T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

SANAL GERÇEKLİK ile ROBOT KOL KONTROL ve SİMÜLASYONU

Eda Derya TOPER

DANIŞMAN
Prof. Dr. Fırat KAÇAR

Elektrik- Elektronik Mühendisliği Anabilim Dalı

Elektrik-Elektronik Mühendisliği, Tezli Yüksek Lisans Programı

Haziran, 2024

TEZ KABUL VE ONAYI

Eda Derya TOPER tarafından, **Prof. Dr. Fırat KAÇAR** danışmanlığında hazırlanan **"SANAL GERÇEKLİK ile ROBOT KOL KONTROL ve SİMÜLASYONU"** başlıklı bu çalışma, jürimiz tarafından **29/07/2024** tarihinde yapılan sınav sonucunda **oy birliği** ile başarılı bulunarak **Yüksek Lisans Tezi** olarak kabul edilmiştir.

Tez Jürisi

		İmza	**Sonuç**
DANIŞMAN	Prof. Dr. Fırat KAÇAR İstanbul Üniversitesi - Cerrahpaşa Elektrik Elektronik Mühendisliği Anabilim Dalı		☐ Kabul ☐ Ret
ÜYE	Doç. Dr. Cengiz Polat UZUNOĞLU İstanbul Üniversitesi - Cerrahpaşa Elektrik Elektronik Mühendisliği Anabilim Dalı		☐ Kabul ☐ Ret
ÜYE	Dr. Öğr. Üyesi Mustafa KONAL Tekirdağ Namık Kemal Üniversitesi Elektrik Elektronik Mühendisliği Anabilim Dalı		☐ Kabul ☐ Ret
ÜYE	Doç. Dr. Muhammed Emin BAŞAK Yıldız Teknik Üniversitesi Gemi Elektrik-Elektronik ve Otomasyonu Anabilim Dalı		☐ Kabul ☐ Ret
ÜYE	Doç. Dr. Serap ÇEKLİ İstanbul Üniversitesi - Cerrahpaşa Elektrik Elektronik Mühendisliği Anabilim Dalı		☐ Kabul ☐ Ret

BEYAN

Bu tez çalışmasının kendi çalışmam olduğunu, tezin planlanmasından yazımına kadar bütün aşamalarda etik dışı davranışımın olmadığını, bu tezdeki bütün bilgileri akademik ve bilimsel etik kuralları içinde elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara kaynak gösterdiğimi ve bu kaynakları da kaynaklar listesine aldığımı, yine bu tezin çalışılması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını ve her türlü hukuki sorumluluğu aldığımı kabul ederim.

Eda Derya TOPER

(İmza)

BÜTÇE DESTEKLERİ

SANAL GERÇEKLİK ile ROBOT KOL KONTROL ve SİMÜLASYONU

Bu çalışma İstanbul Üniversitesi - Cerrahpaşa Bilimsel Araştırma Projeleri Koordinasyon Birimi tarafından desteklenmiştir. Proje numarası: 37183

TEŞEKKÜR

Tez çalışmam süresince her konuda yardımcı olan desteğini hiç esirgemeyen tez danışmanım Prof. Dr. Fırat Kaçar'a, her durumda yanımda olan sevgili aileme, çeşitli katkılarıyla yardımcı olan arkadaşlarıma teşekkür ederim.

Haziran 2024 Eda Derya TOPER

İÇİNDEKİLER

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

TABLO LİSTESİ

Sayfa No

SİMGE VE KISALTMA LİSTESİ

Simgeler	Açıklama
T	: Homojen Dönümüş Matrisi
D	: Yer Değiştirme Vektörü
R	: Rotasyon Matrisi
∇f(θ)	: Tahmini eğim hesabı
α	: Öğrenme oranı
ϵ	: Küçük adım büyüklüğü
J(θ)	: Açı mesafesi

Kısaltmalar	Açıklama
HDM	: Homojen Dönüşüm Matrisi
DH	: Denavit Hartenberg
DOF	: Serbestlik derecesi
PWM	: Sinyal Genişlik Modülasyonu
ROS	: Robot İşletim Sistemi
HCI	: İnsan Robot Etkileşimi
HRC	: İnsan Robot İşbirliği
VR	: Sinyal Gerçeklik
AR	: Artırılmış Gerçeklik
XR	: Karma Gerçeklik
ADC	: Analog – Dijital Dönüştürücü
ROS	: Robot İşletim Sistemi
3B	: 3 Boyut

ÖZET

YÜKSEK LİSANS TEZİ

SANAL GERÇEKLİK ile ROBOT KOL KONTROL ve SİMÜLASYONU

Eda Derya TOPER

İstanbul Üniversitesi-Cerrahpaşa

Lisansüstü Eğitim Enstitüsü

Elektrik- Elektronik Mühendisliği Anabilim Dalı

Elektrik-Elektronik Mühendisliği, Tezli Yüksek Lisans Programı

Danışman: Prof. Dr. Fırat KAÇAR

Endüstri 4.0 ve dijital fabrika; sanal gerçeklik, fiziksel üretim sistemini tasarlamak, simüle etmek, optimize etmek ve bununla uzaktan veya işbirlikçi bir şekilde etkileşim kurmak için temel teknolojileri temsil eder. Endüstri ve uzay uygulamalarındaki son deneyimler, sanal gerçeklik tekniklerinin çok yönlü robot kontrol teknikleriyle, yeni çığırlar açabileceğini göstermiştir. Böylece sanal gerçeklik teknikleri; robot kontrol teknolojisinde, sürdürülebilirlik, test edebilirlik gibi durumlar için fiziksel ortama "uzatılmış bir kol" sağlayarak, yeni bir kullanıcı dostu insan-makine arayüzü kalitesinin yolunu açmaktadır.

Bu çalışmada, özellikle gerçek dünyada var olan bir sistemi uzaktan görerek kontrol etmeye imkan sağlayabilecek; prototip oluşturmadaki maliyet sorunu, iş güvenliği riskleri gibi faktörleri azaltabilecek, bu iki ortamı entegre eden sistemler geliştirmek amaçlandı. Buna göre gerçekte ve sanal ortamda birbirine entegre sistem içerisinde çalışacak mekanizma için günümüzde en çok kullanılan ve hala geliştirilmeye devam eden robot kol seçildi.

Bu kapsamda, oluşturulan robot kolun, Robot İşletim Sistemi (ROS) - Unity3D yazılım programı tabanlı Sanal Gerçeklik (VR) arayüzleriyle entegre çalışması sağlanmıştır. Öncelikle ana eksenleri oluşturan 3 serbestlik derecesine (3DOF) sahip robot kol ayrıntılı işlenmiştir.

Ardından model geliştirilerek 5 serbestlik derecesine (5DOF) sahip mekanizma haline getirilmiştir. Robot kolun hareket mekanizması için ileri ve ters kinematik yöntemler kullanılmıştır. Robot kol hareketinin doğruluğunu ölçmek için ileri kinematik yöntemi olan homojen dönüşüm matrislerinden faydalanılmıştır. Matris hesaplamaları için Python program dilinden faydalanılmıştır. Gerçek dünyada oluşturulan robot kol hareketi için Arduino geliştirme kiti kullanılmıştır. Sanal ortamdaki robot kolun hareket mekanizması için ise gradyan iniş algoritmasından faydalanılmıştır. Sanal ortamdaki robot kol kontrolü için Sanal Gerçeklik Gözlük ve Konsolları kullanılmıştır.

Haziran 2024 , 85. sayfa.

Anahtar kelimeler: Robot Kol, Sanal Gerçeklik, Robot İşletim Sistemi, Kinematik

ABSTRACT

M.Sc. THESIS

CONTROL and SIMULATION of a ROBOTIC ARM with VIRTUAL REALITY

Eda Derya TOPER

İstanbul University-Cerrahpaşa

Institute of Graduate Studies

Department of Electrical and Electronics Engineering

Bir öğe seçin.

Supervisor : Prof. Dr. Fırat KAÇAR

Industry 4.0 and the digital factory represent fundamental technologies for designing, simulating, optimizing, and interacting with physical production systems remotely or collaboratively through virtual reality. Recent experiences in industrial and space applications have demonstrated that virtual reality techniques can open new horizons when combined with versatile robot control techniques. Thus, Virtual reality techniques pave the way for a new quality of user-friendly human-machine interfaces in robot control technology by providing an "extended arm" to the physical environment for situations such as sustainability and testability.

In this study, the aim was to develop systems that integrate these two environments, allowing control of an existing system in the real world remotely, thereby reducing factors like the cost problem in prototyping and occupational safety risks. Accordingly, a robotic arm, which is the most commonly used and still evolving mechanism today, was selected to operate within an integrated system in both real and virtual environments.

In this context, the created robotic arm was integrated to work with Virtual Reality (VR) interfaces based on the Robot Operating System (ROS) and Unity3D software program.

Initially, the robotic arm with 3 degrees of freedom (3DOF), which constitutes the main axes, was detailed. Then, the model was developed into a mechanism with 5 degrees of freedom (5DOF). Forward and inverse kinematic methods were used for the movement mechanism of the robotic arm. Homogeneous transformation matrices, a forward kinematic method, were utilized to measure the accuracy of the robotic arm's movement. Python programming language was used for matrix calculations. An Arduino development kit was used for the movement of the robotic arm created in the real world. For the movement mechanism of the robotic arm in the virtual environment, the gradient descent algorithm was utilized. Virtual Reality Headsets and Consoles were used for the control of the robotic arm in the virtual environment.

June 2024, 85 pages.

Keywords: Robot , Virtual Reality, Arm,Robot Operating System, Kinematic

1. GİRİŞ

Bilgi-iletişim teknolojilerinin gelişmesine paralel olarak internet, kablosuz ağlar, üretim ağları gelişmiş ve endüstriyel üretim güçlü ve köklü değişimler yaşamıştır [1]. 2011 yılında Almanya'da gerçekleştirilen Hannover fuarında ortaya atılan Endüstri 4.0 ve bünyesinde barındırdığı teknoloji ve uygulamaları üretim ekosistemlerine yeni bir soluk getirmiştir. Günümüzde siber-fiziksel sistemlerin, Nesnelerin İnterneti'nin (IoT), bulut bilişimin ve yapay zekanın (AI) entegrasyonuyla öne çıkan Endüstri 4.0, üretim süreçlerinde devrim yaratarak onları daha verimli, esnek ve akıllı hale getiriyor [2].

Endüstri 4.0 ile vurgulamak istenen, makineler arası iletişim ve insan-makine işbirliğidir. Bu doğrultuda, günümüzde kullanılan robotlar artık üretim süreçlerinde aktif rol oynayan ve gerçek zamanlı veri alışverişi yapabilen entegre sistemlerin bir parçası olarak görülmektedir. Robotlar, son yıllardaki toplumsal gelişimi yansıtan mekatronik alanının önemli bir ürünüdür. İlk robotlar, 1960'larda endüstriyel sektöre tanıtılmış olup tehlikeli ortamlarda insanların yerine zorlu ve riskli görevleri yerine getirmek amacıyla kullanılmışlardır [3]. Bugün, imalattaki ilk basit uygulamalarının yanı sıra robotlar sağlık, tıbbi bakım, tarım, gemi yapımı, inşaat ve ulusal güvenlik gibi çeşitli alanlarda da kullanılmaktadırlar [4].

Günümüzde, hemen hemen tüm sektörlerde, işlerin çoğu, ihtiyaca göre farklı Serbestlik Derecelerine (DOF) sahip robotlar veya robotik kollar tarafından yapılmaktadır. Robot kolların pazar payı günden güne artmakla birlikte özellikle endüstride kullanım alanları oldukça yaygındır. Bilgisayar kontrolü ve yapay zekanın hızla gelişmesiyle robot teknolojisi de hızlı bir gelişim yakalamış ve multidisipliner çalışmanın yüksek teknoloji birleşimi haline gelmiştir [5].

Endüstri 4.0, birbirine bağlı sistemlerin kusursuz iletişim ve koordinasyonu düzenlediği akıllı fabrikaların ortaya çıkmasını teşvik ediyor. Akıllı fabrika, günümüz pazarının ihtiyaçlarını karşılayacak esnek ve verimli bir üretim çözümüdür ve çoğu zaman sanal organizasyonlar kuran çeşitli endüstriyel ve endüstriyel olmayan ortaklar arasında entegrasyonu sağlar [2]. Üreticiler, fiziksel varlıkların sanal kopyaları olan dijital ikizlerin kullanımı sayesinde süreçleri simüle edebilir ve optimize edebilir, kaynak israfını en aza indirebilir ve üretkenliği en üst düzeye çıkarabilmektedir.

2002 yılında Michigan Üniversitesi'nin endüstriye yaptığı bir sunumla ortaya atılan "dijital ikiz" modeli bir fiziksel sistem hakkında dijital bir bilgi yapısı oluşturma fikrine dayanır. Bu dijital bilgi, fiziksel

sistemin içindeki bilginin bir "ikizi" olacak ve sistemin tüm yaşam döngüsü boyunca bu fiziksel sistemle bağlantılı olacaktır [6]. Dijital ikiz [7] konseptinin ilk kez önerilmesinin üzerinden neredeyse 15 yıl geçti. Bugüne kadar ürün tasarımı, üretim, prognostik ve sağlık yönetimi ve diğer bazı alanlar dahil olmak üzere farklı endüstrilerde birçok Dijital ikiz uygulaması başarıyla uygulanmıştır [8].

Endüstriyel robotların kullanılması, ürün kalitesini yükseltir ve üretim maliyetlerini düşürür. Ancak robotun amaç ve hedefleri hakkında hiçbir bilgisi olmayan kullanıcı, güvenli bir robottan bile rahatsızlık duyar. Ayrıca robotlu üretim hatlarının dizaynı ve devreye alınması sürecindeki iş güvenliği riskleri ve kurulum maliyetleri değerlendirildiğinde, robot simülasyonun önemi daha iyi anlaşılmaktadır. Sanal gerçeklik teknolojisi; fiziksel ortamlardaki robot kontrol sistemlerinde sürdürülebilirlik, test edebilirlik [9], optimizasyon gibi önemli faktörlerin gerçekleşmesinde kolaylıklar sağlayarak yeni bir kullanıcı dostu insan-makine arayüzü kalitesinin yolunu açmaktadır [10].

Sanal gerçeklik üzerine ilk teknik çalışmalar 1830 yıllarında yaşandı. 1935'te Pygmalion's Spectacles adlı bilim kurgu romanı ile sanal gerçeklik fikri yaygın hale gelmeye başladı. 1985'te Jaron Zepel Lanier sanal gerçeklik teriminin yaygınlaşmasını sağlamakla birlikte VR gözlük ve eldiven satan ilk şirket olan VPL Research'i kurmuştur [11]. 2012 yılında Oculus Rift sanal gerçeklik gözlüklerinin piyasaya sürülmesiyle bugün bildiğimiz şekilde sanal gerçeklik teknolojisiyle tanıştık. Sanal gerçeklik; alanında çok yeni olmakla beraber daha çok oyun, video, sinema gibi alanlarda tanınırlığı artmış ve bu alanlarda gelişmeler sağlanmıştır.

Sanal Gerçeklik (VR) ve Artırılmış Gerçeklik (AR) ve Karma Gerçeklik (XR) gibi yeni Bilgi ve İletişim Teknolojilerinin (BİT) ortaya çıkışı, akademisyenler ve endüstriler için yeni İnsan-Robot Etkileşimi (HRI – Human–Robot Interaction) [12] ve İnsan-Robot İşbirliği (HRC - Human-Robot Collaboration) türlerini keşfetmeleri için umut verici fırsatlar sunmaktadır [13].
Sanal gerçeklik sistemleri genellikle otonom ve yarı otonom sistemler için çeşitli arayüzlerin geliştirilmesini sağladığından dolayı uygun bir teknoloji olarak önerilmektedir. Geçmişte bu tür sistemler, kısmen hem robot yazılım geliştirme hem de sanal ortam altyapısı için ortak yazılım sistemlerinin bulunmayışı nedeniyle tipik olarak "tek seferlik" deneysel sistemler olarak geliştirilmiştir. Daha yakın zamanlarda, hem robot kontrolü (ROS vb.) hem de sanal ortam gösterimi ve etkileşimi (Unity vb.) için ortak çerçeveler ortaya çıkmaya başlamıştır [14].

Bu tezde, sanal ve gerçek ortamda oluşturulan bir model üzerinden, iki ortamı entegre eden bir sistem geliştirmek amaçlandı. Mekanizma modeli olarak günümüzde önemli bir yere sahip olan robot kol seçilmiştir. Yapılacak tez çalışmasına göre robot kol uzaktan gözlemlenerek kontrol edilebilecektir. Buradaki amaç özellikle kullanıcı ya da operatör gerektiren robot sistemlerinde, kullanıcıların uzaktan

çalışmasına imkan sağlayabilmektir. Bu sayede “yerinde görme” işlemlerinin uzaktan yapılabileceği düşünülmektedir. Ayrıca test, deneme ya da deneysel çalışmaların sanal ortamda yapılmasına olanak tanıyarak prototip oluşturmadaki maliyet ve zaman sorunu, iş güvenliği riskleri gibi faktörleri azaltması amaçlanmıştır.

Bu tez çalışma kapsamında, oluşturulan 5 serbestlik derecesine sahip (DOF) [15] robot kolun, Robot İşletim Sistemi (ROS) - Unity3D [16,17] yazılım programı tabanlı Sanal Gerçeklik (VR) [18] arayüzleriyle entegre çalışması sağlanmıştır. Robot kolun hareket yöntemi için ileri ve ters kinematik [19] yöntemler kullanılmıştır.

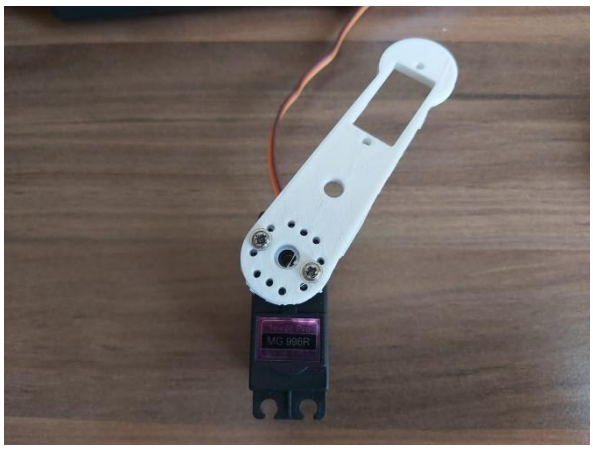
2. KAVRAMSAL ÇERÇEVE

2.1 ROBOT MANİPÜLATÖR KAVRAMI ÜZERİNE

Endüstride insan koluna benzeyen, Fanuc, ABB, KUKA gibi endüstriyel robot [20] örnekleri kullanılıyor. Robot kol genellikle insan kolunun işlevini taklit eden daha spesifik bir yapıya sahiptir. Manipülatör ise daha geniş bir kavram olmakla birlikte, tutma, taşıma gibi çeşitli görevleri yerine getiren çok amaçlı bir anlam taşır. Bu yüzden robot kol yerine daha genel bir kavramı ifade eden manipülatör terimini kullanırız.

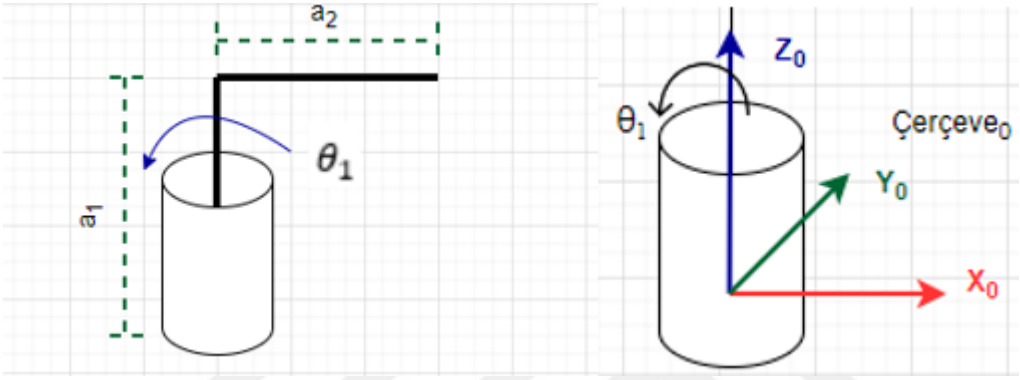
Bir manipülatör, bağlarla birbirine bağlanan bir dizi eklemden oluşur. Eklemler, robot kolunun hareket edebilmesini sağlayan dönme veya doğrusal hareketleri gerçekleştiren mekanizmalardır. Bu eklemler, robot kolunun farklı açılarda ve yönlerde esnek hareket etmesine olanak tanır. Bağlar ise eklemleri birbirine bağlayan ve robot kolunun uzaysal konumunu belirleyen sabit veya hareketli parçalardır. Her bir bağ, eklemler arasında bir köprü görevi görerek manipülatörün uzanma ve tutma işlemlerini gerçekleştirmesine yardımcı olur. Bu yapı, manipülatörün karmaşık ve hassas görevleri yerine getirebilmesini, nesneleri alıp taşımasını ve belirli noktalara yerleştirmesini sağlar.

Şekil 2.1'de tek servo motor kullanılan manipülatör bir eklemden, iki bağdan oluşur. Eklem tipi olarak servo motor hareket yönünden dolayı, döner mafsal eklem türü olduğunu görebiliriz. Döner mafsal eklem türü bağların diğer bağlara göre dönme hareketine izin veren bir mafsaldır. Buradaki manipülatör 1 serbestlik derecesine (1DOF) [15] sahiptir. Serbestlik derecesi eklem sayısına göre belirlenir. Çalışma alanı uç efektörün ulaşabileceği alandır.



Şekil 2.1: 1DOF manipülatör.

Şekil 2.2'de, Şekil 2.1'de yer alan manipülatörün kinematik diyagramı yer almaktadır. Kinematik diyagram dönel eklemlerdeki açı 0° olduğunda manipülatörün aldığı şeklin iki boyutlu görünümüdür. a_1 ve a_2 linkler, θ_1 eklem dönme açısıdır. Servo motor 180° saat yönünde hareket eder. Dönme hareket yönü sağ el kuralına göre belirlenir [15]. Buna göre diyagramda gösterilen θ_1 açısının dönme yönü sağ el kuralına göre motorun pozitif dönme yönünü gösterir. Çerçeve kavramı, matematiksel olarak nesnelerin veya robot parçalarının uzaydaki konum ve yönelimlerini tanımlamak için kullanılan bir referans sistemidir. Bir çerçeve şekildeki gibi genellikle üç dikey eksenden (X, Y, Z) ve bir orijin noktasından oluşur.



Şekil 2.2: 1DOF manipülatörün kinematik diyagramı.

2.1 SANAL GERÇEKLİK

Sanal gerçeklik, bilgisayar ortamında oluşturulan 3 boyutlu resimlerin ve animasyonların teknolojik araçlarla insanların zihinlerinde gerçek bir ortamda bulunma hissini vermesinin yanı sıra, ortamda bulunan bu objelerle etkileşimde bulunmalarını sağlayan teknoloji olarak tanımlanabilir [21].

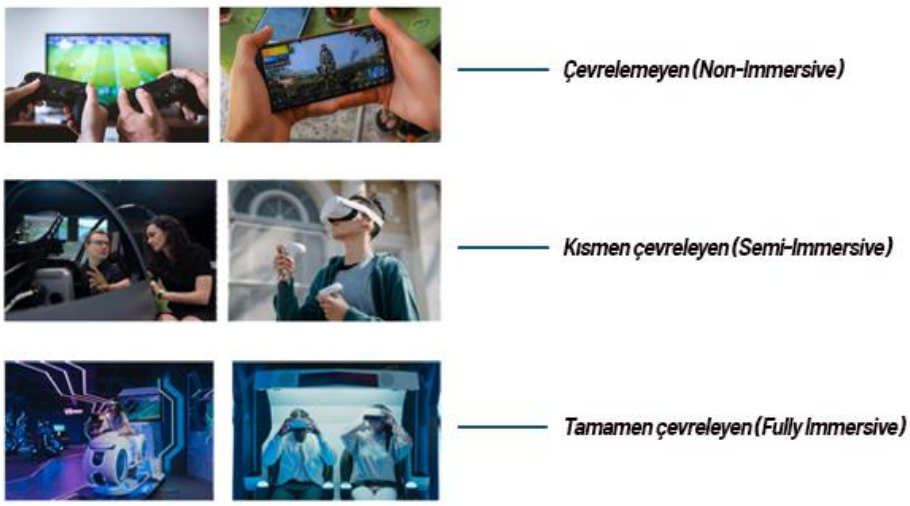
Sanal gerçeklik genellikle özel başlıklar ve çeşitli algılayıcılar kullanılarak gerçekleştirilir. Bu başlıklar, görsel ve işitsel girdileri sunarken, algılayıcılar kullanıcıların hareketlerini izleyerek bu hareketleri sanal ortama aktarır.

Son zamanlarda Sanal Gerçeklik alanında endüstri, sağlık, eğitim gibi alanlarda başlayan yeni çalışmalar görülmektedir. Özellikle endüstriyel robotların kullanımı, insan robot etkileşimi ve insan robot iş birliği üzerine çalışmalar yapılmaya başlandı. Bu çalışmalarda farklı nesne ve senaryolar ile robot davranışlarının simülasyonu oluşturularak test edilmesi amaçlanmıştır. Kullanıcı (operatör) eğitimi, prototip test etme, endüstriyel süreci izleme, iş güvenliği risk analizlerinin yürütülmesi, insan-robot etkileşimi (HRI) [12] ve insan-robot iş birliği (HRC) [13] üzerine robot davranışlarını incelemek gibi gerçek dünyada var olan bir dizi süreç sanal ortamda denenmeye başlandı [9]. Çünkü Sanal

Gerçeklik, gerçekte ciddi maliyet ve zaman kayıplarına yol açan bu süreçleri orijinal sisteme etki etmeden test edebilme imkanı sağlıyor.

Sanal Gerçeklik alanının bir dizi faydası olmasına rağmen, Sanal Gerçeklik gözlüklerinin uzun süreli kullanımı insanlarda baş dönmesi, göz ağrısı, dengesizlik gibi fiziksel sorunlara yol açabiliyor. Bu yüzden günümüzde, bu alanda çokça çalışmalar yapılsa da, uygulamalar piyasaya sürülse de kullanımı hala yaygın değildir.

Sanal gerçeklik için kullanıcıların sanal dünyalarla etkileşim kurmasını sağlayan ve bu deneyimi daha gerçekçi hale getirmek için bazı bileşenler, donanımlar geliştirilmiştir. Şekil 2.3'teki gibi sanal gerçeklik uygulamaları; çevrelemeyen (Non-Immersive), kısmen çevreleyen (Semi-Immersive) ve tamamen çevreleyen (Fully Immersive) olmak üzere üç farklı kategoriye ayrılır [22]. Çevrelemeyen sanal gerçeklik uygulamalarında kullanıcılar düşük oranda kontrole sahiptir. Sanal ortam, genellikle masaüstü bilgisayarlarda, dizüstü bilgisayarlarda veya mobil cihazlarda görüntülenir. Standart giriş cihazları olan fare, klavye, oyun kumandası kullanılarak kontrol edilir. Kısmen çevreleyen uygulamalarda bilgisayar ekranı, büyük projeksiyon ekranları, gözlük veya kulaklık kullanılarak sınırlı sanal deneyim yaşanabilir. Uçuş, sürüş simülatörleri ile eğitim ve askeri simülasyonlar buna örnek olarak verilebilir. Tamamen çevreleyen sanal geçerlik uygulamaları ise görme, ses, dokunma ve koku hissi gibi deneyimler içerebilir. Bunun için VR başlıklarına ve kontrol cihazlarına ek olarak, hareket izleyicileri, haptik cihazlar ve özel kontrol cihazları kullanılır.



Şekil 2.3: Sanal Gerçeklik kategorileri.

3. YÖNTEM

Yapılan tez çalışmasında gerçek dünyada ve sanal ortamda kullanılan manipülatör (robot kol) 5 serbestlik derecesine (5DOF) sahiptir. Ancak bu kısımda öncelikle manipülatörün ana bileşenlerini oluşturan 3 serbestlik derecesi (3DOF) ele alınarak yapılan çalışmalar, gerekli hesaplamalar, bulgular anlatılacaktır.

3.1 KULLANILAN MALZEMELER

3.1.1 Manipülatör Bileşenleri

Şekil 3.1'de gösterildiği gibi tez çalışmasında Arduino Due ve Arduino Mega geliştirme kitlerinin ikisi de denenerek performansları ölçülmüştür. Arduino Mega AtMega2560, Arduino Due ise Atmel SAM3X8E ARM Cortex-M3 mikrodenetleyiciye sahiptir. Arduino Mega'nın saat hızı 16 MHz iken Arduino Due'nin saat hızının 84MHZ olması işlem hızında önemli farklılık oluşturuyor. Ayrıca hassas motor kontrolü açısından Arduino Due'de yüksek çözünürlüklü Analog-Dijital Dönüştürücü (ADC) olması, daha performanslı hale getiriyor. Yine Arduino Due'de daha hızlı işlemci ve daha büyük bellek olması karmaşık algoritmalarda kullanım kolaylığı sağladığı gözlemlenmiştir. Arduino Mega geliştirme kitinin kullanılması amaca hizmet etse de, performans açısından Arduino Due ile daha verimli sonuçlar elde edilmiştir.

Bu çalışmada ana işlevleri yerine getirmesi için 3 adet MG996R servo motor modeli kullanılmıştır. 9-10 kg.cm tork aralığında olması ile düşünülen ağır yüklerin taşınabilmesi için bu motor modeli seçilmiştir. Ayrıca ağır yük taşıması beklenmeyen, temel işlevi yerine getirebilecek 2 adet SG90 mini servo motor modeli de kullanılmıştır.

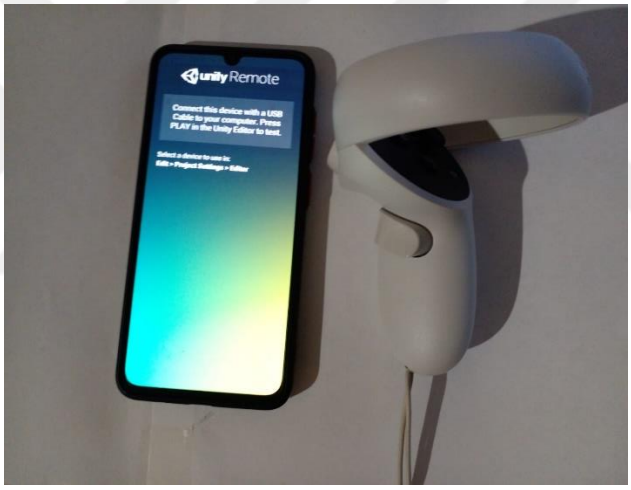
Şekil 3.1: Arduino Mega ve Arduino Due.

3.1.2 Manipülatör Kontrol Araçları

Şekil 3.2'de görüldüğü gibi sanal ortamdaki robot kol hareketleri için Sanal Gerçeklik Konsolları ve mobil ivmeölçer kullanılarak test edilmiştir. Sanal Gerçeklik konsolları dışında uzaktan kontrol için ivmeölçer ve mobil telefonlarda bulunan ivmeölçer ile de deneme çalışmaları yapılmıştır. Buradaki amaç, yaygın olarak kullanılan telefonların bu çalışmada sanal gerçeklik konsolları gibi bir işlevinin olup olmayacağını gözlemleyebilmektir.

Mobil ivmeölçer yöntemi ile robot kolu X, Y, Z eksenlerindeki hareket ettirmek istediğimizde ivmeölçere eğim vermek gerektiğinden dolayı kullanıcı kolaylığı sağlamadığı görülmüştür.

Sanal Gerçeklik konsolları kullanıcılara sanal ortamda etkileşim kurmalarını amaçladığı için daha kapsamlı sistemlerdir. İvmeölçerler daha basit sistemlerdir. Bu yüzden konsol kullanmak kullanıcı kolaylığı sağlamıştır.



Şekil 3.2: Mobil ivme ölçer ve Sanal gerçeklik konsolu.

3.2 KULLANILAN YAZILIM, TASARIM PROGRAMLARI VE ARAÇLAR

3.2.1 Yazılım Programları

Gerçekte oluşturulan manipülatörün motor hareketi için Arduino Mega ve Arduino Due geliştirme kitleri kullanıldığından dolayı Arduino IDE yazılım arayüzü kullanılmıştır. Manipülatör kinematik hesaplamalarının kod karşılığı yazılmıştır.

Manipülatör ileri yön kinematik hesaplamalarında, matris hesaplarında Python programlama dili kullanılmıştır. Basit arayüzü, kullanım kolaylığının olması nedeniyle tercih edilmiştir.

Sanal ortamdaki manipülatörün tasarımı için Unity programından faydalanılmıştır. Yine sanal ortamdaki manipülatörün kinematik hesaplamalarının kod karşılığı, Unity programı aracılığı ile C# program dili üzerinde yazılmıştır.

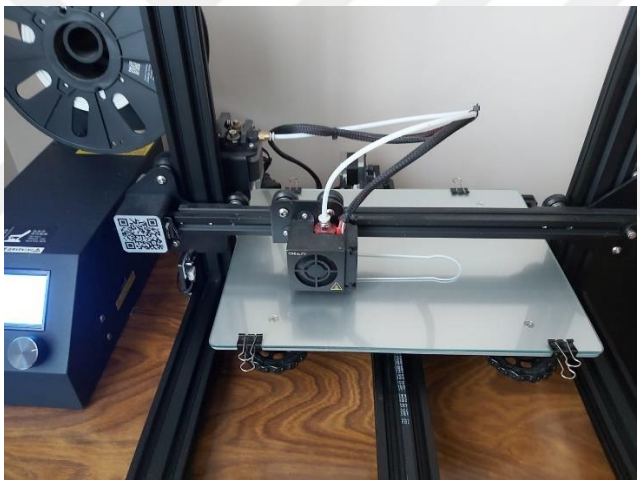
3.2.2 Tasarım Programları

Autocad Fusion 360 programı gerçek dünyada oluşturulan manipülatörün parçalarının tasarımı için kullanılmıştır. Basit arayüzü, kullanım kolaylığının olması nedeniyle tercih edilmiştir.

Sanal ortamdaki robot kolun tasarımı kapsamlı olması gerekmediğinden Unity programının sağladığı bileşenler ile oluşturulmuştur.

3.2.3 3B Printer

Şekil 3.3'te gösterildiği gibi tez çalışmasında manipülatör parçalarının basılması için Creality Mini 10 modeli kullanılmıştır.



Şekil 3.3: Kullanılan 3B Printer.

3.3 MANİPÜLATÖR KİNEMATİĞİ İÇİN UYGULANAN HESAPLAMALAR

3.3.1 İleri Kinematik

İleri kinematik problemi, robot manipülatörünün bireysel eklemleri ile aletin veya uç efektörün konumu ve yönelimi arasındaki ilişkiyle ilgilidir. Eklem değişkenleri; dönel veya rotasyonel eklemler söz konusu olduğunda bağlantılar arasındaki açılar, prizmatik veya kaydırmalı eklemler söz konusu olduğunda ise bağların uzantısıdır [23].

İleri kinematik hesaplamaları, robotik ve mekanik sistemlerin hareketlerini analiz etmek ve kontrol

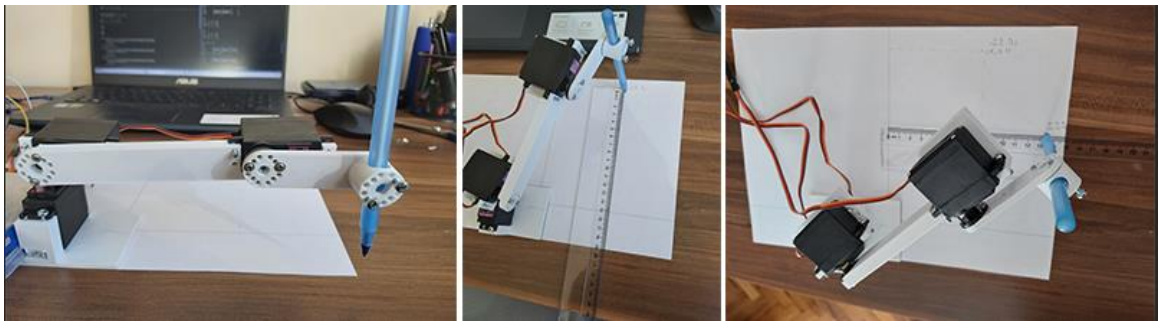
etmek için kullanılır. İleri kinematik, bir robotun veya mekanizmanın belirli bir dizi eklem açısına sahip olduğu durumlarda, uç efektörün veya belirli bir noktanın konumunu ve yönünü belirlemeye çalışır.

Denavit Hartenberg [24] parametreleri ve homojen dönüşüm matrislerinin oluşturulması ileri kinematik hesaplamaları için kullanılan yöntemlerdir.

İleri yön kinematiği [20], bir robotun eklem parametrelerinden (açılar veya yer değişimleri) hareketle, robotun uç noktasının konumunu ve yönelimini belirleme sürecidir. Bu işlem, robot kolunun her bir eklemi için bilinen açısal veya doğrusal pozisyon bilgilerine dayanarak gerçekleştirilir. İleri yön kinematiği, bir dizi matematiksel denklem ve dönüşüm matrisi kullanarak, eklem koordinatlarından kartezyen koordinatlara geçiş yapar.

Robot çalışmalarında izdüşüm hesapları, bir noktanın veya nesnenin bir çerçeveden diğerine nasıl dönüştüğünü veya yansıtıldığını belirler. Bu, özellikle farklı çerçeveler arasındaki ilişkiyi anlamak için önemlidir. Örneğin, bir robot kolunun ucundaki bir nesnenin konumunu ve yönünü belirlemek için hem taban çerçevesi hem de uç çerçevesi kullanılır. Bu çerçeveler, dönüşümler ve izdüşümler hesaplanırken referans noktaları olarak kullanılır.

Robot manipülatör uygulamalarında uç efektörün konumu ve rotasyonu (yönü) önemsiyoruz. Şekil 3.4'de gösterildiği gibi bir servo motor hareket ettiğinde uç efektör ile ilgili iki şey değişir. Uç efektör konumu değişir, uç efektör rotasyonu veya yönü değişir.



Şekil 3.4 : Uç efektör konumu ve yönü.

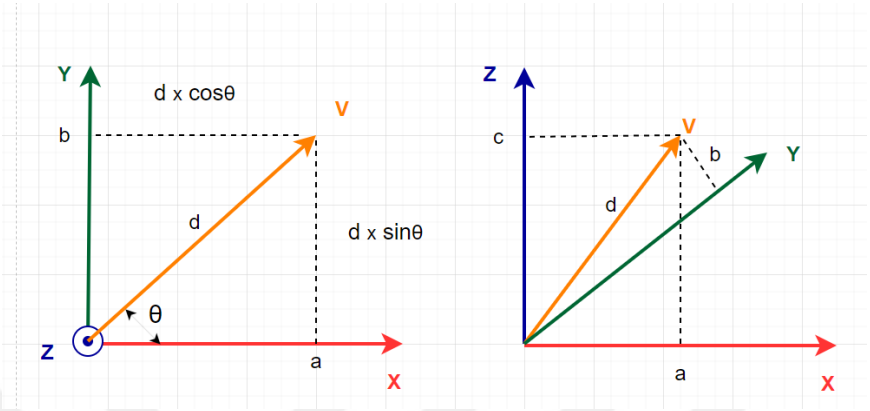
Uç efektörün pozisyonları(konumu): Yer değiştirme Vektörleri ile temsil edilir.

Uç efektörün rotasyon (yönü): Rotasyon - Dönel Matris ile temsil edilir.

3.3.2 Rotasyon Matris

Rotasyon matrisler[25], bir nesnenin belirli bir eksen etrafında döndürülmesini matematiksel olarak temsil eden matrislerdir. Üç boyutlu uzayda bir nesnenin dönme hareketini ifade etmek için kullanılırlar. Rotasyon matrisler, bir çerçevede tanımlanan bir noktanın yeni çerçevede nasıl görüneceğini belirlemek için kullanılır.

Rotasyon matrisleri anlamak için öncelikle izdüşüm kavramını anlamamız gerekiyor. Robot çalışması boyunca açı üzerinden çalışmak yerine izdüşüm fikri ile hareket edeceğiz. Çünkü üç boyutlu uzayda çok sayıda dönüşe sahip olacağız. Öncelikle cebirsel olarak X Y düzleminde V vektörünün izdüşümü θ açısı cinsinden bulalım.



Şekil 3.5: 2 Boyutlu ve 3 Boyutlu uzayda izdüşüm.

Şekil 3.5'de 2 boyutlu X Y düzlemi için Z ekseni üzerinde yani tepeden bakınca oluşan izdüşüm ve 3 boyutlu 3 boyutlu X,Y,Z uzayındaki izdüşüm verilmiştir. Buna göre elde edilen Vektör (V) izdüşümler aşağıdaki gibidir.

$$\begin{matrix} X\ ekseni \\ Y\ ekseni \end{matrix}\begin{bmatrix} a \\ b \end{bmatrix} \qquad \begin{matrix} X\ ekseni \\ Y\ ekseni \end{matrix}\begin{bmatrix} d\ x\ cos\theta \\ d\ x\ sin\theta \end{bmatrix} \tag{3.1}$$

$$\begin{matrix} X\ ekseni \\ Y\ ekseni \\ Z\ ekseni \end{matrix}\begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{3.2}$$

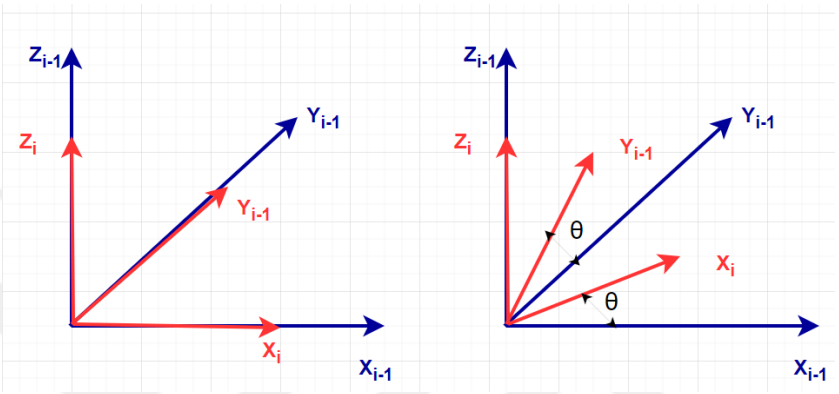
Robot çalışmalarında rotasyon matrisi, çerçevelerin birbiri ile ilişkisinden yola çıkarak iz düşüm hesabı yapmamızı sağlar. Rotasyon matrisi genel anlamda denklem 3.3'teki gibi ifade edilir.

$$R_n^m = \begin{matrix} \\ X_m \\ Y_m \\ Z_m \end{matrix}\begin{matrix} \begin{matrix} X_n & Y_n & Z_n \end{matrix} \\ \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \end{matrix} \tag{3.3}$$

Rotasyon matris görüldüğü gibi $Çerçeve_n$ 'nin $Çerçeve_m$ 'ye göre dönmesinden elde edilen X,Y,Z eksenlerindeki izdüşümünü ifade eder. R ile gösterilir. 3 x 3 matristir.

Z rotasyon matrisi, üç boyutlu uzayda bir nesneyi Z ekseni etrafında döndürmek için kullanılan bir dönüşüm matrisidir. Bir nesnenin Z ekseni etrafında saat yönünde veya saat yönünün tersine döndürülmesi gerektiğinde bu matris kullanılır. Burada (+) pozitif yön saat yönünün tersidir ve çalışmamızda bunu dikkate alarak Z rotasyon matrisi elde edilmiştir.

Robot hesaplamalarında izdüşüm kavramını anlamak için öncelikle Şekil 3.6'deki sol grafikteki gibi birbiri ile çakışan iki çerçeveyi ele alalım. $Çerçeve_i$ Z_{i-1} ekseni etrafında (+) yönde θ açısı kadar döndürülürse şeklin sağ tarafındaki gibi bir görüntü elde ederiz.



Şekil 3.6: Z ekseninde çerçevelerin izdüşümü.

Z rotasyon matrisi:

$$Rz_i^{i-1} = \begin{matrix} \\ X_{i-1} \\ Y_{i-1} \\ Z_{i-1} \end{matrix} \begin{matrix} \begin{matrix} X_i & Y_i & Z_i \end{matrix} \\ \begin{vmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix} \end{matrix} \tag{3.4}$$

X rotasyon matrisi $Çerçeve_i$ X_{i-1} ekseni etrafında (+) yönde θ açısı kadar döndürülürse elde edilir.

$$Rx_i^{i-1} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{vmatrix} \tag{3.5}$$

Y rotasyon matrisi için $Çerçeve_i$ Y_{i-1} ekseni etrafında (+) yönde θ açısı kadar döndürülürse elde edilir.

$$Ry_i^{i-1} = \begin{vmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{vmatrix} \tag{3.6}$$

3.3.3 Yer Değiştirme Vektörü

$$d_n^m = \begin{matrix} X_n^m \\ Y_n^m \\ Z_n^m \end{matrix} \begin{bmatrix} d_{11} \\ d_{21} \\ d_{31} \end{bmatrix} \tag{3.7}$$

Yer değiştirme vektörü [26] uç efektörün konumunun değişimini ifade eder. d ile gösterilir. $Çerçeve_n$ 'nin $Çerçeve_m$ merkezlerinin sırasıyla X,Y,Z eksenlerindeki uzaklıklarını belirtir.

3.3.4 Homojen Dönüşüm Matrisi

$$\text{T=H} = \begin{bmatrix} \begin{bmatrix} . & R & . \\ . & 3x3 & . \end{bmatrix} \begin{bmatrix} d \\ 3x1 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} \end{bmatrix}$$

Homojen dönüşüm matrisi [26], üç boyutlu uzayda bir nesnenin konumunu ve yönelimini tek bir çerçevede ifade eden ve bu nesnenin bir çerçeveden diğerine dönüşümünü sağlayan 4x4 boyutunda bir matristir. Bu matrisler, hem dönme (rotasyon) hem de yer değiştirme (translasyon) bilgilerini içerir. T veya H ile gösterilir.

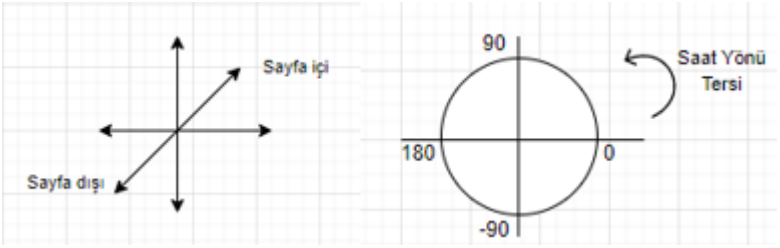
3.3.5 Denavit Hartenberg Kuralı

Denavit ve Hartenberg, iki eklem arasındaki genel dönüşümün dört parametre gerektirdiğini gösterir. Denavit-Hartenberg (DH) parametreleri olarak bilinen bu parametreler, robot kinematiğini tanımlamak için standart haline gelmiştir [23]. Endüstride genellikle Denavit Hartenberg [24] yöntemi ile homojen dönüşüm matrisleri elde edilir.

Öncelikli kurallar:

Eklem sayısından daha fazla çerçeve sayısı olmalıdır. Eklem, bağ, çerçeve numaralandırmaları buna göre yapılır. Örneğin n adet eklem olan bir kolda, çerçeve n+1 tanedir.

X, Y, Z eksenleri birbirine dik ve sağ el kuralına göre belirlenir. Eksenlerin yönleri Şekil 3.7'deki gibi çizilebilir:



Şekil 3.7: X,Y,Z eksen yönleri ve motor dönme yönü.

Şekilden de anlaşılacağı üzere ileri yön kinematiği için, homojen dönüşüm matrisi için motorun kalibrasyonda belirlenen dönme açı aralıklarının 0-180 veya -90 +90 aralığında seçilmesinin bir önemi yoktur. Motor dönme yönleri saat yönünün tersine olacak şekilde hareket edilir.

Denavit-Hartenberg temel kuralları:

Z ekseni döner mafsal için dönme ekseni veya prizmatik mafsal için hareket yönü olmalıdır.

X ekseni, kendisinden önceki çerçevenin Z eksenine dik olmalıdır.

X ekseni, ondan önce gelen çerçevenin Z eksenini kesmelidir.

Y ekseni, çerçeve içerisinde sağ el kuralına uygun şekilde çizilmelidir.

Tutucu, yani uç efektör bir önceki çerçevenin X, Y, Z koordinatlarıyla aynı olmalıdır.

Denavit – Hartenberg Parametre Tablosu:

Satır sayısı manipülatör çerçeve sayısının 1 eksiği kadardır.

Sütunlardaki 4 parametre iki çerçeve arasındaki dönmeyi ve yer değiştirmeyi ifade eder. θ, α dönme parametleridir. r, d yer değiştirme parametreleridir.

θ: X_{n-1} ve X_n eşleşmesi için Z_{n-1} etrafındaki dönmedir.

α: Z_{n-1} ve Z_n eşleşmesi için X_n etrafındaki dönmedir.

r: $Çerçeve_{n-1}$ merkezi ve $Çerçeve_n$ merkezi arasındaki X_n yönü-doğrultusu boyunca mesafedir.

d: $Çerçeve_{n-1}$ merkezi ve $Çerçeve_n$ merkezi arasındaki Z_{n-1} yönü-doğrultusu boyunca mesafedir.

Denavit Hartenberg methodunu homojen dönüşüm matrisi içerisinde gösterimi aşağıdaki gibidir:

$$T_n^{n-1} = \begin{bmatrix} cos\theta_n & -sin\theta_n cos\alpha_n & sin\theta_n sin\alpha_n & r_n cos\theta_n \\ sin\theta_n & cos\theta_n cos\alpha_n & -cos\theta_n sin\alpha_n & r_n sin\theta_n \\ 0 & sin\alpha_n & cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.8)$$

Burada ilk 3x 3 matris rotasyon matrisi, son 1x3 kısmı yer değiştirme vektörünü verir. Yer değiştirme vektörünün olduğu kısım uç efektörün sırasıyla X, Y, Z eksenlerindeki konumunu bize gösterir.

3.3.6 Ters Kinematik

Robotik sistemlerde, genellikle manipülatörün uç noktasının (uç efektör) belirli bir konum ve yönelimde olmasını istediğimiz durumlar vardır. Ters kinematik [19], bu hedef konum ve yönelimi sağlamak için robotun eklemlerinin açısal pozisyonlarını hesaplar [1].

Bir robotun eklem açıları verildiğinde, uç noktasını ve yönünü belirleyen probleme ileri kinematik problemi; robotun verilen bir uç nokta konfigürasyonu için eklem açılarını belirleyen problem ise ters kinematik [19] problemi olarak ifade edilmektedir [20].

Tutucunun merkezinin hedeflenen X,Y,Z konumuna gitmesi için açıların alması gereken değerleri hesaplar. Bunun için kullanılan modele göre geometrik hesaplamalar yapılır ve ters kinematik ile elde edilen denklemler her model için farklılık gösterir. Hareketi sağlayan motorların konumları, bu eklemleri birbirine bağlayan aksların boyutları, manipülatörün fiziksel ulaşabileceği maksimum ve minimum değerlerin belirlenmesi gibi parametreler dikkate alınır.

3.4 GERÇEK DÜNYADAKİ 3DOF MANİPÜLATÖRÜN KALİBRASYONU

İleri veya ters kinematik kullanmadan önce manipülatöre mekanik açıdan zarar gelmemesi için kalibrasyon yapılması önem taşımaktadır. Manipülatör ters kinematik veya ileri kinematik yöntemi ile hareket ettirilmeden önce uç efektör hedef konumları, motorların maksimum ve minumum açı değerleri belirlendi. Bu bölümde manipülatör çalışma alanının belirlenmesi ile ilgili gerekli test ve kalibrasyon çalışmalarına yer verilmiştir. Hedef çalışma öncesinde belirlenen bu sınırlar ana kod çalışmasına eklenmiştir.

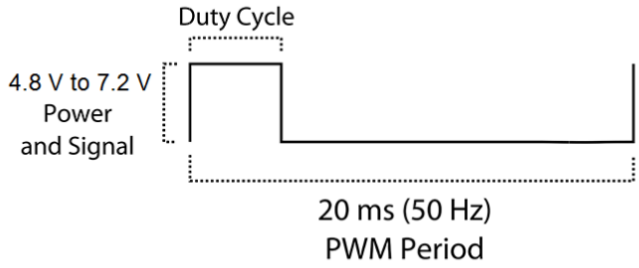
3.4.1 Servo Motor PWM Değerinin Hesaplanması

Servo motorlar, genellikle 1 ms (genellikle 0) ile 2 ms (genellikle 180°) arasında değişen PWM sinyalleriyle kontrol edilirler. Bu süre, motorun şaftının pozisyonunu belirler. Şekil 3.8'de de görüldüğü gibi kullandığımız MG996R Servo motorun veri sayfasını incelediğimizde:

PWM periyodu: 20ms. Yani her 20ms'de sinyal tekrarlanır.

PWM frekansı: 50Hz. 1 Saniyede 50 PWM sinyal döngüsü olduğunu gösterir.

Frekans - Periyot ilişkisi: $$\text{Frekans} = \frac{1}{\text{Periyot}} \quad (3.9)$$



Şekil 3.8: MG996R servo motor veri sayfası(datasheet).

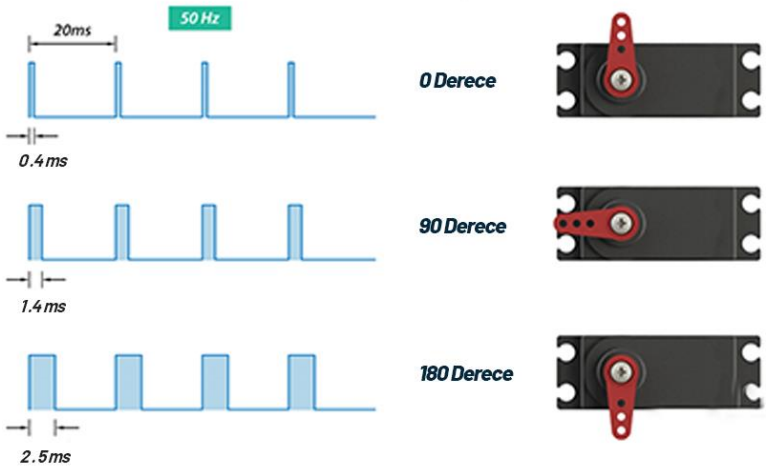
Kullandığımız servo motorların PWM darbe genişlikleri genellikle 1 ms süreli darbeler 0 derece konumuna, 1,5 ms süreli 90 dereceye ve 2 ms süreli 180 dereceye karşılık gelir.

Ancak her servo motorun minimum ve maksimum PWM değerleri, üreticiye ve kalibrasyona bağlı olarak biraz farklı olabilir. Darbelerin minimum ve maksimum süreleri bazen farklı markalara göre değişebilmekle birlikte 0 derece için 0,5 ms, 180 derece için 2,5 ms olabilmektedir.

Kullandığımız servo motorun darbe sürelerini bulabilmek için yapılan testlerin sonucunda aşağıdaki gibi sonuç elde edilmiş olup, bu sonuç ayrıca Şekil 3.9'da gösterilmiştir:

Minimum darbe genişliği: 0.4 ms

Maksimum darbe genişliği: 2.5 ms

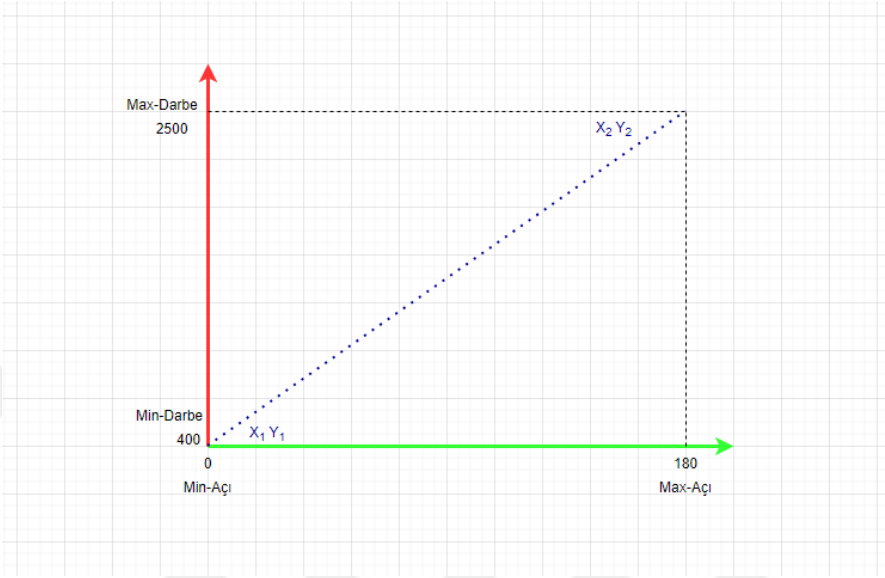


Şekil 3.9: Açıların mikro saniye cinsinden darbe karşılıkları.

3.4.2 Kalibrasyon Kodlarının Oluşturulması

Servo motorlarda, hareketi daha doğru ve hassas şekilde sağlayabilmek için açı yerine mikro saniye cinsinden değer verilmesi önem arz etmektedir. Çünkü açı yerine mikro saniye cinsinden değerlerin

koda işlenmesi, servo motorların veri sayfalarında (datasheet) yer alan görev döngüsü (duty cycle) değerleri ile uyumu kolaylaştırır. Şekil 3.10'da Açıların mikro saniye cinsinden darbe karşılıklarını bir x-y koordinat sistemi üzerinde cebirsel olarak ifadesi yer alıyor. X ekseni maksimum ve minimum açı değerlerini, Y ekseni bu açılara karşılık gelen maksimum ve minimum darbe genişliklerini gösteriyor.



Şekil 3.10: Açı-Darbe ilişkisi.

Buna göre aşağıda elde ettiğimiz formülü koda işlediğimizde servo motorlar belirli açılara karşılık gelen PWM sinyallerini hesaplayacak ve bu sinyallere göre hareket etmesi sağlanacak.

$$Y - Y_1 = \frac{Y_2 - Y_1}{X_2 - X_1}(X - X_1) \qquad (3.10)$$

X1: motorun alacağı minimum açı

X2: motorun alacağı maksimum açı

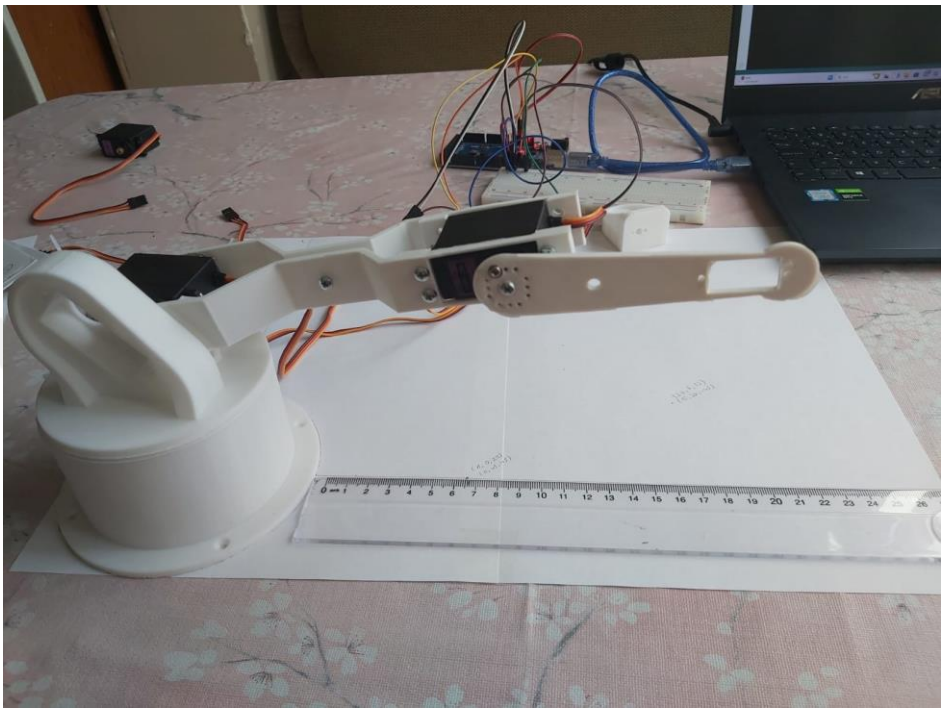
Y1: mikro saniye cinsinden minimum darbe karşılığı

Y2: mikro saniye cinsinden maksimum darbe karşılığı

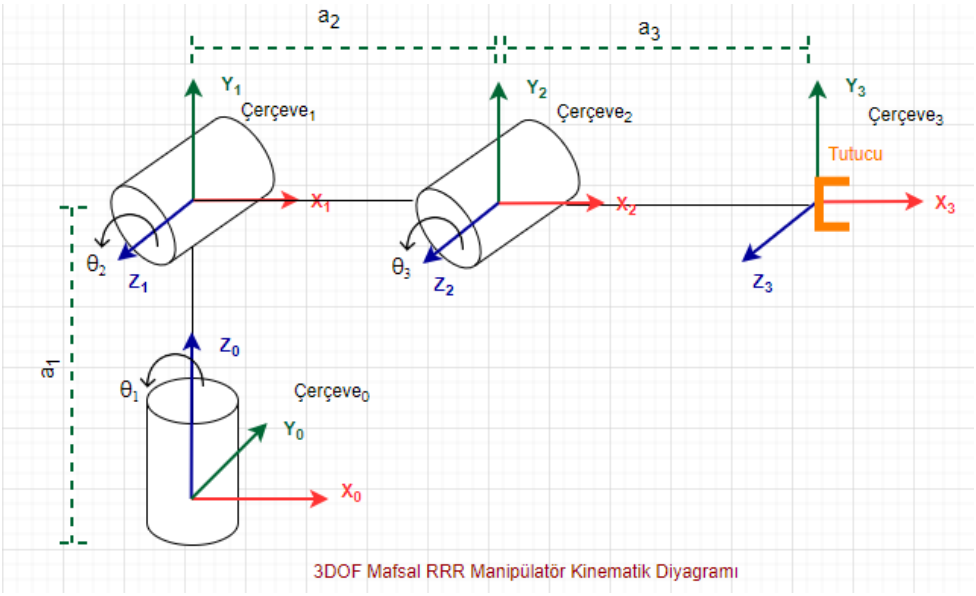
Ek 1'de Arduino kalibrasyon kodlarının bir kısmı yani cebirsel olarak oluşturduğumuz denklemin kod karşılığı yer alıyor. Kodu inceleyecek olursak sadece 3. Servo motorda açı değerlerini minimum -90° maksimum +90° derece arasında belirledik. Çünkü çalışma alanımıza göre 3. Eklem bu açılar arasında hareket etmeli. Bu durum eklem açıları (0,0,0) derece konumunda olduğunda kinematik diyagramlarını etkilemeyecek.

3.5 GERÇEK DÜNYADAKİ 3DOF MANİPÜLATÖRÜN İLERİ KİNEMATİĞİ

Bu tez çalışmasında ele alınan robot manipülatör 5 serbestlik derecesine (5DOF) sahiptir. Ancak bu kısımda öncelikle manipülatörün ana bileşenlerini oluşturan 3 serbestlik derecesi (3DOF) ele alınarak yapılan çalışmalar, gerekli hesaplamalar, bulgular anlatılacaktır.



Şekil 3.11: 3DOF Manipülatör $\boldsymbol{\theta_1}$=0, $\boldsymbol{\theta_2}$=0, $\boldsymbol{\theta_3}$=0 açılarındaki görünümü**.**



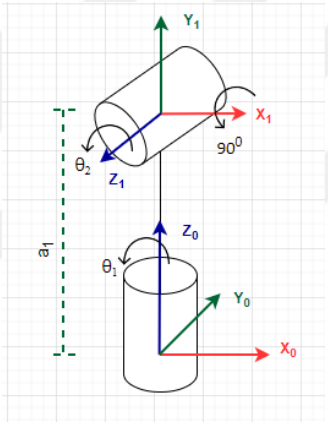
Şekil 3.12: 3DOF Manipülatör Kinematik Diyagram.

Şekil 3.11'da, oluşturulan 3DOF manipülatörün θ_1=0, θ_2=0, θ_3=0 açılarındaki konumunun görüntüsü verilmiştir. İleri kinematik hesaplamaları ile manipülatörün X,Y,Z koordinat sisteminde konumları belirlenip, çalışma alanı içerisinde olup olmadığı denetlendi. İleri kinematik matris hesaplamaları için Python programlama dili kullanıldı. Program kodları Ek 2'de verildiği gibidir.

Şekil 3.12'da oluşturulan 3DOF manipülatörün kinematik diyagramı yer alıyor. Kinematik diyagram [23], eklem açı değerleri 0° olduğunda, bağ ve eklemlerin nasıl bağlantı kurduğunu bize gösterir. Kinematik diyagram [23] tüm eklem ve bağların birbirlerine nasıl bağlandığını gösterir.

3.5.1 Homojen Dönüşüm Matrisinin Bulunması

Homojen Dönüşüm Matrislerinin bulunması için manipülatörün kinematik diyagramda gösterildiği gibi çerçeveler arasındaki ilişkiler incelendi. Bunun için rotasyon matris ve yer değiştirme vektörleri hesaplamaları kullanıldı.



Şekil 3.13: Ç$\boldsymbol{erçeve_0}$ Ç$\boldsymbol{erçeve_1}$ arasındaki kinematik diyagram.

Şekil 3.13'teki kinematik diyagrama göre Ç$erçeve_0$ Ç$erçeve_1$ arasındaki rotasyon matris, yer değiştirme vektörünü bularak homojen dönüşüm matrisini elde edelim:

$$\text{Genel Z rotasyon matris: } R_z = \begin{vmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix} \tag{3.11}$$

Rotasyon matris kinematik diyagrama göre Ç$erçeve_1$ 'in Ç$erçeve_0$ konumuna gelmesi için X_1 ekseninde 90^0 döndürülmesi gerekir. Buna göre R_1^0 aşağıdaki gibi matris hesabı ile bulunur:

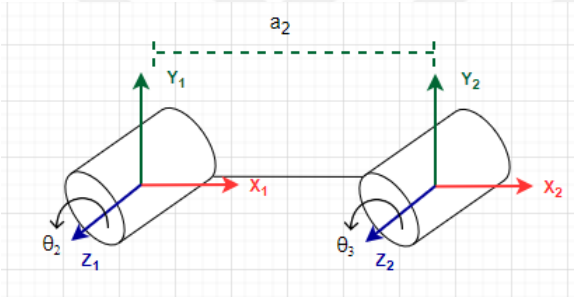
$$R_1^0 = \begin{bmatrix} cos\theta_1 & -sin\theta_1 & 0 \\ sin\theta_1 & cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos90 & -sin90 \\ 0 & sin90 & cos90 \end{bmatrix} \tag{3.12}$$

Yer değiştirme vektörü ise $Çerçeve_1$ merkezi ile $Çerçeve_0$ merkezinin $X_0, Y_{0,} Z_0$ yönündeki mesafelerini ifade eder. Buna göre d_1^0 aşağıdaki gibi bulunur:

$$d_1^0 = \begin{bmatrix} 0 \\ 0 \\ a_1 \end{bmatrix} \tag{3.13}$$

Homojen dönüşüm matrisi:

$$T_1^0 = \begin{bmatrix} cos\theta_1 & 0 & sin\theta_1 & 0 \\ sin\theta_1 & 0 & -cos\theta_1 & 0 \\ 0 & 1 & 0 & a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$



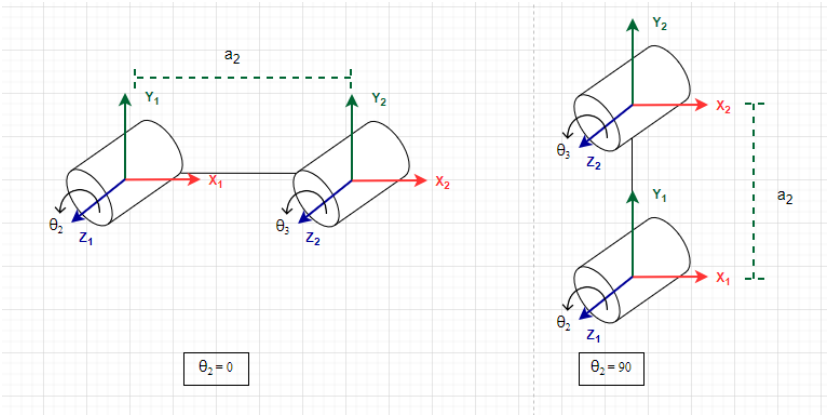
Şekil 3.14: $Çerçeve_1$ $Çerçeve_2$ arasındaki kinematik diyagram.

Şekil 3.14'teki kinematik diyagrama göre $Çerçeve_1$ $Çerçeve_2$ arasındaki rotasyon matris, yer değiştirme vektörünü bularak homojen dönüşüm matrisini elde edelim.

Rotasyon matris kinematik diyagrama göre $Çerçeve_1$ $Çerçeve_2$ arasındaki X, Y, Z yönleri aynı olduğu için herhangi bir dönme meydana gelmez. Bu durum Birim Matris ile ifade edilir. Buna göre R_2^1 aşağıdaki gibi matris hesabı ile bulunur:

$$R_2^1 = \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 \\ sin\theta_2 & cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

Yer değiştirme vektöründe ise $\theta_2 = 0$ ve $\theta_2 = 90^0$ iken Şekil 3.15'teki gibi iki farklı durum ortaya çıkar. Yani a_2 uzunluğunun ekseni değişir.



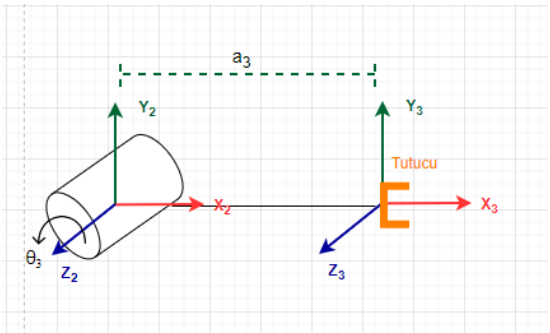
Şekil 3.15: $\boldsymbol{\theta_2 = 0}$ ve $\boldsymbol{\theta_2 = 90^0}$ iken kinematik diyagram

Bu yüzden 0<θ_2<90 aralığında düşünürsek, X ve Y eksenleri üzerinde izdüşüm mesafelerinin yer aldığı görülebilir. Yer değiştirme vektörü d_2^1 aşağıdaki gibi bulunur:

$$d_2^1 = \begin{bmatrix} a_2 cos\theta_2 \\ a_2 sin\theta_2 \\ 0 \end{bmatrix} \tag{3.16}$$

Homojen dönüşüm matrisi:

$$T_2^1 = \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 & a_2 cos\theta_2 \\ sin\theta_2 & cos\theta_2 & 0 & a_2 sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$



Şekil 3.16: Ç$\boldsymbol{erçeve_2}$ Ç$\boldsymbol{erçeve_3}$ arasındaki kinematik diyagram

Şekil 3.16'daki kinematik diyagrama göre Ç$erçeve_2$ Ç$erçeve_3$ arasındaki rotasyon matris, yer değiştirme vektörünü bularak homojen dönüşüm matrisini elde edelim.

Rotasyon matris kinematik diyagrama göre Ç$erçeve_2$ Ç$erçeve_3$ arasındaki X, Y, Z yönleri aynı olduğu için herhangi bir dönme meydana gelmez. Bu durumda Birim Matris ile ifade edilir Buna göre R_3^2 aşağıdaki gibi matris hesabı ile bulunur:

$$R_3^2 = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 \\ sin\theta_3 & cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.18)$$

Yer değiştirme vektöründe ise $\theta_3 = 0$ ve $\theta_3 = 90^0$ iken Şekil 3.13'teki ne benzer şekilde iki farklı durum ortaya çıkar. Yani a_3 uzunluğunun ekseni değişir.

Bu yüzden 0<θ_3<90 aralığında düşünürsek, X ve Y eksenleri üzerinde izdüşüm mesafelerinin yer aldığı görülebilir. Yer değiştirme vektörü d_3^2 aşağıdaki gibi bulunur:

$$d_3^2 = \begin{bmatrix} a_3 cos\theta_3 \\ a_3 sin\theta_3 \\ 0 \end{bmatrix} \quad (3.19)$$

Homojen dönüşüm matrisi:

$$T_3^2 = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 & a_3 cos\theta_3 \\ sin\theta_3 & cos\theta_3 & 0 & a_3 sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.20)$$

Bunun sonucunda 3DOF manipülatörün homojen dönüşüm matrisi T_3^0 aşağıdaki gibi elde edilir.

$$T_3^0 = T_1^0 \, T_2^1 \, T_3^2 \quad (3.21)$$

3.5.2 Denavit Hartenberg Yöntemi ile HDM'nin Bulunması

Denavit-Hartenberg[24] yönteminin uygulanması için kinematik diyagramdaki çerçeveler arası ilişki ele alınır. 3DOF manipülatörün kinematik diyagramı daha önce Şekil 3.12'de gösterilmişti.

Çerçeveler arasındaki ilişkiler incelenerek D-H tablosunu oluşturacak olursak, tablo satır sayısı çerçeve sayısının 1 eksiği kadar olmalıdır. 3DOF serbestlik derecesine sahip örnek manipülatörümüzde 4 çerçeve bulunduğu için tablo satır sayısı 3 olacaktır.

θ parametresi için genel tanıma uygun olarak tek tek ele alalım:

θ_1: X_0 ve X_1 aynı yönde. Hareketsiz durumda dönme yok. Bu yüzden 0'dır. Ancak eklem çalışmaya başladığında θ_1 kadar açısal fark oluştur. Bu durumda θ_1 = 0 + θ_1 olur.

θ_2: X_1 ve X_2 aynı yönde. Hareketsiz durumda dönme yok. Bu yüzden 0'dır. Ancak eklem çalışmaya başladığında θ_2 kadar açısal fark oluştur. Bu durumda θ_2 = 0 + θ_2 olur.

θ_3: X_2 ve X_3 aynı yönde. Hareketsiz durumda dönme yok. Bu yüzden 0'dır. Ancak eklem çalışmaya başladığında θ_3 kadar açısal fark oluştur. Bu durumda θ_3= 0 + θ_3 olur.

α parametresi için genel tanıma uygun olarak tek tek ele alalım:

α_1: Z_0 ve Z_1 eşleşmesi için $Çerçeve_0$ X_1 doğrultusunda (+)90^0 döndürülmelidir. $\alpha_1 = 90^0$

α_2: Z_1 ve Z_2 aynı yönde. $\alpha_2 = 0^0$

α_3: Z_2 ve Z_3 aynı yönde. $\alpha_3 = 0^0$

r *parametresi için genel tanıma uygun olarak tek tek ele alalım:*

r_1: $Çerçeve_0$ merkezinin $Çerçeve_1$ merkezi arasındaki X_1 doğrultusu boyunca uzunluk yoktur. $r_1 = 0$

r_2: $Çerçeve_1$ merkezinin $Çerçeve_2$ merkezi arasındaki X_2 doğrultusu boyunca uzunluk a_2 kadardır. $r_2 = a_2$

r_3: $Çerçeve_2$ merkezinin $Çerçeve_3$ merkezi arasındaki X_3 doğrultusu boyunca uzunluk a_3 kadardır. $r_3 = a_3$

d *parametresi için genel tanıma uygun olarak tek tek ele alalım:*

d_1: $Çerçeve_0$ merkezinin $Çerçeve_1$ merkezi arasındaki Z_0 doğrultusu boyunca uzunluk a_1 kadardır. $d_1 = a_1$

d_2: $Çerçeve_1$ merkezinin $Çerçeve_2$ merkezi arasındaki Z_1 doğrultusu boyunca uzunluk yoktur. $d_2 = 0$

d_3: $Çerçeve_2$ merkezinin $Çerçeve_3$ merkezi arasındaki Z_2 doğrultusu boyunca uzunluk yoktur. $d_3 = 0$

Sonuç olarak elde edilen verilere göre Denavit Hartenberg tablosunu Tablo 3.1'deki gibi elde ederiz:

Tablo 3.1: 3DOF manipülatör D - H tablosu.

n	θ_n	α_n	r_n	d_n
1	θ_1	90	0	a_1
2	θ_2	0	a_2	0
3	θ_3	0	a_3	0

Homojen Dönüşüm Matrislerini, Denavit-Hartenberg tablosuna göre elde edebiliriz.

T_1^0 için D-H tablosundaki 1. Satır ele alınır:

$$T_1^0 = \begin{bmatrix} cos\theta_1 & 0 & sin\theta_1 & 0 \\ sin\theta_1 & 0 & -cos\theta_1 & 0 \\ 0 & 1 & 0 & a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.22)$$

T_2^1 için D-H tablosundaki 2. Satır ele alınır:

$$T_2^1 = \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 & a_2cos\theta_2 \\ sin\theta_2 & cos\theta_2 & 0 & a_2sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.23}$$

T_3^2 için D-H tablosundaki 3. Satırdaki değerler ele alınır:

$$T_3^2 = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 & a_3cos\theta_3 \\ sin\theta_3 & cos\theta_3 & 0 & a_3sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.24}$$

Bunun sonucunda 3DOF manipülatörün homojen dönüşüm matrisi T_3^0 denklem 3.21'de gösterildiği gibi elde edilen matrislerin çarpımı sonucu elde edilir

Sonuçlar rotasyon matris ve yer değiştirme vektörü ile yapılan hesaplamalarla uyumlu çıkmıştır. Şekil 3.17'de Python programında homojen dönüşüm matrisi hesaplaması için yazılan kodun örnek bir sonucu yer almaktadır.

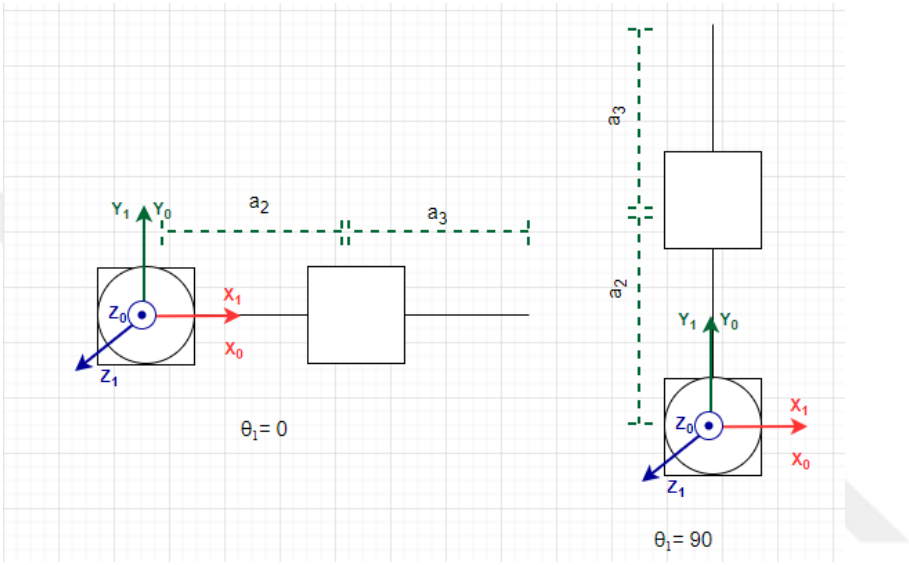
```
 Açı değerleri Theta1: 0, Theta2: 0, Theta3: 0

T0_1

[[ 1.   0.   0.   0. ]
 [ 0.   0.  -1.   0. ]
 [ 0.   1.   0.   8.9]
 [ 0.   0.   0.   1. ]]


T0_2

[[ 1.   0.   0.  16. ]
 [ 0.   0.  -1.   0. ]
 [ 0.   1.   0.   8.9]
 [ 0.   0.   0.   1. ]]


T0_3

[[ 1.   0.   0.  26. ]
 [ 0.   0.  -1.   0. ]
 [ 0.   1.   0.   8.9]
 [ 0.   0.   0.   1. ]]


Uç Efektör Pozisyon:
 [26.   0.   8.9]
```

Şekil 3.17: $\boldsymbol{\theta_1}$ = 0, **$\boldsymbol{\theta_2}$** = 0, **$\boldsymbol{\theta_3}$** = 0 açıları için Python örnek kod sonuçları.

3.6 GERÇEK DÜNYADAKİ 3DOF MANİPÜLATÖRÜN TERS KİNEMATİĞİ

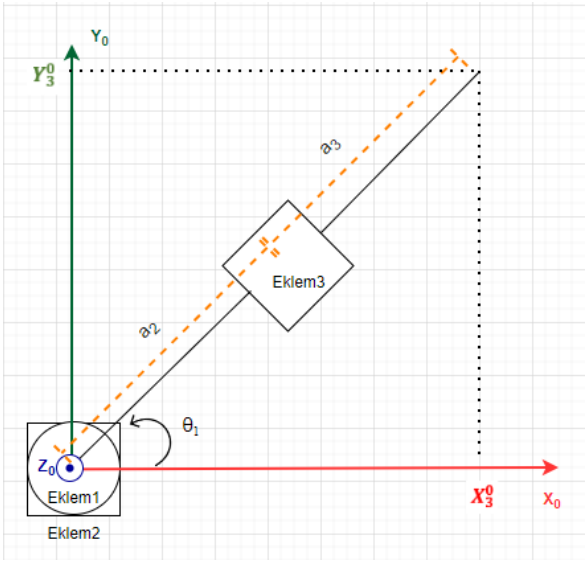
Ters kinematik hesaplaması için ileri yön kinematiğinde olduğu gibi X, Y, Z eksenlerinin ne yönde olduğunun bir önemi yoktur. Ters kinematik hesaplaması için diyagram iki şekilde çizilebilir:

Üstten görünüş – Z ekseninden yani tepeden baktığımızda görünen X Y düzlemi
Yandan görünüş – Z ekseninin dahil olduğu uzay.



Şekil 3.18: 3DOF Manipülatör üstten görünümü.

Tez çalışmasında oluşturduğumuz robot kolun θ_1 = 0 ve θ_1 = 90 iken üstten görünümü Şekil 3.18'deki gibidir. Böyle durumda θ_1 = 0 iken X ekseni üzerindeki uzunluk $a_2 + a_3$ iken, θ_1 = 90 durumunda Y ekseni üzerindeki uzunluk $a_2 + a_3$ olur. Bu durum ters kinematik hesaplamalarında yanlışlığa yol açacağından dolayı θ_1 açısı $0<\theta_1<90$ aralığında düşünülerek üstten görünüm Şekil 3.19'daki gibi çizilir:

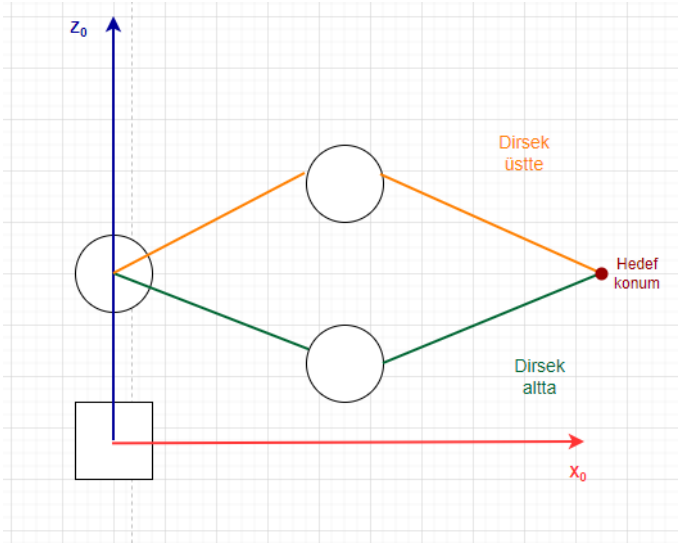


Şekil 3.19: 0<$\boldsymbol{\theta_1}$<90 aralığında 3DOF Manipülatör üstten görünümü.

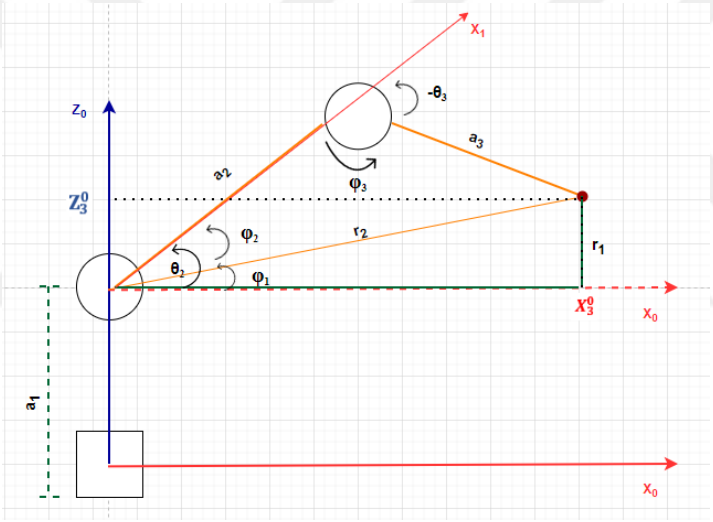
Görüldüğü gibi üstten görünümde θ_2 ve θ_3 açılarının etkisi yok. Burada sadece θ_1 açısını bulacağız:

$$\tan\theta_1 = \frac{Y_3^0}{X_3^0} \quad \theta_1 = \tan^{-1}\frac{Y_3^0}{X_3^0} \text{ olarak elde edilir.} \tag{3.25}$$

Tez çalışmasında oluşturduğumuz robot kolun hedef konuma ulaşması için yandan görünümünde şekildeki gibi iki farklı görünüm ortaya çıkar. θ_2 ve θ_3 açı değer aralıklarına göre Şekil 3.20'de iki farklı yol (üstteki ve alttaki dirsek konumları) kullanarak hedef konuma ulaşan bir robot kolu gösterilmektedir. Bu durum, robot kolunun belirli bir hedef konuma iki farklı yoldan (iki farklı eklem konfigürasyonuyla) ulaşabilmesi anlamına gelir. Bu durum, robot kolunun çalışma aralığını ve esnekliğini artıran eşsiz bir özelliktir. Dirsek Üstte durumunda, dirsek hedef konuma ulaşırken yukarıda kalır. Dirsek Altta durumunda, dirsek hedef konuma ulaşırken aşağıda kalır.



Şekil 3.20: 3DOF Manipülatör yandan görünümü.



Şekil 3.21: 3DOF Manipülatör geometrik hesabı için yandan görünümü.

Tez çalışmasında kullandığımız robot kol modeline ve çalışma alanımıza en uygun özellik dirsek üstte özelliğidir. Buna göre $0<\theta_2<90$ aralığında ve $-90<\theta_3<0$ aralığında düşünerek ters kinematik hesabı Şekil 3.21'e göre yapıldı.

Şekilde görüldüğü gibi θ_3 (-) değer alır çünkü X_1 ekseni altında kalır. Oluşturulan kinematik diyagrama göre θ_2 ve θ_3 açıları için aşağıdaki şekilde denklemler elde edilir.

$$\theta_2 = \varphi_1 + \varphi_2 \tag{3.26}$$

$$r_1 = Z_3^0 - a_1 \tag{3.27}$$

$$\tan\varphi_1 = \frac{r_1}{X_3^0} \qquad \varphi_1 = \tan^{-1}\frac{r_1}{X_3^0} \tag{3.28}$$

$$r_2^2 = r_1^2 + (X_3^0)^2 \quad r_2 = \sqrt{r_1^2 + (X_3^0)^2} \tag{3.29}$$

φ_2 ve φ_3 açı değerleri kosinüs teoremi kullanarak hesaplanır:

$$a_3^2 = a_2^2 + r_2^2 - 2a_2 r_2 \cos\varphi_2 \tag{3.30}$$

$$\varphi_2 = \cos^{-1}\frac{a_2^2 + r_2^2 - a_3^2}{2a_2 r_2} \tag{3.31}$$

$$r_2^2 = a_2^2 + a_3^2 - 2a_2 a_3 \cos\varphi_3 \tag{3.32}$$

$$\varphi_3 = \cos^{-1}\frac{a_2^2 + a_3^2 - r_2^2}{2a_2 a_3} \tag{3.33}$$

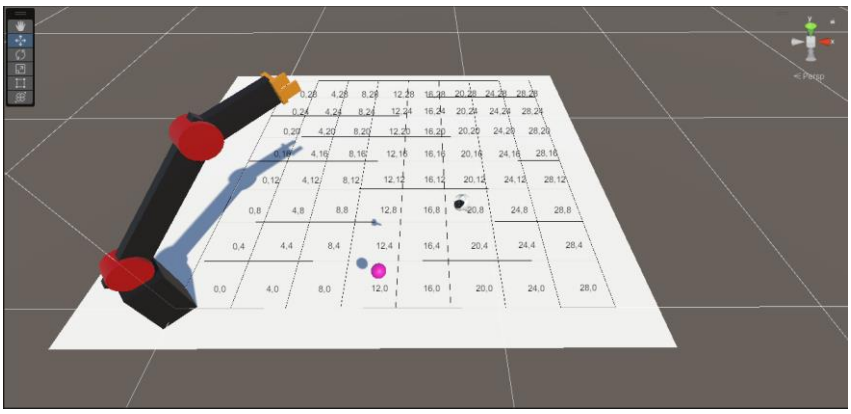
$$-\theta_3 + \varphi_3 = 180 \quad \theta_3 = \varphi_3 - 180 \tag{3.34}$$

Robot kolun istenilen (hedef) konumlara gitmek üzere θ_1, θ_2 ve θ_3 değerlerini hesaplayabilmesi için elde edilen denklemler motorların bağlı olduğu Arduino IDE program diline kod olarak eklenmiştir. Kodlara Ek 3'te yer verilmiştir.

3.7 SANAL ORTAMDAKİ 3DOF MANİPÜLATÖRÜN KİNEMATİĞİ

Şekil 3.22'de sanal ortamda oluşturulan manipülatör yer almaktadır. Gerçekte oluşturulan robot koldan farklı olarak simülasyonda oluşturulan robot kolun geometrik hesaplaması için Gradyan İniş Algoritması [27] baz alınarak gradyan eğim hesaplaması kullanıldı. Bu yöntemin seçilmesinin nedeni simülasyonda bir hedefin olması ve robot kolun bu hedefi efektif bir şekilde yakalamaya çalışmasından kaynaklıdır.

Gradyan iniş, kısıtlanmamış matematiksel optimizasyon için bir yöntemdir. Türevlenebilir çok değişkenli bir fonksiyonun yerel minimumunu bulmak için birinci dereceden yinelemeli bir algoritmadır [28].



Şekil 3.22: Sanal ortamda oluşturulan 3DOF Manipülatör.

Simülasyonda ters kinematik yönteminin kullanılmamasının sebebi, işlemleri basite indirgeyerek sanal ortamda ki robot kol hareketinin sağlanması amaçlanmıştır. Gradyan iniş algoritması ile hedef ve ona ulaşmaya çalışan robot kol uç efektör arasındaki mesafeyi en aza (minimum) indirgemek istemektir. Bu yöntemin kullanılmasının bir başka sebebi de robot manipülatör, uç efektör ve hedef nesne arasındaki mesafeyi minimum düzeyde tutmak için kendisini hareket etmeye zorlayacaktır.

Gradyan iniş algoritması, bu tür optimizasyon problemlerinde kullanılabilir. Örneğin, uç efektörün hedef pozisyona olan uzaklığını minimize etmek için eklem açılarını optimize eder.

Gradyan iniş algoritması, bu gradyan bilgilerini kullanarak her bir eklem açısını iteratif olarak günceller ve hata fonksiyonunu minimize etmeye çalışır. Bu süreç, robot kolunun belirlenen hedef pozisyona en kısa sürede ve en doğru şekilde ulaşmasını sağlar.

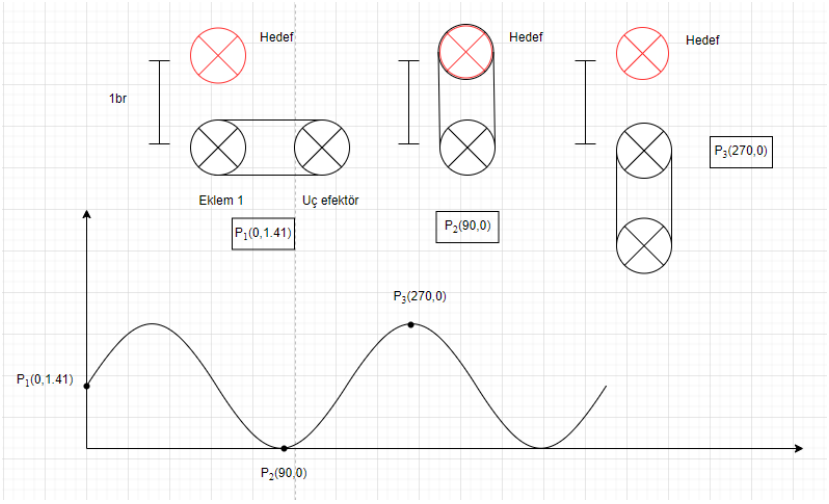
3.7.1 Gradyan Yöntemi

Normalde her bir eklem için eğim hesaplanırken pozisyon hatasının ilgili eklem açısına göre türevi alınır. Ancak türevi hesaplamak, fonksiyonun genel olarak keyfi problemler için garanti edilemeyen belirli matematiksel özellikleri karşılamasını gerektirir. Bu durum basit diyebileceğimiz bir problemi daha karmaşıklaştıracaktır ve gerçek türevine erişmek imkansızdır. Bu yüzden bu çalışmada değerin kaba tahminini verebilmek önemlidir.

Lokal hata: Her bir eklem için hedef pozisyondan sapma miktarını ifade eder.

Global hata: Tüm eklemlerin hedef pozisyondan toplam sapma miktarını ifade eder.

Gradyan hesabını daha iyi anlamak için Şekil 3.23'deki gibi bir eklem ve bir uç efektörden oluşan manipülatör ve bir hedef oluşturuldu.



Şekil 3.23: Sanal 3DOF Manipülatör gradyan hesabı için diyagram.

Gradyan hesabı için önemli notlar:

Uzunlukların birim olarak ifade edilmesinin sebebi simülasyon için kullanılan Unity programında uzunluk tanımlarının bu şekilde olmasından kaynaklıdır.

Hedef ve uç efektör aralarındaki mesafe görselde gradyan [29] hesabının hesaplamanın daha iyi anlaşılması için örnek olarak yazılmıştır.

$P_1(0,1.41)$: $Eklem_1$ açısı 0 iken uç efektörün hedefe uzaklığı 1.41 birimdir.
$P_2(90,0)$: $Eklem_1$ açısı 90° olduğunda uç efektörün hedefe uzaklığı 0 birim olacak. Bu durumda grafikte görüldüğü gibi minimum uzaklığı verir.
$P_3(270,2)$: $Eklem_1$ açısı arttıkça örneğin 270° olduğunda uç efektörün hedefe uzaklığı artacak 2 birim olacak. Bu durumda grafikte görüldüğü gibi tepe noktasına yaklaşır ve maksimum uzaklığı gösterir.

Eğim hesaplaması için açıyı artırdığımızda minimum değere yaklaştığımızı ve 90°'yi geçtiğinde tekrar mesafenin arttığı gözlemlendi. Bu yüzden 0 ve 90° açıları arasında açı değerleri için eğim hesabı yapıldı. $P_1(0,1.41)$ ve $P_2(30,1)$ noktalarından yola çıkarak eğim yaklaşımının hesaplanması denklemdeki gibidir.

Tahmini eğim (Gradyan [29]) hesabı genel formül:

$$\nabla f(p) = \frac{f(p+\Delta x) - f(p)}{\nabla x} \tag{3.35}$$

Çalışmamızda uyguladığımız eğim formülü:

$$\nabla f(\theta) = \frac{f(\theta+\epsilon) - f(\theta)}{\epsilon} \tag{3.36}$$

Gradyan iniş güncelleme adımı:

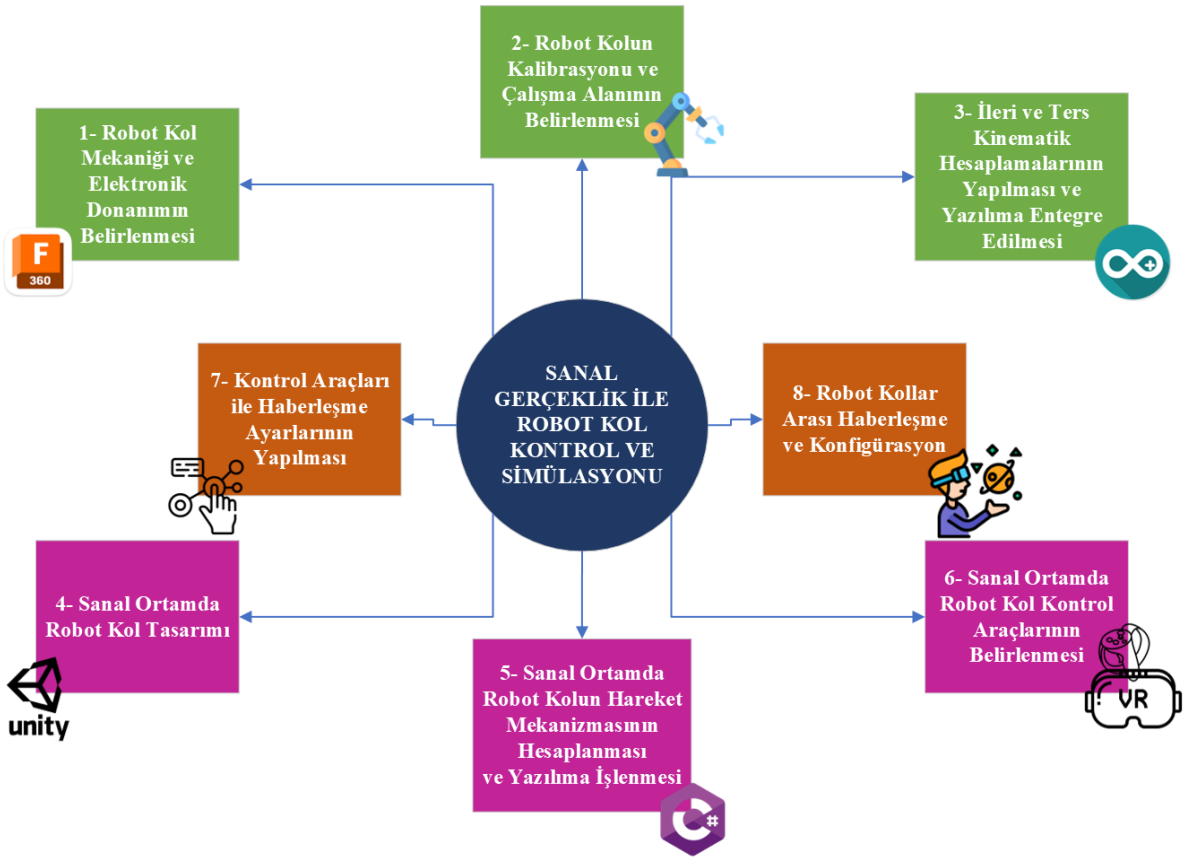
$\theta = \theta - \alpha \cdot \nabla f(\theta)$ $\quad$ $f(\theta)$: açı mesafesi $\quad$ θ: mevcut eklem açısı (3.37)

$\theta = \theta - \alpha \frac{f(\theta+\epsilon) - f(\theta)}{\epsilon}$ $\quad$ ϵ: küçük adım büyüklüğü $\quad$ α: öğrenme oranı

Bu denklemlerde kullanılan $f(\theta)$ açı mesafesini ifade eder. θ mevcut eklem açısıdır. Yazılım kısmında küçük adım büyüklüğü olarak ifade edilen ϵ değeri 0.01 olarak belirlendi. Öğrenme oranını ifade eden α değeri 0.1 olarak atandı.
Sanal ortamda oluşturulan Manipülatör Unity programındaki C# kodlarına Ek 4 'te yer verilmiştir.

4. BULGULAR

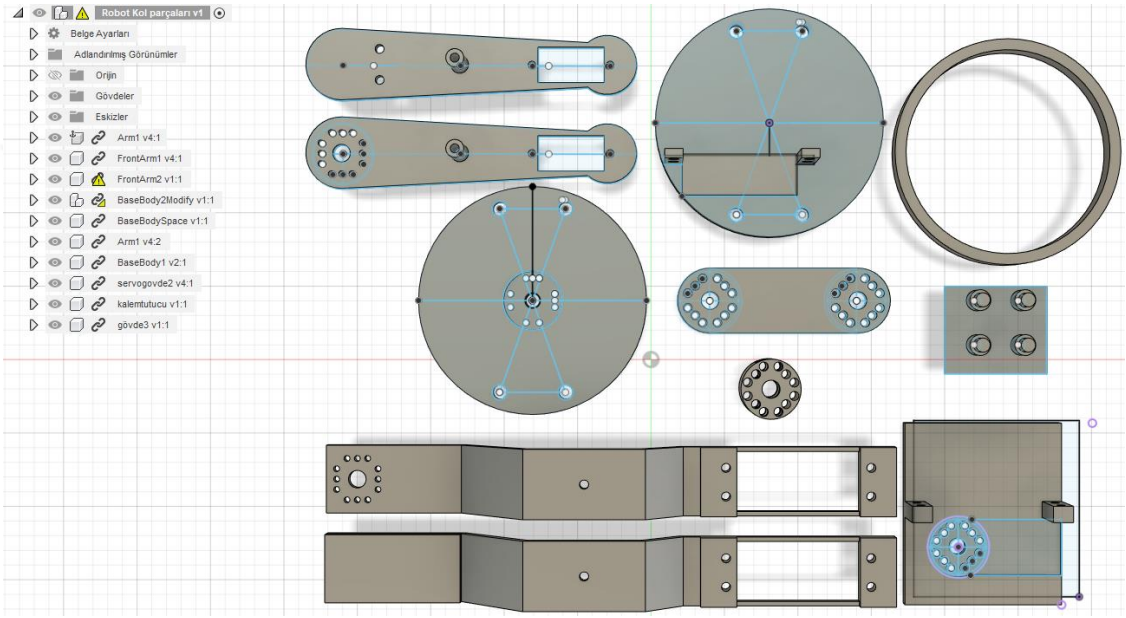
Tez çalışmasının bu kısmında öncelikle 3 serbestlik derecesine (3DOF) sahip robot manipülatör, ardından 5 serbestlik derecesine (5DOF) sahip robot manipülatör için tasarım, kullanılan programlar ve çalışma düzeneği ayrıntılı olarak gösterilmektedir. Bu çalışmada ilk olarak sanal ortamda bulunan hedef cisim sanal gerçeklik konsolları ile hareket ettirilir. Sanal ortamdaki robot manipülatör uç efektörü bu hedefe ulaştığı anda gerçek dünyadaki robot manipülatöre konum bilgileri seri haberleşme ile iletilmektedir. Gerçekte oluşturulan robot manipülatör sanal ortamdan gelen konum bilgilerine göre hareket etmektedir. Çalışma adımları Şekil 4.1'de gösterildiği gibidir.



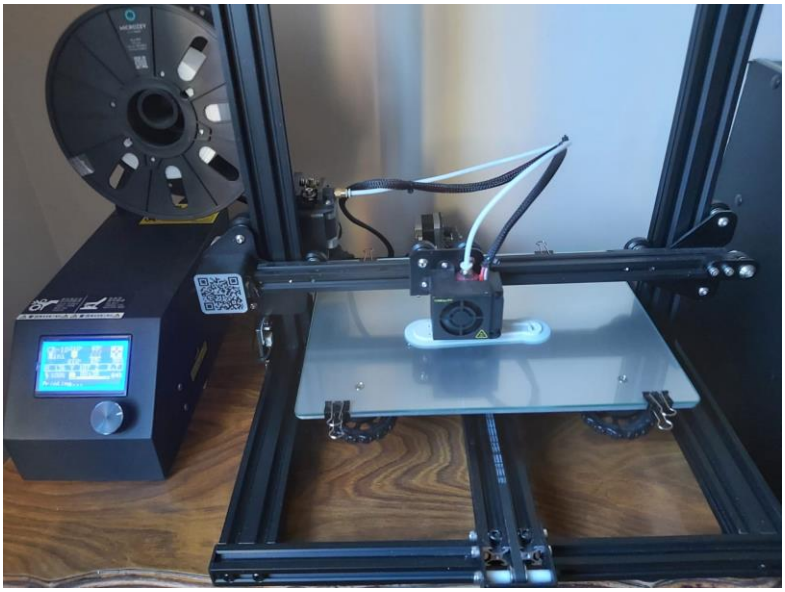
Şekil 4.1: Çalışma adımları.

4.1 GERÇEK DÜNYADA OLUŞTURULAN 3DOF ROBOT MANİPÜLATÖR KİNETMATİĞİ VE KALİBRASYONU

Gerçek dünyada oluşturulan robot kolun parçaları Autocad Fusion 360 programı ile tasarlandı ve 3 Boyutlu Yazıcı ile parçaların basımı gerçekleştirildi. Robot kol modelinin oluşturulmasında GrabCad gibi programlarda bulunan robot kol modellerinden esinlenilmiştir. Bazı parçalar HowToMechatironics adı ile hesabı bulunan cults3d web sitesinden alınmıştır [30,31]. Şekil 4.2'de Autocad Fusion 360 ile tasarlanan robot kol parçaları, Şekil 4.3'te parçaların 3 boyutlu printer ile basılması gösterilmiştir.



Şekil 4.2: Autocad Fusion 360 ile tasarlanan robot kol parçaları.



Şekil 4.3: 3B printer.

4.1.1 Robot Manipülatör Kalibrasyonu ve İleri Kinematiği

Kalibrasyon; robot kolun mekanik açıdan zorlanıp zorlanmadığını kontrol etmek, motorlar başlangıç ve bitiş açılarındayken uç noktalarının istenen doğru koordinatlara ulaşıp ulaşmadığını ölçmek, belirlenen aksların ilk ve son konumlarda doğru eksende gözlemleyebilmek açısından önemlidir. Mekanik montaja geçmeden önce yapılmalıdır. Burada motorun başlangıç ve bitiş açılarına göre uç efektörün motor başlıklarına yani eklemlere bağlanan aksların (bağların) konumlarının doğruluğu kontrol edilir. Buna göre robot kol mekaniğinin düzgün tasarlandı ve çalışma alanına göre hareketini rahat sağlayıp sağlamadığını kontrol etmek açısından kalibrasyon kodları geliştirildi. Robot kolun parçaları (bağlar) kalibrasyona göre motor (eklem) başlıklarına bağlandı. Kalibrasyon ile çalışma alanı göz önünde bulundurularak eklemlerin açı aralıkları ve bu açıların mikro saniye cinsinden karşılıkları aşağıdaki Tablo 4.1'de gösterilmiştir.

Kalibrasyon kodları Ek1'de gösterilmiştir.

Tablo 4.1: Motor açıları ve darbe karşılıkları.

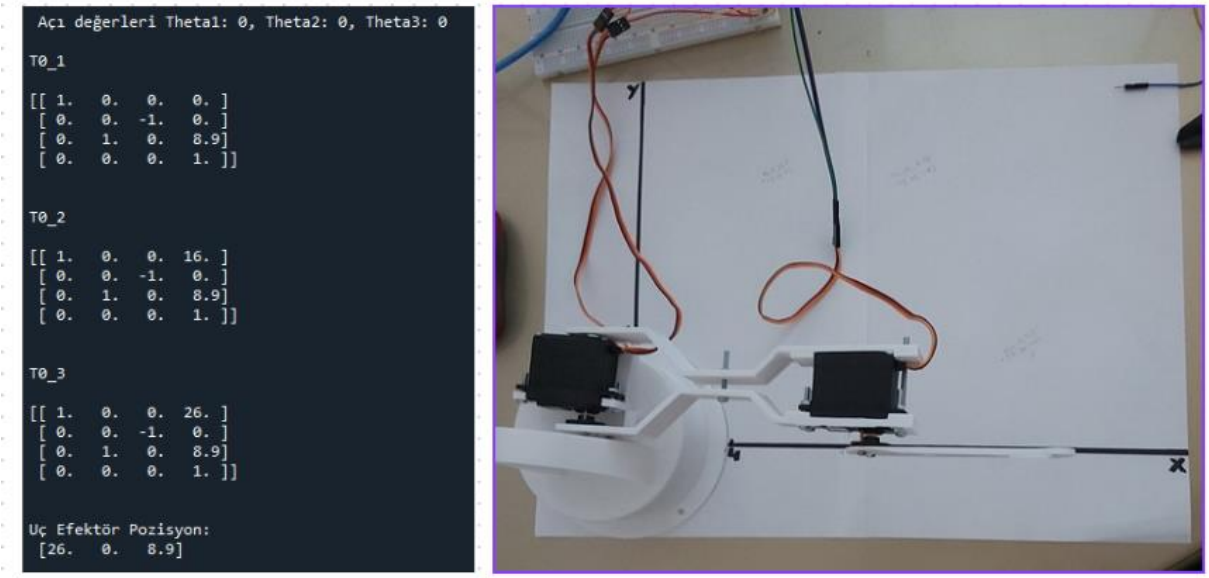
Motor - Eklem	Minimum Açı	Darbe Karşılığı	Maksimum Açı	Darbe Karşılığı
Servo Motor θ_1	0	450	180	2950
Servo Motor θ_2	0	450	180	2450
Servo Motor θ_3	-90	450	90	2450

Kalibrasyon ile gerekli ayarlamalar yapıldıktan ve mekanik montaj gerçekleştirildikten sonra robot kolun çalışma alanını belirleyebilmek için X, Y, Z eksenlerinde ulaşabildikleri maksimum minimum noktaları incelenmiştir. Bunun için İleri kinematik yöntemi kullanılmıştır. Önce Python program dilindeki matris hesabı yapılmıştır. Ardından elde edilen sonuç ile Arduino geliştirme kiti ile doğru konum bilgilerine ulaşıp ulaşmadığı kontrol edilmiştir.

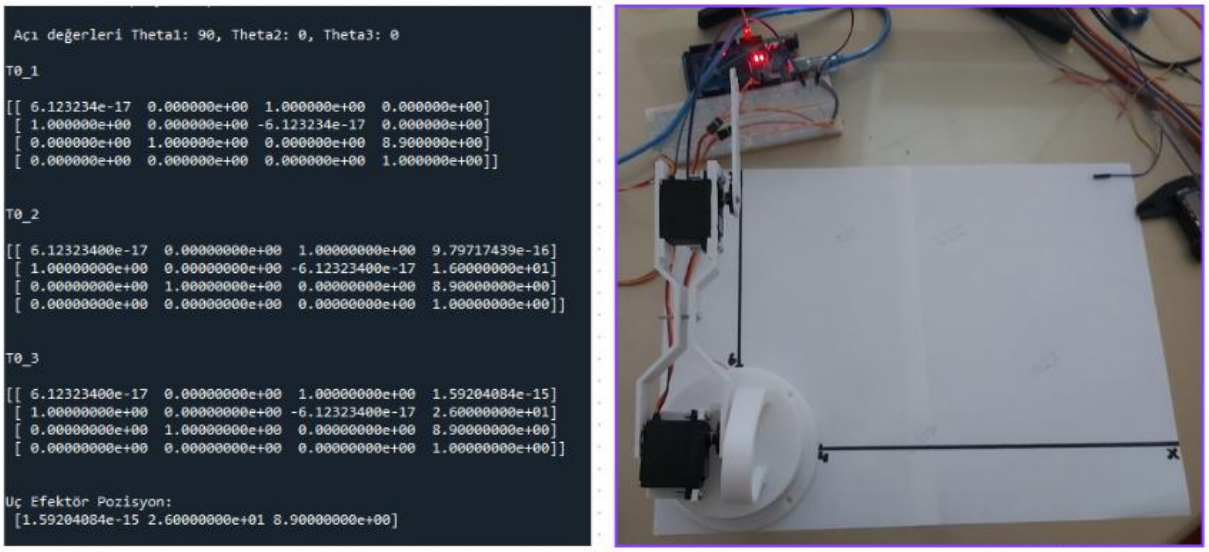
Gerçekte oluşturulan robot kol modelinin her bir aksının (bağ) boyutları a_1 a_2 a_3 boyutları ölçülmüştür. Bu ölçümler ileri kinematik hesaplamasının yer aldığı Python program dillerindeki yazılıma entegre edilmiştir. İleri kinematiğinin hesaplanması için homojen dönüşüm matrislerinden faydalanılmıştır. HDM yöntemine göre belirli açılarda uç efektörün alacağı konumlar hesaplanmış olup, gerçekte oluşturulan robot manipülatöre motorları bu açılar verildiğinde ulaşması gereken konuma gerçekten ulaşıp ulaşmadığı incelenmiştir.

HDM uç efektör pozisyon sonucuna göre, robot kolun çalışma alanının belirlenmesi için yapılan örnek çalışma Şekil 4.4, Şekil 4.5 ve Şekil 4.6'te gösterilmiştir. Şekillerden de görüleceği üzere uç efektörün

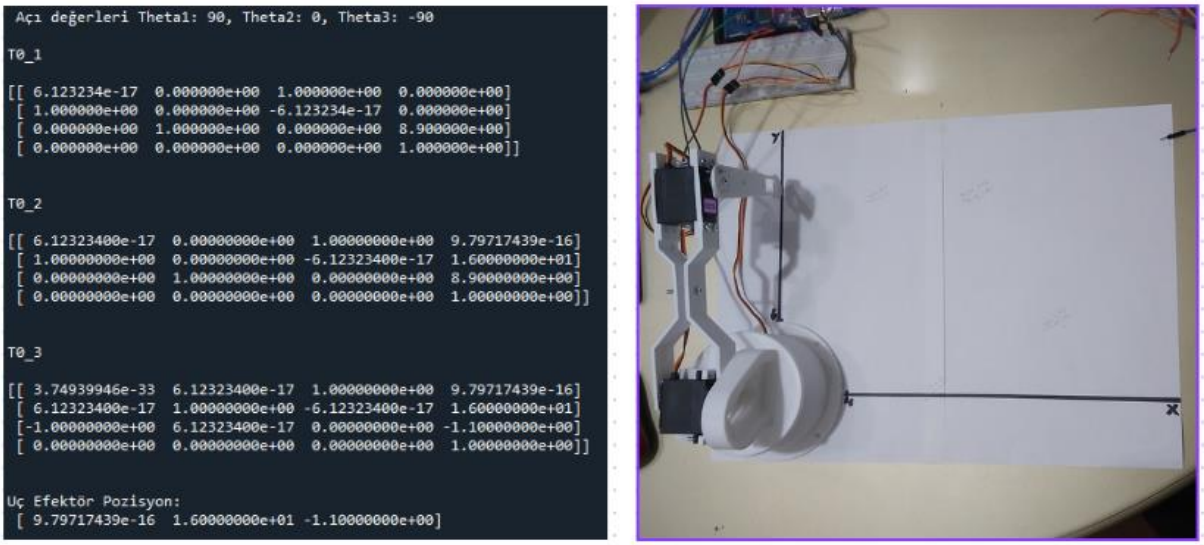
eksenlerdeki maksimum ve minimum pozisyonları belirlenmiştir. Ayrıca HDM hesaplamaları ile pratikte oluşturulan robot manipülatörün uyumlulukları gözlemlenmiştir.



Şekil 4.4: $\boldsymbol{\theta_1}$=0, $\boldsymbol{\theta_2}$=0, $\boldsymbol{\theta_3}$=0 açılarında HDM sonucu ve robot kolun pozisyonu.



Şekil 4.5: $\boldsymbol{\theta_1}$=90, $\boldsymbol{\theta_2}$=0, $\boldsymbol{\theta_3}$=0 açılarında HDM sonucu ve robot kolun pozisyonu.



Şekil 4.6: $\boldsymbol{\theta_1}$=90, $\boldsymbol{\theta_2}$=0, $\boldsymbol{\theta_3}$=-90 açılarında HDM sonucu ve robot kolun pozisyonu.

Sonuç olarak gerçekte oluşturulan robot kolun çalışma alanı, maksimum ve minimum koordinat bilgileri aşağıdaki Tablo 4.2'deki gibidir.

Tablo 4.2: Robot kol eksenlere göre minimum maksimum noktaları.

$\boldsymbol{\theta_1, \theta_2, \theta_3}$	X	Y	Z
0, 0, 0	26	0	8.9
90, 0, 0	0	26	8.9
0, 90, 0	0	0	34.05
0, 0, -90	16	0	-1.95

4.1.2 Robot Manipülatör Ters Kinematiği

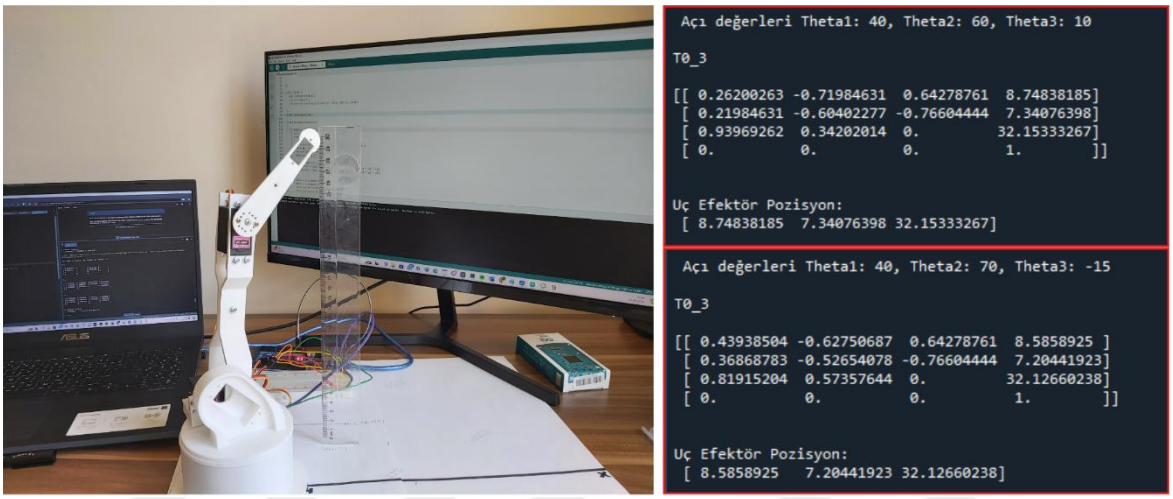
Kalibrasyon işlemi, çalışma alanının belirlenmesi ve ileri kinematik yöntemi ile başlangıç test çalışmalarının yapılmasının ardından hesaplanan ters kinematik yöntemine geçilmiştir.

Gerçek dünyada oluşturulan robot kol için öncelikle HDM ile belirli açılarda gideceği konum bilgileri alındı. İlk olarak ters kinematik hesaplamasının yer aldığı Arduino geliştirme kitinin yazılım kısmında X, Y, Z değerlerine karşılık olarak yazılmıştır. Deneme çalışmaları ile uç efektörün gideceği hedef konuma göre eklemlerin alabileceği açılar yaklaşık olarak gözle incelendi.

Burada önemli olan kısım bu çalışmanın ters kinematik hesabında $\theta_2$ve θ_3 açılarının elde edilebilmesi için dirsek üstte yöntemini baz alınmasıdır. Bu durum aslında çalışma alanımızı daha da sınırlı hale

getiriyor ve θ_3 açısının maksimum 0 minimum -90° aralığında tutmayı sağlıyor. Bu göz ardı edildiği durumda ileri kinematik ve ters kinematik test süreçlerinde uyumsuzluk gözle görülür şekilde olacaktır.

Şekil 4.7'da HDM yöntemine göre yaklaşık olarak aynı olan hedef pozisyon için robot kola farklı açılarda değer verebiliyoruz. Görüldüğü gibi oluşturduğumuz robot manipülatörün yazılım kısmında ters kinematik hesaplaması için dirsek üstte yöntemi kullanıldığı için eklem açıları olan θ_1, θ_2, θ_3 sırasıyla 40°,70°,-15° açılarda dönüş yapmıştır. Eğer ters kinematik hesaplamasında dirsek altta yöntemini uygulasaydık robot manipülatörün eklemleri 40°,60°,10° açı derecelerinde dönüş yapacaktı.



Şekil 4.7: (**40**°,70°,-15°) - (40°,60°,10°) açı derecelerinde HDM uç efektör pozisyonu ve robot hareketi.

Ters kinematik hesabı robot kol modeline, çalışma alanına, ihtiyacına göre şekillenir. Örneğin bizim buradaki çalışmamızda θ_1=0, θ_2= 90° iken sadece θ_3 değerinin değiştiği bir durumda, koda işlenen ters kinematik hesabının eksik kaldığı gözlemlenmiştir. Çünkü aslında X ekseni baz alınarak hesaplanan θ_3 açısı böyle bir durumda Y eksenine kaymıştır. Temelde oluşturulan ters kinematik hesabı bu şekilde gözlemlenen ek durum veya ihtiyaç neticesinde geliştirilmiştir.

Ters kinematik ile beraber çalışma alanı daha sınırlı hale gelmiş olup eklemlerin alabileceği maksimum minimum açı değerleri aşağıdaki Tablo 4.3'te verilmiştir.

Tablo 4.3: Motor açıları ve darbe karşılıkları.

Motor - Eklem	Minimum Açı	Darbe Karşılığı	Maksimum Açı	Darbe Karşılığı
Servo Motor θ_1	0	400	180	2500
Servo Motor θ_2	0	400	90	1450
Servo Motor θ_3	-90	400	0	1450

4.2 SANAL ORTAMDA OLUŞTURULAN 3DOF ROBOT MANİPÜLATÖR MODELİ ve KONTROLÜ

Gerçek dünyada oluşturulan robot kol mekanizması ve senaryosu ile uyumlu robot manipülatörün sanal ortamdaki tasarımına geçmeden önce Unity3D [5,6] programındaki koordinat sistemini anlamak gerekiyor. Çünkü gerçek dünyadaki ve sanal ortamdaki koordinat sistemleri farklı olduğu için veri gönderimi ve alımı sırasında doğru dönüşümlerin yapılabilmesi için önemlidir. Robot manipülatörler birbirleri ile etkileşim halinde olduklarından dolayı çalışma içerisinde gerekli dönüşümler gerçekleştirildi.

4.2.1 Sanal Ortam ve Gerçek Dünyadaki Koordinat Sistemleri

Unity'de Koordinat Sistemi (Sol El Kuralı):

Unity, sol el koordinat sistemini kullanır. Bu sistemde:

- X ekseni sağa doğru pozitif yönde uzanır.
- Y ekseni yukarı doğru pozitif yönde uzanır.
- Z ekseni ileri doğru pozitif yönde uzanır.

Bu durumda, başparmağınızı X ekseni boyunca sağa doğru, işaret parmağınızı Y ekseni boyunca yukarı doğru ve orta parmağınızı Z ekseni boyunca ileri doğru tuttuğunuzda sol el kuralını uygulamış olursunuz.

Gerçek Dünya Koordinat Sistemi (Sağ El Kuralı):

Gerçek dünyada ve birçok mühendislik uygulamasında sağ el koordinat sistemi kullanılır. Bu sistemde:

- X ekseni sağa doğru pozitif yönde uzanır.
- Y ekseni yukarı doğru pozitif yönde uzanır.
- Z ekseni size doğru pozitif yönde uzanır.

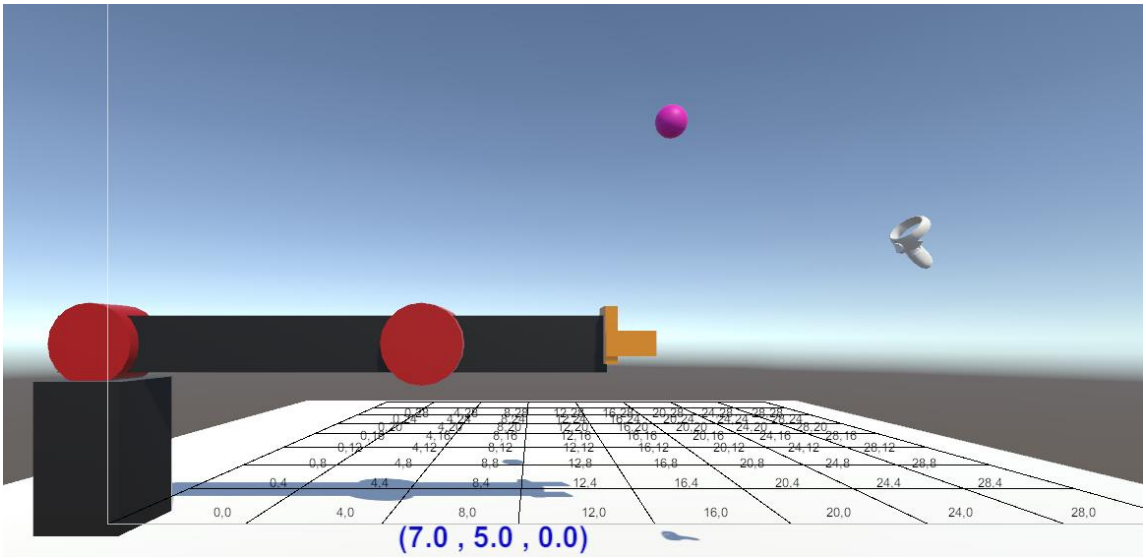
Bu durumda, başparmağınızı X ekseni boyunca sağa doğru, işaret parmağınızı Y ekseni boyunca yukarı doğru ve orta parmağınızı Z ekseni boyunca size doğru tuttuğunuzda sağ el kuralını uygulamış olursunuz.

Sonuç olarak sanal ortamdaki X, Y, Z koordinatları gerçek dünyada sırasıyla X, Z, Y şeklindedir.

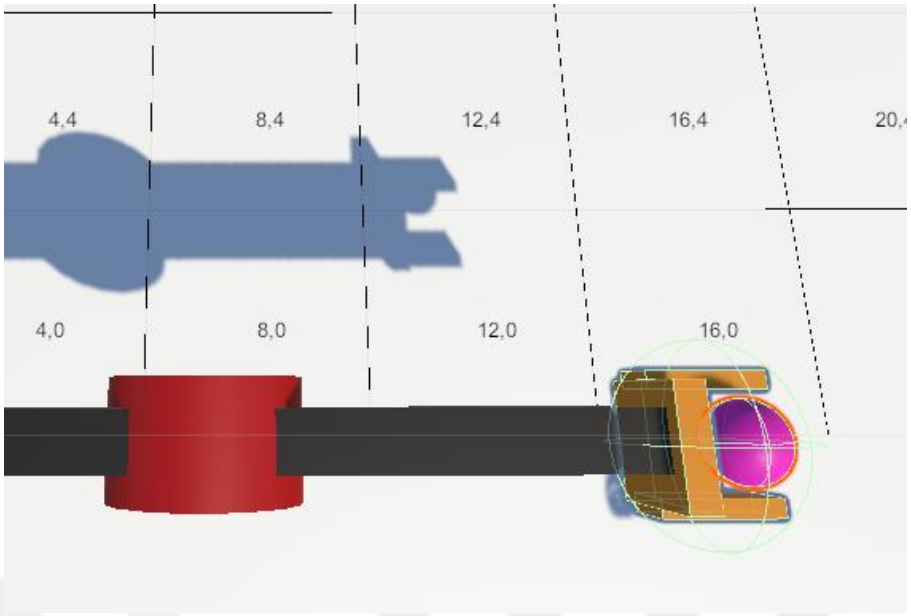
4.2.2 Sanal Ortamdaki Robot Kol Modeli

Sanal ortamdaki manipülatör için rahat hareketin sağlanması ve görüntü olarak hareketin anlaşılabilmesi için gerçek robot kol modelinin boyutlarının ¼'ü olacak şekilde belirlendi. Sanal robot kol modelinin tasarımı ve yazılımı için Unity programı kullanıldı. Boyutların gerçekteki robot kol boyutlarının ¼'ü olacak şekilde belirlenmesinin bir sebebi de kullanılan Unity programında boyutların birim – metre olmasından kaynaklıdır. Gerçek dünyada kullandığımız robot kol modelinin boyutlarını santimetre veya milimetre cinsinden hesaplıyoruz.

Şekil 4.8'de sanal ortamdaki robot manipülatörün eklemlerin θ_1=0, θ_2=0, θ_3=0 derece olduğu durumdaki konumunu göstermektedir. Gerçekteki ile uyum sağlayacak şekilde 3 eklemli olarak tasarlanmıştır. Öncelikle C# ile Unity oyun sahnesinde görüldüğü gibi bir grid oluşturulmuştur ki uç efektörün pozisyonu bu şekilde anlaşılabilmektedir. Yine şekilde mavi ile yazılı kısım hedef olarak belirlenen pembe kürenin pozisyonunu (sırasıyla X, Y, Z koordinatlarını) göstermektedir. Pembe küre şeklinde belirlenen hedef cisim sanal gerçeklik konsolu ile hareket ettirilmiştir. Gradyan hesabı ile hareket eden sanal robot manipülatör bu hedefe yaklaşmaya çalışır. Hedef olarak belirlenen pembe küre robot kol uç efektörün etkileşim alanına girdiği anda uç efektör koordinat bilgileri gerçek dünyada oluşturulan robot kola iletilir. Koordinatlar iletildiğinde mavi yazı kırmızıya döner ve koordinat bilgileri gerçek dünya ile uyumlu olacak şekilde gerekli dönüşümler yazılım kısmında gerçekleştirilmiştir. Uç efektör etkileşim alanı Şekil 4.9'da gösterilmiştir

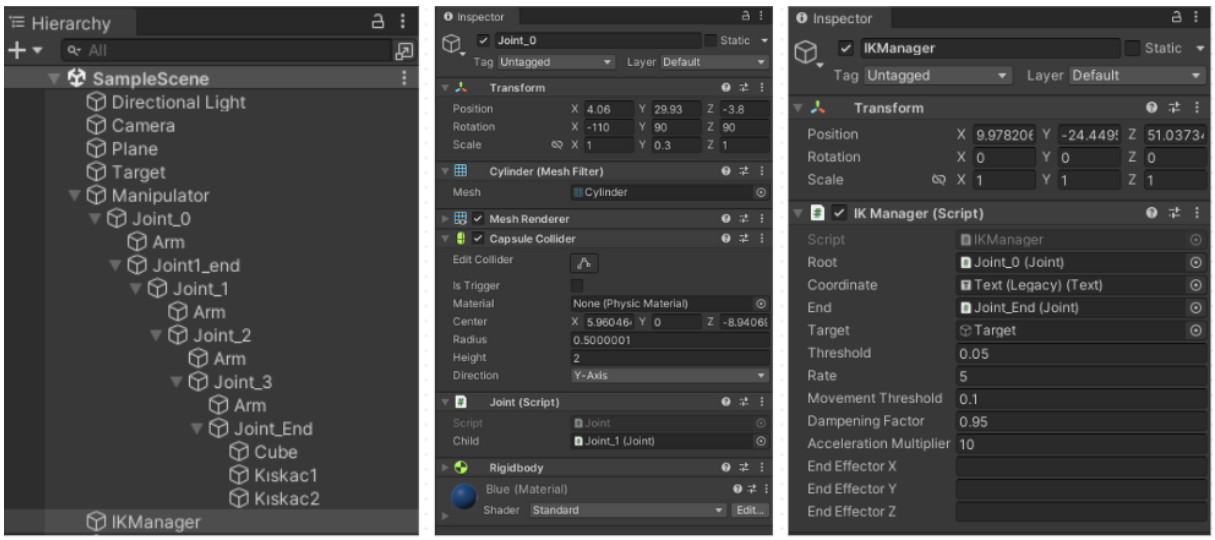


Şekil 4.8: Sanal ortamdaki robot $\boldsymbol{\theta_1}$=0, $\boldsymbol{\theta_2}$=0, $\boldsymbol{\theta_3}$=0 durumdaki konumu.



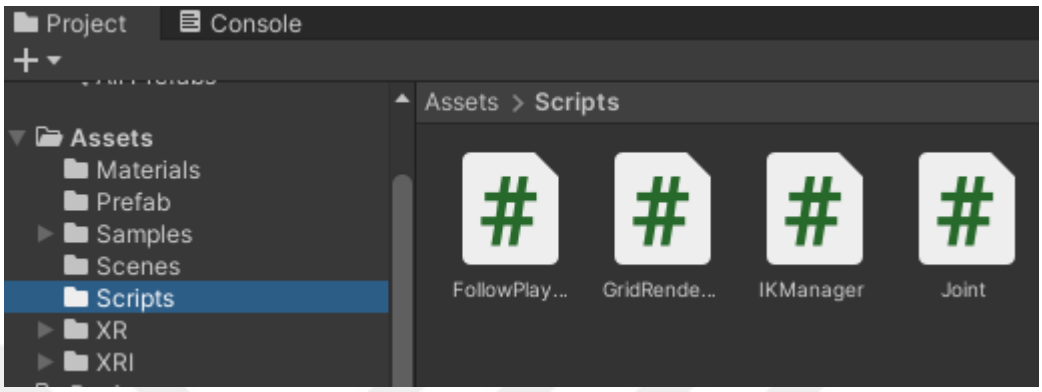
Şekil 4.9: Uç efektör – hedef etkileşim alanı.

Unity sahnesinde görülen robot koldaki kırmızı silindir nesneleri eklemleri ifade eder. Tabanda gözle görülmeyen ama siyah cismin içerisinde taban eklemi de oluşturulmuştur. Yazılımda "joint" şeklinde adlandırılan ve gradyan eğim hesabına göre hareketi sağlayan bu kırmızı silindir nesnelerine atanmıştır. Aşağıdaki Şekil 4.10'da Unity tasarım kısmında robot manipülatörün bileşenlerini oluşturan nesnelerin belirli bir düzen içerisindeki hiyerarşik dizilimi yer alıyor. Ayrıca bu eklemlere yani nesnelere gerçek dünyadaki fizik kurallarına uygun şekilde davranmalarını sağlamak amacıyla bileşenler eklediğimiz, gerekli tanımlamaları yaptığımız kontrolör kısmı yer alıyor. IKManager adında boş bir nesne oluşturup, C# ile yazılan gradyan (eğim) hesaplamasını içeren kodlar bu nesneye atanmıştır. Yine bu nesneye ait kontrolör görüntüsü şekilde gösterilmiştir.



Şekil 4.10: Sanal ortamdaki robot eklemlerin dizilim ve işlevi.

Şekil 4.11'de script adıyla tanımladığımız proje klasöründe grid oluşturulması, kameranın nesne takibi, manipülatöre ait gradyan eğim hesabı, tasarlanan eklem nesnelerine fizik kazandırmak için oluşturulan yazılım dosyalarının görüntüsü yer alıyor. Gradyan eğim hesabının yer aldığı kodlara Ek 4'te yer verilmiştir.

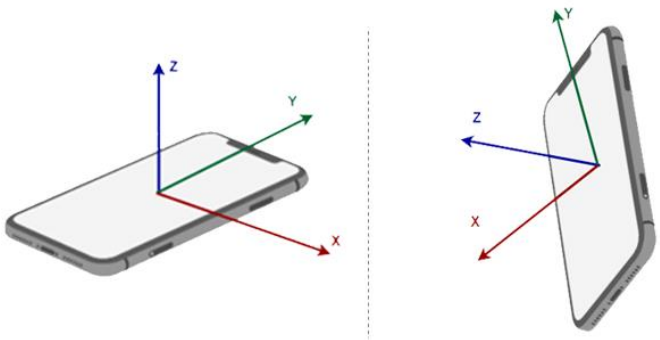


Şekil 4.11: Sanal ortam yazılım dosyaları

4.2.3 Mobil ivmeölçer ile Robot Kol Kontrolü

Sanal ortamdaki robot manipülatör kontrol araçları ile sağlanıyor. Bunun için projenin de ana konusu olan sanal gerçeklik gözlüğü ve kontrol araçları (konsol) kullanıldı. Ancak günümüzde sanal gerçeklik kullanımının yaygın olmamasından kaynaklı mobil ivme ölçer kullanarak ta deneme çalışmaları yapılmıştır.

Mobil telefonlardaki ivmeölçer yönü mobil telefon yatay ve dikey konumdayken Şekil 4.12'de gösterilmiştir. Görüldüğü gibi yatay ve dikey konumdayken yönler değişmez ama bu eksenlerde hareket sağlayabilmek için mobil telefonlara eğim vermek gerekir.

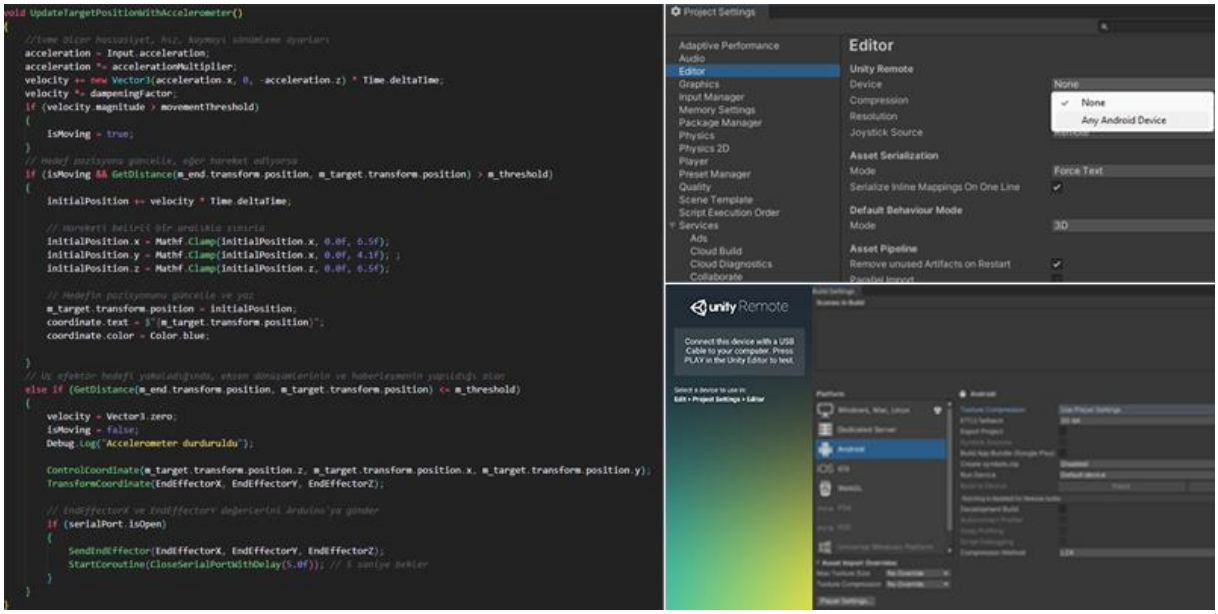


Şekil 4.12: Mobil ivme ölçer eksenlerin yönü.

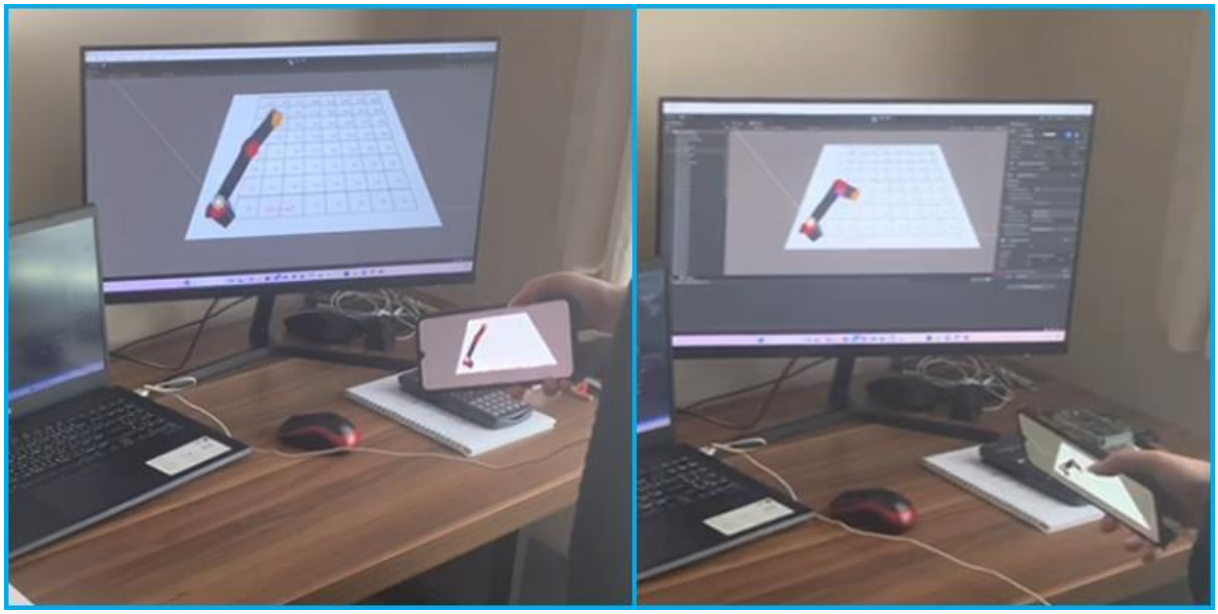
Mobil ivme ölçer kullanımı X ve Y eksenin kullanımında idealdir. Direksiyon gibi işlev görür ve cismin hareketi rahatlıkla kontrol edilebilir. Bu çalışmada X, Y, Z eksenlerinin üçü de kullanıldığı için mobil

ivme ölçer ile robot kol kontrol ve hareketi zorlaşmıştır. Çünkü istenilen eksenlerde hareketin sağlanması için mobil telefonun sürekli olarak eğilip bükülmesi gerekmektedir. Bu durum kullanıcı tarafından pratikte yön algısının karışmasına neden olmaktadır.

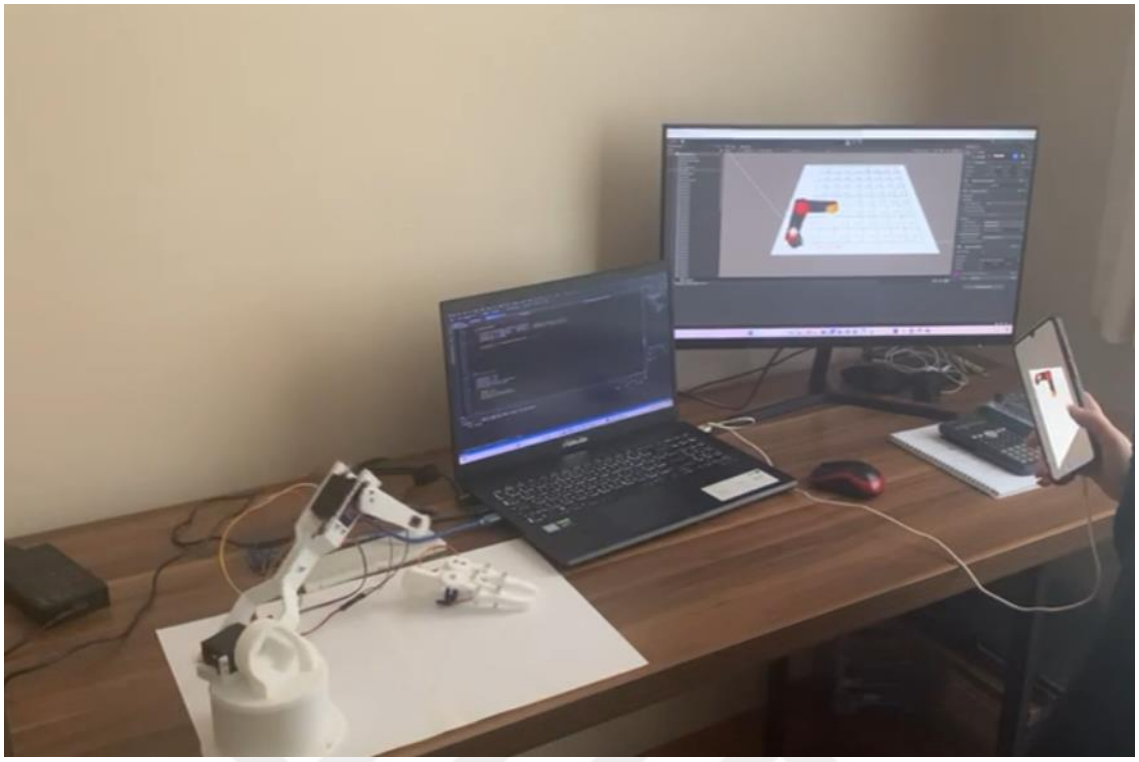
İvmeölçer ile ilgili kod parçası ve Unity program ayarları Şekil 4.13'teki gibidir. 3 serbestlik derecesine (3DOF) sahip robot kol kontrolünün mobil ivme ölçer ile sağlandığı çalışmanın örnekleri Şekil 4.14 ve 4.15'te gösterilmiştir.



Şekil 4.13: Unity programında mobil ivme ölçer ayarlamaları ve yazılımı.



Şekil 4.14: Sanal ortamdaki 3DOF Manipülatörün mobil ivme ölçer ile hareketi.



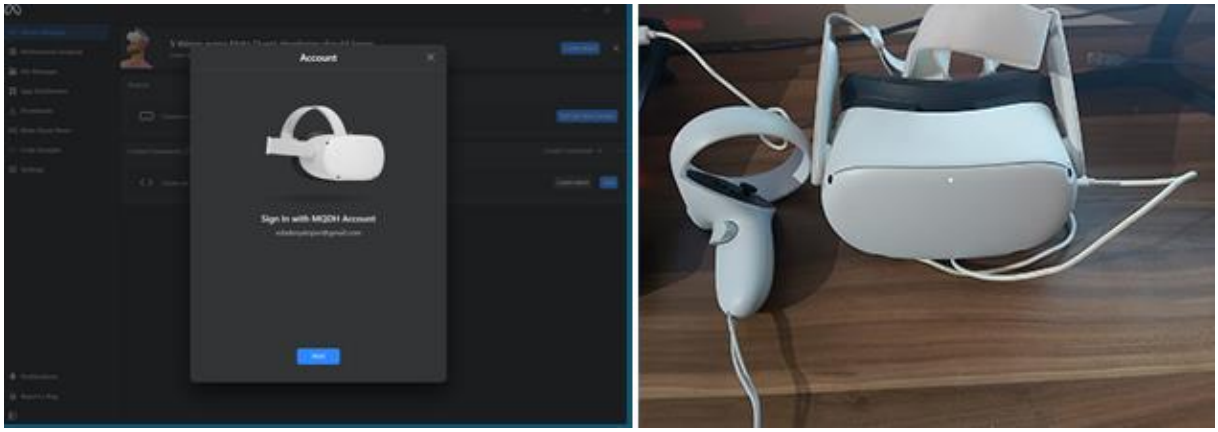
Şekil 4.15: Sanal 3DOF Manipülatörden gelen koordinatlara göre gerçek manipülatör hareketi.

4.2.4 Sanal Gerçeklik Konsolları ile Robot Kol Kontrolü

Sanal ortamdaki robot kol kontrolü proje ana amacına da uygun olarak sanal gerçeklik konsolu ile sağlandı. Sanal gerçeklik konsolu ile hedef cisim (pembe küre) hareket ettirildi. Sanal ortamdaki robot kol uç efektörün (tutucu) hedefi yakalaması beklendi.

Sanal gerçeklik konsolu zaten sanal ortam için oluşturulmuş cihazlar olduğu için koordinat sistemleri de buna uyum sağlayacak şekildedir. Sağa sola hareket X ekseninde, yukarı aşağı hareket Y ekseninde ve ileri geri hareket Z eksenindedir. Bu durum kullanıcı kolaylığı sağlamakla birlikte hareketi kolaylaştırmaktadır.

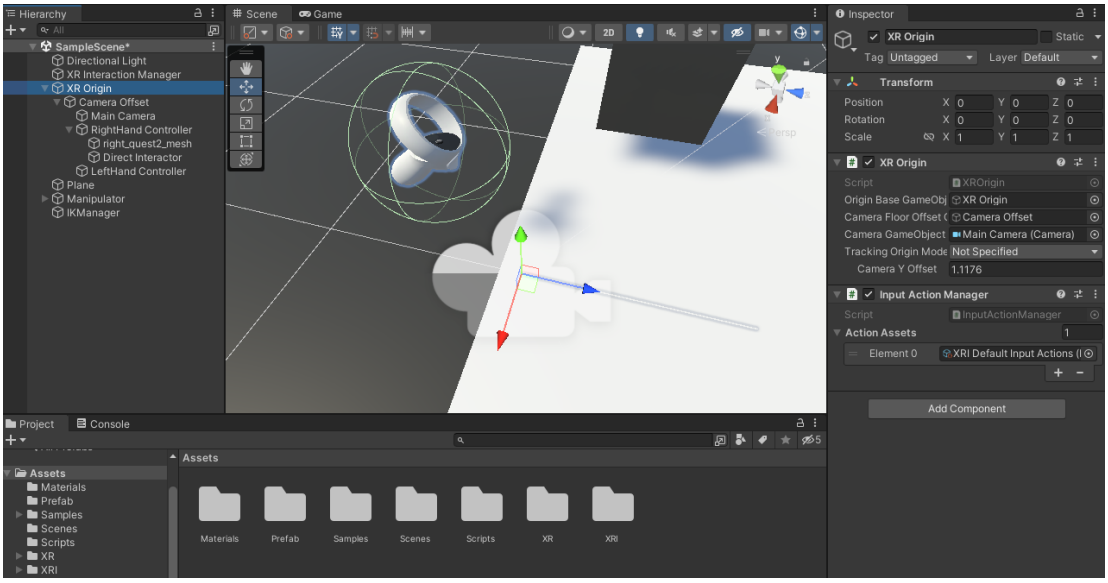
Bu tez çalışmasında sanal ortamda deneme çalışmaları için Meta Quest 2 markası kullanılmıştır. Sanal gerçeklik gözlüğü ve konsollarını kullanmadan önce temel kurulum ve Unity ayarlarının yapılması gerekmektedir. Kurulum aşamasında öncelikle kullanılacak donanımın markasına ait programın indirilmesi gerekiyor. Unity üzerinde proje geliştirmek üzere kullanılacağı için geliştirici için olan yazılım programı indirildi. Şekil 4.16'daki Meta geliştirici uygulaması üzerinde kullanılan donanımın kurulumu, geliştirici ayarları yapılmıştır.



Şekil 4.16: Meta Quest 2 geliştirici programı donanım yapılandırması.

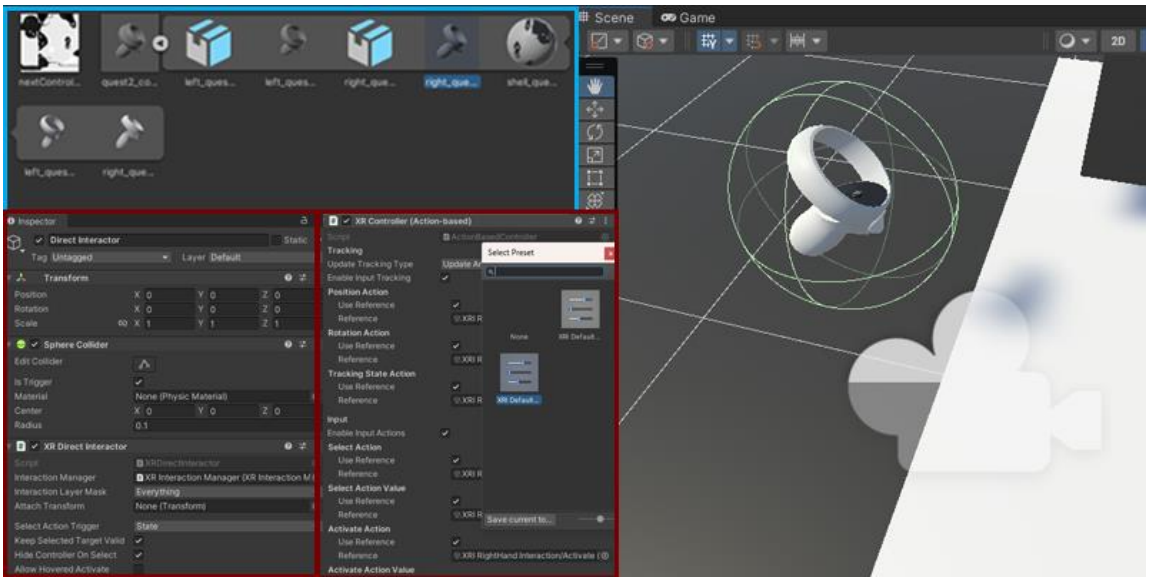
Unity package panager kısmında “XR-Plug-in Management” ve “XR Interaction ToolKit” paketlerinin indirilmesi gerekmektedir. XR Plug-in Management ve XR Interaction Toolkit, Unity'de XR (Extended Reality) uygulamaları geliştirmek için kullanılan iki önemli araçtır. XR Plug-in Management Unity programının bu çalışma için kullandığımız Meta platformuna entegre olmasını sağlar. Sanal gerçeklik için kullanılan donanımının Unity üzerinde geliştiriciler için gerekli olan temel ayarları, bağımlılıkları, yapılandırmaları otomatik yönetir. XR Interaction Toolkit aracı için sanal gerçeklik uygulamalarında kullanıcı etkileşimlerini kolayca oluşturmak ve yönetmek için kullanılır.

Unity üzerinde sanal gerçeklik için gerekli paket kurulumlarının gerçekleştirilmesinin ardından sanal gerçeklik için Şekil 4.17’deki sanal gerçeklik bileşenleri, tasarımlar ve bu tasarımlara gerekli işlevler eklendi.



Şekil 4.17: Unity programını sanal gerçeklik ortamına uygun hale getirmek için temel ayarlamalar.

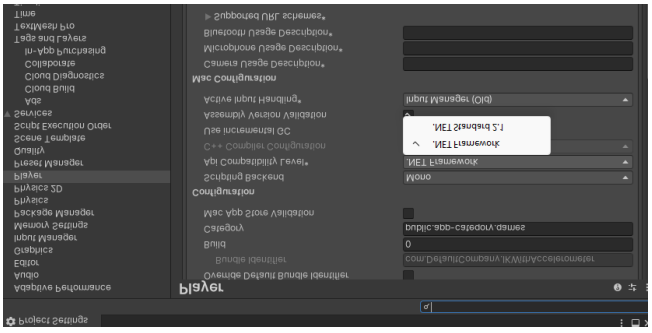
Çalışmada kontrol aracı olarak sadece sağ konsol kullanıldığı için bu konsola ait nesne şablonu, XR yapılandırmaları sadece bu konsol üzerinde yapılmıştır. Temel ayarlamalar Şekil 4.18'de gösterildiği gibidir. Konsolun sanal ortamdaki materyal, tasarım, şablonu için yine kullanılan donanım markası olan Meta sitesinden alınmıştır. Nesneyi yakalama işlevi için XR bileşeni olan "Direct interactor" eklenmiştir. Konsol etkileşim alanı şekilde görüldüğü gibi konsol etrafındaki yeşil alandır bu ihtiyaca göre artırılıp azaltılabilir. Yazılım için XR paketlerinin yüklenmesi ile beraber gelen yazılım araçları kullanılmıştır, ekstra çalışmaya gerek duyulmamıştır.



Şekil 4.18: Unity programı üzerinde sanal gerçeklik konsolu tasarım ve ayarlamaları.

4.2.5 Sanal Ortamdaki ile Gerçek Dünyadaki Robot Kolların Haberleşmesi

Sanal ortam ile gerçek dünyadaki robot kolların haberleşmesi, bu çalışmada seri haberleşme ile sağlandı. Unity C# yazılım dosyasında Arduino geliştirme kiti ile seri iletişim sağlamak için System.IO.Ports kütüphanesinden faydalanıldı. Kütüphanenin kullanılabilmesi için Şekil 4.19'daki gibi Unity ayarlarından .Net Framework olanını seçmek gerekiyor.

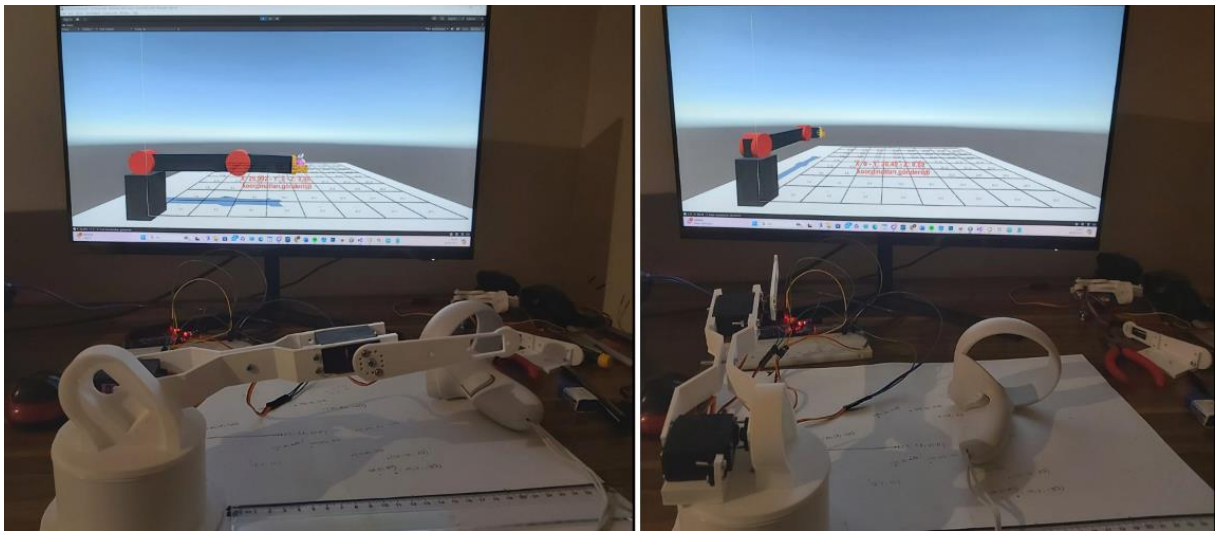


Şekil 4.19: Haberleşme için Unity program ayarlaması

Sanal ortamdan gerçek dünyadaki robot kola doğru konum, koordinat ve ölçülerin gitmesi için yazılım kısmında gerekli dönüşümler daha önce bahsedildiği gibi yapıldı. Haberleşme için Arduino kitinin bağlı olduğu port girişi ve bant genişliği tanımlandı. Dönüşümün yapıldığı kod bloğu aşağıdaki Şekil 4.20'deki gibidir. Şekil 4.21'de Sanal ve Gerçek ortamdaki (0,0,0) ve (90,0,0) açılarında konumu gösterilmiştir.

```
void StartSerialCommunication()
    {
        if (GetDistance(m_end.transform.position, m_target.transform.position) <= m_threshold)
        {


            ControlCoordinate(m_end.transform.position.z, m_end.transform.position.x, m_end.transform.position.y);
            EndEffectorX = (m_target.transform.position.x * 4).ToString();
            EndEffectorY = (m_target.transform.position.z * 4).ToString();
            EndEffectorZ = (m_target.transform.position.y * 4).ToString();
            if (serialPort.IsOpen && (EndEffectorX != lastSentX || EndEffectorY != lastSentY || EndEffectorZ != lastSentZ))
            {
                SendEndEffector(EndEffectorX, EndEffectorY, EndEffectorZ);
                Debug.Log($"X: {EndEffectorX} - Y: {EndEffectorY} - Z: {EndEffectorZ} koordinatları gönderildi");
                coordinate.text = $"X: {EndEffectorX} - Y: {EndEffectorY} - Z: {EndEffectorZ}\n koordinatları gönderildi";
                coordinate.color = Color.red;
                coordinate.fontSize = 30;
                coordinate.fontStyle = FontStyle.Bold;
                lastSentX = EndEffectorX;
                lastSentY = EndEffectorY;
                lastSentZ = EndEffectorZ;
            }


        }
    }

    //string tipinde gönderilen Koordinatların decimal olacak şekilde Arduino'ya iletilmesi
void SendEndEffector(string x, string y, string z)
    {
        try
        {
            string endEffectors = $"{x.Replace(',', '.')},{y.Replace(',', '.')},{z.Replace(',', '.')}\n";
            byte[] data = System.Text.Encoding.UTF8.GetBytes(endEffectors);
            serialPort.Write(data, 0, data.Length);
            Debug.Log($"Write data: {data}, {x}, {y},{z}");
            Debug.Log($"Sent coordinates: {x.Replace(',', '.')}, {y.Replace(',', '.')},{z.Replace(',', '.')}");
        }
        catch (System.Exception e)
        {
            Debug.LogWarning($"Error sending angles: {e.Message}");
        }
    }
```

Şekil 4.20: Unity programında haberleşme için yazılan kod bloğu.



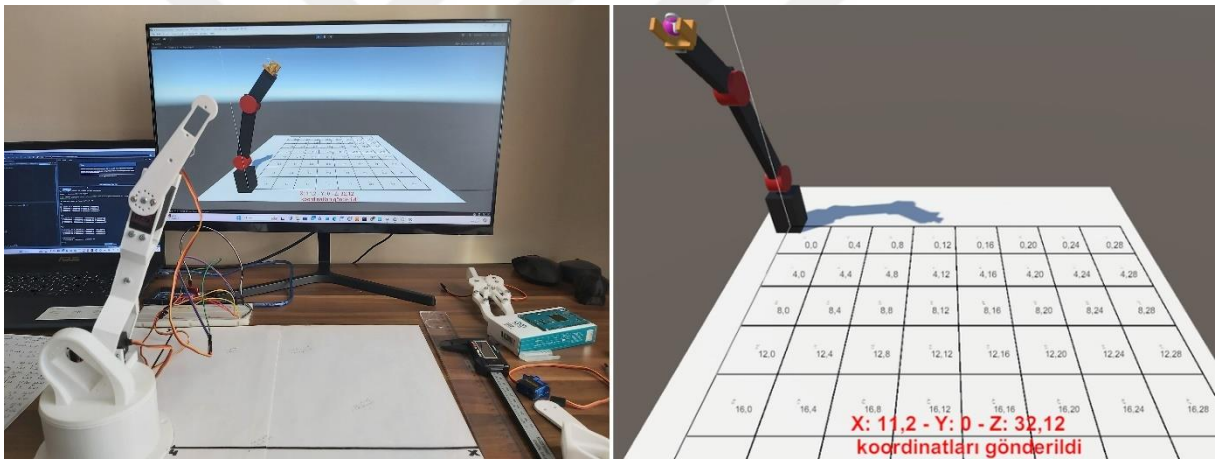
Şekil 4.21: Sanal ve Gerçek ortamdaki (0,0,0) açılarında ve (90,0,0) açılarında konumu.

4.3 GERÇEK VE SANAL DÜNYADAKİ ROBOT KOLLARIN ETKİLEŞİMLİ HAREKETİ

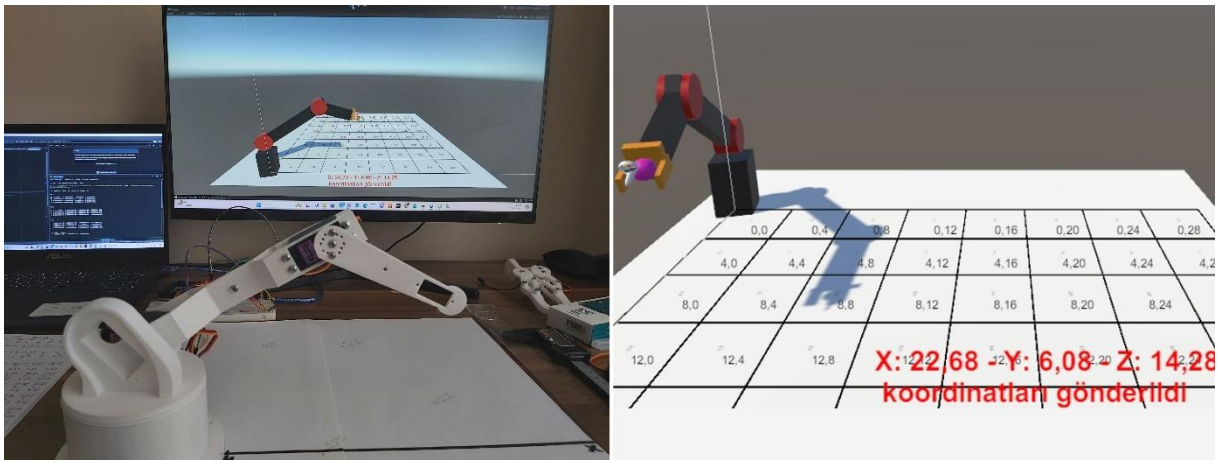
Aşağıdaki şekillerde sanal ortamdaki robot manipülatörün hedef cisimi yakaladığında gerçek dünyadaki robot manipülatörün gelen koordinat bilgilerine göre hareketi iki farklı açıdan gösterilmiştir. Sol taraftaki görsel ile iki ortamdaki robot kolun eklem hareketlerinin benzerliği gösterilmek istenmiştir. Sağ taraftaki görsel ise farklı sanal ortam kamera açısından görünümü olup hem hedef cismin hem de sanal gerçeklik konsolunun pozisyonları gösterilmek istenmiştir. Bu konumlar ile robot kol eklemlerinin aldığı açılar gözlemlenerek homojen dönüşüm matrislerinde doğruluk payları incelenmiştir.

Uygulama Meta Quest 2 sanal gerçeklik gözlükleri ile denenmiştir, görselin düzgün ve kaliteli gözükmesi açısından Unity programı ekranından alınmıştır.

Şekil 4.22'de (0, 70°, -15°), Şekil 4.23'de (15°, 30°, -45°) açılarındaki pozisyonu gösterilmiştir.

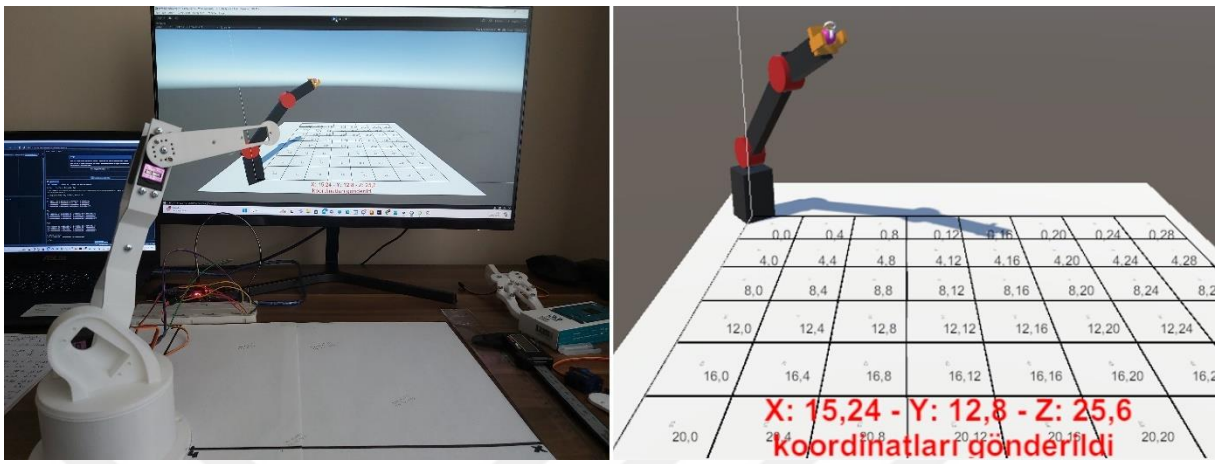


Şekil 4.22: 0, 70°, -15° açılarındaki pozisyonu.

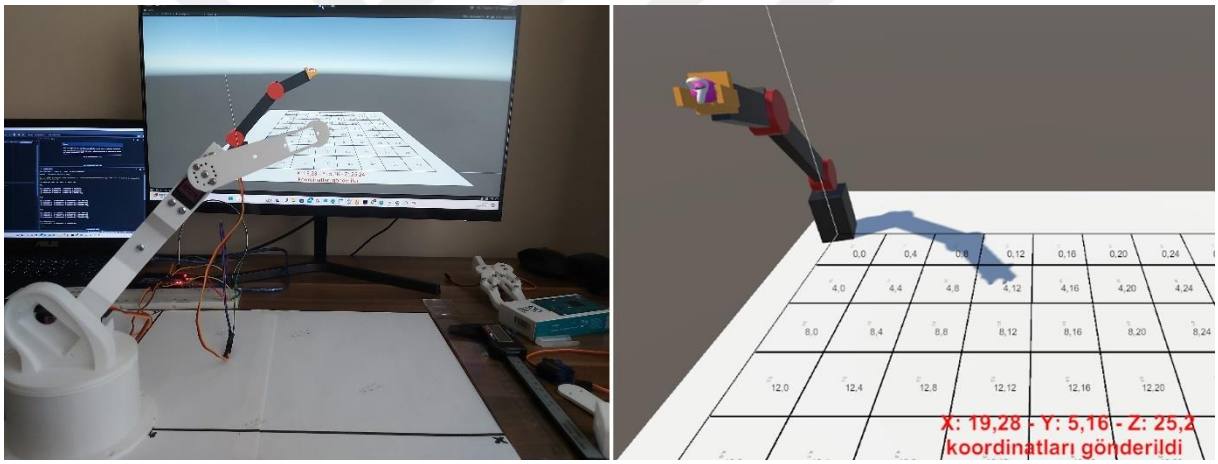


Şekil 4.23: 15°, 30°, -45° açılarındaki pozisyonu.

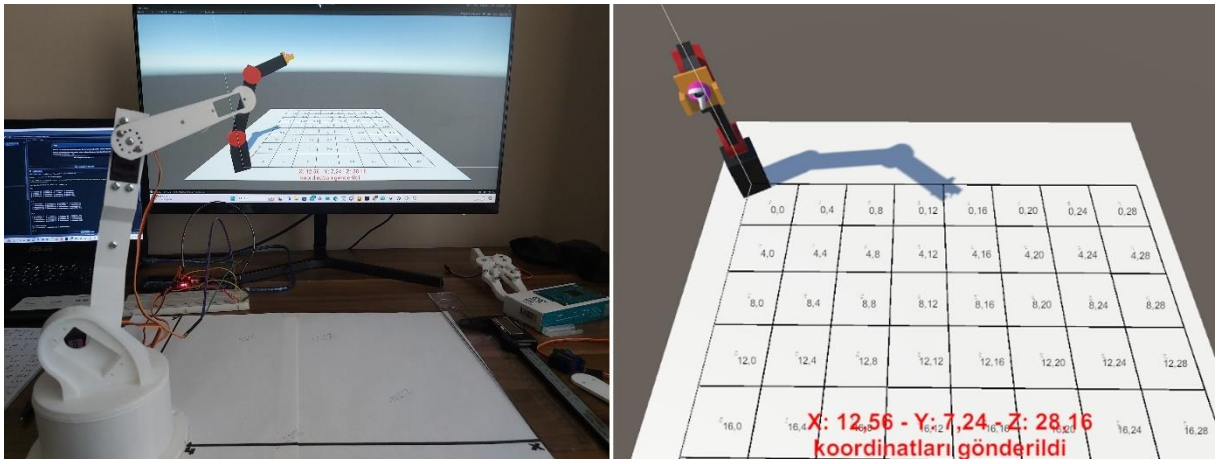
Şekil 4.24’de (40, 40°, 0), Şekil 4.25’de (15°, 45°, -15°), Şekil 4.26’da (30°, 70°, 45°) açılarındaki pozisyonu gösterilmiştir.



Şekil 4.24: 40°, 40°, 0 açılarındaki pozisyonu.

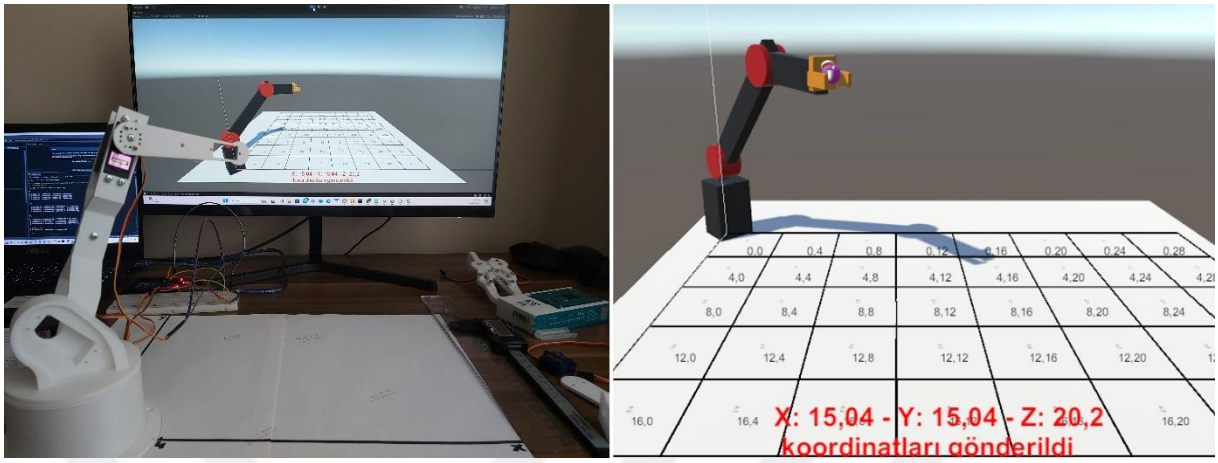


Şekil 4.25: 15°, 45°, -15° açılarındaki pozisyonu.

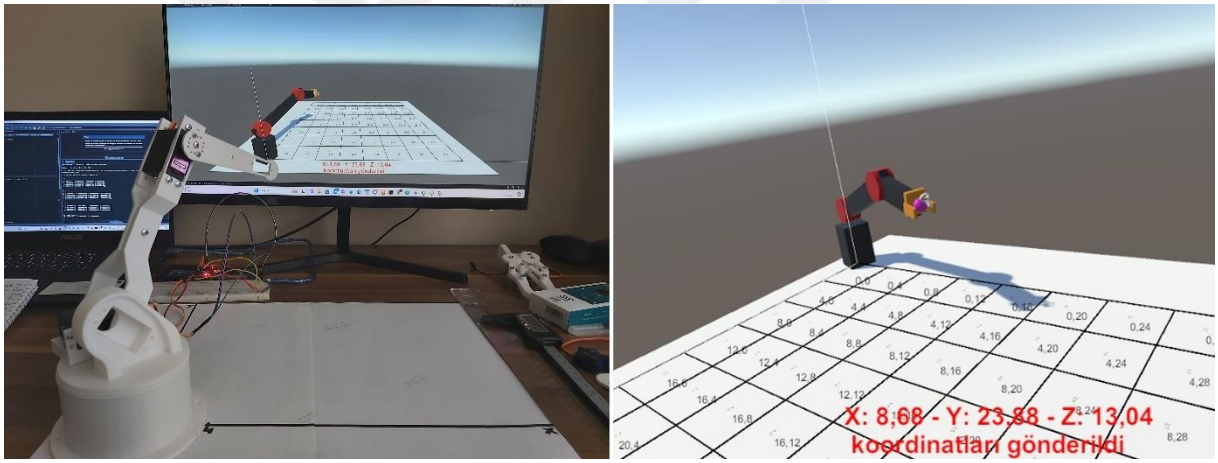


Şekil 4.26: 30°, 70°, -45° açılarındaki pozisyonu.

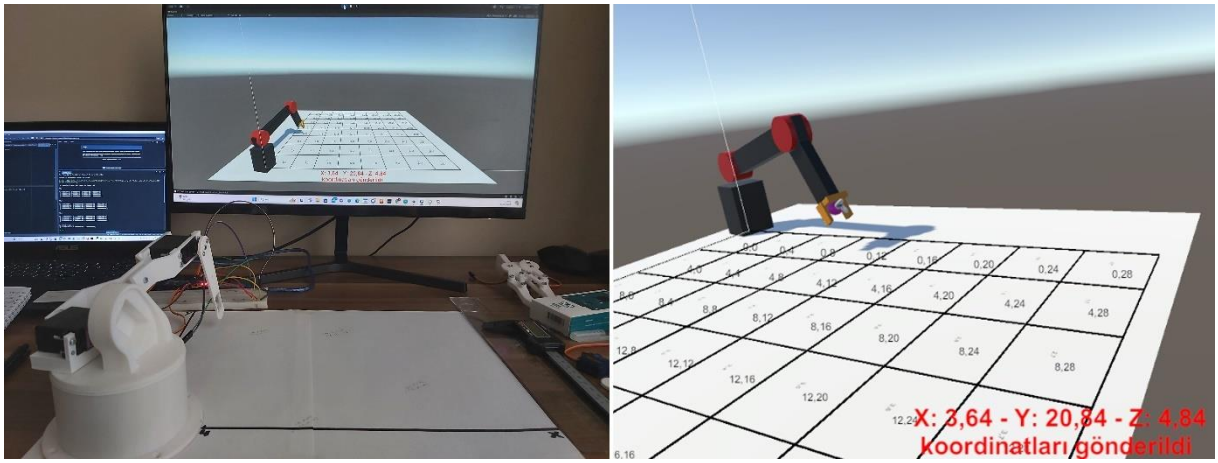
Şekil 4.27'de (45°, 45°, -45°), Şekil 4.28'de (70°, 15°, -15°), Şekil 4.29'da (80°, 15°, -70°) açılarındaki pozisyonu gösterilmiştir.



Şekil 4.27: 45°, 45°, -45° açılarındaki pozisyonu.

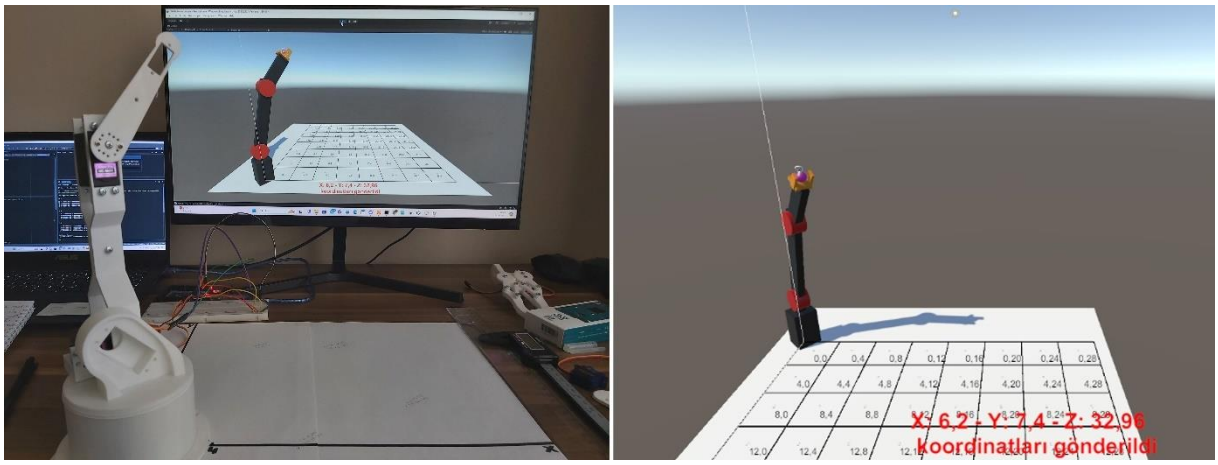


Şekil 4.28: 70°, 15°, -15° açılarındaki pozisyonu.

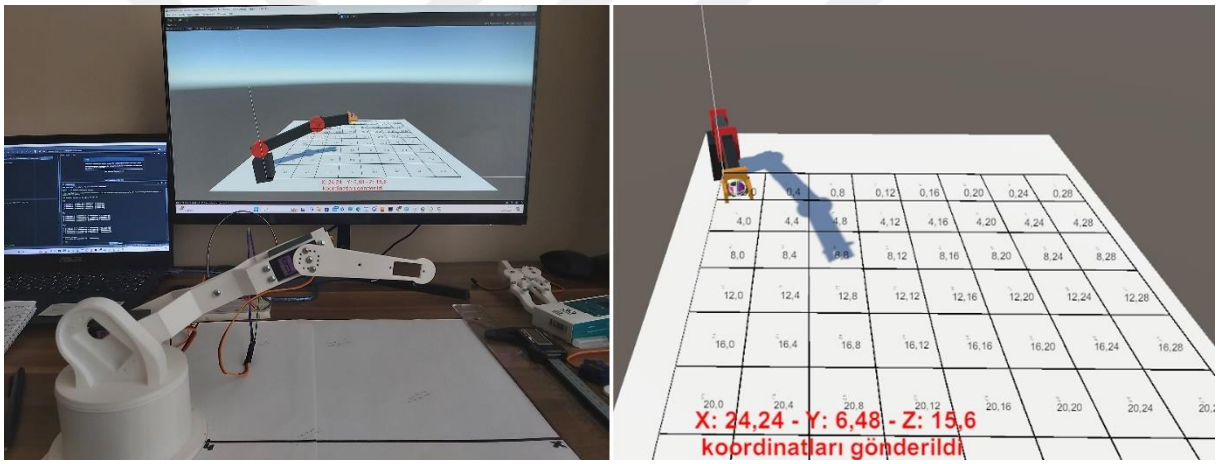


Şekil 4.29: 80°, 15°, -70° açılarındaki pozisyonu.

Şekil 4.30’da (50°, 70°, -5°), Şekil 4.31’de (15°, 15°, 0) açılarındaki pozisyonu gösterilmiştir.



Şekil 4.30: 50°, 70°, -5° açılarındaki pozisyonu.



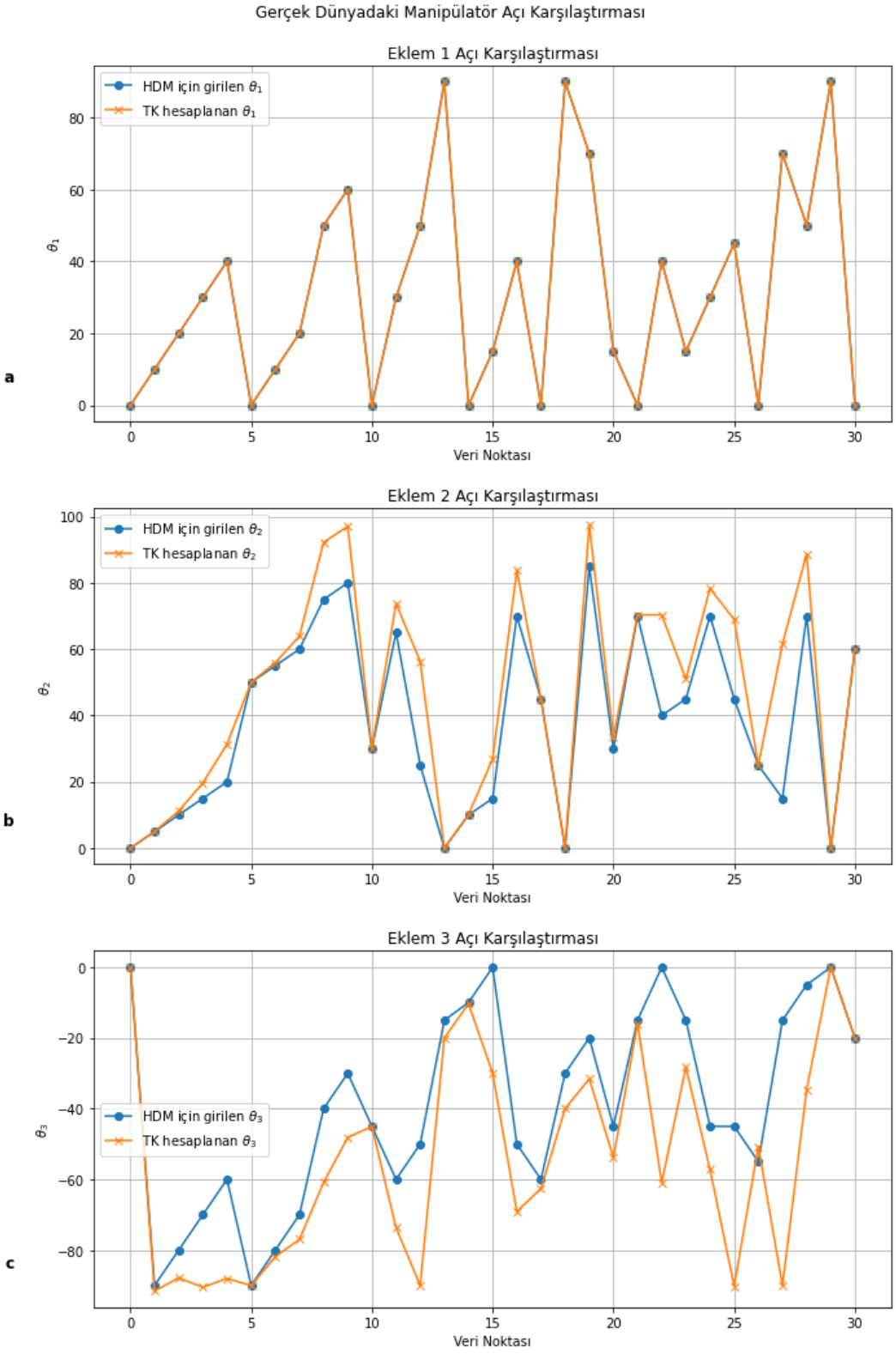
Şekil 4.31: 15°, 15°, 0 açılarındaki pozisyonu.

Gerçek dünyada ve sanal ortamdaki manipülatör çalışma alanı içerisinde test edildi. Yapılan test çalışmasında ileri kinematik yöntemiyle girilen açı değerleri ile her iki ortamda ölçülen açı değerleri kıyaslandı. Gerçek ortamdaki robot kolun açı değerleri için ters kinematik yöntemi ile hesaplanan açı değerleri alınmıştır. Sanal ortamdaki için ise gradyan hesabı ile hesaplanan açı değerleri alınmıştır.

Aynı şekilde ileri kinematik yöntemiyle hesaplanan koordinat bilgileri ile her iki ortamda ölçülen koordinat bilgileri alınmıştır. Gerçek dünyadaki robot kolun koordinat ölçümü için ölçü aletleri kullanılmıştır. Sanal ortamdaki robot kol pozisyonu oluşturulan grid sistemi ile ölçülmüştür.

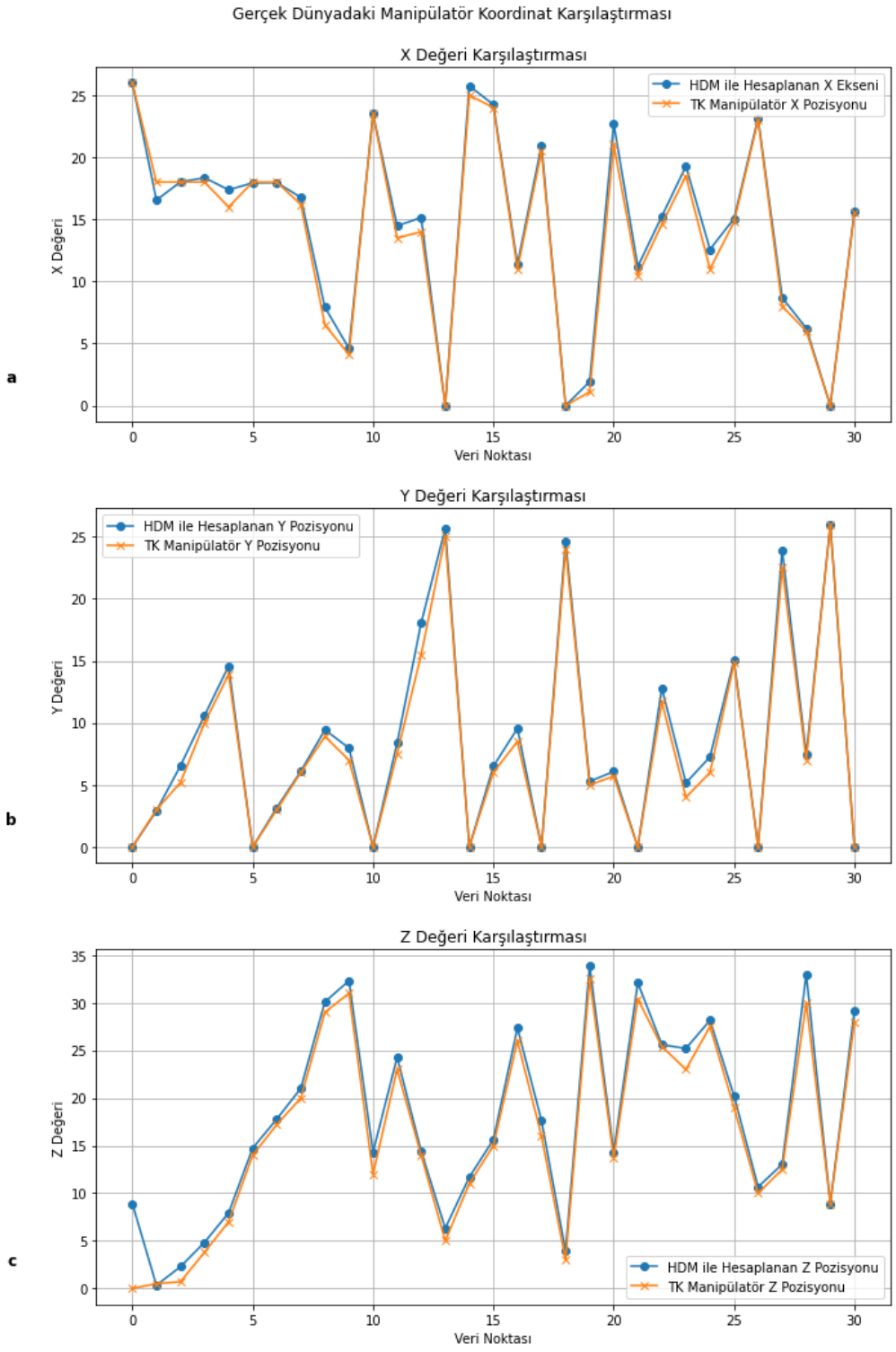
Şekil 4.32’teki grafikte, gerçek dünyadaki robot kol hareketi için ileri kinematik (homojen dönüşüm matrisi) hesabı için girilen açı değerleri ile ters kinematik yöntemiyle hesaplanan açı değerlerinin karşılaştırma sonuçları gösterilmiştir. Şekil 4.32a’da 1. Eklemin açısı θ_1, Şekil 4.32b’de 2. Eklemin açısı

θ_2, Şekil 4.32c'de 3. Eklemin açısı θ_3 değerleri için ileri kinematik hesabı için girilen ve ters kinematik sonucu ölçülen değerlerin karşılaştırması gösterilmektedir.



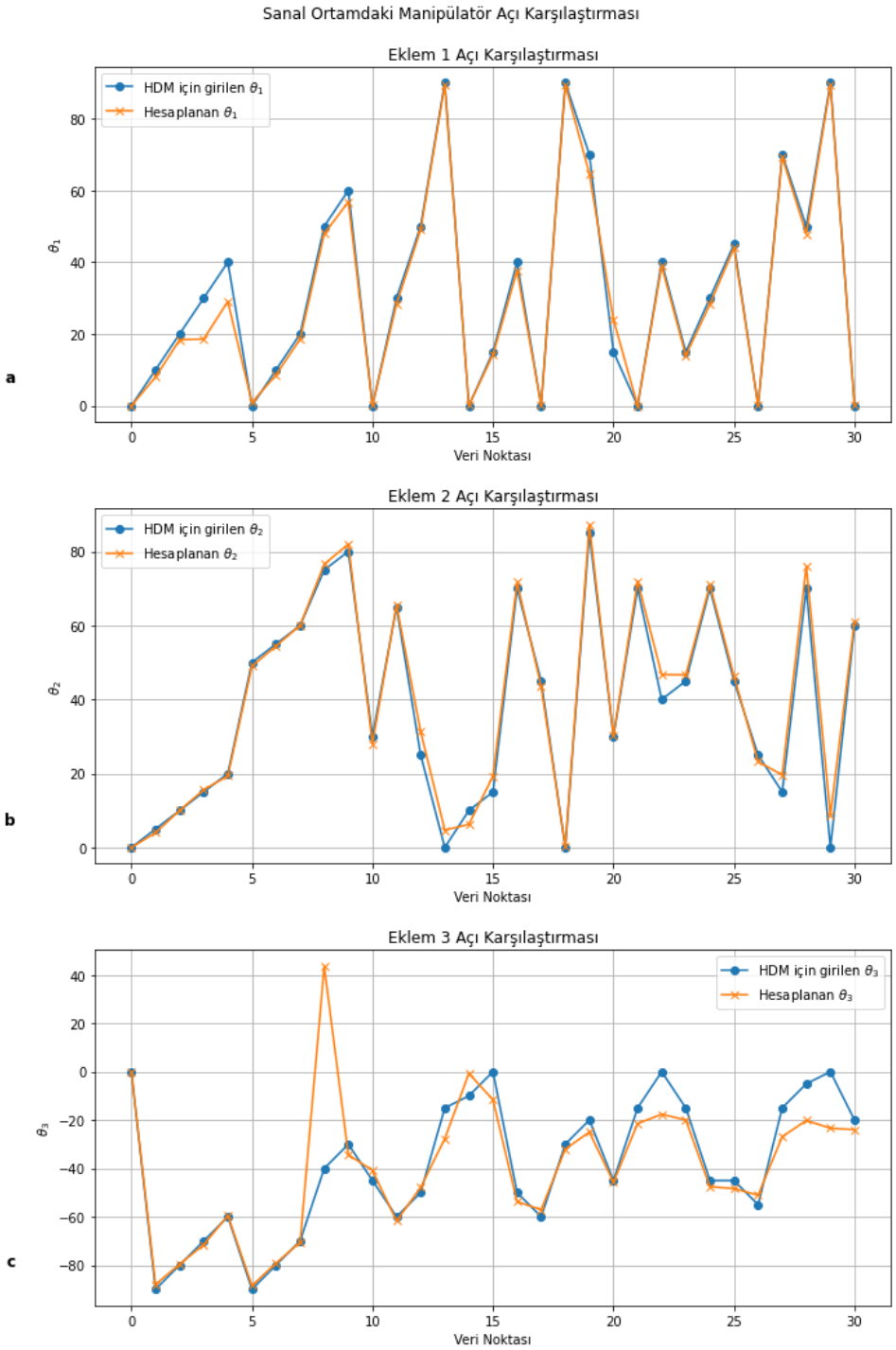
Şekil 4.32: Gerçek dünyadaki manipülatör için İK-TK açı karşılaştırması.

Şekil 4.33'teki grafikte, gerçek dünyadaki robot kol hareketi için ileri kinematik yöntemiyle (homojen dönüşüm matrisi) hesaplanan koordinat bilgileri ile ters kinematik hesabı için girilen bu değerler sonucunda hareket eden robot kolun pozisyonu ölçülerek doğruluğu kıyaslanmıştır. Şekil 4.33a'da X, Şekil 4.33b'de Y, Şekil 4.33c'de Z ekseni konumu için girilen ve ölçülen değerler kıyaslanmıştır.



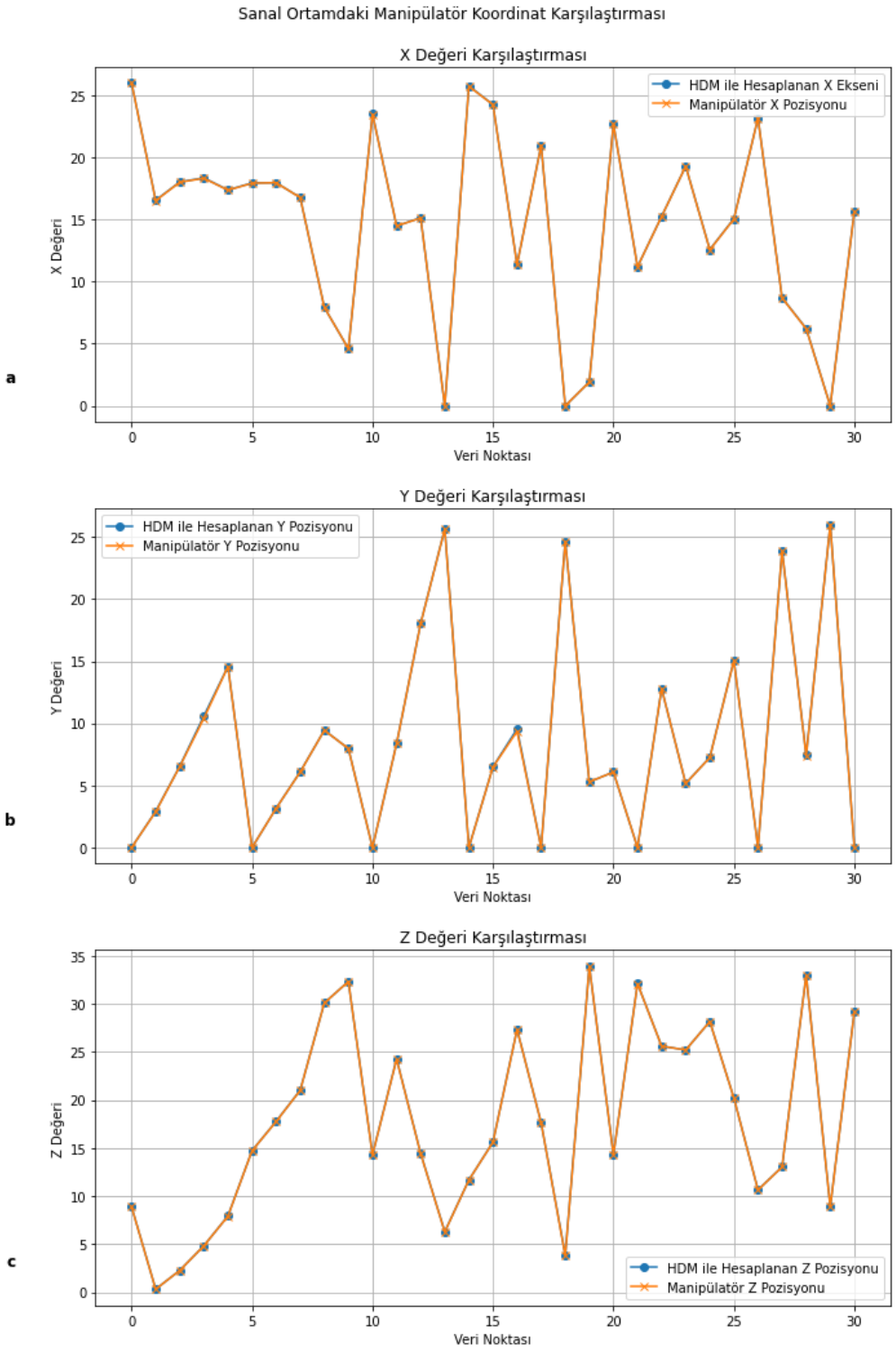
Şekil 4.33: Gerçek dünyadaki manipülatör için İK-TK koordinatlarının karşılaştırması.

Şekil 4.34'teki grafikte, sanal ortamdaki robot kol hareketi için ileri kinematik (homojen dönüşüm matrisi) hesabı için girilen açı değerleri ile robot kol hareketi sonucu aldığı açı değerlerinin karşılaştırma sonuçları gösterilmiştir. Şekil 4.34a'da 1. Eklemin açısı θ_1, Şekil 4.34b'de 2. Eklemin açısı θ_2, Şekil 4.34c'de 3. Eklemin açısı θ_3 değerleri için girilen ve ölçülen değerlerin karşılaştırması gösterilmektedir.



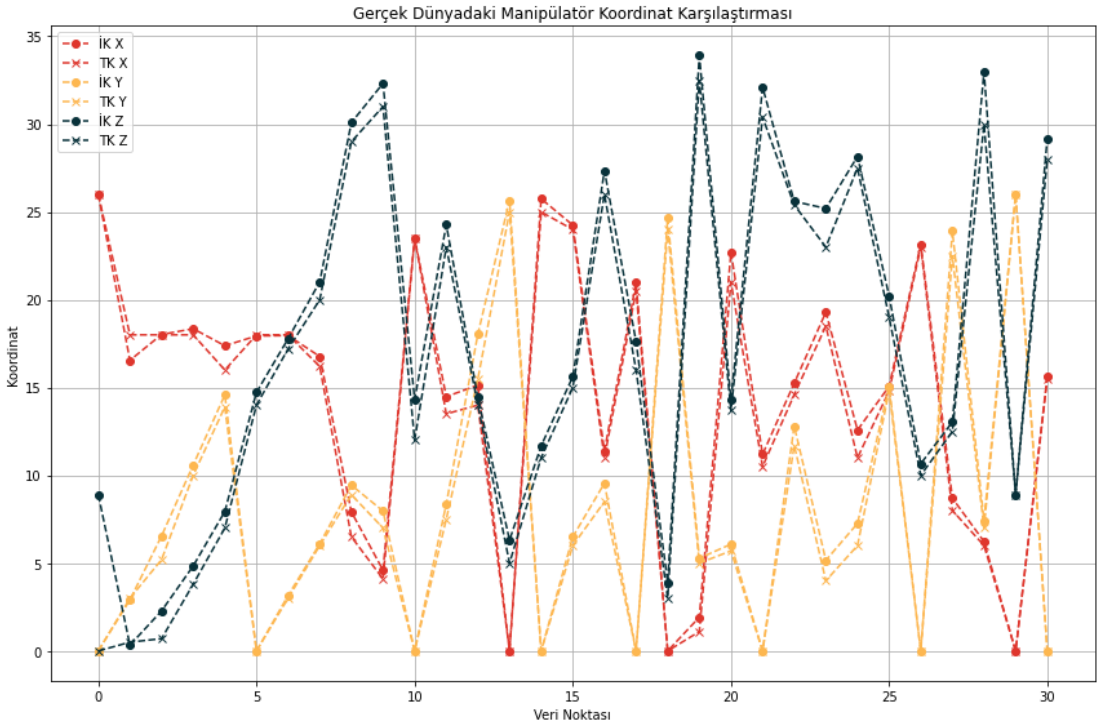
Şekil 4.34: Sanal ortamdaki manipülatör için girilen – ölçülen açı karşılaştırması.

Şekil 4.35'teki grafikte, sanal ortamdaki robot kol hareketi için ileri kinematik yöntemiyle (homojen dönüşüm matrisi) hesaplanan koordinat bilgileri ile bu değerler sonucunda hareket eden robot kolun pozisyonu ölçülerek doğruluğu kıyaslanmıştır. Şekil 4.33a'da X, Şekil 4.33b'de Y, Şekil 4.33c'de Z ekseni konumu için girilen ve ölçülen değerler kıyaslanmıştır.

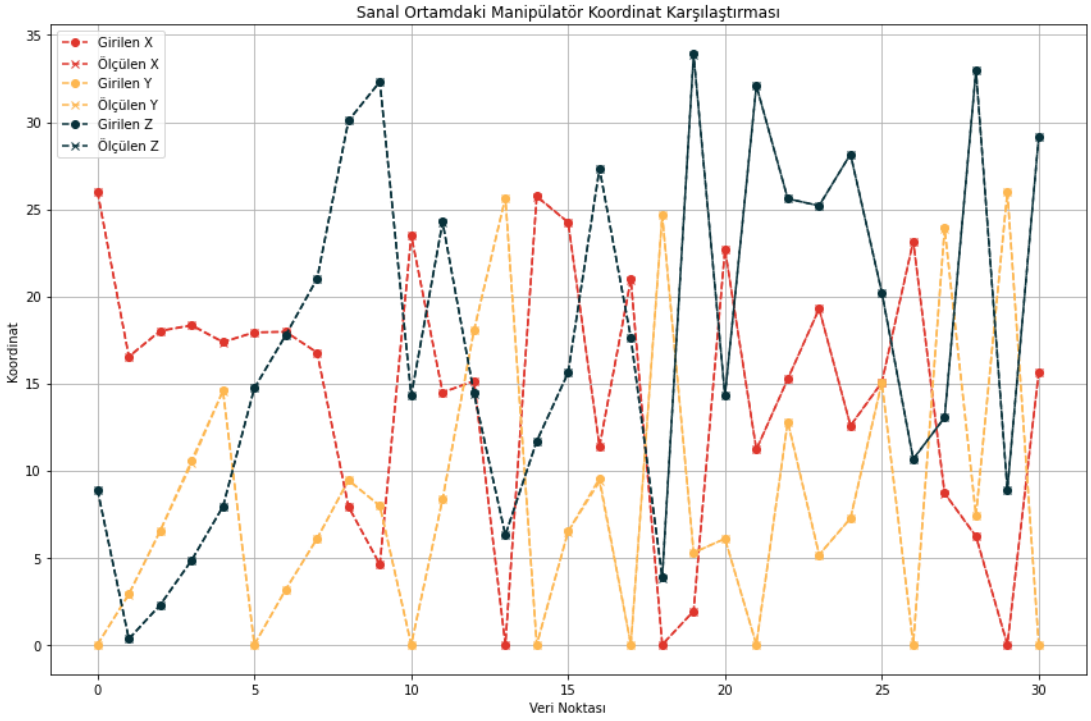


Şekil 4.35: Sanal ortamdaki manipülatör için girilen – ölçülen koordinatlarının karşılaştırması.

Şekil 4.36'da gerçek dünyadaki, Şekil 4.37'de sanal ortamdaki robot manipülatörlere ait girilen ve ölçülen koordinat bilgilerinin karşılaştırması tek grafikte gösterilmiştir.



Şekil 4.36: Gerçek dünyadaki manipülatör için koordinat karşılaştırmasının tek grafikte gösterilmesi.

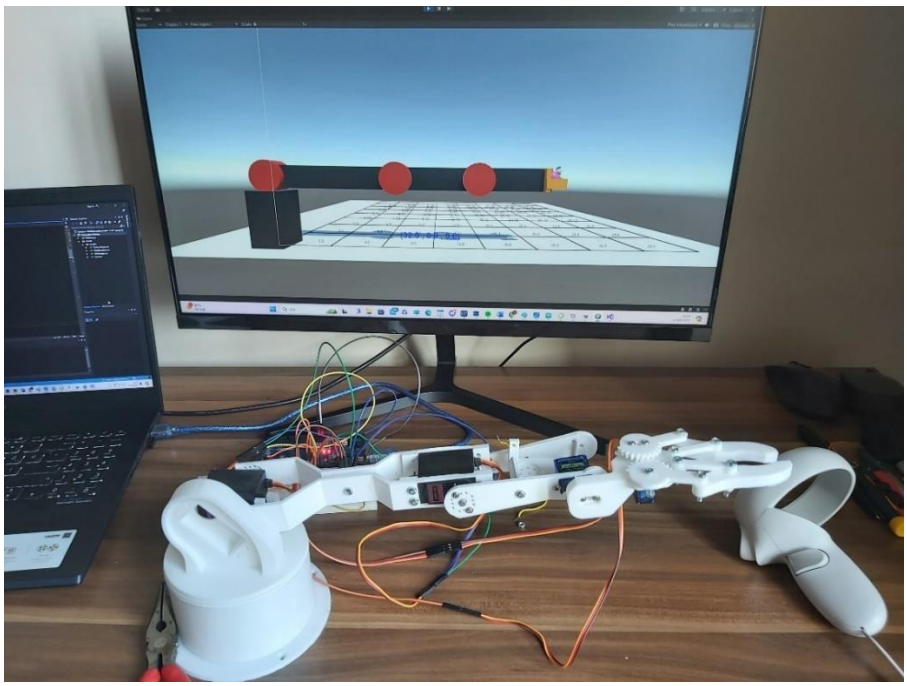


Şekil 4.37: Sanal ortamdaki manipülatör için koordinat karşılaştırmasının tek grafikte gösterilmesi.

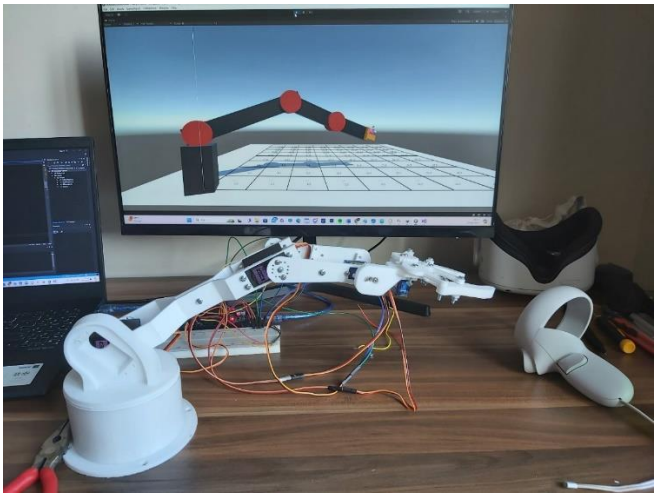
4.4 GERÇEK DÜNYADA VE SANAL ORTAMDA 5DOF ROBOT KOL MODELİ

3 serbestlik derecesine (3DOF) sahip için yapılan çalışmalar, hesaplamalar baz alınarak model tasarımı geliştirilmiş olup sanal gerçeklik gözlüğü kullanarak Unity ve Arduino arasında haberleşme sağlanmıştır. Daha önce de belirtildiği gibi ana çalışma, hesaplamalar 3 serbestlik derecesindeki robot manipülatör üzerine yapılmıştır. 5 serbestlik derecesine sahip robot manipülatörde eklenen iki eklem modelin amacına uygun olacak şekilde esneklik sağlanarak oluşturulmuştur. 4. Eklem ve 5. Eklem 0 - 180° açı aralığında hareket etmektedir. Tutucu işlevi için eklenen 5. Eklem belirli konuma geldiğinde açısal olarak hareket etmektedir. Burada kullanılan çalışma modelinin gereksinimine uygun olacak şeklide ivmeölçer, sınır anahtarı, hall etkisi gibi sensörler tercih edilerek daha hassas ve güvenilir şekilde hareketi sağlanabilir. Bu çalışmada sensör kullanımına gerek duyulmadı.

Gerekli kalibrasyon ayarları, çalışma alanının belirlenmesi, İleri kinematik hesaplamaları ve ters kinematik hesapları 3DOF robot manipülatör için anlatıldığı gibi yapılmıştır. Sanal ortamdaki robot manipülatör için sadece tasarım kısmında değişiklik yapılmıştır. Çünkü yazılım kısmı daha fazla eklem olabilme olasılığına göre ayarlanmıştır. Gerekli tanımlamaların yapılmasının ardından sorunsuz bir şekilde çalıştığı gözlemlenmiştir. Şekil 4.38'de 5DOF robot kolun 0, 0, 0, 0, 0 açılarındaki pozisyonu gösterilmiştir. Şekil 4.39'da (20°, 20°, -40°, 0, 50°) açılarındaki, Şekil 4.40'da (50°, 40°, -80°, 0, 80°)açılarındaki, Şekil 4.41'de (45°, 80°, -90°, 10°, 100°) açılarındaki pozisyonu gösterilmiştir.

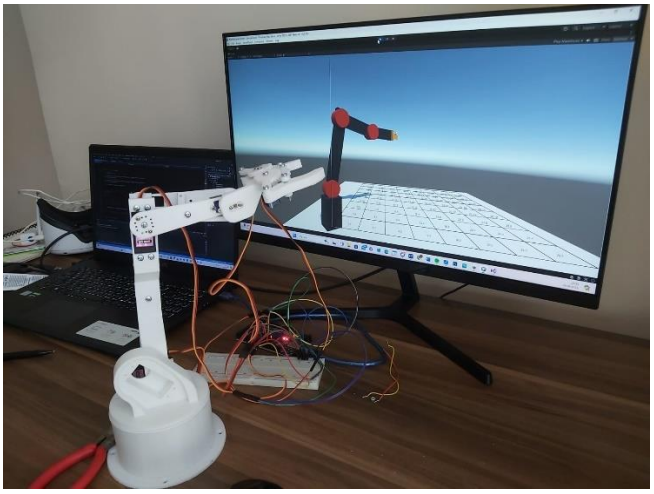


Şekil 4.38: 0, 0, 0, 0, 0 açılarındaki pozisyonu.



Şekil 4.39: 20°, 20°, -40°, 0, 50° açılarındaki pozisyonu.

Şekil 4.40: 50°, 40°, -80°, 0, 80° açılarındaki pozisyonu.



Şekil 4.41: 45°, 80°, -90°, 10°, 100° açılarındaki pozisyonu.

5. TARTIŞMA

Yapılan tez çalışmasında özellikle endüstriyel alanda kullanımı giderek yaygınlaşan, gelişen robot kontrol teknolojisinden faydalanılmıştır. Endüstri 4.0 ile beraber insan robot etkileşimi baz alınarak, ihtiyaca yönelik farklı serbestlik derecelerine (DOF) sahip robot kollar oluşturulmaktadır. Günümüzde hemen her sektörde kullanılan bu robotlar, yapay zeka ve diğer teknolojilerle daha verimli, esnek ve akıllı hale gelmektedir [3,4]. Sanal Gerçeklik alanında, literatür incelendiğinde günümüzde en çok tartışılan konu insan robot etkileşimi ve insan robot iş birliği [10] robot kol modelleri üzerine çeşitli çalışmaların yapıldığı görülmektedir.

Literatürde sanal gerçeklik (VR) alanında endüstri, sağlık, eğitim gibi alanlarda başlayan yeni çalışmalar görülmektedir. Bu çalışmaların dışında yapılan tez çalışmasına da benzer şekilde özellikle endüstriyel robotların kullanımı, insan robot etkileşimi ve insan robot iş birliği [10] üzerine çalışmalar bulunmaktadır. Sanal gerçeklik (VR-Virtual Reality) alanının tanınmaya başlamasıyla gerçek dünyada kullanılan birçok unsurun kullanıcı (operatör) eğitimi, prototip test etme, endüstriyel süreci izleme, iş güvenliği risk analizlerinin yürütülmesi, insan-robot etkileşimi (HRI) [9] ve insan-robot iş birliği (HRC) [13] üzerine robot davranışlarını incelemek gibi çalışmalar yapılmıştır. Gerçekte var olan orijinal sistemlere müdahale edilmeden maliyet ve zaman kayıplarına yol açan bu süreçler sanal ortamlarda test edilmiştir.

Sanal ortamda kullanılan Unity 3D programının [7], literatürde benzer çalışmalar için de tercih edildiği görülmektedir. Unity ve ROS arasında bir köprü mekanizması, basitleştirilmiş arayüzü daha kısa sürede etkileyici bir VR ortamında geliştirmelerine olanak tanımaktadır [12,16]. Gerçekte kullanılan bir robot kolunun sanal ortamda oluşturulması, kolaylık ve etkileşim açısından iyileştirmeler sunar ve akademik derslerde kullanılabilir. Sanal laboratuvarlar oluşturularak öğrencilerin teorik açıklamaları daha iyi anlaması sağlanabilir.

Sanal Gerçeklik (VR) alanının yeni olması ile beraber literatür incelendiğinde yapılan her çalışma yetersiz, bir ilk niteliğinde ve gelişime açık projeler olduğu görülmektedir. Bu nedenle, bu tez çalışması, sanal gerçeklik ve işbirlikçi robot kontrolü alanında yeni bir bakış açısı sunmayı hedeflemektedir. Yapılan tez çalışmasında ve literatürde de görüleceği üzere, Sanal gerçeklik (VR) tekniklerinin gelecekteki programlama ve robotik sistemlerin uzaktan kontrol uygulamaları için büyük vaatler sunmaktadır [17,18].

6. SONUÇ VE ÖNERİLER

Gerçek dünyada oluşturulan robot manipülatörün, bu çalışma kapsamında gösterilen ters kinematik hesaplarının aslında eksik kaldığı görülmüştür. Bazı özel durumlar meydana gelebilir; eksen yönleri pozisyona göre değişebilir ve geometrik hesaplamalarda bazı konumlarda açılar hesaplanamayabilir. Bu durum, her robot kol modelinde farklılık gösterebilir. Bu nedenle, çalışma amacını göz önünde bulundurarak sürekli iyileştirmeye yönelik test çalışmalarına devam edilmesi gerektiği öngörülmüştür. Ölçümlerin daha doğru olabilmesi açısından donanım geliştirilebilir ve sensörlerden faydalanılabilir. Bu şekilde, daha hassas ve daha doğru bir robot kol modeli oluşturulabilir. Bu çalışmada sanal ve gerçek ortam etkileşimi baz alındığı için oluşturulan model tasarım ve donanımı üzerine prototip oluşturacak nitelikte ek bir çalışma yapılmamıştır.

Sanal ortamda kontrol ilk olarak mobil ivmeölçer kullanılarak denenmiştir. Her ne kadar mobil ivmeölçer kullanımı tez çalışmasının kapsamı içinde yer almasa da, herkesin elinde bulunan bir cihaz olan telefonu bir kontrol aracı olarak denemek istedik. Mobil ivmeölçer veya ivmeölçer sensörü ile eksenlerde hareket, bu cihaza eğim vererek sağlanabiliyor. Ancak, bu durumun kullanıcı açısından sanal ortamdaki manipülatörün kontrolünü zorlaştırdığı gözlemlenmiştir. Bu nedenle, sanal ortam için tasarlanmış ve daha karmaşık bir yapıya sahip olan, tezin ana amacına da uygun olan sanal gerçeklik konsolu kullanılarak çalışmalara devam edilmiştir. Sanal gerçeklik konsollarına alternatif olarak mobil ivme ölçer yerine; fare, yön tuşları veya oyun konsolları tercih etmek daha hassas güvenilir bir kontrolü de sağlar.

Sanal ortamdaki manipülatörün aks boyutları, gerçektekine uyacak şekilde orantılı olarak küçültülmüştür; bu çalışma için 1/4 oranında küçültme gerçekleştirildi. Çünkü sanal ortam ile gerçek boyutlar farklılık göstermektedir. Gerçekteki robot kol modelinin boyutları milimetre veya cinsinden belirlenerek oluşturulurken, sanal ortamda boyutlar birim metre olarak geçmektedir. Gerçek dünya ile sanal ortamın etkileşimli çalışacağı bir düzende, sanal ortam, gerçek dünyadaki çalışma alanına göre sınırlandırılmalıdır.

Gerçekte oluşturulan manipülatör, sanal ortamdan gelen konum bilgilerine göre hareket etse de, gerçek ve simülasyon modelleri hedefe ulaştıktan sonra aynı görünüme sahip olmayabiliyordu. Bu durumun nedeni, sanal ortamda kullanılan programın sol el kuralına göre dönme hareketi yapması olarak tespit edildi. Gerçek dünyada sağ el kuralı geçerli iken, sanal ortamda sol el kuralının geçerli olması, sanal

ortamdaki eksen yönlerinin gerçekteki eksen yönlerinden farklı olmasına sebep oluyor. Örneğin, gerçek uzayda X, Y, Z eksenleri, sanal ortamda sırasıyla X, Z, Y şeklinde tanımlanıyor. Ayrıca, sanal ortamda eklem dönme hareketi saat yönündeyken, gerçek dünyada saat yönünün tersine hareket sağlanıyor. Bu çalışmanın temelini koordinat sistemleri, eksenlerin yönleri, bu eksenlerdeki dönme hareketi oluşturduğu için ölçümlerin doğru ve gerçek fiziksel koşullara uygunluğunun sağlanmasında zorluk yaşanmıştır. Çünkü çalışma süresince iki ortamın da sistemine adapte olmak, sürekli iki yönlü düşünmek gerekmektedir. Bu yüzden, sanal ortamda bir grid sistemi oluşturuldu ve kamera mümkün mertebe sabit tutularak çalışmalara devam edildi. Sanal ortamdan alınan koordinat bilgileri gerçek koşullara uyacak şekilde gerekli dönüşümler yapılarak gönderildi.

Robot manipülatörlerin çalışma sonuç grafiklerinden de görüleceği üzere sanal ortamdaki manipülatör gerçek dünyadakine kıyasla çok uyumludur. Sanal ortamdaki manipülatörde uç efektör konumlarının eksenleri ileri kinematik yöntemi ile hesaplanan eksenlerle neredeyse bire bir aynıdır. Genellikle 0.01 - 0.03 aralığında sapma olurken, en fazla 0.19'luk bir sapma gözlemlenmiştir. Gerçek dünyadaki manipülatörde ise 0.5 – 3cm aralığında sapmaların meydana geldiği gözlemlenmiştir. Açılarda ise özellikle 3. eklemin açıları farklılık göstermiştir, ancak uç efektör beklenen ve doğru eksen konumlarında yer aldığı için bu sapmalar göz ardı edilmiştir.

Sanal ortamdaki sonuçların gerçek dünyadakine kıyasla daha uyumlu olmasının sebebi olarak, sanal ortamda fizik koşullarının daha esnek olmasını söyleyebiliriz. Sanal ortamda, fiziksel modeller idealize edilmiştir ve bu durum, manipülatörün hareketlerinin daha öngörülebilir ve tutarlı olmasını sağlar. Gerçek dünyada karşılaştığımız elektronik donanımda meydana gelen ısınma, haberleşme gecikmeleri, mekanik sürtünmeler, motorun çektiği akımın değişkenliği, yapılan ölçüm ve kullanılan ölçüm aletlerinin doğruluğu gibi parametreler sanal ortamda göz ardı edilir. Bu nedenle, sanal ortamda elde edilen sonuçlar daha tutarlı ve öngörülebilirdir.

KAYNAKLAR

[1]. D'Souza, A., Vijayakumar, S., & Schaal, S. (2001). Learning inverse kinematics. *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium*. 10.1109/iros.2001.973374

[2]. Saydulu Kolasani (2024). Unleashing Exponential Intelligence: Transforming Businesses through Advanced Technologies. International Journal of sustainable Development Through AI, ML and IoT. https://ijsdai.com/index.php/IJSDAI/article/view/42/26

[3]. Mikolajczyk T, Mikołajewska E, Al-Shuka HFN (2022). Recent advances in bipedal walking robots: review of gait, drive, sensors and control systems. *Sensors 2022*, 22(12). 10.3390/s22124440

[4]. Gia-Hoang Phan (2024). Integrating long short-term memory for optimal control of 6-DOF welding robot arm. 10.1177/16878132241260525

[5]. Mu Lin, Lijun Sun, Yu Ding. (2020). Construction of Robotic Virtual Laboratory System Based on Unity3D. *IOP Conference Series: Materials Science and Engineering*. 10.1088/1757-899X/768/7/072084.

[6]. Michael Grieves (2016). Origins of the Digital Twin Concept. 10.13140/RG.2.2.26367.61609

[7]. Havard, V., Jeanne, B., Lacomblez, M., & Baudry, D. (2019). Digital twin and virtual reality: a co-simulation environment for design and assessment of industrial workstations. *Production & Manufacturing Research*, 7(1), [472–489]. 10.1080/21693277.2019.1660283.

[8]. Tao, F., Zhang, H., Liu, A., & Nee, A. Y. C. (2019). Digital Twin in Industry: State-of-the-Art. *IEEE Transactions on Industrial Informatics*, 15(4), [2405–2415]. 10.1109/tii.2018.2873186

[9]. Bolano, G., Juelg, C., Roennau, A., & Dillmann, R. (2019). Transparent Robot Behavior Using Augmented Reality in Close Human-Robot Interaction. *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. 10.1109/ro-man46459.2019.8956296

[10]. Freund, E., & Rossmann, J. (1999). Projective virtual reality: bridging the gap between virtual reality and robotics. *IEEE Transactions on Robotics and Automation*, 15(3), [411–422]. doi:10.1109/70.768175

[11]. Lanier, J., & Biocca, F. (1992). An Insider's View of the Future of Virtual Reality. *Journal of Communication*, 42(4), [150–172]. 10.1111/j.1460-2466.1992.tb00816.x

[12]. Mizuchi, Y., & Inamura, T. (2017). Cloud-based multimodal human-robot interaction simulator utilizing ROS and unity frameworks. *2017 IEEE/SICE International Symposium on System Integration (SII)*. 10.1109/sii.2017.8279345

[13]. Dianatfar, M., Latokartano, J., & Lanz, M. (2021). Review on existing VR/AR solutions in human–robot collaboration. *Procedia CIRP* [407–411]. 2020.05.259 10.1016/j.procir.2020.05.259.

[14]. R. Codd-Downey, P. Mojiri Forooshani, A. Speers, H. Wang and M. Jenkin. (2014). From ROS to Unity: leveraging robot and virtual environment middleware for immersive teleoperation. 10.1109/icinfa.2014.6932785.

[15]. Kim, H.-S., & Song, J.-B. (2014). Multi-DOF Counterbalance Mechanism for a Service Robot Arm. *IEEE/ASME Transactions on Mechatronics*, 19(6), [1756–1763]. 10.1109/tmech.2014.2308312

[16]. Sita, E., Horvath, C. M., Thomessen, T., Korondi, P., & Pipe, A. G. (2017). ROS-Unity3D based system for monitoring of an industrial robotic process. *2017 IEEE/SICE International Symposium on System Integration (SII)*. 10.1109/sii.2017.8279361.

[17]. Roldán, J. J., Peña-Tapia, E., Garzón-Ramos, D., de León, J., Garzón, M., del Cerro, J., & Barrientos, A. (2018). Multi-robot Systems, Virtual Reality and ROS: Developing a New Generation of Operator Interfaces. *Robot Operating System (ROS)*, [29–64]. 10.1007/978-3-319-91590-6_2

[18]. Marzano, A., Friel, I., Erkoyuncu, J. A., & Court, S. (2015). Design of a Virtual Reality Framework for Maintainability and Assemblability Test of Complex Systems. *Procedia CIRP*, 37, [242–247]. 10.1016/j.procir.2015.08.067.

[19]. Lee, C. s. g., & Ziegler, M. (1984). Geometric Approach in Solving Inverse Kinematics of PUMA Robots. *IEEE Transactions on Aerospace and Electronic Systems*, AES-20(6), [695–706]. 10.1109/taes.1984.310452

[20]. S. KuCuk dan Z. Bingul, (2004). The inverse kinematics solutions of industrial robot manipulators. *Proceedings of the IEEE International Conference on Mechatronics*. 10.1109/ICMECH.2004.1364451

[21]. Y Kayabaşı (2002). Sanal gerçeklik ve eğitim amaçli kullanilmasi. *Turkish Online* 4(3), [151 – 166]. http://www.tojet.net/volumes/v4i3.pdf#page=151

[22] Adriana Salatino, Claudio Zavattaro ,Roberto Gammeri, Emanuele Cirillo, Maria Luisa Piatti , Maria Pyasik, Hilary Serra, Lorenzo Pia, Giuliano Geminiani, Raffaella Ricci (2023). Virtual reality rehabilitation for unilateral spatial neglect: A systematic review of immersive, semi-immersive and non-immersive techniques. 10.1016/j.neubiorev.2023.105248

[23] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar (2005). *Robot Modeling and Control*. https://www.researchgate.net/profile/Mohamed_Mourad_Lafifi/post/How_to_avoid_singular_configurations/attachment/59d6361b79197b807799389a/AS%3A386996594855942%401469278586939/download/Spong+-+Robot+modeling+and+Control.pdf

[24]. Hayat, A. A., Chittawadigi, R. G., Udai, A. D., & Saha, S. K. (2013). Identification of Denavit-Hartenberg Parameters of an Industrial Robot. *Proceedings of Conference on Advances In Robotics - AIR '13*. 10.1145/2506095.2506121

[25] Hock, O., Drgona, P., & Paskala, M. (2014). Simulation model of adjustable arm using Denavit-Hartenberg parameters. 10.1109/elektro.2014.6847

[26]. Tadej Bajd, Matjaž Mihelj, Jadran Lenarcic, A. Stanovnik, Marko Munih (2010). Homogenous transformation matrices 10.1007/978-90-481-3776-3_2

[27] Sharma, A. (2018). Guided Stochastic Gradient Descent Algorithm for inconsistent datasets. *Applied Soft Computing.* 10.1016/j.asoc.2018.09.038

[28]. G. Oke; Y. Istefanopulos (2001). .Gradient-descent based trajectory planning for regulation of a two-link flexible robotic arm. *2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Proceedings.* 10.1109/aim.2001.936807

[29] De Luca, A., & Oriolo, G. (1990). The Reduced Gradient Method for Solving Redundancy in Robot Arms. IFAC Proceedings Volumes. 10.1016/s1474-6670(17)51725-5

[30] Anonim (2023, Kasım 27). Arduıno Based Robot Arm. *Cults.* https://cults3d.com/en/3d-model/various/arduino-based-robot-arm-howtomechatronics

[31] Allen N. (2019, Mayıs 20). Robotic Arm. *GrabCad*. https://grabcad.com/library/robotic-arm-163

EKLER

EK 1. Arduino kalibrasyon kodları

```
void Motor_Control() // Eklem Hareketi
{
  servoMotor1.writeMicroseconds(Theta(0.0));
  delay(3000);
  servoMotor1.writeMicroseconds(Theta(180.0));
  delay(3000);
}
//motor açısı 0-180 arası
int Theta(float Angle)
{
  int Compare;
  int min_comp=400;
  int max_comp=2500;
  int min_angle=0;
  int max_angle=180;
  Compare = (((max_comp-min_comp) / (max_angle - min_angle))*(Angle-min_angle)) +
min_comp;
  return Compare;
}
```

EK 2. Python Homojen Dönüşüm Matris hesaplaması

```
T0_1 = [
    [np.cos(T1), 0, np.sin(T1), 0],
    [np.sin(T1),0, -np.cos(T1), 0],
    [0, 1, 0, d1],
    [0, 0, 0, 1]
]
T1_2 = [
    [np.cos(T2), -np.sin(T2), 0, r2 * np.cos(T2)],
    [np.sin(T2), np.cos(T2),0 , r2 * np.sin(T2)],
    [0, 0, 1, 0],
    [0, 0, 0, 1]
]
T2_3 = [
    [np.cos(T3), -np.sin(T3), 0, r3 * np.cos(T3)],
    [np.sin(T3), np.cos(T3),0 , r3 * np.sin(T3)],
    [0, 0, 1, 0],
    [0, 0, 0, 1]
]
T0_2 = np.dot(T0_1, T1_2)
T0_3 = np.dot(T0_2, T2_3)
end_effector_position = T0_3[:3, 3]
```

EK 3. Arduino Ters Kinematik kodları

```
void CalculateKinematic()
{
      X=26.01;
      Y=0.01;
      Z=8.9;

      T1 = atan2(Y, X); //Eq.1
      r1=Z-a1;
      r2=sqrt((X*X)+(r1*r1)); // Eq. 3
      phi1=atan2(r1,X);

      // Cos değeri [-1,1] aralığında olmalıdır
      acosInput1 = ((a2 * a2) + (r2 * r2) - (a3 * a3)) / (2 * a2 * r2);
      acosInput2 = ((a2 * a2) + (a3 * a3) - (r2 * r2)) / (2 * a2 * a3);

      phi2 = acos(acosInput1); // Eq. 2
      phi3 = acos(acosInput2);

        // (90,0,T3) konumunda eksenler değişiyor. Burada Theta3 açısı farklı şekilde hesaplanır
      if(0.04>=X && X>=0.00)
      {

          if( r1 >= 0.00 && r1 <= 0.04 && r2 >= 0.00 && r2 <= 0.04)
          {
            T2=0.0;
            T3=0.0;
          }
          else{
              eq=a2+a1-Z;
              T3=atan2(Y,eq);
              T3=-1*T3;
            }
      }
      else{
          T2 = phi1 + phi2; // Eq. 6
          T3 = phi3 - PI;
      }
        angle_T1= T1*180.0/PI;
        angle_T2= T2*180.0/PI;
        angle_T3= T3*180.0/PI;

        Motor_Kinematik_Control();

}
```

EK 4. Unity Gradyan hesaplama kodları

```
    //Root of the armature
     public Joint m_root;

     //End effector
     public Joint m_end;
     //hedef nesne
     public GameObject m_target;

     //end effector hedef arası minimum mesafe eşiği
     public float m_threshold = 0.05f;
     //kök ekleme uygulanan döndürme oranı (daha hızlı dönmesi için koyduk)
     public float m_rate = 5.0f;
     public int m_steps = 20;

     //yazdırmak için bileşenler
     public Text Distance1Text;
     public Text Distance2Text;

     //Kök eklem dönüşüne göre mesafe değişiminin eğimi hesaplanır
     float CalculateSlope(Joint _joint)
     {
         float deltaTheta = 0.01f;
         float distance1 = GetDistance(m_end.transform.position, m_target.transform.position);
         Distance1Text.text = distance1.ToString();
         _joint.Rotate(deltaTheta);


         float distance2 = GetDistance(m_end.transform.position, m_target.transform.position);
         Distance2Text.text = distance2.ToString();
         _joint.Rotate(-deltaTheta);
         return (distance2 - distance1) / deltaTheta;
     }


     void Update()
     {
         for (int i = 0; i < m_steps; i++)
         {
             if (GetDistance(m_end.transform.position, m_target.transform.position) > m_threshold)
             {
                 Joint current = m_root;
                 while (current != null)
                 {
                     float slope = CalculateSlope(current);
                     current.Rotate(-slope * m_rate);
                     current = current.GetChild();
                 }

             }
         }


     }

     //öklid
     float GetDistance(Vector3 _point1, Vector3 _point2)
     {
         return Vector3.Distance(_point1, _point2);
     }
```

İNTİHAL RAPORU İLK SAYFASI

Eda Derya Toper

ORİJİNALLİK RAPORU

%8	%5	%4	%4
BENZERLİK ENDEKSİ	İNTERNET KAYNAKLARI	YAYINLAR	ÖĞRENCİ ÖDEVLERİ

BİRİNCİL KAYNAKLAR

1	Submitted to The Scientific & Technological Research Council of Turkey (TUBITAK) Öğrenci Ödevi	%1
2	acikbilim.yok.gov.tr İnternet Kaynağı	%1
3	Submitted to University of Bristol Öğrenci Ödevi	%1
4	www.coursehero.com İnternet Kaynağı	%1
5	Submitted to Marmara University Öğrenci Ödevi	<%1
6	Erkalkan, Ercan. "Esnek Programlama yaklasimlari Ile Oyun gelistirme", Marmara Universitesi (Turkey), 2021 Yayın	<%1
7	Submitted to Beykent Universitesi Öğrenci Ödevi	<%1
8	Submitted to Southern University And A & M College Öğrenci Ödevi	<%1

ÖZGEÇMİŞ