizasyonu gibi yetenekler kazandırılmıştır. Örümcek robotun çeşitli hareketleri düzgün, dengeli ve kontrollü bir şekilde yapabilmesi için ayakların birbiriyle senkron çalışması gerekmektedir. Her bir gruptaki ayaklar birbiri ile senkron çalışır iken, hareket eden grubun dışındaki ayaklar, harekete başlamadan önceki bulundukları konumdaki ayak tabanı pozisyonunu korumaktadır. Genellikle yapılan çalışmalarda robot bacakları iki gruba ayrılmış ve bu şekilde hareket ettirilmiştir (Sun ve diğ., 2017). Bu sayede diğer üç ayak havada iken hareket etmeyen üç bacak yerde kalarak örümcek robotun devrilmeden hareket ettirilmesi sağlanır.




Şekil 4.1. Robot üzerindeki bacak grupları

Bunun için örümcek robotun sahip olduğu altı ayak; Şekil 4.1'de gösterildiği gibi iki gruba ayrılarak sırasıyla hareket ettirilmiştir. Robotta bulunan ayak grupları yukarıdaki şekilde gösterildiği gibidir. Birbiri ardına sıralanan ayaklar sırası ile sarı ve beyaz gruba alınarak yapılan hareketin dengeli ve homojen olması sağlanmıştır.




Şekil 4.2. Algoritma üzerindeki koordinat sistemleri

Kinematik denklem hesapları için robotun üzerindeki çeşitli noktalara ve robotun hareket ettiği uzaya hayali koordinat sistemleri konulmuştur (Şekil 4.2). Her bir bacağın gövdeye bağlandığı noktaya LegX koordinat sistemi yerleştirilmiştir. Ayrıca robot üzerindeki koordinat sistemlerinin referans alacağı robotun dışında bulunan hayali bir orijin koordinat sistemi oluşturulmuştur. Robot kontrolünde asıl amaç, ayak taban pozisyonunu istenilen yörüngede hareket ettirmektir. Ayak tabanının istenilen yörüngede hareket ettirilebilmesi için gerekli olan $\theta_1$, $\theta_2$ ve $\theta_3$ eklem açılarının hesaplanması gerekir. Bu açı değerleri ayak taban pozisyonun LegX'e göre ters kinematik çözümü ile bulunur. Ayrıca gövde üzerinde Body isminde robot gövdesinin pozisyon bilgilerinin bulunduğu koordinat sistemi bulunmaktadır.

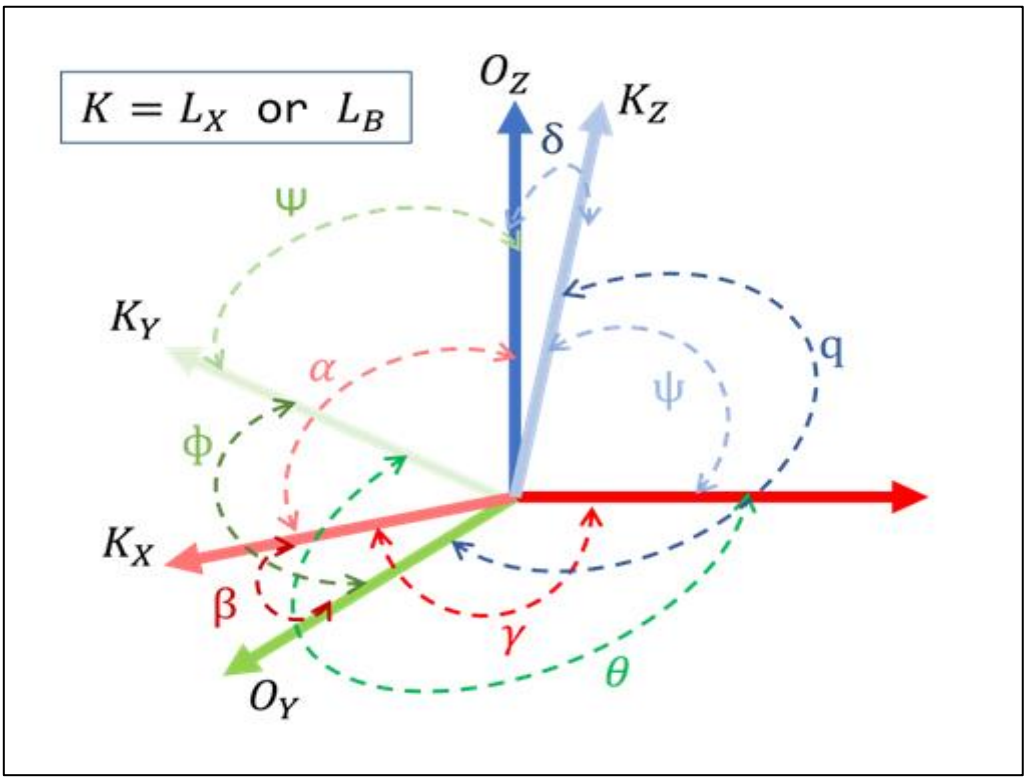
Ayak tabanının orijin koordinat sistemine göre pozisyonu Denklem (4.1) ile bulunur.

$$ {}^{O}_{BASE}P = P_{(X_{OB},Y_{OB},Z_{OB})} \tag{4.1} $$

Ayağın gövdeye bağlandığı noktanın (LegX) orijin koordinat sistemine göre pozisyonu Denklem (4.2) ile hesaplanır.

$$ {}^{O}_{LX}P = P_{(X_{OL},Y_{OL},Z_{OL})} \tag{4.2} $$

Şekil 4.3'deki koordinat sistemlerinin orijin koordinat sistemine göre dönme matrisi Denklem (4.3) bulunur.




Şekil 4.3. Merkezleri çakışık yönelimleri farklı iki koordinat sistemi

$$ {}^{O}_{K}R = \begin{vmatrix} |X_K||X_O|\cos\gamma & |Y_K||X_O|\cos\theta & |Z_K||X_O|\cos\psi \\ |X_K||Y_O|\cos\beta & |Y_K||Y_O|\cos\phi & |Z_K||Y_O|\cos q \\ |X_K||Z_O|\cos\alpha & |Y_K||Z_O|\cos\Psi & |Z_K||Z_O|\cos\delta \end{vmatrix} \tag{4.3} $$

Ayak tabanının orijin koordinat sistemine dönme matrisi ${}^{O}_{BASE}R$ ile ayağın gövdeye bağlandığı noktanın orijin koordinat sistemine göre dönme matrisi ise ${}^{O}_{LX}R$ ile isimlendirilmiştir.

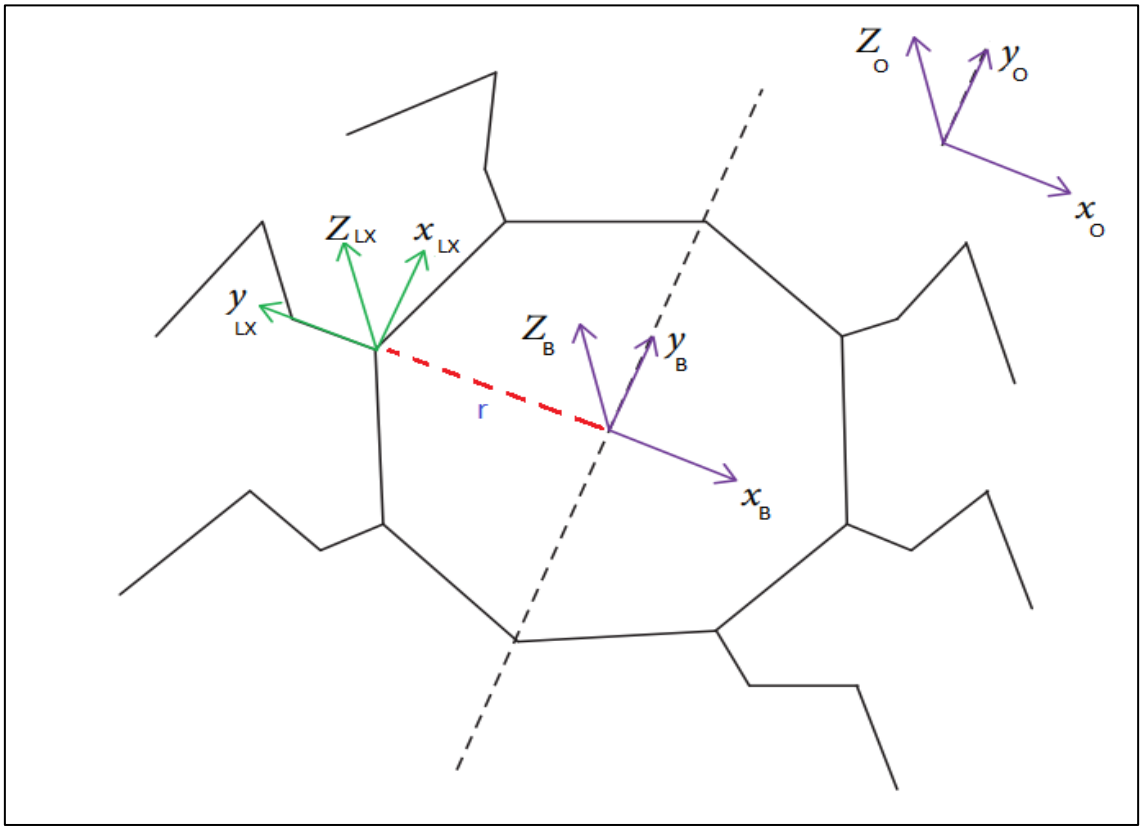
Dönme matrisi, {B} koordinat sisteminin {A} koordinat sistemine göre yönelimini açıklar. Bu dönme matrisi bilindiği takdirde B koordinat sistemine göre tanımlanmış bir noktayı A koordinat sistemine göre tanımlamak mümkündür (Bingül ve Küçük, 2015). Bu sebeple ayak tabanının ayağın gövdeye bağlandığı koordinat sistemine göre pozisyonu Denklem (4.4) ile hesaplanmıştır.

$$ {}^{LX}_{BASE}P = {}^{O}_{LX}R^{-1} \, {}^{O}_{BASE}P \tag{4.4} $$

Ayak tabanının orijin koordinat sistemine göre pozisyonu ise Denklem (4.5) ile hesaplanmıştır.

$$ {}^{O}_{B}P = {}^{O}_{LX}R \, {}^{LX}_{BASE}P \tag{4.5} $$

Her bir ${}^{O}_{LX}P$ değeri ise robotun gövdesin de bulunan koordinat sistemi ve orijin koordinat sistemi arasındaki dönüşümler ile hesaplanır.




Şekil 4.4. LX, Gövde ve orijin koordinat sistemi

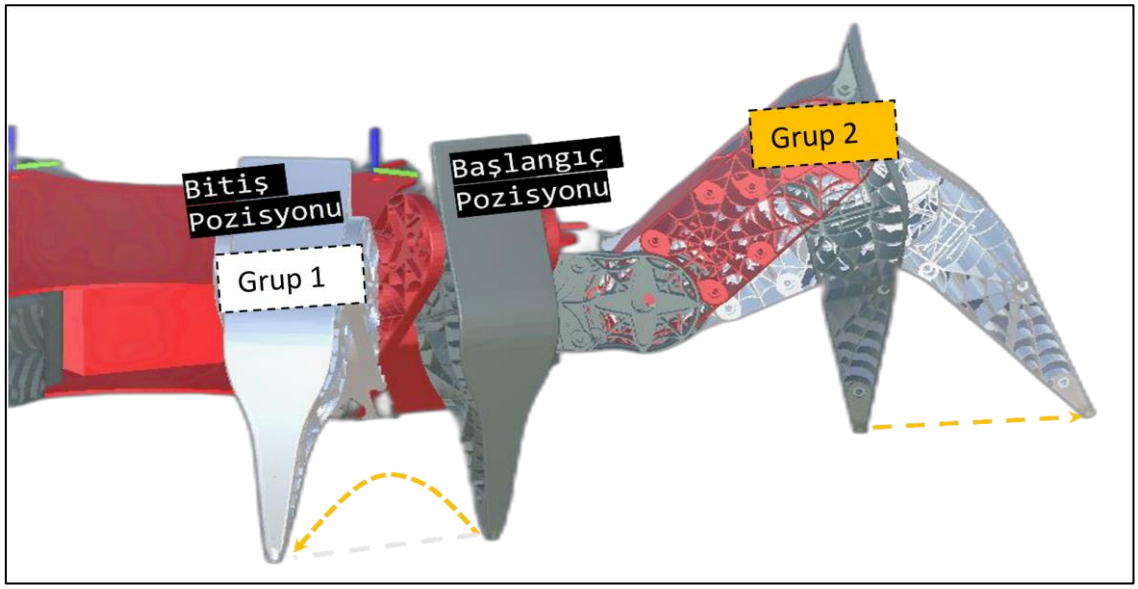
Şekil 4.4'da gösterildiği gibi LX koordinat sistemi orijin koordinat sistemine (4.6) denklemi ile hesaplanır. Yukarıda şekilde gösterilen r değeri ${}^{B}_{LX}P$ ile ifade edilir. Bu değer

LX koordinat sisteminin gövdeye göre olan konumudur. Literatürde vakumlu duvara tırmanan bir örümcek robotun kinematik hesaplarında ve dönüşümünde de buna benzer bir yaklaşımda bulunulmuştur (Heming ve diğ., 2019). LX koordinat sisteminin robotun gövde ile bağlantısının kurulması; robot gövdesinin farklı açılarda bulunduğu durumlarda bacakların buna ayak uydurmasına sebep olur.

$$ {}^{O}_{LX}P = {}^{O}_{B}R \; {}^{B}_{LX}P + {}^{O}_{B}P \tag{4.6} $$

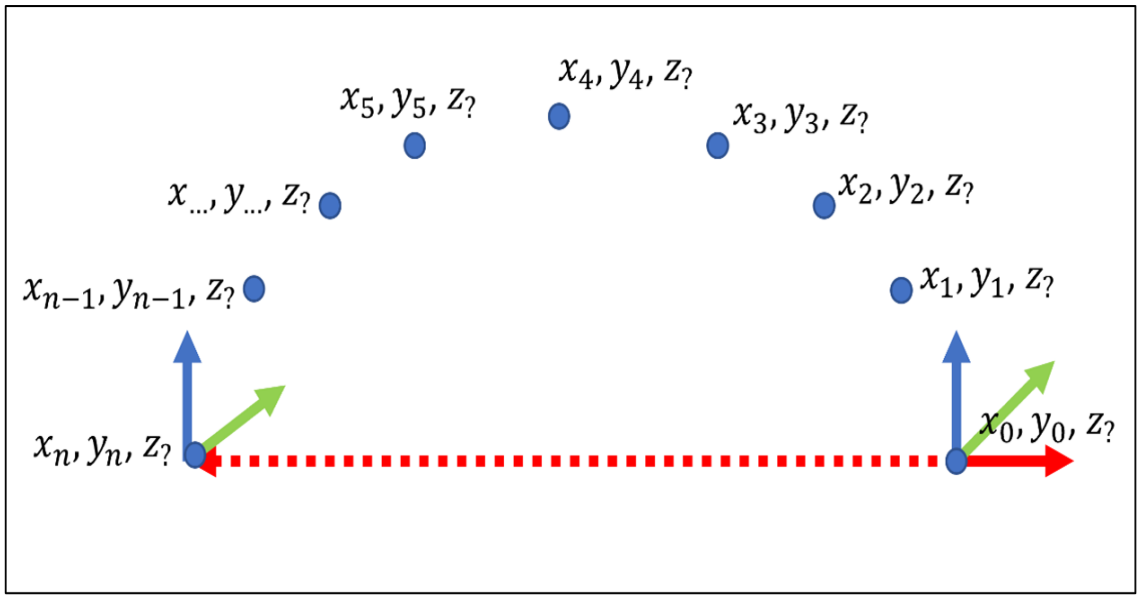
### 4.1. Yürüyüş Algoritması

Önceki bölümlerde bahsedildiği gibi yürüyüş algoritmasında robot bacakları iki farklı gruba ayrılarak hareket ettirilmektedir. Hareketler nizami ve sıralı bir şekilde gerçekleştirilmektedir.




Şekil 4.5. Farklı gruplardaki bacakların hareket şekli

Şekil 4.5'de robottaki iki farklı grupta bulunan bacakların robotun yürüme esnasında yaptığı hareketler gösterilmektedir. Gri renkte olan bacaklar, robotun harekete başlamadan önceki pozisyonunu göstermektedir. Beyaz renkte olanlar ise bacakların bir adımlık yürüme hareketinden sonraki pozisyonunu göstermektedir. Şekil 4.5'te gösterildiği gibi bir grup bacak, robot hareketi esnasında robotun devrilmemesi için yere basılı kalırken bir grup bacakta robotun mesafe kat etmesi için hareket eder. Aynı zamanda yere basılı kalan bacaklar, robotun kat etmesi istenen mesafe kadar gövdeyi ileri taşır.




Şekil 4.6. Grup 1'e ait bacağın yörüngesi

Şekil 4.6'de Grup 1'e ait bacağın yörüngesi gösterilmektedir. Bu yörünge LegX koordinat sistemine ait olan bir bacağın ayak tabanının izlediği yörüngedir. Burada bacak tabanı robotun kat etmesi istenen mesafe kadar LegX koordinat sisteminin *X* ekseninde hareket ettirilir. Örnek olarak robot *K* mesafesi kadar hareket ettirilmek istensin. Hareketin sürdürüldüğü her bir *t* örneğinde ayak tabanı *X* ekseninde $x_1$ mesafesi kadar, *Y* ekseninde ise $\pm\, y_1$ mesafesi kadar hareket ettirilir. Eğer gidilecek mesafenin yarısı tamamlanmış ise *Y* ekseninde $-\, y_1$ mesafesinde ayak tabanı aşağı doğru hareket ettirilir.

Robot, orijin koordinat sisteminin *X* ekseninde *K* mesafesi kadar hareket ettirilmek istendiğinde aşağıdaki adımlar sırasıyla çalıştırılır. Aşağıdaki adımlar çalıştırılmadan önce her bir bacağın ayak taban pozisyonu orijin koordinat sistemine göre bilinmektedir.

1. Gövde üzerinde bulunan Body koordinat sisteminin pozisyonu orijin koordinat sistemine göre *X* ekseninde $x_1$ mesafesi kadar Denklem (4.7) ile hareket ettirilir.

$$ {}_{BODY}^{O}P \;=\; {}_{BODY}^{O}P(x + x_1, y\,, z) \tag{4.7}$$

2. Grup 1'de bulunan her bir bacağın ayak tabanındaki koordinat sistemi, orijin koordinat sistemine göre *X* ekseninde $x_1$ mesafesi kadar Denklem (4.8) ile hareket ettirilir.

$$ {}_{BASE}^{0}P \;=\; {}_{BASE}^{O}P(x + x_1, y\,, z) \tag{4.8}$$

3. Grup 1'de bulunan her bir bacağın ayak tabanındaki koordinat sistemi, orijin koordinat sistemine göre *Y* ekseninde Denklem (4.9) ile hareket ettirilir.

3.1 Eğer *X* ekseninde hedeflenen yolun yarısı tamamlanmamış ise, Ayak tabanı *Y* ekseninde + $y_1$ mesafesi kadar hareket ettirilir.

3.2 Eğer *X* ekseninde hedeflenen yolun yarısı tamamlanmış ise, Ayak tabanı *Y* ekseninde $-y_1$ mesafesi kadar hareket ettirilir.

$$ {}^{0}_{BASE}P \; = \; {}^{O}_{BASE}P(x, y \; \pm y_1 \; , z) \tag{4.9} $$

4. İlk üç adım işlendikten sonra sistemin ters kinematiği çözülür. Bunun için aşağıdaki adımlar sırasıyla işlenir. Bu adımlar her bir gruptaki bacaklar için yapılır.
5. Ayağın gövdeye bağlandığı noktanın orijin koordinat sistemine göre ${}^{0}_{LX}R$ dönme matrisi Denklem (4.3) ile hesaplanır.
6. Gövdedeki koordinat sisteminin orijin koordinat sistemine göre ${}^{0}_{BODY}R$ dönme matrisi Denklem (4.3) ile hesaplanır.
7. ${}^{0}_{BODY}P$ (Gövdedeki koordinat isteminin orijin koordinat sistemine göre pozisyonu) ve ${}^{BODY}_{LX}P$ (Bir bacağın gövdeye bağlandığı koordinat sisteminin, gövdedeki koordinat sistemine göre pozisyonu) kullanılarak bacağın gövdeye bağlandığı koordinat sisteminin pozisyon değeri orijin koordinat sistemine göre Denklem (4.10) ile hesaplanır.

$$ {}^{O}_{LX}P \; = \; {}^{BODY}_{LX}P \, {}^{0}_{BODY}R + {}^{O}_{BODY}P \tag{4.10} $$

8. Denklem (4.10) kullanılarak bir bacağın gövdeye bağlandığı koordinat sisteminin orijin koordinat sistemine göre dönüşüm matrisi aşağıdaki gibi Denklem (4.11) ile hesaplanır.

$$ {}^{O}_{LX}T = \begin{bmatrix} {}^{0}_{LX}R & {}^{O}_{LX}P \\ 0\;0\;0 & 1 \end{bmatrix} \tag{4.11} $$

9. Denklem (4.11)'un tersi alınır ise Denklem (4.12)'deki dönüşüm matrisi elde edilir. Bu matris sayesinde orijin koordinat sistemine göre bilenen herhangi bir pozisyon değeri LegX koordinat sistemine göre hesaplanır.

$$ {}^{LX}_{O}T \; = \; {}^{O}_{LX}T^{-1} \tag{4.12} $$

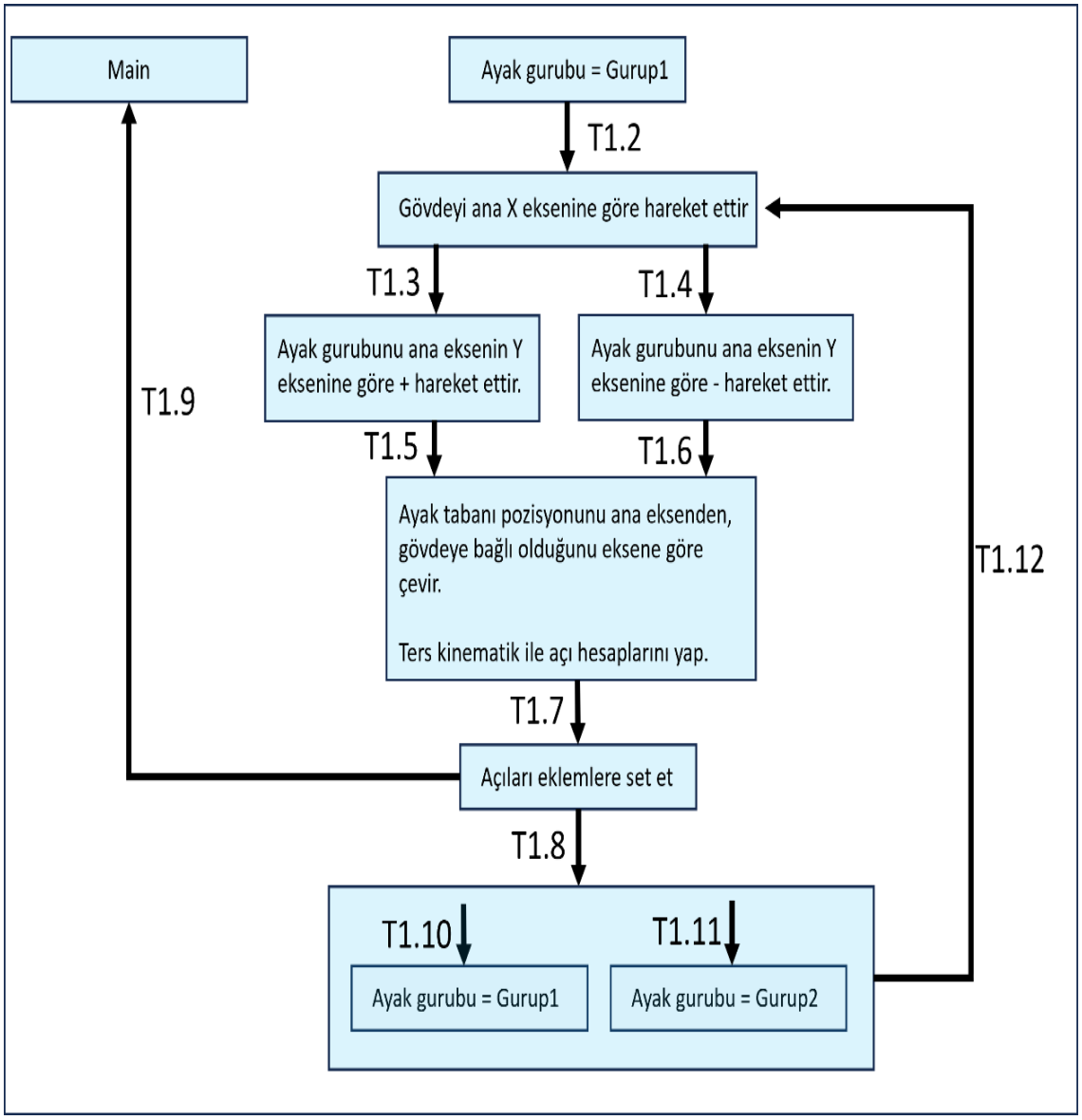
10. Denklem (4.12) kullanılarak orijin koordinat sistemine göre pozisyon bilgisi bilinen her bir ayak tabanı pozisyonu LegX koordinat sistemlerine göre hesaplanır.

$$ {}^{LX}_{BASE}P = {}^{LX}_{O}T \; {}^{O}_{BASE}P \tag{4.13} $$

11. Denklem (4.13)'de bulunan pozisyon bilgileri için ters kinematik denklemleri çözülerek $\theta_1$, $\theta_2$ ve $\theta_3$ açıları hesaplanır.
12. Eğer gidilecek mesafe kat edilmiş ise ayak tabanı sabit kalacak gruplar değiştirilerek birince adıma geçilir. Eğer mesafe kat edilmemiş ise ilk dört adım sırasıyla tekrarlanır.

Tablo 4.1. Yürüyüş algoritması geçiş tablosu

| | |
|---|---|
| T1.1 | durum = yürüme |
| T1.2 | koşulsuz |
| T1.3 | 0 ≤ adımX ve adımX ≤ k/2 |
| T1.4 | k/2 ≤ adımX ve adımX ≤ k |
| T1.5 | koşulsuz |
| T1.6 | koşulsuz |
| T1.7 | koşulsuz |
| T1.8 | durum ≠ duruş veya adımX ≠ k |
| T1.9 | durum = duruş ve adımX = k |
| T1.10 | Ayak grubu = Grup 2 |
| T1.11 | Ayak grubu = Grup 1 |
| T1.12 | koşulsuz |

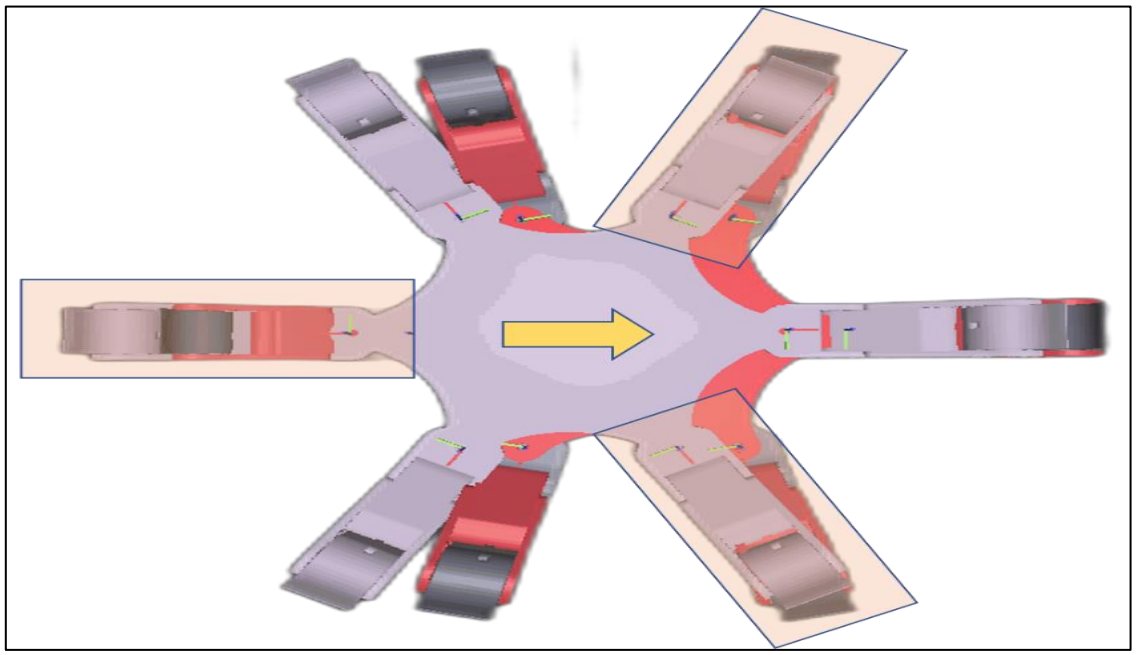



Şekil 4.7. Yürüyüş algoritması durum makinesi

Şekil 4.7'da yürüyüş algoritmasının durum makinesi gösterilmektedir. Tablo 4.1'de ise bu durum makinesindeki geçiş şartları verilmiştir. İlk adımda şekilde gösterildiği gibi hareket edecek olan ayak grubu yapılandırılır. Sonrasında gövde hareket ettirilir. Bir sonraki adımda ilgili grup bacağı yukarı veya aşağı yönde hareket ettirilir. Tabloda da gösterildiği gibi dur komutu gelmiş ise ilgili hareketin bitmesi beklenir. Herhangi bir harekete başlamadan önce robotun başlangıç konumunda olması önemlidir. Algoritmalar bir hareketi tamamlayıp bitirmek ve belirli bir başlangıç konumundan çalıştırılmak üzere geliştirilmiştir. Tablo 4.1'de şartsız geçişler ile gösterilmektedir. Bu geçişler herhangi bir işlem gerçekleştirildikten sonra diğer işleme geçmek için oluşturulmuştur.

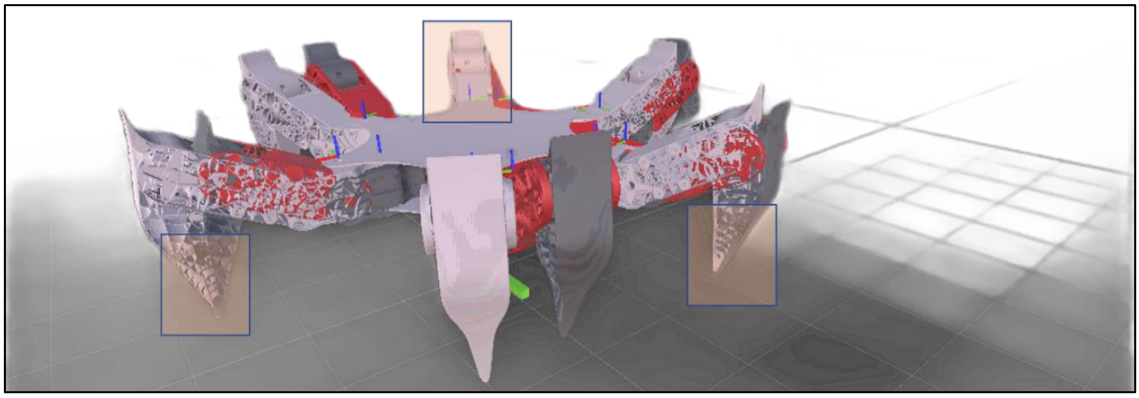
## 4.2. Gövde Hareketi Algoritması

Gövde hareket algoritması robotun kendi etrafında dönebilmesi için geliştirilmiştir. Geliştirilen bu algoritma ile robot yürüyüş yönünü değiştirmektedir.



Şekil 4.8. Robot üstten görünüşü

Şekil 4.8'de gösterildiği üzere sarı gruba ait bacakların ayak tabanları bulunduğu konumları korumaktadır. Gövde hareket ettirildiğinde bu ayaklar bulundukları ayak tabanı konumlarını korumak için yukarıdaki şekilde gösterildiği gibi bir hareket yapar. Şekilde gösterildiği gibi her bir bacağın gövdeye bağlandığı eksende bulunan koordinat sistemi gövdeyle birlikte yer değiştirirken, bastığı yeri koruyan veya korumayan grup bacaklar sayesinde hareket oluşur.



Şekil 4.9. Robot yandan görüşünü

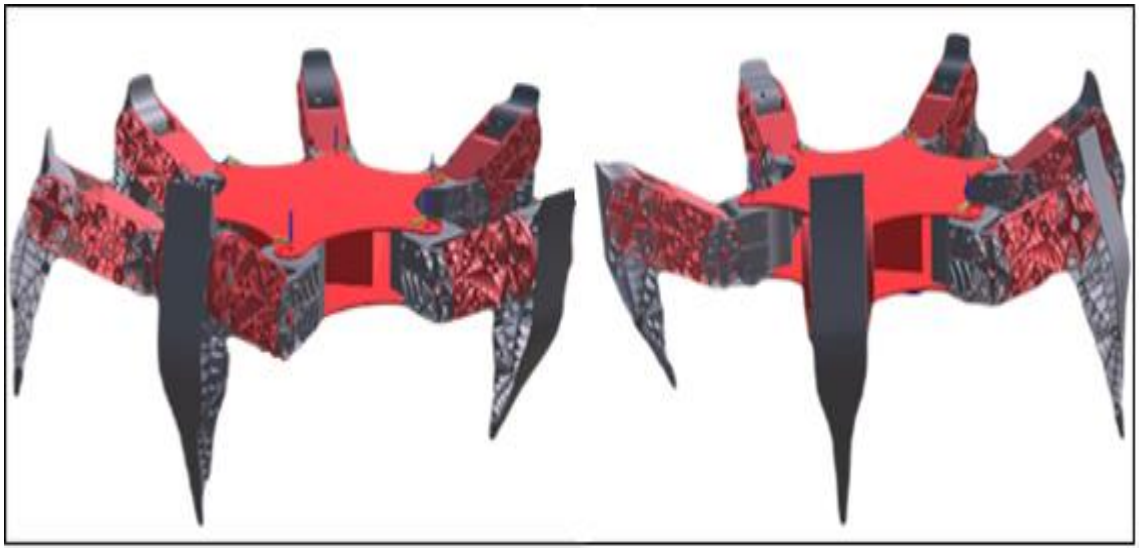
Şekil 4.9'de gösterildiği gibi bir grup bacak bastığı yeri korumakta, diğer bacaklar ise hareket etmektedir. Böylece gövde belirli bir mesafe ileri girmekte ve robota yürüyüş yaptırılmaktadır.

Sadece robotun gövdesini bir *x* ekseninde, *y* ekseninde veya *z* ekseninde hareket ettirmek nispeten daha kolaydır. Bu durumda gövde hareket ettirilmeden önce her bir ayağın bastığı yerin pozisyonu ${}^{O}_{BASE}P$ orijin koordinat sistemine göre bir hafızaya alınır. Daha sonra her bir bacağın gövdeye bağlandığı kısımdaki koordinat sistemleri orijin koordinat sistemine göre *k* mesafesi kadar hareket ettirilir. Koordinat sistemlerinin almış oldukları yeni pozisyon değerleri de ${}^{O}_{LX}P$ ile isimlendirilir.

Denklem (4.3) kullanılarak orijin koordinat sisteminin LegX koordinat sistemine göre dönüşüm matrisi ${}^{LX}_{O}R$ hesaplanır. Orijin koordinat sisteminin bulunduğu konum ${}^{O}_{LXORG}P$, LegX koordinat sistemine göre hesaplanır. Bulunan konum ve dönme matrisi ile ayak tabanının bulunduğu pozisyon yeni LegX koordinat sistemine göre Denklem (4.14) ile hesaplanır.

$$ {}^{LX}_{BASE}P = {}^{LX}_{O}R\,{}^{O}_{BASE}P + {}^{O}_{LXORG}P \tag{4.14} $$

Denklem (4.14) ile hesaplanan pozisyon bilgisi ters kinematik bölümünde bulunan denklemler ile çözülerek $\theta_1$ , $\theta_2$ ve $\theta_3$ değerleri hesaplanır.



Şekil 4.10. *Y* ekseninde negatif hareket

Hesaplan $\theta_1, \theta_2$ ve $\theta_3$ değerleri istenilen bir eksen hareket ettirildiğinde robot Şekil 4.10 da gösterilen hareketleri gerçekleştirir. Şekil 4.10'da gösterildiği gibi sadece robotun gövdesi hareket ettirilmek istendiğinde tüm ayaklar yere basar ve bir itme hareketi yapar. Bu itme hareketi koordinat sistemlerinin kaymasından dolayı yapılan ters kinematik çözümleriyle bulunan açı değerlerinin bacaklardaki eksenlere verilmesi ile ortaya çıkar.

### 4.3. Dönme Hareketi Algoritması

Robotun bulunduğu konumda dönüş yapabilmesi için bu çalışmada bir algoritma geliştirilmiştir. Bu algoritmada gövde de bulunan her bir koordinat sistemi istenilen açı değeri kadar *Z* ekseninde döndürülür. Yürüyüş algoritmasında olduğu gibi bir grup bacak bastığı yeri korurken bir grup bacakta dönüş hareketi esnasında bir dönüş adımı atar. Bu durum yürüyüş algoritmasında olduğu gibi robotun dengede tutulabilmesi için yapılır.

Örneğin robot bulunduğu konumda *K* derecesi kadar döndürülmek istenirse aşağıdaki adımlar çalıştırılır.

1. Gövde üzerinde bulunan Body koordinat sisteminin pozisyonu orijin koordinat sistemine göre Z ekseninde $k_1$ derecesi kadar Denklem (4.15) ile döndürülür.

$$ {}^{0}_{BODY}R \;=\; {}^{O}_{BODY}R(x, y, z \,+\, k_1) \tag{4.15}$$

2. Grup 1'de bulunan her bir bacağın ayak tabanı koordinat sistemi, orijin koordinat sistemine göre Z ekseninde $k_1$ derecesi kadar Denklem (4.16) ile döndürülür.

$$ {}^{0}_{BASE}R \;=\; {}^{O}_{BASE}R(x, y\,, z + k_1) \tag{4.16}$$

3. Grup 1'de bulunan her bir bacağın ayak tabanı koordinat sistemi, orijin koordinat sistemine göre Z ekseninde $k_1$ derecesi kadar Denklem (4.16) döndürülür.

$$ {}^{0}_{BASE}P \;=\; {}^{O}_{BASE}P(x, y \,\pm y_1, z) \tag{4.17}$$

4. İlk üç adım işlendikten sonra sistemin ters kinematiği çözülür. Bunun için aşağıdaki adımlar sırasıyla işlenir. Bu adımlar her bir gruptaki bacaklar için yapılır.
5. Ayağın gövdeye bağlandığı noktanın orijin koordinat sistemine göre ${}^{0}_{LX}R$ dönme matrisi Denklem (4.3) ile hesaplanır.

6. Gövdedeki koordinat sisteminin orijin koordinat sistemine göre ${}^{0}_{BODY}R$ dönme matrisi Denklem (4.3) ile hesaplanır.
7. ${}^{0}_{BODY}P$ (Gövdedeki koordinat isteminin orijin koordinat sistemine göre pozisyonu) ve ${}^{BODY}_{LX}P$ (Bir bacağın gövdeye bağlandığı koordinat sisteminin, gövdedeki koordinat sistemine göre pozisyonu) kullanılarak bacağın gövdeye bağlandığı koordinat sisteminin pozisyon değeri, orijin koordinat sistemine göre pozisyonu Denklem (4.18) ile hesaplanır.

$$ {}^{O}_{LX}P = {}^{BODY}_{LX}P\,{}^{O}_{BODY}R + {}^{O}_{BODY}P \tag{4.18} $$

8. Denklem (4.18) kullanılarak bir bacağın gövdeye bağlandığı koordinat sisteminin orijin koordinat sistemine göre dönüşüm matrisi aşağıdaki gibi Denklem (4.19) ile hesaplanır.

$$ {}^{O}_{LX}T = \begin{bmatrix} {}^{0}_{LX}R & {}^{O}_{LX}P \\ 0\ 0\ 0 & 1 \end{bmatrix} \tag{4.19} $$

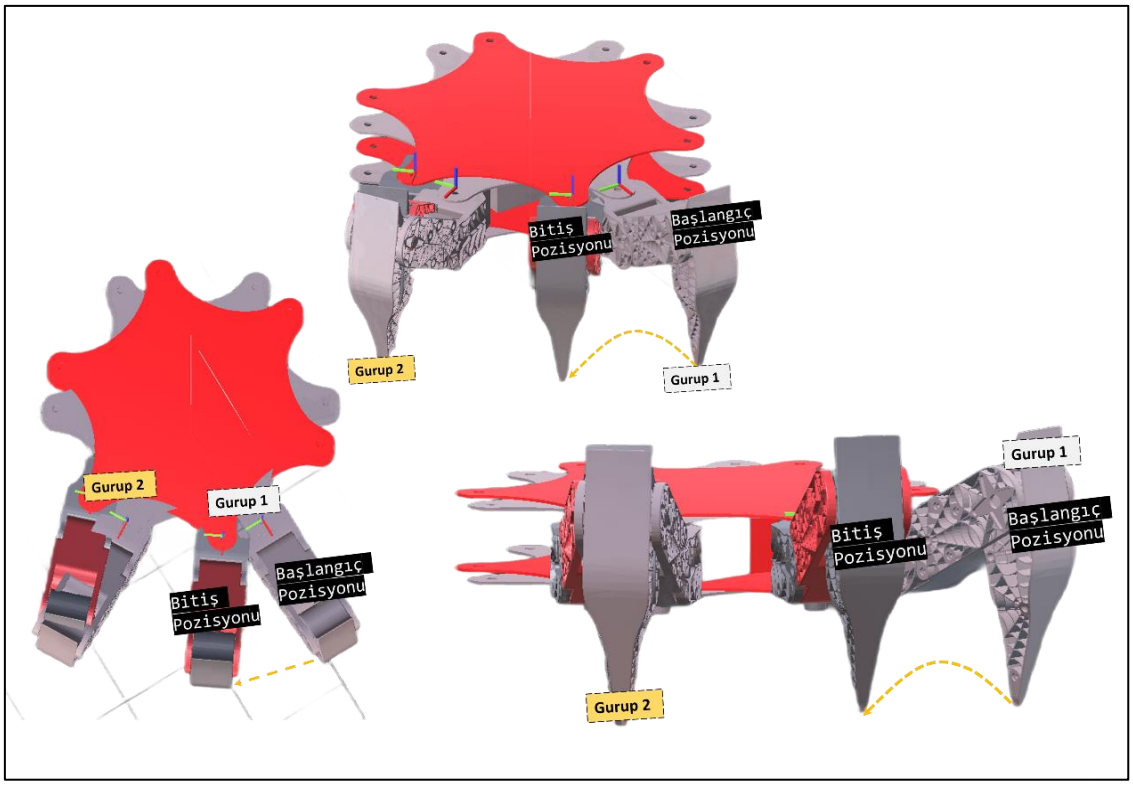
9. Denklem (4.19)'in tersi alınır ise Denklem (4.20)'da ki dönüşüm matrisi elde edilir. Bu matris sayesinde orijin koordinat sistemine göre bilenen herhangi bir pozisyon değeri LegX koordinat sistemine göre hesaplanır.

$$ {}^{LX}_{O}T = {}^{O}_{LX}T^{-1} \tag{4.20} $$

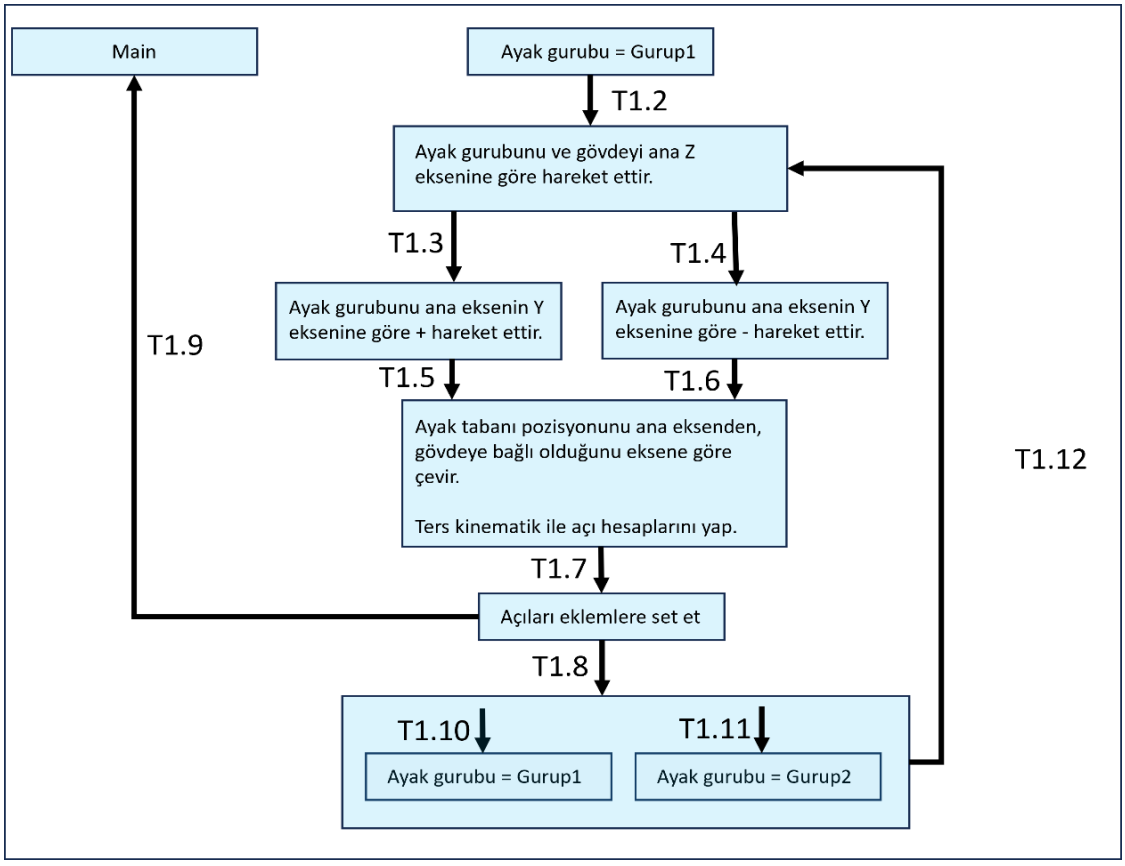
10. Denklem (4.20) kullanılarak orijin koordinat sistemine göre pozisyon bilgisi bilinen her bir ayak tabanı pozisyonu LegX koordinat sistemlerine göre hesaplanır.

$$ {}^{LX}_{BASE}P = {}^{LX}_{O}T\ {}^{O}_{BASE}P \tag{4.21} $$

11. Denklem (4.21) 'de bulunan pozisyon bilgileri için ters kinematik denklemleri ile çözülerek $\theta_1$, $\theta_2$ ve $\theta_3$ açıları hesaplanır.
12. Eğer gidilecek mesafe kat edilmiş ise ayak tabanı sabit kalacak gruplar değiştirilerek birince adıma geçilir. Eğer mesafe kat edilmemiş ise ilk dört adım sırasıyla tekrarlanır.




Şekil 4.11. Dönme işlemi




Şekil 4.12. Dönme hareket algoritması durum makinesi

Şekil 4.11'de dönme algoritması uygulandığında oluşan bir kesit gösterilmektedir. Şekil 4.11'de gösterildiği gibi iki farklı grup iki farklı hareket yapmaktadır. Grup 1'deki bacaklar bir dönme adımı atarken grup 2'deki bacaklar ise bulunduğu yeri korur. Algoritmada anlatıldığı gibi grup 1'e ait bacakların ayak tabanları dönme esnasında $Y$ ekseninde hareket ettirilir.

Tablo 4.2. Dönüş algoritması geçiş tablosu

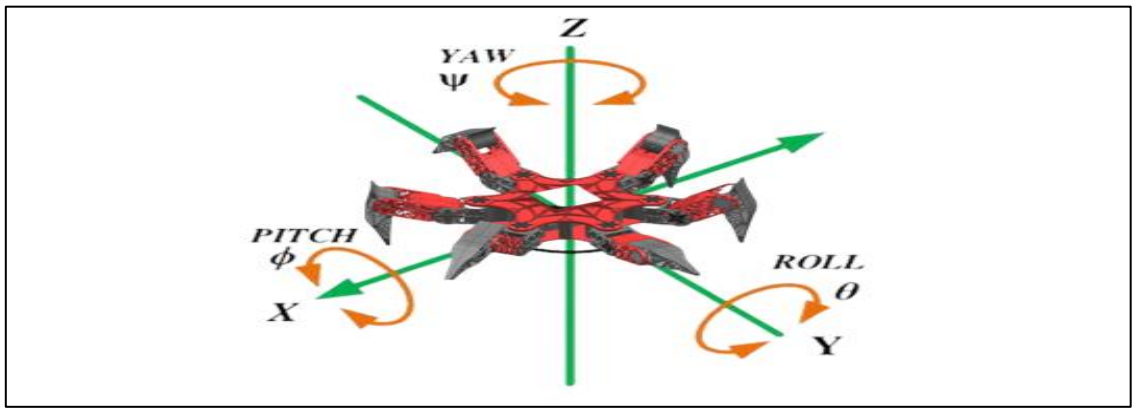
| | |
|---|---|
| T1.1 | durum = dönme |
| T1.2 | koşulsuz |
| T1.3 | 0 ≤ adımZ ve adımZ ≤ k/2 |
| T1.4 | k/2 ≤ adımZ ve adımZ ≤ k |
| T1.5 | koşulsuz |
| T1.6 | koşulsuz |
| T1.7 | koşulsuz |
| T1.8 | durum ≠ duruş veya adımZ ≠ k |
| T1.9 | durum = duruş ve adımZ = k |
| T1.10 | Ayak grubu = Grup 2 |
| T1.11 | Ayak grubu = Grup 1 |
| T1.12 | Null |

Şekil 4.12'da yürüyüş algoritmasının durum makinesi gösterilmektedir. Tablo 4.1'de ise bu durum makinesindeki geçiş şartları verilmektedir. Şekilde gösterildiği gibi ilk önce bulunduğu yeri korumayacak olan ayak grubu yapılandırılır. Sonrasında gövde hareket ettirilir. Bir sonraki adımda ilgili grup bacağı yukarı veya aşağı yönde hareket ettirilir. Tabloda da gösterildiği gibi 'duruş' komutu gelmişse ilgili hareketin bitmesi beklenir. Herhangi bir harekete başlamadan önce robotun başlangıç konumunda olması önemlidir.

Algoritmalar bir hareketi tamamlayıp bitirmek ve belirli bir başlangıç konumundan çalışacak şekilde geliştirilmiştir. Tablo 4.1'de şartsız geçişler koşulsuz ile gösterilmektedir. Bu geçişler herhangi bir işlem gerçekleştirildiğinden sonra diğer işleme geçmek için konulan geçişlerdir.

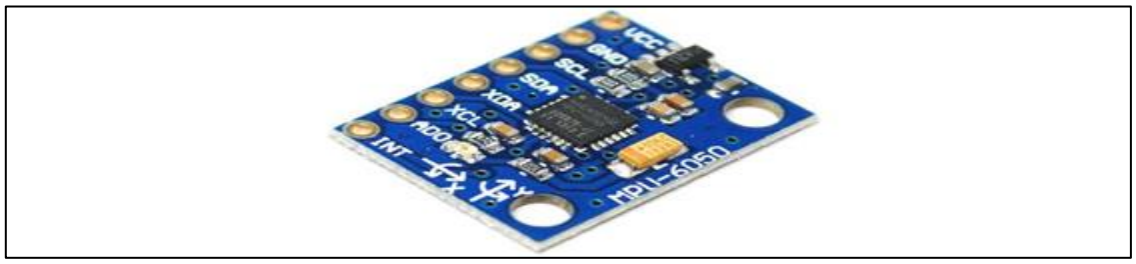
### 4.4. Dengede Durma Algoritması

Bu bölümde robotun bulunduğu eğimli bir yüzeyde dengede kalabilmesi için yapılan çalışmalar anlatılmıştır. Örümcek robotun eğimli bir yüzeyde dengede kalabilmesi için Şekil 4.13'de verilen gövde üzerindeki *pitch* ve *roll* açı değişimlerini hesaplanmıştır.




Şekil 4.13. Örümcek robot üzerindeki roll, pitch, yaw dönme eksenleri

Şekil 4.13'de gösterildiği gibi *pitch* açı değişimi, robot gövdesinin *X* ekseni etrafında, *roll* açı değişimi de *Y* ekseni etrafında ve *yaw* açı değişimi de *Z* ekseni etrafındaki açı değişimidir.



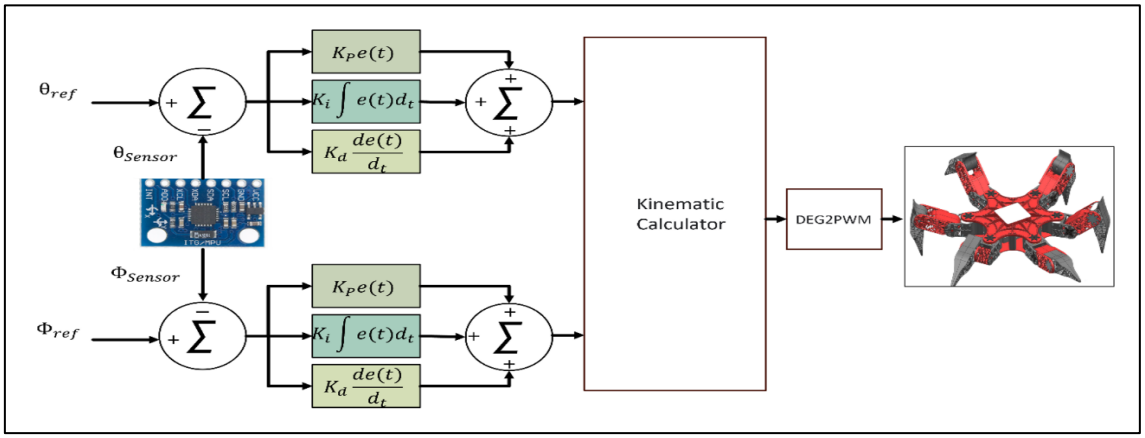

Şekil 4.14. MPU6050 6 eksen ivme ve gyro sensor

*Roll*, *pitch* ve *yaw* açı değişimleri hesaplanabilmesi için Şekil 4.14' de gösterilen MPU6050 sensörü kullanılmıştır. Sensörden alınan açısal ivme ölçer değerleri kullanılarak Denklem (4.22) ve (4.23) ile *roll* ve *pitch* açı değişimleri aşağıdaki gibi hesaplanmıştır (URL-2).

Denklemlerde kullanılan $A_x$, $A_y$ ve $A_z$ değerleri yerçekimi ivmelerini ifade eder. İlgili A değeri cihazının ilgili ekseni boyunca yerçekimi tarafından etkilenen ivmeyi gösterir. Ölçülen ivme "$g$" birimi cinsinden sağlar.

$$pitch = tan^{-1}\left(\frac{A_x}{\sqrt{{A_y}^2+{A_z}^2}}\right) \tag{4.22}$$

$$roll = tan^{-1}\left(\frac{A_y}{\sqrt{{A_x}^2+{A_z}^2}}\right) \tag{4.23}$$



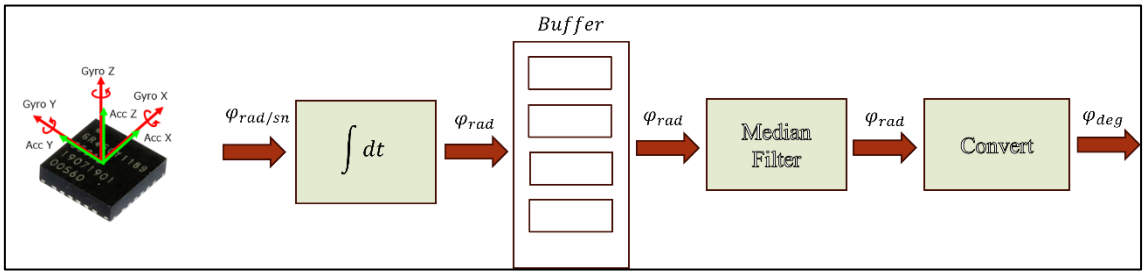

Şekil 4.15. Dengede durma algoritması

Hesaplanan açı değerleri Şekil 4.15'de gösterilen algoritma ile robotu dengede tutmak için kullanılmıştır. Sensörden gelen *roll* ve *pitch* açı değerleri ile robotun durması istenilen referans açı değerleri PID algoritmasından geçerek kinematik dönüşüm gerçekleştirilir.

### 4.5. Yön Stabilizasyonu

Örümcek robotun yürüdüğü yüzeydeki sürtünmeden, robotun montajı sırasında bacaklar arasında oluşan montaj farkından veya servo motorlarda bulunan dişli boşluklarından dolayı gerçekleştirilen robot üzerinde yürüme sırasında yürüdüğü yönde sapmalar gerçekleşebilmektedir.

Bu durumun önlenebilmesi için yön stabilizasyonu algoritması geliştirilmiştir. Robotun yürüdüğü yöndeki kaçıklığın anlaşılabilmesi için *Z* ekseni üzerindeki *yaw* açı değişimi hesaplanmıştır.




Şekil 4.16. *Yaw* açı değişim hesabı

*Yaw* açı değişimi Şekil 4.16'da gösterildiği gibi hesaplanmıştır. Sensörden gelen veriler birim zamanda değişen açısal hız datalarıdır. Bu değerlerin integrali alınarak açısal hız değişimlerinden açı değerleri elde edilmiştir. Elde edilen değerlerde gürültüler oluşabileceğinden veriler bir buffer içerisinde biriktirilmiştir. Buffer'daki dataların median filtresi alınarak *yaw* açı değeri elde edilmiştir. Daha sonra hesaplanan *yaw* açı değeri ile gövdenin yürüme yönündeki kaçıklık elde edilmiştir. Yürüme sırasında bu kaçıklığın tersine gövde döndürülerek yön stabilizasyonu sağlanmış olunur.

Bu bölümde sonuç olarak her bir hareket algoritmasında temel bir mantığın çalıştığı görülmüştür. Her bir hareket algoritmasında tüm bacaklar aynı anda çalışmamakta, bacaklar iki farklı gruba ayrılarak senkron çalışmaktadır ve temel olarak hareketler dönüşüm matrislerinden ve kinematik çözümlerden oluşmaktadır.

# 5. YAZILIMSAL BİLEŞENLER

Bu bölümde tez kapsamında geliştirilen yazılımın özelliklerinden, programlama mimarilerinden bahsedilmiştir. Robotun kontrolü için gerekli olan uzak kontrol yazılımı ve mesaj paketleri açıklanmıştır.

## 5.1. Yazılımsal Özgün Özellikler

Herhangi bir hazır kütüphane kullanılmadan kinematik denklemlerin hesaplanması, koda dökülmesi ve robotun kontrol edilmesi bu tez çalışmasının özgün yönlerinden biridir. Bu özgün yönlerin yanı sıra yazılım mimarisinde nesne yönelimli programlama (NYP) mimarisi kullanılmıştır. Böylece altı bacak için birden fazla kod yazılmadığı, temelde tek bir bacak üzerine kod yazıldığı için kodun performansından, hızından ve kapladığı alandan tasarruf elde edilmiştir. Temelde tek bir bacağa ait bir kontrol sınıfı yazıldığı için bu örümcek robotun bacak sayısında istenildiği gibi değişikliklere gidilebilmektedir. İstenildiği taktirde robot dört bacak veya farklı sayılardaki bacaklar ile kontrol edilebilmektedir. NYP mimarisinin yanı sıra yazılım parametrik fonksiyonlar ve değişkenler kullanılarak yazılmıştır. Her bir bacağın uzuvları sabit değişkenlerle değil parametrik isimler ile yazıldığından yazılım farkı tasarıma sahip robotlarda da çalışabilir hale getirilmiştir.

### 5.1.1. NYP

NYP, bilgisayar programlama dünyasında kullanılan bir paradigmadır. Bu paradigma, bir programın kodunu nesneler olarak adlandırılan yapısal birimlere ayırmak ve bu nesnelerin birbirleriyle iletişim kurmasına olanak tanımak için tasarlanmıştır (Gamma ve diğ., 1994).

NYP, birçok farklı programlama dili tarafından desteklenmektedir. Bu diller arasında Java, C++ (Lippman ve diğ., 2012), Python (Letscher ve Goldwasser, 2007) ve C# gibi popüler diller yer almaktadır. NYP, kodun daha düzenli ve okunaklı hale gelmesini sağlar. Ayrıca, kodun yeniden kullanılabilirliğini artırır, bakımını kolaylaştırır ve büyük projelerde daha iyi bir organizasyon sağlar.

NYP programlama, bir nesnenin özelliklerini ve davranışlarını tanımlayan sınıfların kullanılmasıyla çalışır. Örneğin, bir arabayı modelleyen bir sınıf, arabaların sahip olduğu özellikleri (renk, marka, model, hız vb.) tanımlayabilir ve arabaların yapabileceği eylemleri (sürüş, frenleme, park etme vb.) de belirtebilir.

NYP programlama, birçok farklı kavramı içerir. Bu kavramlar arasında miras, sarmalama, çok biçimlilik ve soyutlama yer alır. Miras, bir sınıfın diğer bir sınıftan özellikleri veya davranışlarını almasıdır. Sarmalama, bir nesnenin özelliklerini ve davranışlarını bir arada tutan bir kavramdır. Çok biçimlilik, aynı sınıftan türetilen nesnelerin farklı şekillerde davranabildiği bir kavramdır. Soyutlama ise, bir nesnenin sadece önemli olan özelliklerinin ve davranışlarının belirtilmesidir. NYP, birçok farklı programlama problemi için uygun bir paradigmadır, özellikle büyük ve karmaşık projeler için uygundur (Albahari ve Albahari, 2012).

Bu tez kapsamında yapılan algoritmalarda EK-A ve EK-B'de gösterilen nesneler oluşturulmuştur. EK-A'da gösterilen leg sınıfı; name, "alpha", beta ve gama gibi değişkenlere sahiptir. Name bu sınıftan üretilen her bir bacak nesnesinin ismini tutmaktadır. Bu nesnede bulunan değişkenlerden bazıları, her bir eklemin tutulduğu "alpha", "beta" ve "gama" değişkenleridir. Bu değişkenlerin değeri EK-A'da gösterilen her bir leg sınıfı içerisinde bulunan "İnverseKinematicsForLegBase" fonksiyonu ile hesaplanır. Ayrıca fonksiyon LegX koordinat sisteminin, orijin koordinat sistemine göre pozisyonunun hesaplanması için "GetAlphaPosForOrigin" fonksiyonuna sahiptir. Bacağın ayak tabanının ise LegX koordinat sistemine göre hesaplanması için "UpdateLegBaseFORG" fonksiyonu bulunmaktadır. Ayrıca ayak tabanının bulunduğu pozisyonu, orijin koordinat sistemine göre hareket ettirmek için "MoveLegBasePoint" fonksiyonuna sahiptir. Ters kinematik fonksiyonunun çağrıldığı, bacağın sahip olduğu eklemlerdeki açı değerlerinin güncellenmesi gibi işlemlerin yapıldığı "update" fonksiyonu da leg sınıfının içerisine yazılmıştır.

EK-A'da verilen leg sınıfı sadece bir bacağı tanımlarken EK-B'de verilen "hexapod" sınıfı ise örümcek robotun bütününü tanımlamaktadır. EK-B'de gösterildiği gibi "hexapod" sınıfı içerisinde robotta bulunan 6 bacak, "leg" sınıfı kullanılarak tanımlanmıştır. Tüm bacaklar "hexapod" sınıfı içerisinde "legs" dizisine aktarılarak programlama sırasında her bir bacağın kontrolü kolaylaştırılmıştır. "Hexapod" sınıfı daha

önce algoritmalar bölümünde anlatıldığı gibi her bir grup bacağı hareket ettirmek için MoveLegGroup fonksiyonuna sahiptir. Bu fonksiyon EK-B'de gösterildiği gibi "legs" dizisi ile leg bir sınıfına ait fonksiyonlara ulaşmaktadır. Bunun yanı sıra "hexapod" "sınıfı robotun yürümesi, dönmesi gibi algoritmaların çalıştırıldığı fonksiyonlara da sahiptir.

Yazılan "leg" ve "hexapod" sınıfı sayesinde istenildiği takdirde robotun sahip olduğu bacak sayısı artırılabilir veya eksiltilebilir. Bu işlem sadece "hexapod" sınıfının içerisinde gösterilen "legs" dizisine yeni bir bacak ekleyerek veya çıkartılarak yapılır. Bunun yanı sıra eklenecek yeni algoritma veya fonksiyon, doğrudan leg sınıfı içerisine yazılarak sistemde bulunan tüm bacaklara bunun otomatik bir şekilde geçmesi sağlanır.

### 5.1.2. Parametrik Programlama

Parametrik programlama yazılımdaki esnekliği artırır. Esneklik, bir modelin farklı senaryolara uygun hale getirilmesini ve farklı kullanım durumlarına uygun yazılımların geliştirilmesini mümkün kılar. Parametrik programlama, esnekliği sağlamak için bir dizi farklı yöntem kullanır. Bunlar arasında değişkenlerin sabitlenmesi, değişkenlerin sınırlandırılması, yeni değişkenlerin tanımlanması ve parametrelerin matematiksel işlemlerle birleştirilmesi yer alır. Bu yöntemler, bir modelin esnekliğini artırarak farklı senaryolara uygun hale getirilmesini sağlar. Esneklik, yazılım geliştirme sürecinde büyük bir avantajdır. Örneğin, bir parametrik modelin parametreleri farklı senaryolara uygun hale getirilebilir. Bu, bir ürünün tasarımında veya bir üretim hattının optimize edilmesinde kullanılabilir. Örneğin, bir otomobil üreticisi, aynı platformu kullanarak farklı modeller tasarlayabilir ve bu modellerin farklı özelliklerini parametrelerin optimize edilmesiyle elde edebilir. Bu da maliyet tasarrufu sağlar ve müşterilere farklı seçenekler sunar. Esneklik, yazılımın kullanımı sırasında da fayda sağlar. Örneğin, bir parametrik model, farklı senaryolara uygun hale getirilebilir ve bu senaryolara uygun sonuçlar üretebilir. Bu, bir ürünün performansını optimize etmek için kullanılabilir veya bir işletmenin verimliliğini artırmak için kullanılabilir. Esnekliğin artırılması, kullanıcılara daha fazla seçenek sunar ve yazılımın daha fazla kullanılabilir hale gelmesini sağlar. Sonuç olarak, parametrik programlama, matematiksel modellerin kullanımıyla birçok fayda sağlar ve esnekliği artırarak farklı senaryolara uygun hale getirilebilir. Esnekliğin artırılması, yazılımın geliştirme sürecinde ve kullanım sırasında büyük bir avantajdır ve kullanıcılara daha fazla seçenek sunar (Martin, 2008).

Bu tez çalışmasında yazılımın farklı uzunluklara veya kinematiğe sahip robotlarda da çalışabilmesi için oluşturulan parametre sınıfı EK-C'de gösterilmektedir. EK-C'de gösterilen $\text{coxia}_X$, $tibia_X$, $tibia_H$, $femuar_X$ ve $femuar_H$ değişkenler kinematik bölümünde anlatılan değişkenlerdir. Bu değerler ilgili robotun bacağındaki uzunluklara göre belirlenerek, geliştirilen yazılım istenilen örümcek robotta çalıştırılır. Ayrıca EK-C her bir bacağın gövdeye bağlandığı LegX koordinat sisteminin gövdedeki koordinat sistemine göre pozisyonları ve açı değerleri de parametreye eklenmiştir. Bu değerler de değiştirilerek farklı gövde yapılarına sahip örümcek robotlarda tanımlanabilir.

### 5.2. Uzak Kontrol Yazılımı

Bu tez kapsamında gerçekleştirilen robotun herhangi bir masaüstü ile web tarayıcı sayesinde kontrol edilmesi için yazılımlar geliştirilmiştir. Robotun bir web tarayıcısından kontrol edilebilmesi için MQTT, React gibi arayüzler kullanılmıştır.

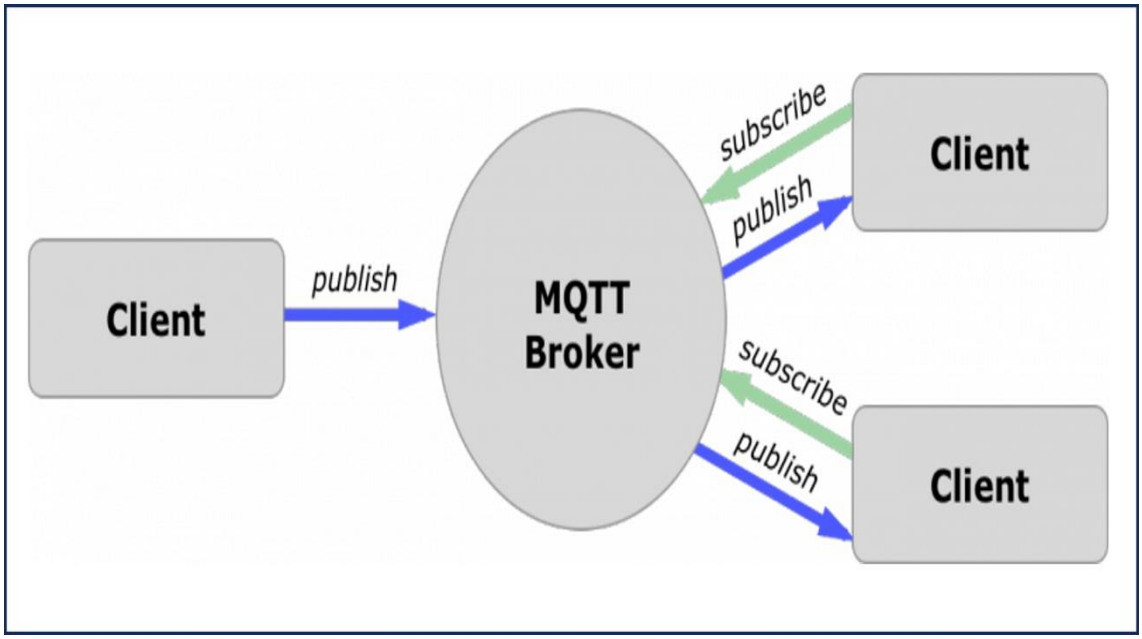
#### 5.2.1. MQTT

MQTT; özellikle internet üzerinden haberleşen uygulamalar için hedeflenen hafif ve basit bir mesajlaşma protokolüdür (Smart, 2020). MQTT kullanılarak internet üzerinden makinalar arasında iletişim kurulabilir. Hemen hemen tüm bulut sistemleri; akıllı nesnelere ulaşmak, onlardan veri okumak, onlara mesaj göndermek için MQTT protokolü kullanılırlar.

Bu protokol, istek-yanıt yapısına dayalı HTTP protokolüne karşıt olarak yayıncı abone yapısında TCP/IP bağlantısı kurulur. TCP/IP protokolünün yazılabildiği Linux; Windows, Android gibi işletim sistemlerinde çalışır (Hillar, 2017).

#### 5.2.2. MQTT Yayıncı Abone Mimarisi

MQTT yayıncı abone mimarisi Şekil 5.1'de gösterilmektedir. MQTT sunucusu en kilit unsurdur. Asıl görevi yayıncılardan aldığı mesajları herhangi bir konuya üye olan abonelere göndermektir. Sunucu mesajı aldıktan sonra mesaj içerisindeki konu bilgisine göre mesajı ilgili konuya abone olan cihazlara gönderir.




Şekil 5.1. MQTT yayıncı-abone mimarisi

Bir aygıt sunucuya veri göndermek istediğinde bu işleme yayın denir. Bir aygıt sunucudan veri almak istediğinde bu işleme abone olmak denir.

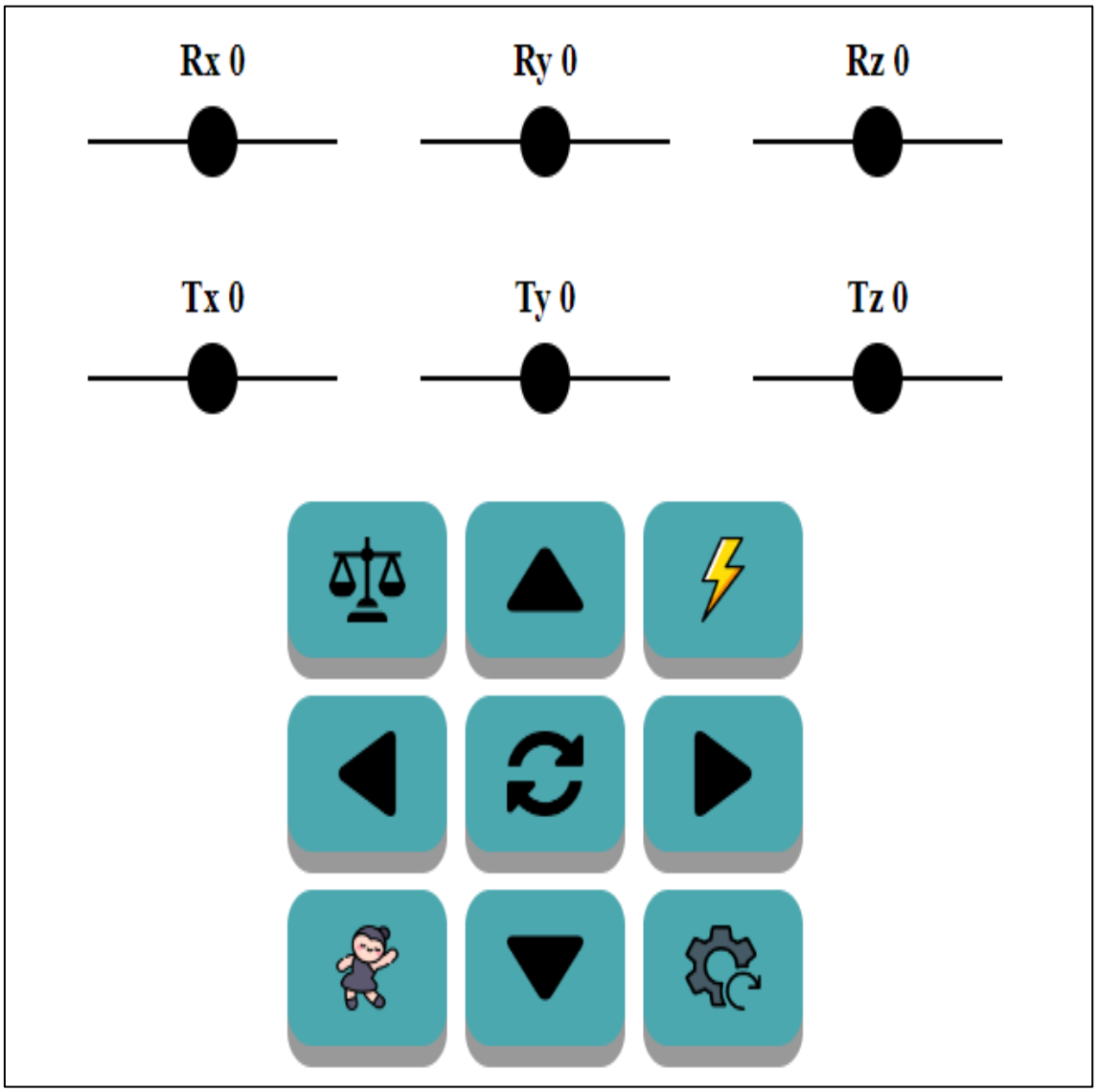
Nesneler birbiriyle direkt iletişim kurmazlar. Bunun yerine sunucuya bağlanırlar. Her bir istemci bir yayıncı, abone veya her ikisi de olabilir. Aboneler verileri belli bir konu adıyla yayınlar ve aboneler bu konu adı aracılığıyla sunucuya abone olarak verilir.

### 5.2.3. HiveMQ

HiveMQ, MQTT protokolünü kullanarak internete bağlı cihazlar arasında güvenli ve hızlı veri iletişimini sağlayan bir MQTT sunucusudur (Hillar, 2017). HiveMQ, ölçeklenebilir bir yapıya sahiptir ve yüksek performanslı bir MQTT sunucusu olarak tanınmaktadır. HiveMQ, MQTT istemcileriyle birlikte kullanılır. MQTT istemcisi, cihazların HiveMQ ile iletişim kurmasını ve veri gönderip almasını sağlar. HiveMQ, çeşitli lisans modelleri sunar. HiveMQ açık kaynaklı bir sürümü bulunmaktadır. Bu sürüm, ücretsiz olarak kullanılabilir ve sınırlı bir özellik setine sahiptir. Daha gelişmiş özellikler için, HiveMQ'nun ücretli sürümlerini kullanmanız gerekebilir. Bu çalışmada HiveMQ'nin ücretsiz versiyonu seçilmiştir. Gömülü sistemin MQTT üzerinden gelecek mesajları dinleyebilmesi için "/spider/ESP32" başlığı oluşturuldu ve ESP32 işlemcisinin bu konu başlığına abone olması sağlanmıştır.

### 5.2.4. Web Arayüzü

Robotun bir bilgisayar üzerinden kontrol edilebilmesi için React programlama dili kullanılarak bir web arayüzü geliştirilmiştir.




Şekil 5.2. Uzak kontrol arayüzü

Şekil 5.2'de oluşturulmuş web arayüzü gösterilmektedir. Bu web arayüzü ile kullanıcı durumları bölümünde açıklanan her bir işlem yapılabilmektedir. Bu arayüz React yazılım dili kullanılarak gerçekleştirilmiştir. Kullanıcı web arayüzünde herhangi bir butona bastığında ilgili işlem NodeJS ile yazılmış bir server yazılımına bildirilir. Bu server ise gelen işlemi MQTT sunucu birimine bildirir. MQTT sunucu birimi ise gelen mesajı ESP32'e göndererek STM32'e kadar bir komut dizisi gönderir.

### 5.2.5. Mesaj Paket Yapısı

STM32'e gönderilen mesaj İntel HEX formatındadır. Intel HEX formatı bir ARM mimarisine sahip, işlemciyi güncellemek için kullanılan bir format yapısıdır.

Tablo 5.1. Intel HEX formatı

| | | | | |
|---|---|---|---|---|
| *ll* | *aaaa* | *tt* | *dd…* | *ccc* |

Genel dosya formatı Tablo 5.1'de verilmektedir. Intel HEX formatında her bir satır ":" işareti ile başlar. "*ll*" ile ifade edilen onaltılık veri bölümü, "*dd...*" satırında kaç byte veri bulunduğunu belirten 1 byte'lık bir onaltılık sayıdır. "*aaaa*" ile ifade edilen onaltılık veri bölümü mesajı kimin gönderdiğini belirtir. "*aaaa*" değeri "0000" ise, bu durumda mesaj ana kontrol birimine aittir. *aaaa*" değeri "FFFF" ise, bu durumda mesaj alt kontrol birimine aittir. "*tt*" ile ifade edilen kısım komut mesajı olduğu belirtir. Bu kısım "0F" değerindedir. "*dd*" ile ifade edilen kısım bir komut tipini, belirtir (dönme, yürüme, dans etme). "*cc*" ile ifade edilen onaltılık kayıt bölgesi; ilgili onaltılık kayıt satırının kontrol değerini temsil eder. Bu kontrol değeri, ilgili kayıt satırındaki verilerin doğruluğunu sağlamak ve iletişim hatasını tespit etmek için kullanılır. Bu değerin hesaplanması genellikle ilgili verilerin toplamının belirli bir yönteme göre hesaplanmasıyla gerçekleştirilir. Bu hesaplamada kullanılan yöntem, veri bütünlüğünü kontrol etmeye ve iletişim hatalarını tespit etmeye yardımcı olur. Kontrol değeri gelen mesaj paketi içindeki dataların toplanması ve tersinin alınması ile hesaplanır. Örneğin :0200000F0801FE mesajın son 2 karakteri olan kontrol değeri, kaydı oluşturan diğer veriler kullanılarak aşağıdaki örnekteki gibi hesaplanır.

0x01 + 0xFF ^ (0x02 + 0x00 + 0x00 + 0x0F + 0x08+ 0x01) = 0xFE

### 5.2.6. Komutlar

Uzak arayüzden gelen komutlara göre ESP32'nin STM32'e göndermiş olduğu komutlar Tablo 5.2'de gösterilmektedir. Tabloda bir data ve iki data byte'lık mesaj paketleri bulunmaktadır. Bir byte'lık mesaj paketinde sadece komut bilgisi giderken, iki byte'lık bir mesaj paketinde komut bilgisinin yanında gövdenin kaç birim öteleneceği veya döndürüleceği ile ilgili bilgiler gönderilir.

Tablo 5.2. Mesaj komutları

| Bir data byte'lık mesaj paketleri | | Açıklama |
|---|---|---|
| Cmd_Go_Forw | 0x01 | Robotu ileri yönde hareket ettirir. |
| Cmd_Go_Left | 0x02 | Robotu sol yönde hareket ettirir. |
| Cmd_Go_Rote | 0x03 | Robotu olduğu yönde döndürür. |
| Cmd_Go_Rigt | 0x04 | Robotu sağ yönde hareket ettirir. |
| Cmd_Go_Back | 0x05 | Robotu geri yönde hareket ettirir. |
| Cmd_Go_Stop | 0x06 | Robotu durdurur. |
| Cmd_Reset | 0x0F | Robota reset atar. |
| İki data byte'lık mesaj komutları | | Açıklama |
| Cmd_Step_Rx | 0x07 | Robot gövdesini *X* ekseninde döndürür. |
| Cmd_Step_Ry | 0x08 | Robot gövdesini *Y* ekseninde döndürür. |
| Cmd_Step_Rz | 0x09 | Robot gövdesini *Z* ekseninde döndürür. |
| Cmd_Step_Tx | 0x0A | Robot gövdesini *X* ekseninde öteler. |
| Cmd_Step_Ty | 0x0B | Robot gövdesini *Y* ekseninde öteler. |
| Cmd_Step_Tz | 0x0C | Robot gövdesini *Z* ekseninde öteler. |

Bir byte data gönderilen komutların mesaj paket yapısı Tablo 5.3'de gösterilmektedir. Tabloda gösterildiği gibi gönderilen data byte sayısı 1 olduğundan *ll* değeri 0x01'dir. Bunun yanı sıra gönderilen mesaj bir ana kontrolcü mesajı (ESP32'den gelen mesaj) olduğu için *aaaa* değeri *0x0000'*dır. *tt* ise bir komut mesajı olduğu için 0x0F'dir. *dd* değeri komut tipini belirtir. *cc* ise komut mesajının checksum değeridir.

Tablo 5.3. Bir data byte'lık mesaj paket yapısı

| Ana kontrol birimi mesajı | | | | |
|---|---|---|---|---|
| *:ll* | *aaaa* | *tt* | $dd_0$ | *cc* |
| :01 | 0000 | 0F | $0xdd_0$ | CS |

Ana kontrol birimine cevap olarak Tablo 5.4'deki mesaj paketi döndürülür. Tabloda gösterildiği gibi gönderilen data byte sayısı 1 olduğundan *ll* değeri 0x01'dir. Bunun yanı sıra gönderilen mesaj bir alt kontrolcü birimi mesajı (STM32'den gelen mesaj) olduğu için *aaaa* değeri *0xFFFF'*dır. *tt* ise bir komut mesajı olduğu için 0x0F'dir. $dd_0$ değeri komut tipini belirtir. *cc* ise komut mesajının checksum değeridir.

Tablo 5.4. Bir data byte'lık mesaj paket yapısı

| Alt kontrol birimi mesajı | | | | |
|---|---|---|---|---|
| ***:ll*** | ***aaaa*** | ***tt*** | $\boldsymbol{dd_0}$ | ***cc*** |
| :01 | FFFF | 0F | $0xdd_0$ | CS |

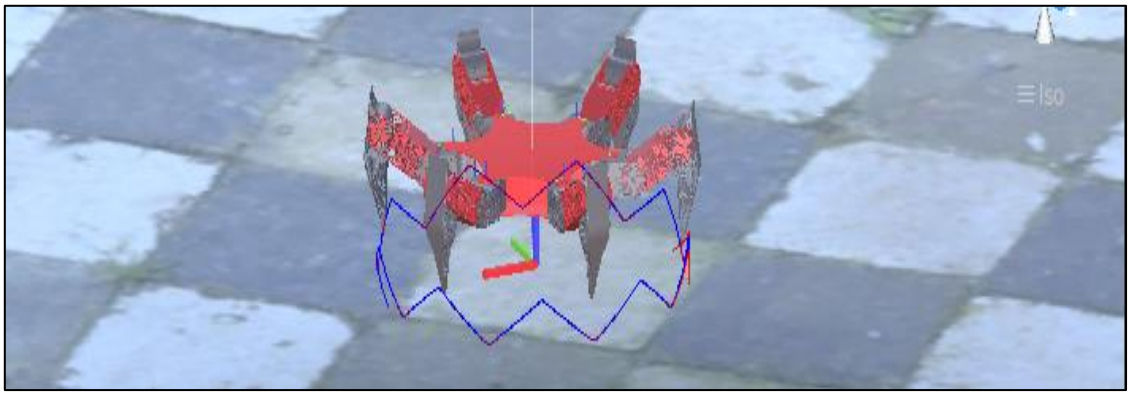
Tablo 5.5. İki data byte'lık mesaj paket yapısı

| Ana kontrol birimi mesajı | | | | |
|---|---|---|---|---|
| ***:ll*** | ***aaaa*** | ***tt*** | $\boldsymbol{dd_0\ dd_1}$ | ***cc*** |
| :02 | 0000 | 0F | $0xdd_0\ 0xdd_1$ | CS |

İki byte datanın gönderildiği mesaj paket yapısı ise Tablo 5.5'de gösterilmektedir. Tabloda gösterildiği gibi gönderilen byte sayısı 2 olduğundan *ll* değeri 0x02'dir. Bunun yanı sıra gönderilen mesaj bir ana kontrolcü mesajı (ESP32'den gelen mesaj) olduğu için *aaaa* değeri *0x0000'*dır. *tt* ise bir komut mesajı olduğu için 0x0F'dir. $dd_0$ değeri komut tipini belirtir. $dd_1$ değeri ise gövdenin ötelenme veya döndürülme değeridir.

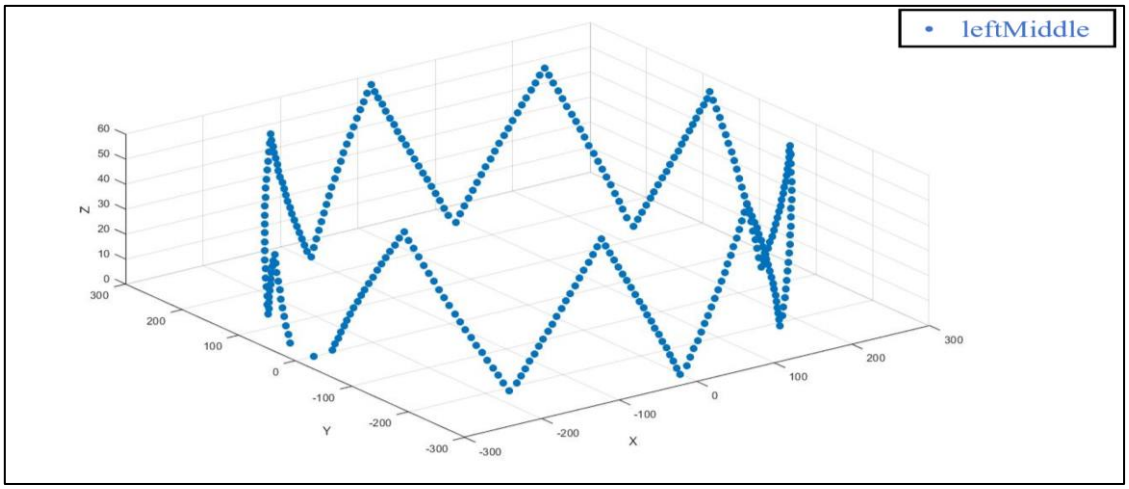
## 6. DENEYSEL SONUÇLAR

Geliştirilen algoritmalar çalıştırılarak ayak tabanları için üretilen yörüngeler incelenmiştir. Robotun 360 derecelik dönüşü için her bir ayak taban pozisyonun bulunduğu konuma başarılı bir şekilde geri döndüğü görülmüştür.



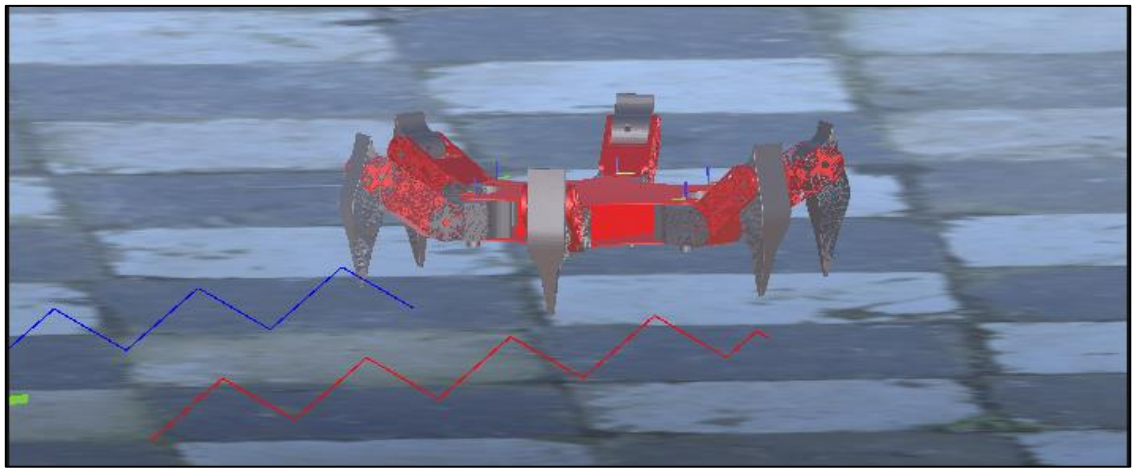
Şekil 6.1. Unity 3D ortamında bir bacağın dönme yörüngesi

Şekil 6.1'de robotun dönme hareketi esnasında geçmişe yönelik oluşan yörüngeleri çizdirilmiştir. Şekilde gösterildiği gibi robot bacağının ayak tabanı çembere yakın bir yörünge izlemektedir. Bu yörünge robotun bulunduğu konum etrafında dönmesi ile ortaya çıkan bir sonuçtur.



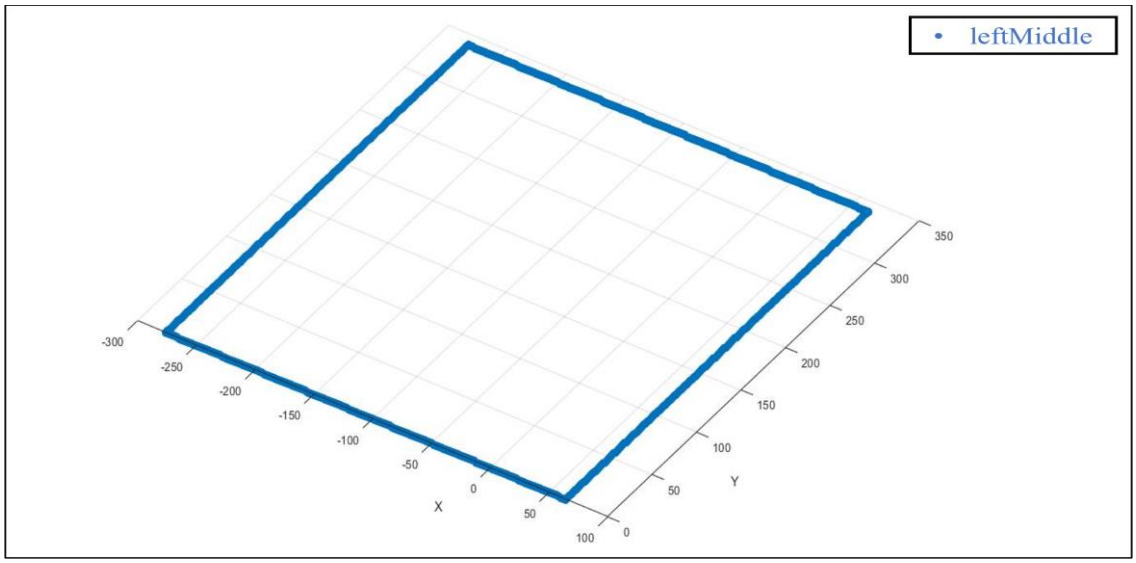

Şekil 6.2. Robotun solundaki orta bacağın ayak tabanının pozisyonu

Şekil 6.2'de ise leftMiddle isimli bacağın ayak tabanının pozisyon değeri çalıştırılan dönme algoritması sonucunda değişikliği gösterilmektedir. Şekilde gösterildiği gibi ayak taban pozisyonu $X$, $Y$ ve $Z$ eksenlerinde değişmektedir.



Şekil 6.3. Unity 3D ortamında iki farklı gruptaki bacakların yürüme yörüngesi

Yürüme algoritmalarının çalıştırılması sırasında iki farklı gruptaki ayak tabanlarının konumları geçmişe yönelik Şekil 6.3'te gösterilmektedir. Yürüme algoritmasında istenildiği gibi bir gruptaki bacak adım hareketi yaparken diğer gruptaki bacağın konumunu sabit tuttuğu görülmektedir.




Şekil 6.4. Bir bacağın ayak tabanının yüreme esnasında konum değişimi

Robotun yürüme algoritması testi sırasında ilk önce ileri yönde 300 cm, kendi sağı yönünde 340 cm, daha sonra geri yönde 300 cm ve kendi solu yönünde 340 cm ilerletilerek bir bacağın ayak tabanın *X* ve *Y* eksenlerindeki pozisyon değişikliği Şekil 6.4'deki gibi elde edilmiştir. Şekilde gösterildiği gibi ayak tabanının başarılı bir şekilde başlangıç konumuna döndüğü görülmüştür.

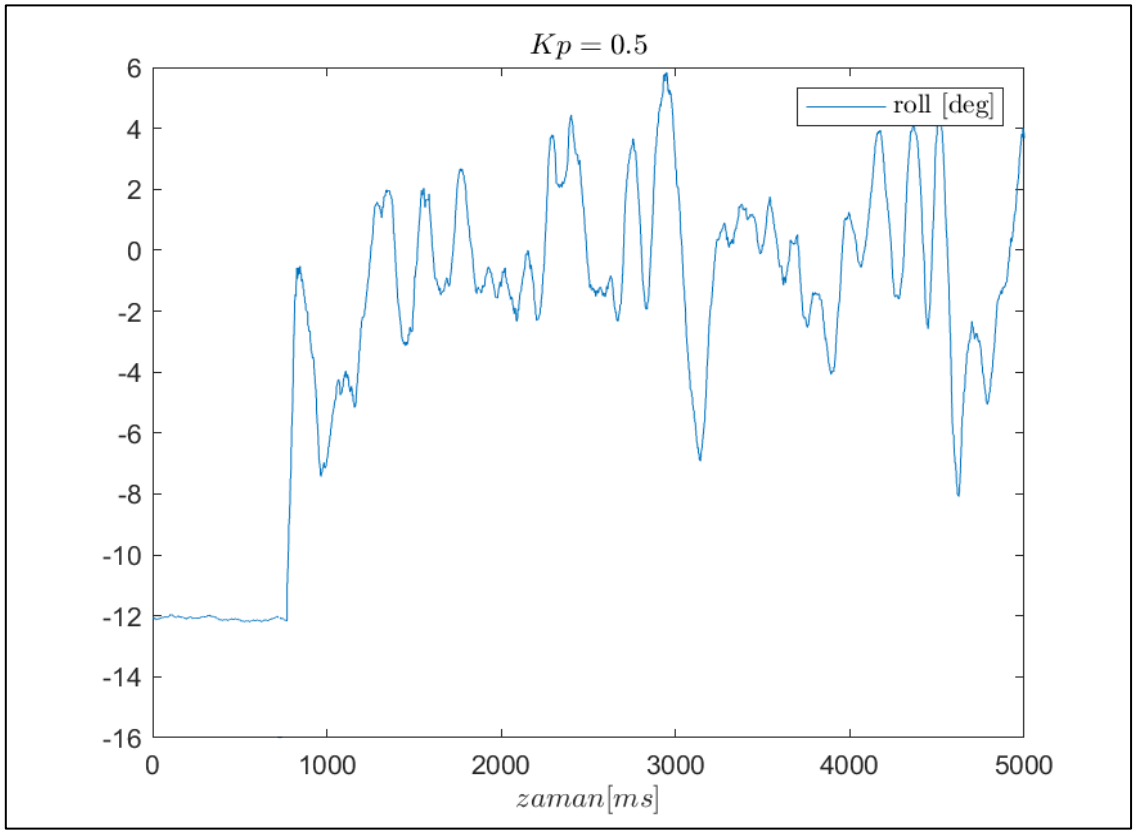
Şekil 6.5. Örümcek robotun gerçekleştirilmesi

Şekil 6.6. Dengede durma algoritması için oluşturulan test düzeneği

Örümcek robotun hareket ettirilmesi ve kontrolü için herhangi bir tablo oluşturmadan kinematik model çıkartılarak parametrik yüksek hareket kabiliyetine sahip bir kontrol algoritması başarılı bir şekilde geliştirilmiştir. Ayrıca algoritma 3D oyun motoru olan Unity 3D ortamında geliştirilerek, simülasyonu yapılmıştır. Simülasyon sırasında

doğruluğu ve çalışması test edilen kod, bir ARM tabanlı işlemciye aktarılarak Şekil 6.5'de gösterilen örümcek robot gerçekleştirilmiştir. Gerçekleştirilen örümcek robotun, simülasyonda test edilen hareketleri başarılı bir şekilde tamamladığı gözlemlenmiştir.

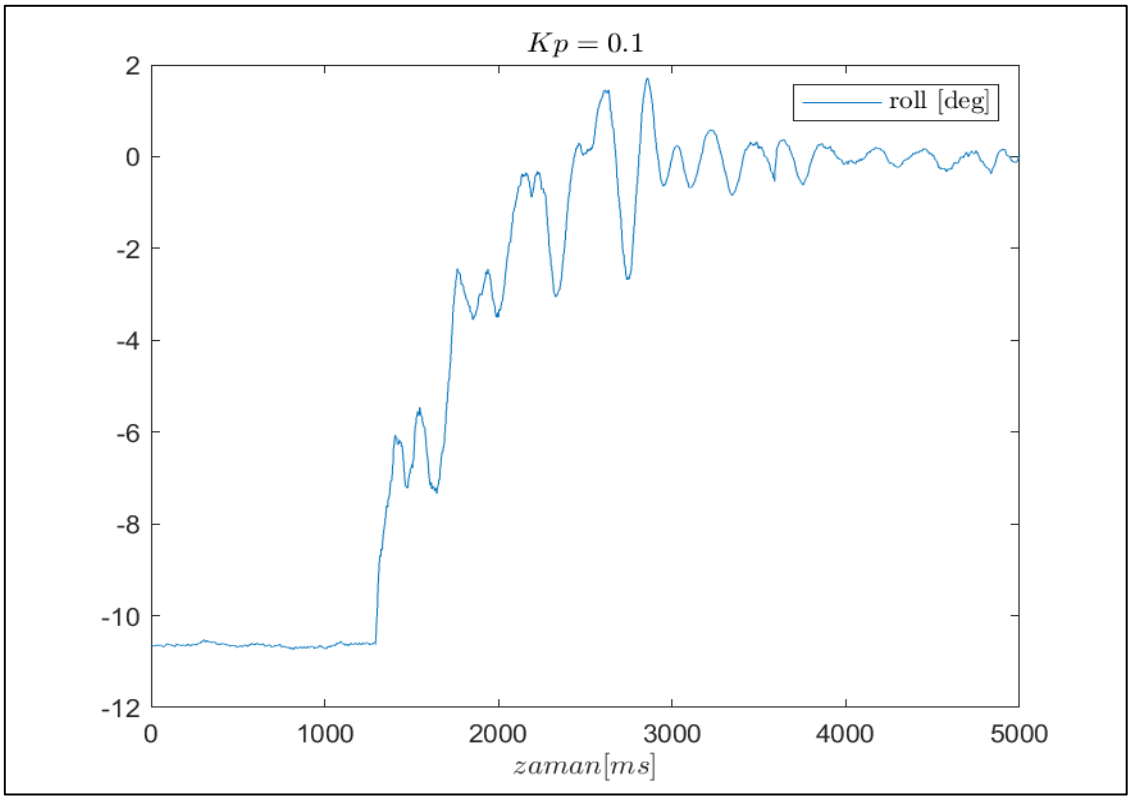
Dengede durma algoritmasının etkinliğini test etmek amacıyla Şekil 6.6'de gösterildiği gibi robot, eğimli bir yüzeye yerleştirilmiş ve sonrasında uzak arayüzden dengede kalma modu aktif edilerek sonuçlar gözlemlenmiştir. Deneyler sırasında, ilk olarak $K_p$ katsayısı değiştirilmiş, ardından $K_d$ katsayısı ve son olarak $K_i$ katsayısı değiştirilmiştir. Her katsayı değişikliği sonrasında test, baştan yeniden yapılmıştır. PID denetleyici katsayılarının bu şekilde değiştirilmesi, robotun eğimli bir yüzeyde dengede kalma yeteneğini nasıl etkilediğini incelemek için kullanılmıştır. PID katsayılarının özellikle, test edilen yüzeyin eğimi nedeniyle *roll* açısındaki etkinin gözlemlendiği bu deneyde, robotun gövdesindeki *Y* ekseni etrafındaki *roll* açı değişimi izlenmiştir.




Şekil 6.7. $K_p$ = 0,5 değeri için *roll* açı değişimi

Oluşturulan test düzeneğinde, öncelikle $K_p$ katsayısı 0,5 belirlenerek robotun tepkisi gözlemlenmiştir. Robotun gövdesindeki *Y* ekseni üzerindeki *roll* açı değişimi, Şekil 6.7'de gösterildiği gibi elde edilmiştir.

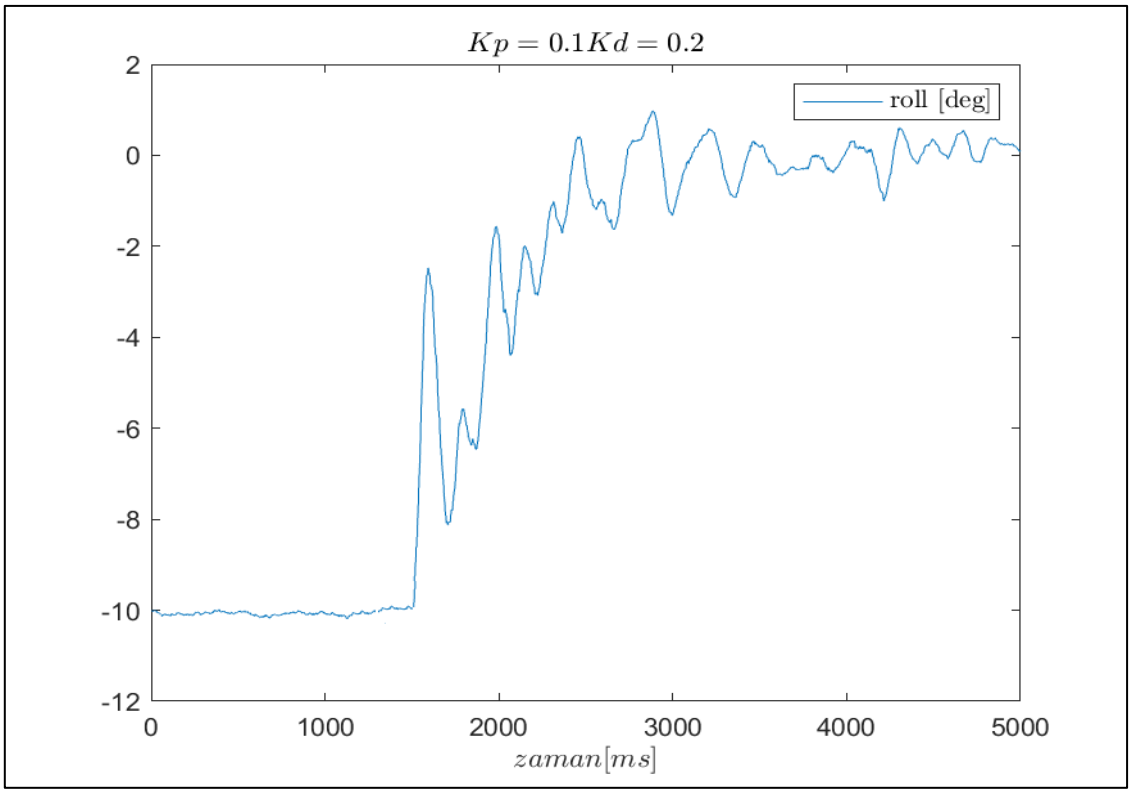
Robotun, gövdesini paralel bir konumda tutma çabası gözlemlenmiş, ancak tam anlamıyla paralel durumda kalamayarak osilasyona girdiği gözlemlenmiştir. Şekilde görüldüğü gibi, robot gövdesini paralel bir şekilde tutmaya çalışırken *roll* açısının -6 ile +6 derece arasında osilasyonlar gösterdiği belirlenmiştir.



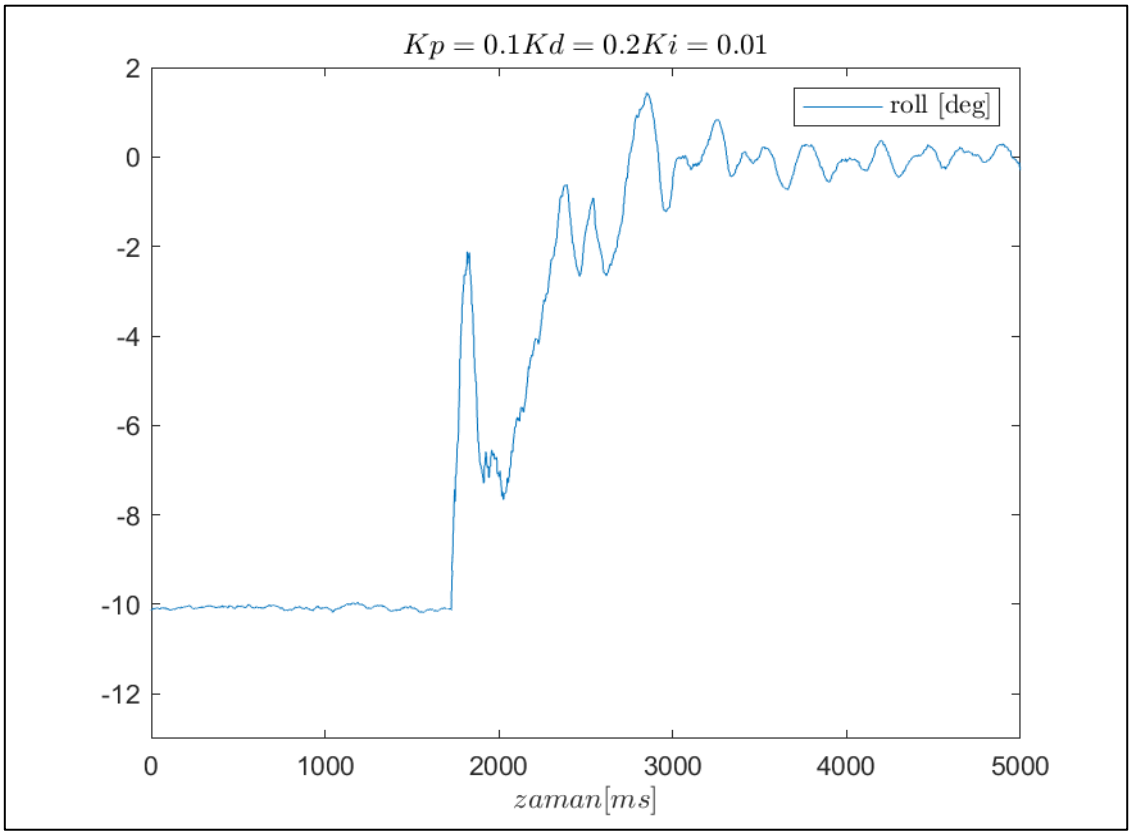

Şekil 6.8. $K_p$ = 0,1 değeri için *roll* açı değişimi

Bir sonraki adımda $K_p$ katsayısı 0,1 belirlenerek robotun tepkisi gözlemlenmiştir. Robotun gövdesindeki *Y* ekseni üzerindeki *roll* açı değişimi, Şekil 6.8'de gösterildiği gibi elde edilmiştir. Robotun, gövdesini paralel bir konumda tutma çabası gözlemlenmiştir. $K_p$ değerinin küçültülmesi osilasyonunda azalmasına sebep olmuştur. $K_p$ değeri 0,1 iken sistemin gövdesinin paralel bir konuma geldiği ve kalıcı durum hatasının -0,4 ile +0,4 derece arasında değiştiği gözlemlenmiştir.

$K_p$ değerinden sonra $K_d$ katsayısının 0,2 belirlenmesiyle *roll* açı değişimi Şekil 6.9'da gösterildiği gibi gözlemlenmiştir. Diğer testlerde olduğu gibi robotun gövdesini paralel bir konumda tutma çabası gözlemlenmiştir. $K_d$ değerinin eklenmesiyle osilasyonun azaldığı ve kalıcı durum hatasının 0,1 dereceye düştüğü gözlemlenmiştir.




Şekil 6.9. $K_p$ = 0,1 ve $K_d$ = 0,2 değeri için *roll* açı değişimi




Şekil 6.10. $K_p$ = 0,1 $K_d$ = 0,2 ve $K_i$ = 0,01 değeri için *roll* açı değişimi

Bir sonraki işlemde $K_i$ katsayısının eklenmesi ile denge algoritmasının tepkisi Şekil 6.10'da gösterildiği gibi görülmüştür. Diğer testlerde olduğu gibi robotun gövdesini paralel bir konumda tutma çabası gözlemlenmiştir. Sisteme $K_i$ parametresinin çok etkisi olmadığı gözlemlenmiştir. Kalıcı durum hatasının 0,1 derecede değiştiği gözlemlenmiştir.

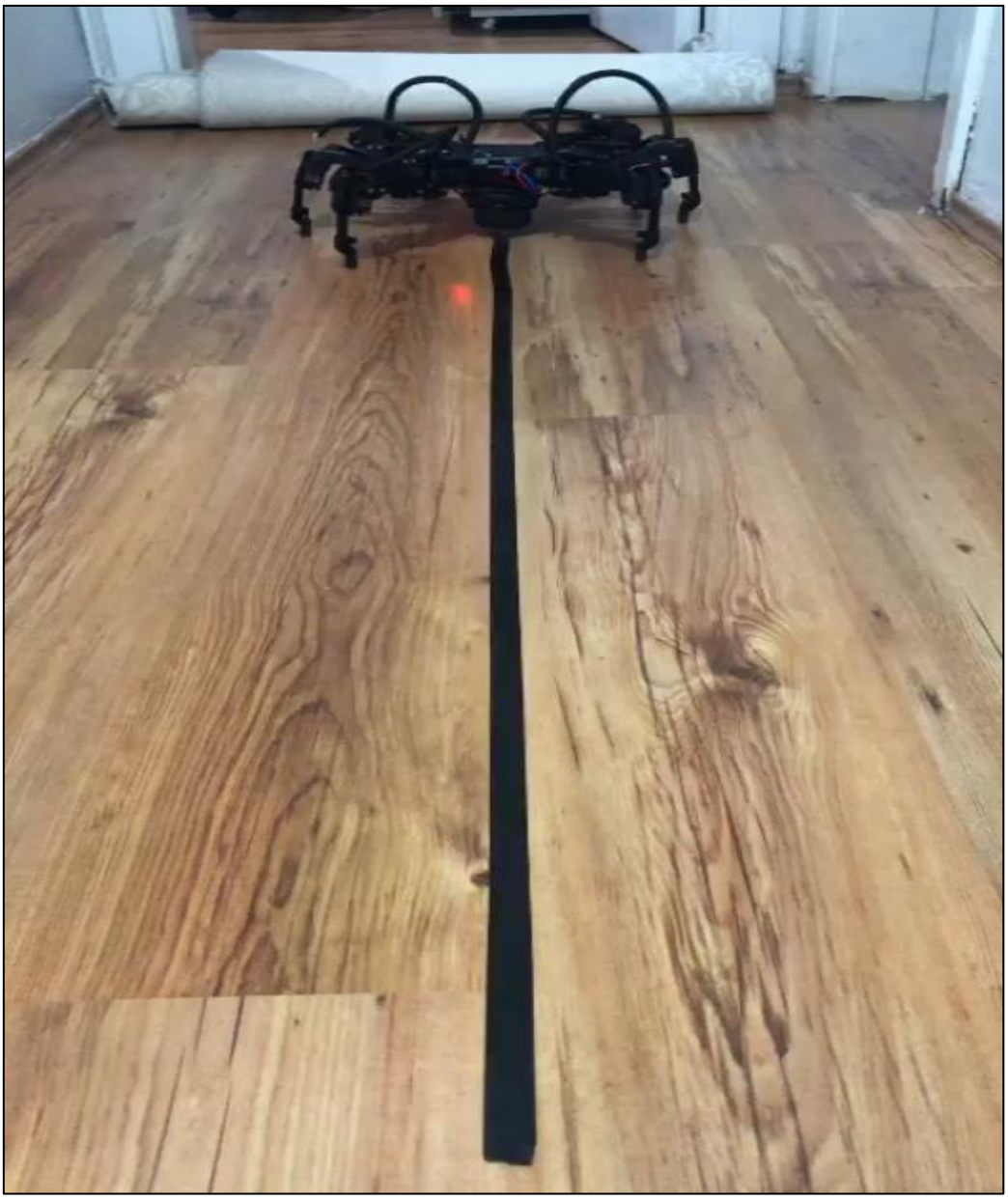
Şekil 6.11. Dengede durma algoritmasının test düzeneği üzerindeki etkisi

Denge algoritmasının sistemi, eğik bir yüzeyde Şekil 6.11'de gösterildiği gibi sistem gövdesini yere paralel olacak şekilde tuttuğu görülmüştür. Robot üzerinde bulunduğu eğimli bir yüzeyde gövdesini paralel tutacak şekilde bir kinematik model çözümü yaparak bacaklarını ilgili pozisyona götürmüştür.

Şekil 6.12. Denge algoritmasının testi durum1

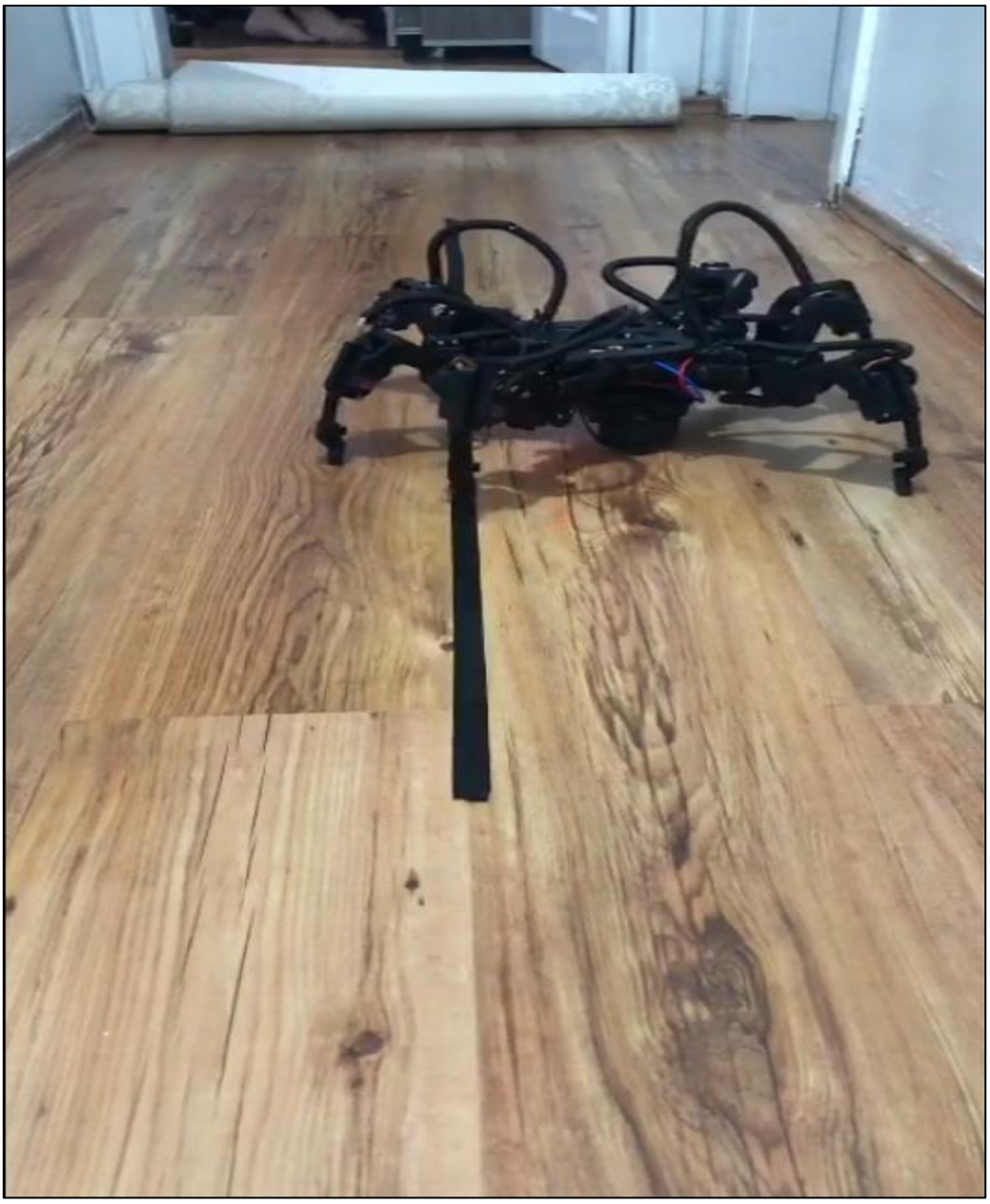
Şekil 6.13. Denge algoritmasının testi durum2

Sistemin bir platform üzerinde dengesini test etmek için robot Şekil 6.12 ve Şekil 6.13'de gösterildiği gibi test düzeneği kurulmuştur. Test düzeneğinde robotun üzerine içi dolu bir bardak konulmuştur. Daha sonra şekillerde gösterildiği gibi platform hareket ettirilerek bardağın durumu gözlemlenmiştir. Robotun denge algoritması sayesinde gövdesini yüzeye paralel tuttuğu ve bu sayede bardağı düşürmeyip içerisindeki sıvıyı dökmediği gözlemlenmiştir.



Şekil 6.14. Robot yön stabilizasyonu test etmek için kurulan düzenek

Robot montaj edilirken her bir bacağın aynı şekilde bağlanamamasından, servo motorlardaki dişlilerde bulunan boşluklardan ve üzerinde yürüdüğü yüzeyin kayganlığından dolayı robot düz bir şekilde yürüyememektedir. Robot belirli bir yöne doğru yürüdüğünde yürüme işlemini bitirdiği durumda başlangıç pozisyonuna göre kaçıklığı ölçmek için Şekil 6.14'de gösterilen test düzeneği kurulmuştur.

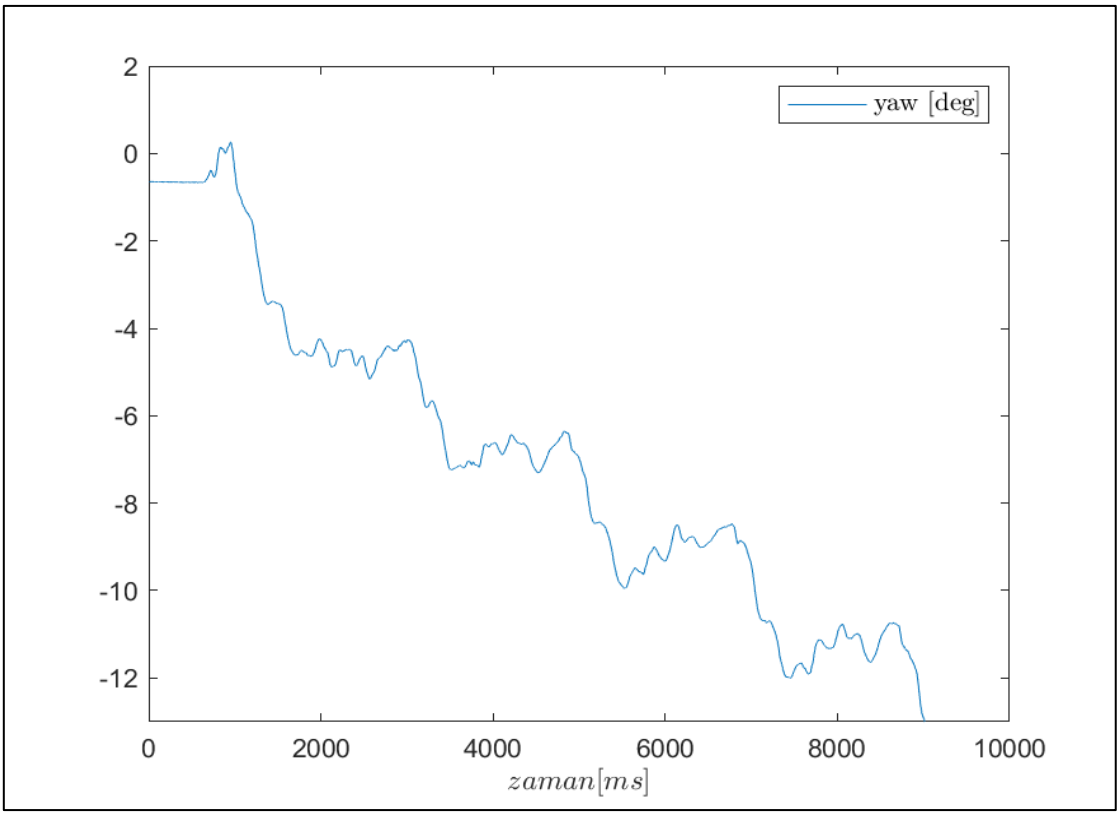


Şekil 6.15. Yön stabilizasyonu olmadığı durumda robotun yürüyüşü

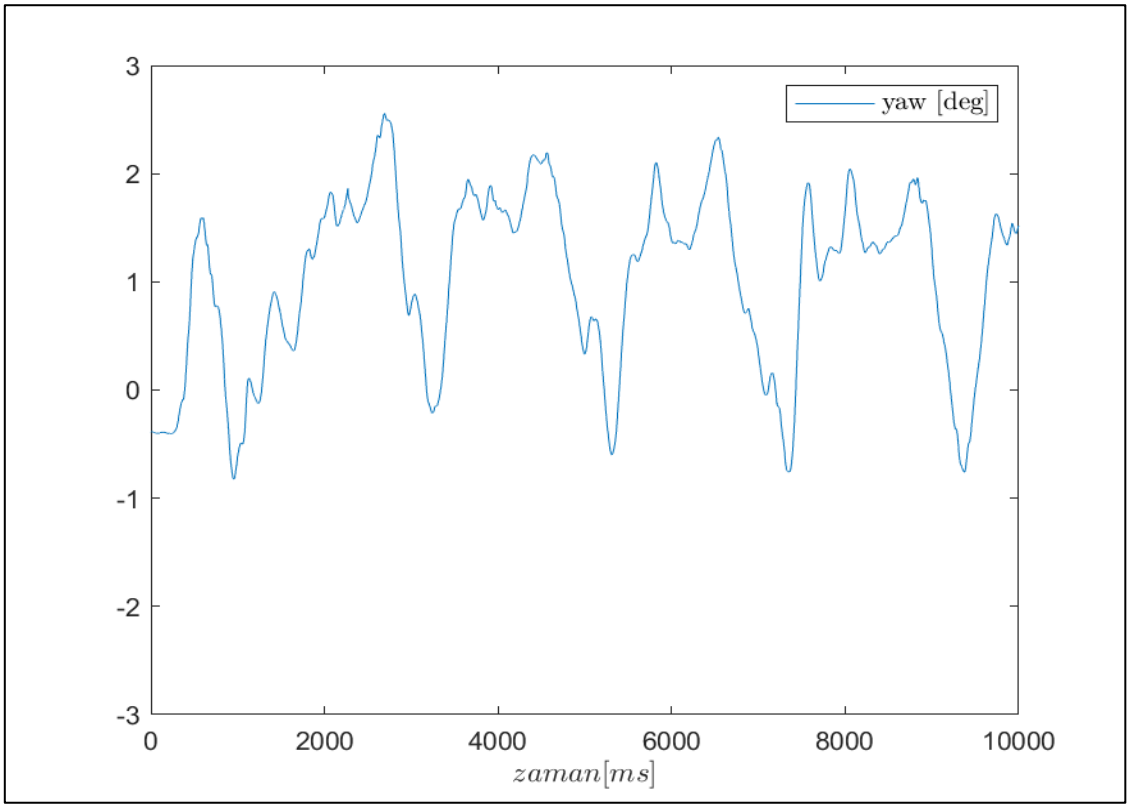
Yön stabilizasyonu aktif edilmeden robot test düzeneğine konulmuş ve belirli bir süre hareket ettirilmiştir. Yön stabilizasyonu olmadığı durumda robot yürüme işlemini bitirdiğinde Şekil 6.15'de gösterildiği gibi hareketi sonlandırmıştır. Robotun belirli bir süre sonrasında yürüme yönünün başka bir yöne doğru kaydığı gözlemlenmiştir.

Şekil 6.16. Yön stabilizasyonu olduğu durumda robotun yürüyüşü

Yön stabilizasyonu aktif edilerek robot test düzeneğine konulmuş ve belirli bir süre hareket ettirilmiştir. Yön stabilizasyonu aktif olduğu durumda robot yürüme işlemini bitirdiğinde Şekil 6.16'de gösterildiği gibi hareketi sonlandırmıştır. Robotun harekete başladığı yönü koruduğu ve bu şekilde hareketi bitirdiği gözlemlenmiştir.




Şekil 6.17. Yön stabilizasyonu olmadığı durumda gövde *yaw* açı değişimi




Şekil 6.18. Yön stabilizasyonu olduğu durumda gövde *yaw* açı değişimi

Yön stabilizasyonu aktif olmadığı durumda robot üzerinde bulunan IMU sensöründen gelen *yaw* açı değeri Şekil 6.17'da gösterilmiştir. Şekilde gösterildiği gibi robotun yürüme yönünde sapmalar oluşmaktadır. Yürüyüş bittiği durumda başlangıç pozisyonundaki yürüyüş yönüne göre 12 derecelik bir sapmanın olduğu gözükmektedir. Yön stabilizasyonu aktif olduğu durumda ise IMU sensöründen gelen *yaw* açı değerleri Şekil 6.18'de gösterilmiştir. Şekilde gösterildiği gibi robot yürüyüş yönünde kaymalar olduğu durumda yürümeyi kayan yönün tersine döndürerek istenilen yürüyüş yönünü takip ettiği gözlemlenmiştir.

# 7. SONUÇLAR VE ÖNERİLER

Tez kapsamında, altı bacaklı bir örümcek robotun yürüme, dönme ve eğimli yüzeylerde denge kontrolü için sanal bir ortamda algoritmalar geliştirilmiş ve bu algoritmalar gerçek bir robot üzerinde başarıyla uygulanmıştır. Geliştirme sürecinin sanal bir ortamda yapılmasının, doğrudan fiziksel bir sistem üzerinde geliştirme yapmaya göre avantajları deneyimlenmiştir. Sanal bir ortamda geliştirme yapılarak birçok senaryo hızlı bir şekilde test edilmiş ve robot üzerindeki etkisi kısa sürede gözlemlenmiştir. Sanal ortamdan gerçek bir sisteme kodun minimum değişiklikle aktarılması için, sanal ortamda gerçekleştirilen kodun geliştirildiği ortama ait kütüphanelerinin kullanılmaması gerektiğinden Unity 3D ortamında herhangi bir hazır matematik kütüphanesi kullanılmadan geliştirmeler yapılmıştır.

Geliştirilen algoritmalar gerçek bir örümcek robota aktarıldıktan sonra sistem üzerinde yürüme ve dönme algoritmaları test edilmiştir. Yapılan testler doğrultusunda sanal ortam ve gerçek ortamda robotun benzer hareketler yaptığı görülmüştür. Robotun montaj sürecinde her bir bacağın aynı hassasiyet ile bağlanamaması, servo motorlardaki dişli boşluklar ve üzerinde yürüdüğü kaygan yüzey nedeniyle, robotun harekete başladığı yönden saparak yürüdüğü gözlemlenmiştir. Bu sebep ile robotun yürüdüğü yönü koruyabilmesi gövdesini sapma yönünün tersi yönünde döndürebilmesi için yön stabilizasyonu algoritması geliştirilmiştir. Yön stabilizasyonu algoritmasının aktif edildiği durumda robotun yürüme konumunu koruduğu gözlemlenmiştir.

Tez kapsamında, robotun eğimli yüzeylerde dengede durabilmesi için PID tabanlı bir algoritma geliştirilmiştir. Farklı PID katsayıları kullanılarak robotun eğimli yüzeylerdeki tepkisi gözlemlenmiş ve uygun katsayılar belirlenmiştir. Bu katsayılardan $K_p$ ve $K_d$ katsayılarının sistem üzerinde daha etkili olduğu görülmüştür. Yüksek katsayı değerleri için sistemin osilasyona girdiği denge pozisyonunu bulamadığı gözlemlenmiştir. Belirlenen PID katsayılarıyla, robotun üzerine bir bardak su konularak eğimli bir yüzeyde test edildiğinde başarılı bir şekilde dengesini koruyabildiği ve suyu dökmediği gözlemlenmiştir. Robot üzerinde bulunan IMU sensöründen gelen bilgiler ile robotun gövdesinin *roll* ve *pitch* eksenlerinde 0,1 derecelik hassasiyet ile dengede kaldığı görülmüştür.

Örümcek robotun kontrolü için MQTT tabanlı bir web arayüzü geliştirilmiş ve bu sayede sabit bir IP veya Wi-Fi bağlantısına ihtiyaç duyulmadan herhangi bir bilgisayar üzerinden kontrol edilebilen bir bulut tabanlı robot başarıyla elde edilmiştir.

Tez kapsamında gerçekleştirilen örümcek robota eklenmesi planlanan yeni özelliklerle robotun geliştirilmesi hedeflenmektedir. Robot üzerine eklenecek sensörler ile robotun bulunduğu konumun bilinmesi, bir harita üzerinde istenilen yörüngeleri izlemesi ve robotun önüne çıkan engellerin aşarak etrafından dolaşması hedeflenmektedir. Robot üzerine bir kamera entegre edilerek robotun kontrol edildiği web arayüzünden robotun izlenmesi hedeflenmektedir.

Tez kapsamında gerçekleştirilen robotun simülasyon ve gerçekleştirilmiş halinde yapılan testler doğrultusunda çekilen videolar URL-3 adresine, kodları ise URL-4 adresine yüklenmiştir.

## KAYNAKLAR


Addison, P., Manning, J., Nugent, T. (2019). *Unity Game Development Cookbook Essentials for Every Game* (1st ed.). Sebastopol CA: O'Reilly.

Albahari, J., Albahari, B. (2012). *C# 5.0 in a Nutshell* (5th ed.). Sebastopol CA: O'Reilly.

Arshad, H., Chun, L., Khor, C. (2012). Virtual Robot Kinematic Learning System a New Teaching Approach. *Journal of Convergence Information Technology*, *7*(14), 54-64. DOI:10.4156/jcit.vol7.issue14.7

Bingül, Z., Küçük, S. (2015). *Robot Kinematiği* (3. basım). İstanbul: Birsen Yayınevi.

Ekelund, J. (2018). Balancing and Locomotion of a Hexapod Robot Traversing Uneven Terrain. Master Thesis, Lund University, Department of Automatic Control, Lund.

Erkol, H. (2015). Robot Kinematik Denklemlerinin FPGA ile Çözülmesi ve Çok Eklemli Bir Robota Uygulanması. Doktora Tezi, Karabük Üniversitesi, Fen Bilimleri Enstitüsü, Karabük, 405868.

Gamma, E., Helm, R., Johnson, R., Vlissides J. (1994). *Design Patterns Elements of Reusable Object-Oriented Software* (1st ed.). Indianapolis: Addison-Wesley.

Hillar, G. (2017). *MQTT Essentials A Lightweight IoT Protocol* (1st ed.). Birmingham: Packt Publishing Limited.

Jackson, S. (2014). *Mastering Unity 2D Game Development* (1st ed.). Birmingham: Packt Publishing Limited.

Jocqué, R., Schoeman, A., Sophia, A. (2001). *African Spiders an Identification Manual* (1st ed.). United Kingdom: Pemberley Natural History Books BA.

Khoswanto, H., Sugiharto, K. N., Sandjaja, I. N., Thiang, T. (2018). Forward and Inverse Kinematic of a Manipulator Simulator Software Using Unity Engine. *Journal of Telecommunication Electronic and Computer Engineering*, *10*(2), 173–176.

Letscher, D., Goldwasser, M. (2007). *Object-Oriented Programming in Python* (1st ed.). Birmingham: Packt Publishing.

Lippman, S., Lajoie, J., Moo, B. (2012). *C++ Primer* (5th ed.). Westford: Addison-Wesley.

Martin, R. (2008). *Clean Code a Handbook of Agile Software Craftsmanship* (1st ed.). Boston: Pearson.

Oner, V. (2021). *Developing IoT Projects with ESP32 Automate Your Home or Business with Inexpensive Wi-Fi Devices* (1st ed.). Birmingham: Packt Publishing Limited.

Pakdel, M. (2020). *Advanced Programming with STM32* (1st ed.). London:lektor.

Roth, J. (2019). Trajectory Regulation for Walking Multipod Robots. *International Journal on Advances in Systems and Measurements*, *12*(3), 265-278.

Sun, J., Ren, J., Wang, B., Chen, D. (2017). Hexapod Robot Kinematics Modeling and Tripod Gait Design Based on the Foot End Trajectory. *IEEE International Conference on Robotics and Biomimetics*, Macau, Macao, 12-13 Aralık 2022.

Tedeschi, F., Carbone, G. (2014). Design Issues for Hexapod Walking Robots. *Robotics*, *3*(2), 181-206. DOI:10.3390/robotics3020181

Thilderkvist, D., Svensson, S. (2015). Motion Control of Hexapod Robot Using Model-Based Design. Master Thesis, Lund University, Department of Automatic Control, Lund.

Thorn, A (2021). *Learning C# by Developing Games with Unity* (6th ed.). Birmingham: Packt Publishing Limited.

URL-1: https://en.wikipedia.org/wiki/Spider_anatomy, (Ziyaret Tarihi: 16 Ekim 2022).

URL-2: https://www.analog.com/en/app-notes/an-1057.html, (Ziyaret Tarihi: 30 Ekim 2022).

URL-3: https://www.youtube.com/channel/UCudzSjT42CCZeGFAgstJX4g, (Ziyaret Tarihi: 16 Eylül 2023).

URL-4: https://github.com/enesvardar/, (Ziyaret Tarihi: 16 Eylül 2023).

Urvaev, I., Spirkin, A., Bazykin, S. (2022). Kinematic Control of The Hexapod Robot. *IEEE 23rd International Conference of Young Professionals in Electron Devices And Materials*, Altai, Russian Federation, 30-31 Haziran 2022.

Xu, S., He, Bin., Hu, H. (2019). Research on Kinematics and Stability of a Bionic Wall-Climbing Hexapod Robot. *Hindawi Applied Bionics and Biomechanics*, *1*(1), 1-17, DOI:10.1155/2019/6146214

Yamağan, İ. (2013). Altı Bacaklı Bir Robot için Dinamik Simülatör Tasarımı. Yüksek Lisans Tezi, Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Karabük, 334623.

# EKLER

**EK-A**

```
class Leg
{
   /* alpha açısının değiştirildiği eklem*/
   public Transform alpha;

   /* beta açısının değiştirildiği eklem*/
   public Transform beta;

   /* gama açısının değiştirildiği eklem*/
   public Transform gama;

   /* bacak tabanın orgindeki kordinat sistemine göre pozisyonu*/
   public MyVector3 legBaseFORG;

   /* bacak tabanın gövdeye bağlandığı yerdeki kordinat sistemine göre pozisyonu*/
   public MyVector3 legBaseFCCP;

   /* bacağın gövdeye bağlandığı noktadaki kordinat
   sistemininin, gövdedeki kordinat sitemine göre pozisyonu */
   public MyVector3 legCCP;

   /* bacağın gövdeye bağlandığı noktadaki
   kordinat sistemininin, gövdedeki kordinat sitemine göre açısal konumu*/
   public MyVector3 legLocalEulerAngles;

   /* alpha açısının radyan değeri */
   public float alphaAngleRad = 0;

   /* beta açısının radyan değeri */
   public float betaAngleRad = 0;

   /* gama açısının radyan değeri */
   public float gamaAngleRad = 0;

   /* her bir bacağın isminin tutulduğudeğişken (leftBack,leftMiddle....rightFront)*/
   public string name;

   /* sınıf yapılandırma fonksiyonu */
   public Leg(GameObject leg, float _endOfset)
   {
      // code …
   }

   public void ForwardKinematicsForLegBase()
   {
        /* Bu fonksiyon ile her bir bacak tabanın pozisyo
```

```
        bilgisi bacağın ana kordinat sistemine göre
        Q1,Q2 ve Q3 açıları ile güncellenmesini sağlar.
        /*
       // code …
   }

   private void İnverseKinematicsForLegBase()
   {
        /*
        Bu fonksiyon ile bacağın ana koordinat
        sistemine göre bacak tabanın pozisyonu
        için gerekli  Q1,Q2 ve Q3 deperleri bulunur
        /*
       // code …
   }

   private MyVector4 GetAlphaPosForOrigin()
   {
        /*
        Bu fonkisyon ile bacağın ana ekseninin
        (bacağın gövdeye bağlandığı yerdeki eksen)
        origine göre pozisyon bilgsi okunuyor
        /*
       // code …
   }

   public void UpdateLegBaseFORG(float ofsetZ)
   {
      // code …
   }

   public void MoveLegBasePoint(float _endOfsetX, float _endOfsetY, float
_endOfsetZ)
   {
        /*
        Bu fonkisyon ile bacak tabanı pozisyonu
        ofset değerleri kadar hareket ettirilir.
        /*
        // code …
   }

   public void Update()
   {
        Bu fonksiyon ile alpha beta ve gama açılarıle ilgili objeler hareket ettirliyor.
   }
}
```

**EK-B**

```
class Hexapod
{
    /*
      init code ...
    */

    public Hexapod()
    {
        legs = new List<Leg>
        {
            new Leg(GameObject.Find("leftBack"), Parameters.endOfset),
            new Leg(GameObject.Find("leftMiddle"), Parameters.endOfset),
            new Leg(GameObject.Find("leftFront"), Parameters.endOfset),
            new Leg(GameObject.Find("rightBack"), Parameters.endOfset),
            new Leg(GameObject.Find("rightMiddle"), Parameters.endOfset),
            new Leg(GameObject.Find("rightFront"), Parameters.endOfset)
        };
    }

    public void MoveHexapodBodyDir(float step, Direction dir)
    {
        /*
        Bu fonkisyon robotun gövdesini hareket ettirmek için kullanılır.
        */
        // code ...
    }

    public void MoveHexapodBodyXYZ(float stepX, float stepY, float stepZ)
    {
        /*
        Bu fonksiyon robot gövdesini x y z ekseninde hareket ettirmek için kullanılır
        */
        // code ...
    }

    public void RotateHexapodBodyXYZ(float stepX, float stepY, float stepZ)
    {
        /*
        Bu fonksiyon robot gövdesini x y z ekseninde
        döndürmek ettirmek için kullanılır
        */
        // code ...
    }
    public void SetLocalPositionHexapodBody(MyVector3 value)
    {
```

```
    /*
    Bu fonksiyon ile gövde pozisyonu istenilen bir konuma set edilir
    */
    // code ...
}

public void SetLocalEulerAnglesHexapodBody(MyVector3 value)
{
    /*
    Bu fonksiyon ile gövde rotatini istenilen bir değere set edilir
    */
    // code ...
}

public void MoveLegGroup(int group, int step, Direction dir)
{
    for (int i = 0; i < step; i++)
    {
        switch (group)
        {
            case (int)LEG_GROUP.firstly:
                legs[(int)(LEG_NAME.leftBack)].MoveDirLegBasePoint(dir);
                legs[(int)(LEG_NAME.rightMiddle)].MoveDirLegBasePoint(dir);
                legs[(int)(LEG_NAME.leftFront)].MoveDirLegBasePoint(dir);
                break;

            case (int)LEG_GROUP.secondly:
                legs[(int)(LEG_NAME.rightBack)].MoveDirLegBasePoint(dir);
                legs[(int)(LEG_NAME.leftMiddle)].MoveDirLegBasePoint(dir);
                legs[(int)(LEG_NAME.rightFront)].MoveDirLegBasePoint(dir);
                break;

            default:
                break;
        }
    }
}

public void Walking(Direction dir, bool contFlag)
{

}

public void RotateLegGroup(int group,float rotateZ)
{
    switch (group)
    {
```

```
            case (int)LEG_GROUP.firstly:
                legs[(int)(LEG_NAME.leftBack)].UpdateLegBaseFORG(rotateZ);
                legs[(int)(LEG_NAME.rightMiddle)].UpdateLegBaseFORG(rotateZ);
                legs[(int)(LEG_NAME.leftFront)].UpdateLegBaseFORG(rotateZ);
                break;

            case (int)LEG_GROUP.secondly:
                legs[(int)(LEG_NAME.rightBack)].UpdateLegBaseFORG(rotateZ);
                legs[(int)(LEG_NAME.leftMiddle)].UpdateLegBaseFORG(rotateZ);
                legs[(int)(LEG_NAME.rightFront)].UpdateLegBaseFORG(rotateZ);
                break;

            default:
                break;
        }
    }

    public void Rotating(bool contFlag)
    {

    }

    public void Update()
    {
        foreach (var joint in legs)
        {
            joint. Update();
        }
    }
}
```

**EK-C**

```
class Parameters
{
  public static float coxiaX = 45;
  public static float tibiaX = 65.8f;
  public static float tibiaH = 0;
  public static float femuarX = 65.0f;
  public static float femuarH = 131.00085f;

  public static float endOfset = 0;

  public static MyVector3 bodyLocalEulerAngles = new MyVector3(0, 0, 0);
  public static MyVector3 bodyLocalPosition = new MyVector3(0, 0, 110);

  public static float lenght = coxiaX + tibiaX + femuarX;

  public static MyVector3 lbEulerAngles = new MyVector3(0, 0, 120);
  public static MyVector3 lmEulerAngles = new MyVector3(0, 0, 180);
  public static MyVector3 lfEulerAngles = new MyVector3(0, 0, 240);
  public static MyVector3 rbEulerAngles = new MyVector3(0, 0, 60);
  public static MyVector3 rmEulerAngles = new MyVector3(0, 0, 360
  public static MyVector3 rfEulerAngles = new MyVector3(0, 0, 300);

 public static MyVector3 lbContCntrPnt = new MyVector3(-68, 120.0f, 0);
 public static MyVector3 lmContCntrPnt = new MyVector3(-140, 0, 0);
 public static MyVector3 lfContCntrPnt = new MyVector3(-68, -120.0f, 0);
 public static MyVector3 rbContCntrPnt = new MyVector3(68, 120.0f, 0);
 public static MyVector3 rmContCntrPnt = new MyVector3(140, 0, 0);
 public static MyVector3 rfContCntrPnt = new MyVector3(68, -120.0f, 0);
 // code…
}
```

## KİŞİSEL YAYIN VE ESERLER

Kuncan, M., **Vardar**, **E.**, Kaplan, K., Ertunç, M. (2017). Turkish Handwriting Recognition System Using Multi-Layer Perceptron. *Journal of Mechatronics and Artificial Intelligence in Engineering, 1*(2), 41-52, DOI:10.21595/jmai.2020.21502

**Vardar**, **E.**, Ertunç, M. (2023). Bir Hexapod Örümcek Robot Tasarımı. *Incohis Spring Organizing Committee*, İstanbul, Türkiye, 20-21 Mayıs 2023.

**Vardar**, **E**., Kaplan, K., Ertunç, M. (2017). Handwriting Character Recognition by Using Fuzzy Logic. *Turkish Journal of Science & Technology, 12*(2), 71-77.

**Vardar**, **E**., Kaplan, K., Ertunç, M. (2018). Ball and Beam Control Based on NARMA L-2 Controller. *International Journal of Control and Automation, 11*(12), 91-102.

## ÖZGEÇMİŞ

İlk, orta, lise ve üniversite öğrenimini Kocaeli'nde tamamladı. Kocaeli Derince Merkez Bankası Anadolu Lisesi sayısal bölümünden 2012 de mezun olduktan sonra 2013 yılında girdiği Kocaeli Üniversitesi Mühendislik Fakültesi Mekatronik Mühendisliği Bölümünü 2017 yılında tamamladı. 2018'den itibaren Turkuaz Elektromekanik firmasında Mekatronik Mühendisi olarak çalışmaya devam etmektedir. 2020 yılında Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Mekatronik Mühendisliği Anabilim Dalı'nda Yüksek Lisans eğitimine başlamıştır.