T.C.

MARMARA ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

GÖRÜNTÜ İŞLEME VE YAPAY ZEKA İLE OTOMATİK HEDEF ALAN TARET

Hazırlayanlar

Yusuf Kini – 171219007

Hüseyin Özgür Ceylan – 170219036

Tez Danışmanı

Doç.Dr. Uğur DEMİR

İstanbul, 2023

T.C.

MARMARA ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME TEZİ KABUL VE ONAY BELGESİ

Bölümümüz 171219007 ve 170219036 numaralı öğrencileri Yusuf Kini, Hüseyin Özgür Ceylan, "Görüntü İşleme ve Yapay Zeka ile Otomatik Hedef Alan Taret" başlıklı Bitirme Tezi çalışması aşağıdaki jüri üyeleri tarafından Mekatronik Mühendisliği Bölümü 'nde Bitirme Tezi olarak Oy Birliği / Oy Çokluğu ile kabul edilmiştir.

Danışman: Doç.Dr. Uğur Demir

İmzası :

Tez Jürileri

1. Üye : İmzası :

2. Üye : İmzası :

3. Üye : İmzası :

Tezin sunulduğu tarih :

Bitirme Projesi dersi kapsamında yapılan bu Tez çalışması, ilgili jüriler tarafından değerlendirme sonucunda Mekatronik Mühendisliği Bölümü 'nde Bitirme Tezi çalışması olarak kabul edilmiştir.

Bölüm Başkanı :

İmzası :

DOĞRULUK BEYANI

Bitirme Tezi olarak sunduğumuz bu çalışmayı tüm akademik kurallara ve Marmara Üniversitesi Teknoloji Fakültesi Bitirme Projesi Yönergesi ‘ne uygun olarak gerçekleştirdiğimizi ve sunduğumuzu, bu kurallar ve ilkelere aykırı hiçbir yol ve yardıma başvurmaksızın bizzat hazırladığımızı beyan ederiz.

Tezimizle ilgili yaptığımız beyana aykırı bir durum saptanması durumunda ortaya çıkacak tüm ahlaki ve hukuki sonuçlara katlanacağımızı bildiririz.

Yusuf Kini İmzası :

Hüseyin Özgür Ceylan İmzası :

TEŞEKKÜR

Lisans tezimizde bize bilgi birikimi ve yol göstericiliğiyle yardımcı olan, zor zamanlarımızda hiçbir zaman yardımını esirgemeyen, kıymetli zamanını ayırıp sabırla ve büyük bir ilgiyle bizlere faydalı olmak için elinden gelenin fazlasını sunan, bir sorun yaşadığımız zaman çekinmeden yanına gidebildiğimiz, iyi insanlığıyla, güler yüzüyle, bizlere verdiği değer ve saygıyla Sayın Danışmanımız Doç.Dr. Uğur DEMİR ‘e , Sayın Dr.Öğr.Üyesi Barış DOĞAN ‘a ve Sayın Dr.Öğr.Üyesi Gazi AKGÜN ‘e teşekkürlerimizi borç biliriz.

ÖZET

Bu projede, algıladığı nesnenin takibini yapan ve algılanan nesneye yönelik plastik nerf mermisi fırlatan bir Taret tasarlamak amaçlanmıştır. Tasarlanacak robot üzerinde, hedefin nesne takibini yapabilmesi amacıyla harici kamera olarak EpocCam uygulaması üzerinden Telefon kamerası kullanılacaktır. Kullanılan kamera ile video akışı üzerinden algılanan görüntüler (veya imgeler) bilgisayar ‘da bulunan OBS Studio üzerinden Python tarafına aktarılacaktır. Python tarafında yazılan kod dizini, OpenCV teknikleri ve eğitilen YOLOv8 modeline dayanarak, hedef nesnenin X ve Y eksenlerindeki koordinat bilgileri kullanılan Arduino Nano mikrodenetleyicisi tarafına aktarılacaktır. Hedefin X ve Y koordinatlarını alan Arduino Nano mikrodenetleyicisi, Taret Robot ‘un gerekli rotasyonları yapması ve hedefi takip etmesi için Taret üzerindeki servoları döndürecektir. Taret haznesinde bulunan nerf mermisi, servo motor yardımıyla ileriye ittirilecektir ve 2 DC motor vasıtasıyla hızlanma kazanarak hedefi vurması sağlanacaktır. Görüntünün işlenmesi bilgisayarın dahili ekran kartı ile gerçekleştirilecektir. Robot ‘un tasarımı FUSION 360 platformu üzerinden yapılacaktır. Tasarımı tamamlanan Taret Robot ‘un bütün parçaları FDM teknolojisine sahip 3B yazıcı ile üretilecektir.

ÖNSÖZ

Üzerinde çalışması tamamlanan Taret Projesinin montajı yapılmış ve Statik Stres analizi Fusion 360 üzerinden gerçekleştirilmiştir. İmalat ve montaj öncesinde, çizim ile ilgili gerekli ölçüm ve analizler yapılarak Robot ‘un mekanik aksamında olabilecek olası sorunların önüne geçilmeye çalışılmıştır. Görüntü işleme ve Nesne Tespiti tarafında, kullanılması gereken teknikler öğrenilmiştir. Veri tabanına yeşil elma resimleri tanıtılmış ve bu resimlere gerekli etiketleri eklenerek YOLOv8 ile eğitim yapılmıştır. Dahili kamera olarak, piksel kalitesi (1920 x 1080p) ve video akışında daha iyi bir görüntü kalitesi elde edebilmek amacıyla EpocCam uygulaması üzerinden iPhone 11 telefonu kullanılması uygun görülmüştür. Veri tabanına tanıtılan ve eğitilen Yeşil elma ‘nın daha iyi algılanabilmesi ve hedefin tespit edilebilmesi noktasındaki doğruluk değerinin arttırılması için Python tarafındaki parametreler deneme yanılma yöntemiyle en uygun değeri alabilmesi amacıyla değiştirilmiştir. Robot ‘un Arduino tarafındaki yazılımında olası hatalardan kaçınmak ve Robot ‘un mekanik kısıtlamalarını göz önünde bulundurmak kaydıyla, teorik bilgilere ve pratik üzerinden gerçekleştirilen gözlemlere dayanarak gerekli kapsülleme teknikleri uygulanmış ve rotasyon kodları yazılmıştır.

İÇİNDEKİLER

KISALTMALAR LİSTESİ

PWM : Pulse-Width Modulation

DC : Direct Current

Hz : Hertz

USB : Universal Serial Bus

USART : Universal Synchronous/Asynchronous Receiver/Transmitter

PC : Personal Computer

PWR : Power

GND : Ground

RGB : Red Green Blue Color Model

HSV : Hue Saturation and Value

IDE : Integrated Development Environment

UART : Universal Asynchronous Receiver/Transmitter

COM : Communications Port

CV : Computer Vision

Mbps : Megabits Per Second

RPM : Revolutions Per Minute

IC : Integrated Circuit

EMF : Electromotive Force

AC : Alternating Current

A : Amper

mm : millimeter

GUI : Graphical User Interface

YOLO : You Only Look Once

LTI : Linear Time-Invariant

ŞEKİL LİSTESİ

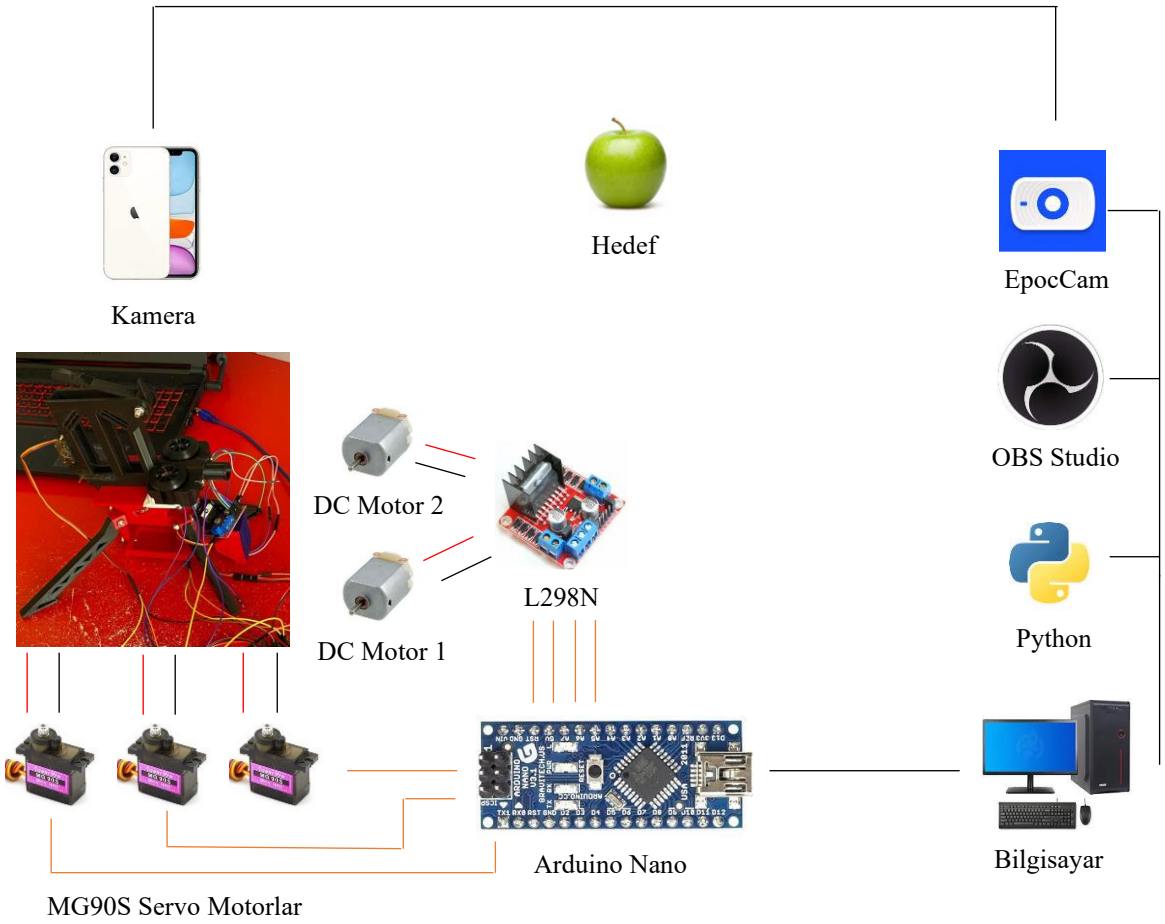
TABLO LİSTESİ

BÖLÜM 1

1.1 Giriş

Teknolojinin hızla gelişmesiyle birlikte Robotik alanındaki gelişmeler ve yenilikler Endüstri ‘den insanların kişisel hobilerine kadar çeşitli alanlarda yapılan çalışmaları kategorize etmiş ve bu alana olan ilgiyi arttırmış bulunmaktadır. Görüntü İşleme teknolojisi ve robotik entegrasyonu, makinelerin başarabileceği şeylerin sınırlarını zorlayarak eski tarihlerdeki bilim kurgu alemiyle sınırlı olan sofistike işlevsellikleri günümüzde mümkün kılmıştır. Bu gelişmeler kategorisinde yer alan taret robotları olağanüstü kreasyonlar, uyarlanabilirlik, çeviklik ve etkileşimli yetenekler sergilemektedir.

Genellikle savaş taretleri olarak adlandırılan taret robotları, akıllı sistemler ve mekanik çeviklik arasındaki sinerjiyi gözler önüne sermektedir. Bu sistemler, hedeflerini olağanüstü hassasiyetle takip etmek ve tespit etmek amacıyla tasarlanmıştır. İster Endüstriyel alanda kullanılmak için tasarlansın ister hobi amaçlı olarak tasarlansın, Taret Robotları mekanik tasarım inceliklerini ve Görüntü İşleme teknolojilerini içerisinde barındıran bir robotik sistem olarak karşımıza çıkmaktadır.



Şekil 1.1 Tasarlanan sistemin şema gösterimi

1.2 Projenin Amacı

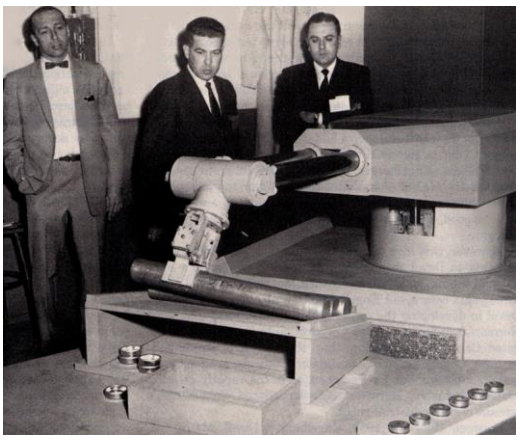
Bu Tez bağlamında amaç, gelişmiş Görüntü İşleme tekniklerinden, YOLOv8 teknolojisinden ve Arduino tabanlı mikrodenetleyici ‘den entegre bir şekilde yararlanarak bir Nerf Taret ‘in gelişimini keşfetmektir. Güçlü Görüntü İşleme algoritmalarından yararlanarak taret sistemi, yazılımın ve mekanik aksamın birlikte oluşturduğu entegrasyon ile hedefleri gerçek zamanlı olarak tespit etme ve takip etme yeteneğini kazanır.

Bu projede, Çağdaş Robotik Literatürüne yenilik ve farklı bir bakış açısı kazandırmak amacıyla hedeflediğimiz projeyi gerçek hayatta somutlaştırıyoruz. Bu tez, Görüntü İşleme vizyonunun robotik sistemlerle birleştirilmesindeki pratik uygulamalarını göstermeye ve Taret Robotiği alanındaki Literatüre katkı sunmaya çalışmaktadır.

1.3 Literatür Özeti

Robotiğin evrimi, otomatların ve mekanik cihazların insan veya hayvan hareketlerini taklit etmek için üretildiği eski uygarlıklara kadar uzanır. Bununla birlikte, modern robotik anlayışı 20. yüzyılda şekillenmeye başladı. 1950 ‘lerde “robotik” terimi, “Robotiğin Üç Yasasını (Three Laws of Robotics)” öneren Amerikalı bilim kurgu yazarı ve biyokimyager Isaac Asimov tarafından icat edildi. Robotların etik davranışlarına odaklanan bu yasalar, günümüzde Yapay Zeka ve Robotiğin entegrasyonu konusundaki tartışmaları etkilemeye devam ediyor.

Robotik tarihinin temel anlarından biri, 1961 ‘de George Devol ve Joseph Engelberger tarafından tanıtılan Unimate ‘in geliştirilmesiydi. Unimate, bir montaj hattında kullanılan, üretim süreçlerinde devrim yaratan ve dünya çapındaki endüstrilerde otomasyon için zemin hazırlayan ilk endüstriyel robot oldu.



Şekil 1.2 Unimate robot

1970 ‘lerde ve 1980 ‘lerde mikroişlemciler ve bilgi işlem gücündeki gelişmeler, robotiği yeni olasılık alanlarına itti. Robotlar, öncelikle kontrollü ortamlarda kullanılmaktan dinamik ve yapılandırılmamış alanları keşfetmeye geçiş yaparak uzay araştırmaları, tıp ve eğlence gibi alanlardaki uygulamaların önünü açtı.

Robotların çevrelerini algılamasını ve etkileşimde bulunmasını sağlamak için çok önemli olan Bilgisayar Görüşü Alanı (The Field of Computer Vision), 20. yüzyılın sonlarında önemli bir ilerleme kaydetti. Görüntü işleme ve model tanımadaki atılımlar, robotların görsel verileri yorumlamaları, özerkliklerini ve uyarlanabilirliklerini arttırmaları için zemin hazırladı.

Geçmişte yapılan çalışmalar ve hayata geçirilenler, günümüzdeki robotik ufkunu genişletti. Makine Öğrenimi (Deep Learning), Yapay Zeka (AI) ve gelişmiş sensörler gibi en yeni teknolojileri entegre etti. İşbirlikçi robotlar (cobots) çeşitli endüstrilerdeki insanlarla birlikte çalışırken, otonom araçlar ve dronlar çeşitli ortamlardaki robotik sistemlerin yeteneklerini sergiliyor.

Robotik alanındaki gelişmeler devam ettikçe AI, sensörler ve malzemelerdeki gelişmeler ise robotik alanındaki gelişmelerin üzerine dahil edildiğinde, robotların başarabilecekleri sınırlar gün geçtikçe zorlanmakta ve daha farklı bir boyuta evrilmektedir. Bu sayede Teknoloji, endüstri ve günlük yaşamdaki yeni sınırların kapıları açılmaktadır.

Robotlar ve robotik sistemler, benzersiz inovasyon ve entegrasyonun damgasını vurduğu bir çağın eşiğinde durmaktadır. Makine öğrenimi ve sinir ağları gibi gelişen teknolojiler, robotikte bir rönesans başlattı ve makinelerin insanlarla daha sorunsuz bir şekilde öğrenmesine, uyum sağlamasına ve iş birliği yapmasına olanak sağladı. Robotiğin önemli ve odak noktası, robotlara yalnızca işlevsel yetenekler değil aynı zamanda bilişsel yetenekler de kazandırmaya yöneldi. Duygusal açıdan akıllı makinelerin ve kişiselleştirilmiş etkileşimlerin önünü açtı. Robotik Bilimi sınırları aşmaya devam ettikçe, en ileri teknolojilerin birleşimi, robotların yalnızca verimlilik araçları değil aynı zamanda günlük yaşamlarımızda yol arkadaşları, eğitimciler ve işbirlikçileri olduğu bir gelecek vaat ediyor.

1.4 İş – Zaman Çizelgesi

Proje ‘nin İş – Zaman çizelgesi Tablo 1.1 ‘de sunulmuştur.

Tablo 1.1 İş – Zaman Çizelgesi

İŞ-ZAMAN ÇİZELGESİ / WORK-TIME TABLE																
İş No	**Yapılacak İşler**	**EKİM**				**KASIM**				**ARALIK**				**OCAK**			
		H1	H2	H3	H4	H 1	H 2	H3	H 4	H 1	H 2	H3	H4	H1	H2	H3	H4
1	Literatür taraması	x	x														
2	Taret Tasarımı ve Montajı		x	x	x												
3	OpenCV kütüphanesi incelenmesi ve metot belirlenmesi				x	x											
4	Bilgisayarın mikrodenetleyiciyle haberleşmesi ve ilk testlerin yapılması				x	x	x										
5	Hedef Nesnenin Tespitinin Gerçekleştirilmesi					x	x	x	x	x							
6	Kamera üzerinden gelen imgelerin işlenmesi							x	x	x							
7	Robot Kontrolünün Yapılması							x	x	x	x	x	x				
8	Robotun Nesne Takibinin ve taret fırlatmasının gerçekleştirilmesi											x	x	x	x		
9	Tuning işlemlerinin gerçekleştirilmesi												x	x	x	x	
10	Tez raporunun yazılması														x	x	x

1.5 Bütçe ve Gerekçesi

Proje ‘nin yapılması amacıyla temin edilen ürünler, fiyatlar ve gerekçeleri Tablo 1.2 ‘de sunulmuştur.

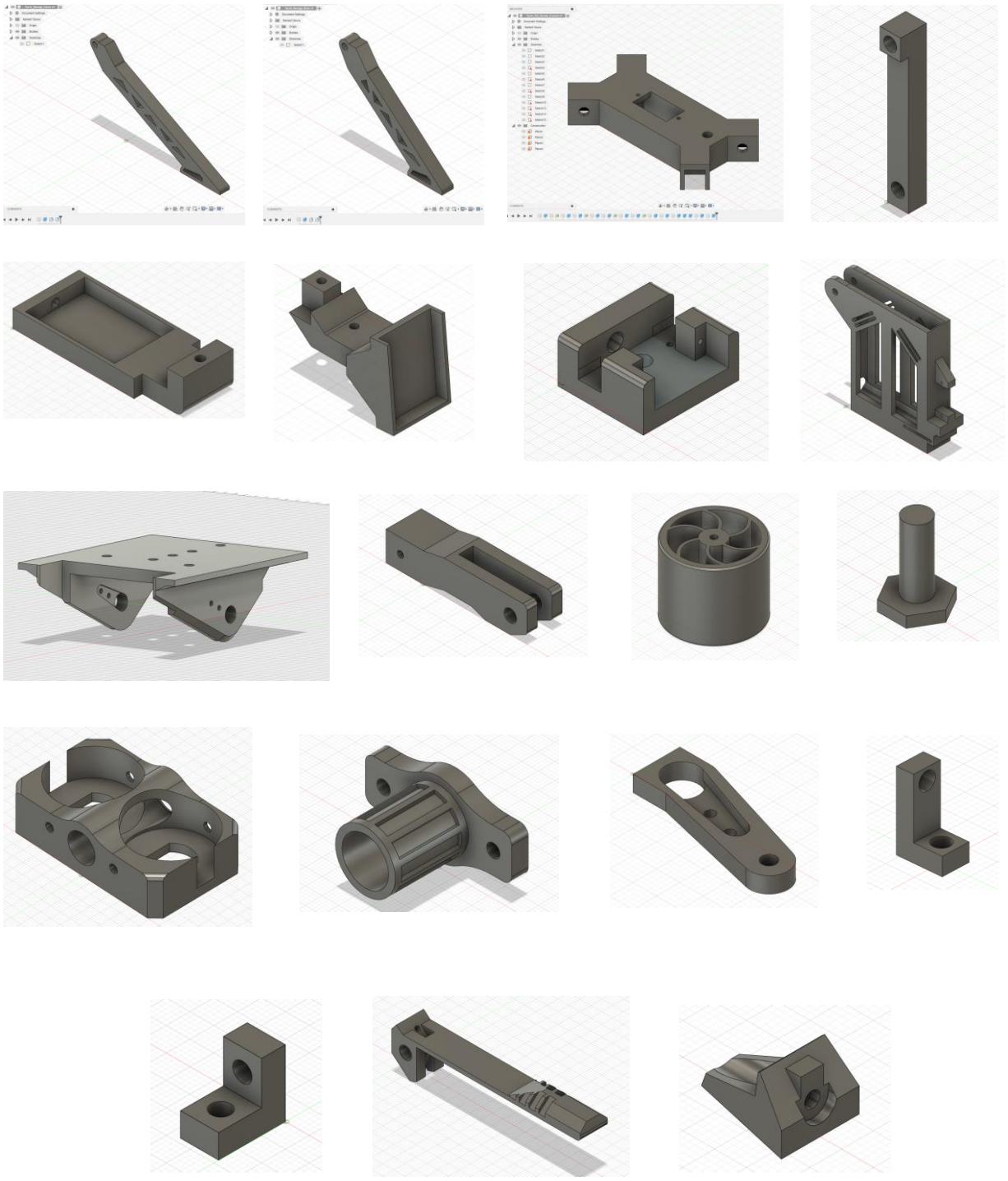
Tablo 1.2 Bütçe ve Gerekçe Tablosu

Temin Edilen Ürünler	**Ürün Fiyatı (TL + KDV)**	**Gerekçesi**	**Toplam Miktar**	**Toplam Maliyet (TL + KDV)**
3B (FDM) Baskı Maliyeti	3000	Tasarımın üretimi ve imalatı	1	9345,86
Arduino Nano	180	Mikrodenetleyici ihtiyacının karşılanması	1	
MG90S Servo Motor	95,18	Taret rotasyonu, mermi ittirme ve malzeme yedeklemek için alınacaktır	6	
DC Motor	37,30	Mermi ateş sistemi	5	
Metrik Civata	2	Montajın sabitlenmesi	100	
Metrik Somun	1	Montajın sabitlenmesi	50	
Jumper Kablo	25	Elektronik devrenin kablolanması	6	
L298N	61.09	Motor sürücü kartı	1	
5V 3A Adaptör	200	Devrenin beslenmesi	1	
12V 1A Adaptör	81,32	Sürücü kartı ve DC motorların beslenmesi	1	
Mekanik Kumpas	200	Sistemdeki parçaların ölçülmesi	1	
Mini Manuel El Matkabı	163,58	Parçalara delik açılması veya genişletilmesi	1	
5x5 Delikli Plaket	4,48	Elektronik devre kurulumu	3	
5x7 Epoxy Çift Taraflı PCB	16,78	Elektronik devre kurulumu	3	
1x40 12mm Erkek Header	2,24	Elektronik devre kurulumu	4	
1x40 180C Dişi Header	4,48	Elektronik devre kurulumu	4	
Soldex 0.75mm 200gr Lehim Teli	366,09	Elektronik devre lehiminin yapılması	1	
Lötfett Felder Lehim Pastası 50gr	103,67	Elektronik devre lehiminin yapılması	1	
Class Havya Temizleme Teli ZD-12W	129,31	Elektronik devre lehiminin yapılması	1	
DC125B3 Mini ON-OFF Anahtar 3P	6,71	Güç kaynağı veya sistemdeki akım kontrolü	4	
DC254K Jak	8,95	Elektronik devrenin ara bağlantısı	4	
Konnektör	2,61	Elektronik devrenin ara bağlantısı	4	
Taret mermisi	4,5	Taret mermisi	20	
Mini breadboard	5,59	Elektroniklerin takılması	4	
Kargo Giderleri	254	Siparişin ulaşma maaliyeti		
Harici Gereksinimler	3155,18	Herhangi bir hata durumuyla karşılaşıldığında farklı parçaların temin edilmesi, stok işlemleri ve ulaşım giderleri gibi harici harcamaların yapılması		

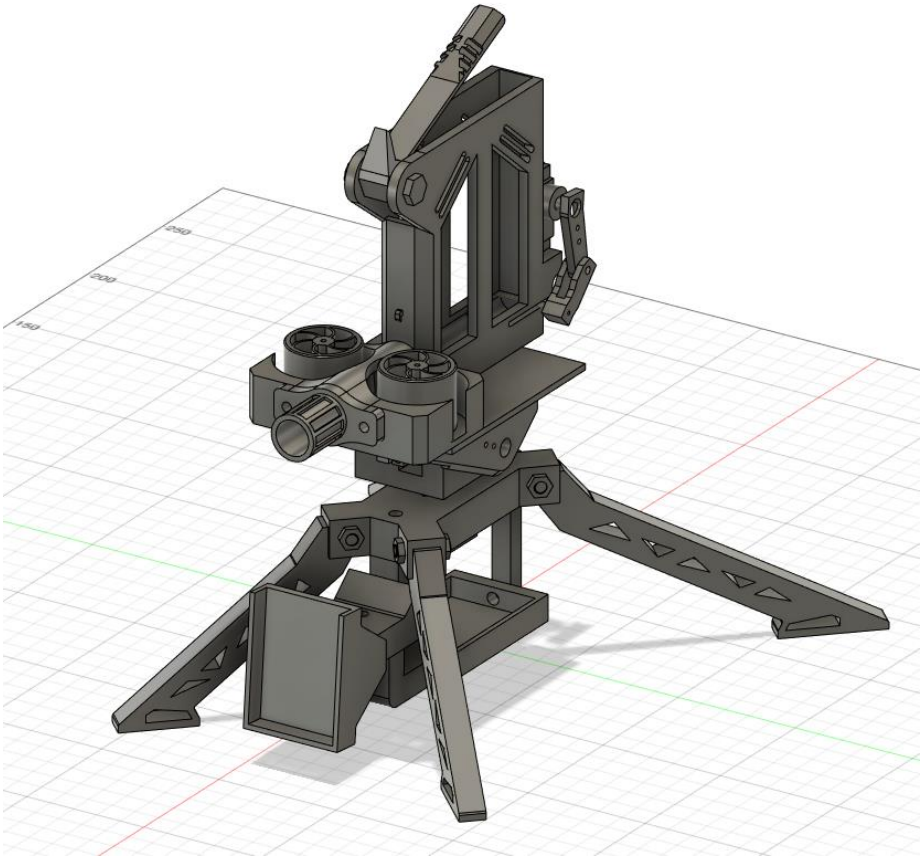
BÖLÜM 2

2.1 Mekanik Sistemin Tasarımı

Taret robotun tasarımı Autodesk Fusion 360 üzerinden çizilmiştir. Sistemde toplam 24 parça bulunmaktadır.



Şekil 2.1 Sistemin mekanik parçaları



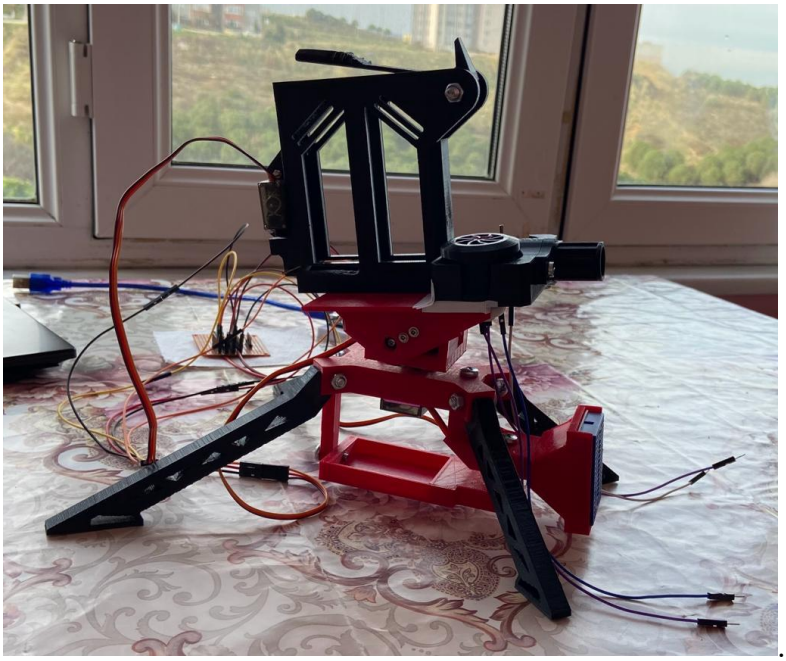
Şekil 2.2 Taretin Fusion 360 içerisindeki montajlanmış görseli

Tablo 2.1 Sistemin mekanik parça tablosu

Uzun Taret Bacağı	2
Kısa Taret Bacağı	2
Alt Gövde	1
Alt Gövde Ara Parça1	1
Alt Gövde Ara Parça2	1
Alt Gövde Ara Parça3	1
Alt Ara Parça	1
Üst Gövde Tutucu	1
Üst Gövde	1
Taret Haznesi	1
Taret Namlusu	1
Geri Tepme Kolu1	1
Geri Tepme Kolu2	1
DC Motor Tekerleği	2
DC Motor Tutucu	2
DC Motor Haznesi sabitleyici	2
DC Motor Haznesi	1
Dart Tutucu	1
Dart Tutucu Aparatı	1

2.2 Taret Robotun Görünümü

Montajlanan Taret ‘in görünümü Şekil 2.3 ve Şekil 2.4 ‘de verilmiştir.



Şekil 2.3 Taret Robot’un görünümü 1

Şekil 2.4 Taret Robot’un görünümü 2

2.3 Sistemin Stres Analizleri

Statik Analiz Nedir?

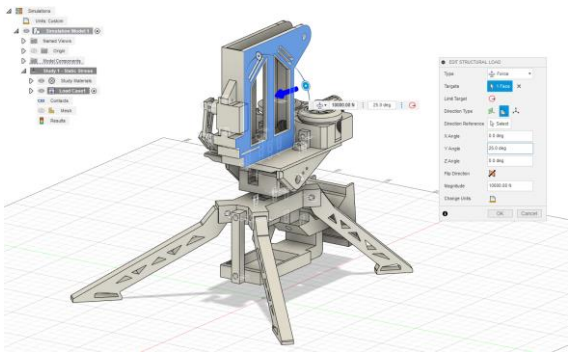
Statik analiz, mühendislik projelerinde kritik bir değerlendirme aracı olarak kullanılır. Tasarlanan sistemlerin yapısal bütünlüğü ve performansı hakkında öngörüler sunmaktadır. Görüntü İşleme (CV) ve Arduino destekli bir Nerf Taretini geliştirme bağlamında statik analiz, yapısal deformları, parçanın dayanabileceği kuvveti, stres miktarını ve emniyet katsayısı gibi parametreleri teorik olarak öngörebilmeyi ve statik yükleme koşulları altında oluşabilecek deformasyonların incelenmesini sağlamaktadır.

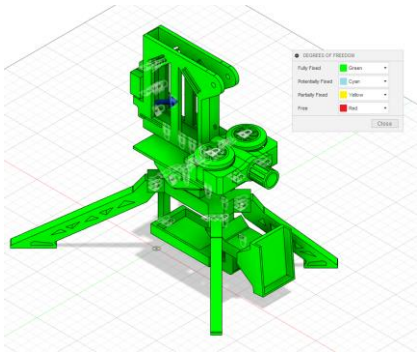
Bu analizler sayesinde, potansiyel zayıflıkları tahmin etmeyi, yapısal tasarımı optimize etmeyi, operasyon sırasında taretin güvenilirliğini ve kararlılığını sağlamayı amaçlıyoruz. Bu bölüm, taret tasarımındaki mekanik sağlamlığın değerlendirilmesinde statik analiz kullanımının önemini gözler önüne serecektir.

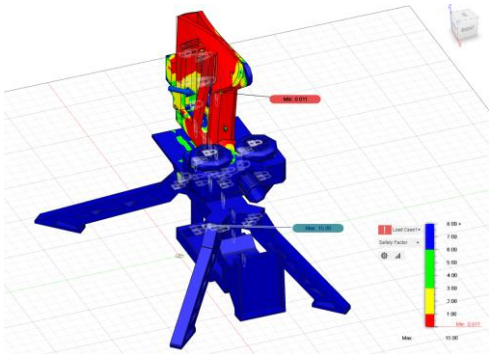
Taret robotun statik stres analizi Fusion 360 üzerinden gerçekleştirilmiştir. Taret ‘in montajlı (assembly) hali üzerinden ve bazı kritik parçaların tek başına incelenmesiyle gerekli analiz teknikleri uygulanmıştır. Robot ‘un parçalarına ve montajlı haline, kuvvet değeri 5000 N – 20.000 N arasında değişkenlik gösterecek şekilde ve farklı açılardan kuvvetin uygulanmasıyla statik analiz yapılması uygun görülmüştür. Bu prensibin uygulanmasıyla birlikte farklı koşullarda Robot ’un mekanik değerlendirilmesindeki farklılıklar üzerinde durulması ve gözlem yapılması amaçlanmıştır.

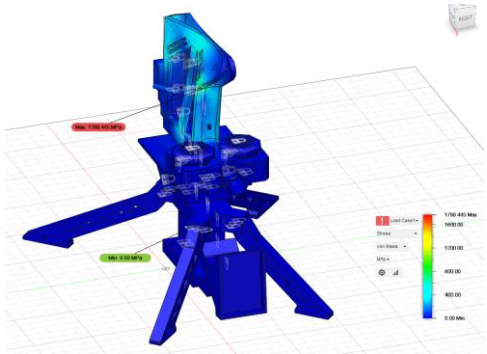
Robotun bütün parçalarının malzeme yapısı Fusion 360 ‘da “ABS Plastic” seçilmiştir.

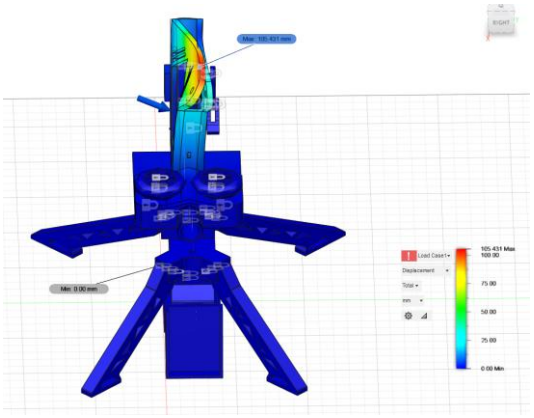
Statik Analiz 1: Sisteme uygulanacak kuvvet 10.000 N olarak belirlenmiştir. Kuvvet yüzeyi ve açısı belirlendikten sonra sistemin yapısal kısıtlamaları belirlenir. Bu işlemler sonucunda sistemin güvenlik katsayısı, stres ve yer değiştirme analiz sonuçları elde edilmiştir.





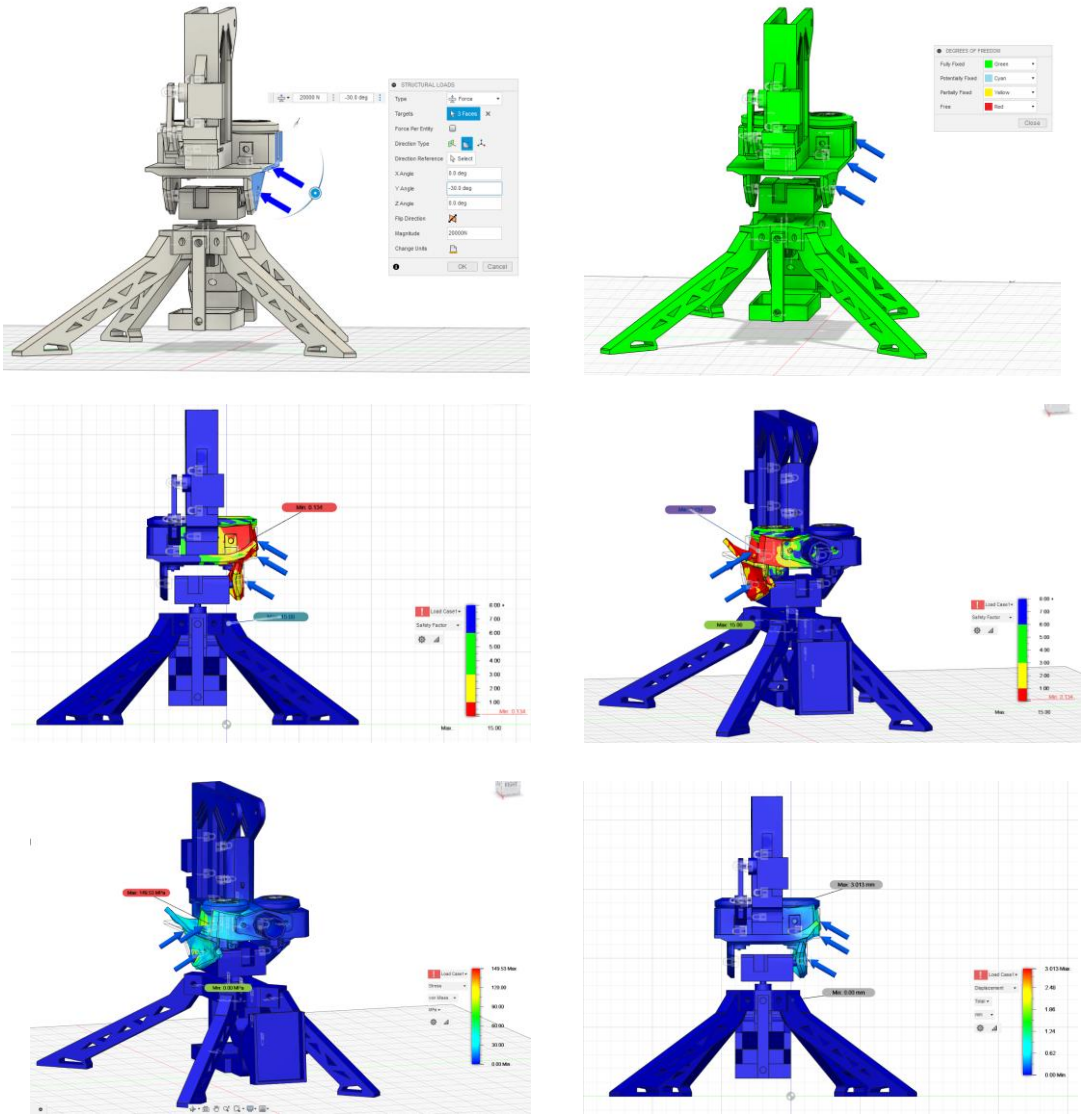






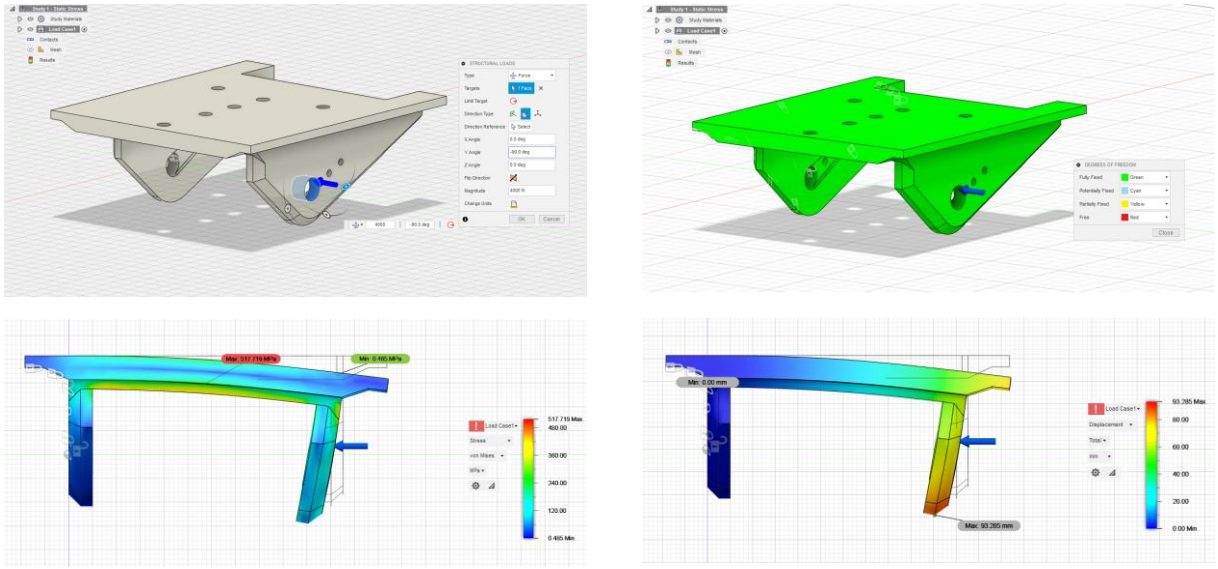
Şekil 2.5 Sistemin statik analiz sonuçları 1

Statik Analiz 2: Sisteme uygulanacak kuvvet 20.000 N olarak belirlenmiştir ve sistemin ikincil statik analizleri gerçekleştirilmiştir.



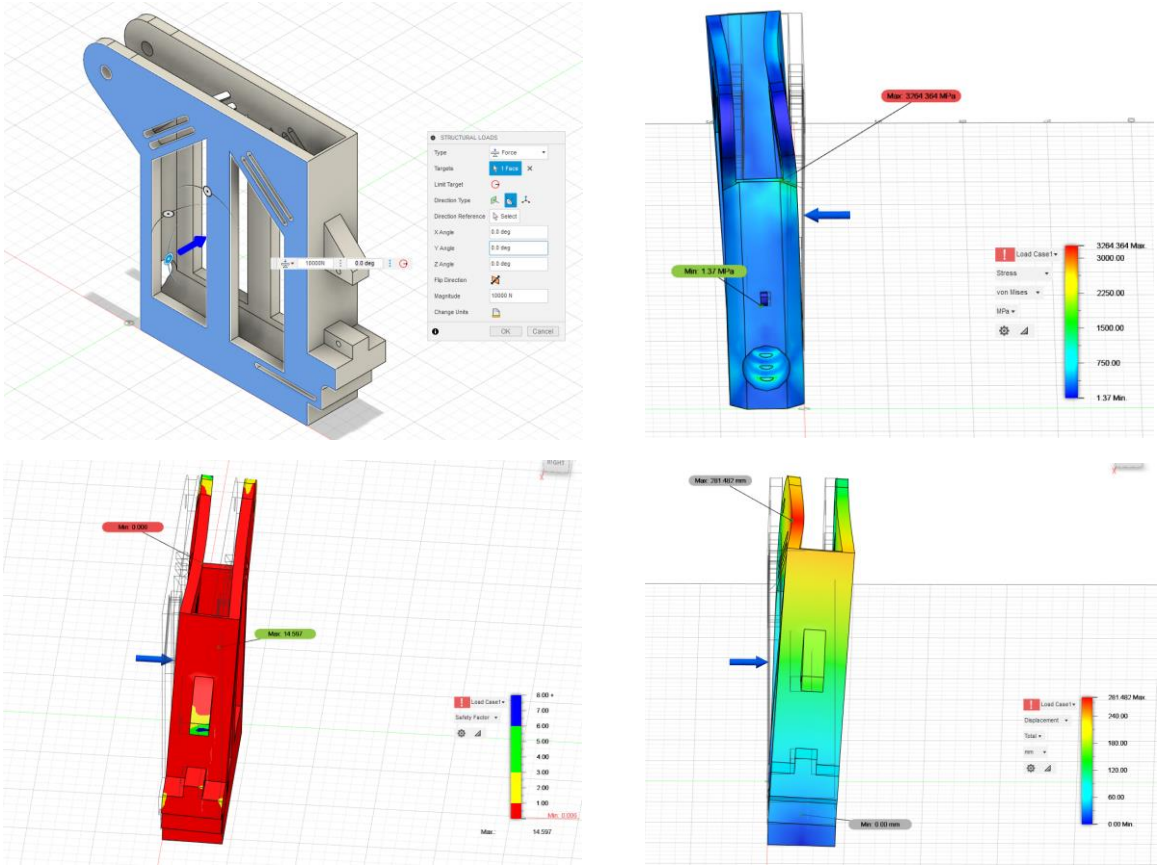
Şekil 2.6 Sistemin statik analiz sonuçları 2

Statik Analiz 3: Üst gövdeye uygulanacak kuvvet 4.000 N olarak belirlenmiştir ve mekanik parçanın statik analizleri gerçekleştirilmiştir.



Şekil 2.7 Mekanik parçanın statik analiz sonuçları 1

Statik Analiz 4: Taret haznesine uygulanacak kuvvet 10.000 N olarak belirlenmiştir ve mekanik parçanın statik analizleri gerçekleştirilmiştir.



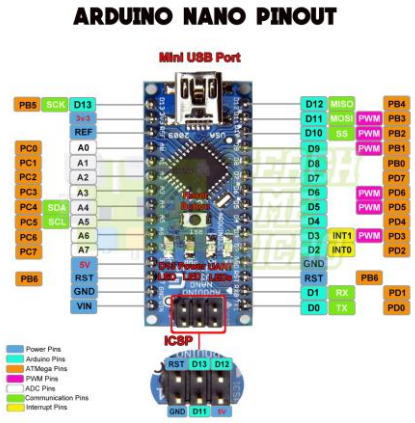
Şekil 2.8 Mekanik parçanın statik analiz sonuçları 2

BÖLÜM 3

3.1 Sistemde Kullanılan Malzemeler

3.1.1 Arduino Nano Mikrodenetleyici

ATmega328P temelli Arduino Nano mikrodenetleyici kartı, kompakt form faktörü ve zengin özellikleriyle ünlüdür. Bu da onu elektronik prototipleme ve gömülü sistem projelerini geliştirmenin temel taşı haline getirir. 14 dijital Giriş/Çıkış pini ve 8 analog pine sahip olan Arduino Nano, sensörler, aktüatörler ve çevre birimleri ile arayüz oluşturmak için çok yönlü bağlantı sunar. 32KB flash belleği, 2KB SRAM ve 1KB EEPROM 'u, program depolama ve veri işleme gibi işlemler için geniş alan sağlar. Arduino Nano, diğer cihazlarla kesintisiz etkileşimi kolaylaştıran UART, SPI ve I2C dahil olmak üzere bir dizi iletişim protokolünü destekler. USB arayüzü kolay programlamaya ve güç kaynağına olanak sağlarken, düşük güç tüketimi ve modüllerle çeşitli uyumlulukları, verimliliğin ve genişletilebilirliğin çok önemli olduğu projeler için ideal bir seçim haline getirmektedir.



Şekil 3.1 Arduino Nano ve Pin Dizilimi

3.1.2 Arduino Nano USB Mini Kablo

Arduino Nano USB kablosu, Nano kartı ile bilgisayar veya uyumlu cihazlar arasında veri aktarımını sağlar. USB 2.0 standardıyla uyumlu olan bu kablo, yüksek hızlı veri iletimi sağlayarak güvenilir iletişime olanak tanımaktadır. Bir ucu USB Type-A konnektörü ve diğer ucu Mini-B veya Micro-B konnektörü barındırır. Veri aktarım özelliklerinin yanı sıra bu kablo Arduino Nano ‘nun PC vasıtasıyla 5V ile beslenmesinde de işlem görmektedir. Ayrıca, 480 Mbps ‘ye kadar veri aktarım hızı sağlayabilmektedir.

Şekil 3.2 Arduino Nano USB Mini Kablo

3.1.3 MG90S Servo Motor

MG90S servo motor, robotik, RC (uzaktan kumanda) araçlarında, hassas ve kontrollü hareket gerçekleştirmesi gereken çeşitli elektronik projelerde yaygın olarak kullanılan kompakt ve çok yönlü bir motordur. MG90S 'in orijinal veya pahalı servo modellerindeki dişliler metaldir. Bundan dolayı, boyutu küçük olsa bile sağlam performansıyla anılır. PWM sinyali ile çalışarak 0 – 180 derece arasında hareketini gerçekleştirir. 4.8V – 6V arasında çalışan bu güçlü servolar 4.8V ile beslendiğinde 1.8kg/cm, 6V ile beslendiğinde ise 2.2kg/cm durma torku (stall torque) ile çalışır.

Şekil 3.3 MG90S servo motorun görünümü

3.1.4 Mini DC Motor (3-6V)

Genellikle bir hobi veya oyuncak motor olarak adlandırılan 3-6V DC motor, hobi projelerinde, oyuncaklarda ve küçük ölçekli uygulamalarda yaygın olarak kullanılan bir bileşendir. Bu motor, güç kaynağı seçeneklerinde esneklik sağlayarak 3V – 6V arasındaki güç kaynakları ile beslenebilir. Yaygın olarak 5V güç kaynağı ile çalıştırılır. 6V 'da yüksüz akımı 80mA olarak belirtilir fakat üretilen modele ve markalara göre mini DC motorun özellikleri değişebilmektedir. Genellikle 3 – 6V aralığında beslenen hobi DC motorlar yaklaşık 2000 ile 10.000 RPM aralığında çalışabilmektedir.

Şekil 3.4 Hobi DC motor görünümü

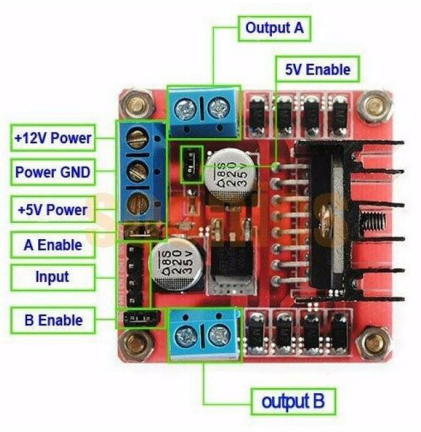
3.1.5 L298N Motor Sürücü Kartı

L298N, endüktif yükleri, DC motorları, step motorları ve diğer yüksek akım veya yüksek voltajlı cihazları sürmek için tasarlanmış ve içerisinde bir çift H-köprüsü barındıran motor sürücü entegre devresidir (IC).

Özellikleri:

- Çift H – Köprü Tasarımı: L298N, Çift H – Köprü tasarımı ile iki DC motoru çift yönlü (saat yönünde ve saat yönü tersinde) olarak döndürebilmektedir.
- Yüksek Akım Kapasitesi: Köprü başına 2A 'e kadar olan tepe (peak) akımlarını (uygun soğutucu olmak kaydıyla) işleyebilir ve daha yüksek akım gerektiren motorlara güç sağlayabilir.
- Geniş Voltaj Aralığı: 4,5V ile 46V aralığında çalışan motorlar için bu sürücü kartı kullanılabilmektedir. Bu da onu çeşitli motor uygulamaları için çok yönlü kılar.

- Dahili Diyotlar: Endüktif yükleri sürmek için içerisinde bulunan geri dönüş diyotları (flyback diodes), arka EMF ‘ye karşı koruma sağlar.
- Lojik Girişler: TTL ve CMOS mantık seviyeleriyle uyumlu olduğundan mikrodenetleyiciler, Arduino kartları ve diğer kontrol sistemleriyle kolayca arayüz oluşturabilirler.



Şekil 3.5 L298N ve pin bilgisi

3.1.6 5V Kablolu Güç Adaptörü

5V kablolu adaptör veya güç kaynağı, duvar prizindeki AC akım voltajını 5V ‘luk sabit DC çıkış voltajına dönüştüren bir cihazdır.

Taret Robot ‘da bulunan 3 adet MG90S servo motoru beslemek için 5V 3A adaptör kullanılmaktadır. Ayrıca, DC motorları ve L298N motor sürücü kartını beslemek için ayrı bir 5V 1A veya 12V 1A kablolu güç adaptörü kullanılmaktadır.

Özellikleri:

- Çıkış Gerilimi: 5V giriş gerektiren çeşitli elektronik cihazlara güç sağlamak için 5V DC çıkış sağlar.
- Akım Değeri: Farklı cihazların güç taleplerini karşılamak için çıkışına farklı amper değerleri sağlar.
- Konnektör Türü: USB Tip-A (Type-A), jack tipi (2.1 mm veya 2.5 mm vb.) veya diğer özel konnektörler ile güç sağlaması gereken cihaza veya gücünü iletecek dişi – erkek kabloya bağlanabilmektedir.
- Güvenlik Özellikleri: Cihazları güç dalgalanmalarından korumak için aşırı akım koruması, kısa devre koruması ve voltaj regülasyonu içerebilir.

Şekil 3.6 Kablolu güç adaptörü

3.1.7 Mini Işıksız ON-OFF Anahtar 3P

3A akıma dayanabilen, 3 kutuplu (3P) konfigürasyona sahiptir. Üzerinde bir adet Açma (ON) / Kapama (OFF) anahtarı bulunmaktadır. 3 kutuplu konfigürasyonu ile aynı anda birden fazla ayrı elektrik devresinden geçebilecek akımı kontrol edebilmektedir. Dayanıklı malzemelerden yapılmış olup, uzun ömürlü olacak ve devreleri açıp kapatırken tekrar tekrar kullanılacak şekilde tasarlanmıştır.

Şekil 3.7 3P mini anahtarın görünümü

3.1.8 Kablolu Dişi Jak

2,1 mm dişi jak kablosu, elektronikte güç bağlantıları için yaygın olarak kullanılan bir konnektör tipidir. Güç adaptörünün 2,1 mm erkek jak çıkış bağlantısıyla uyumlu olacak şekilde birbirine bağlanır. Dişi jak ‘ın diğer tarafında ise PWR ve GND olacak şekilde 2 adet kablosu bulunmaktadır. Bu kabloların kırmızı olanı pozitif hattı, negatif olanı ise toprak hattını temsil etmektedir.

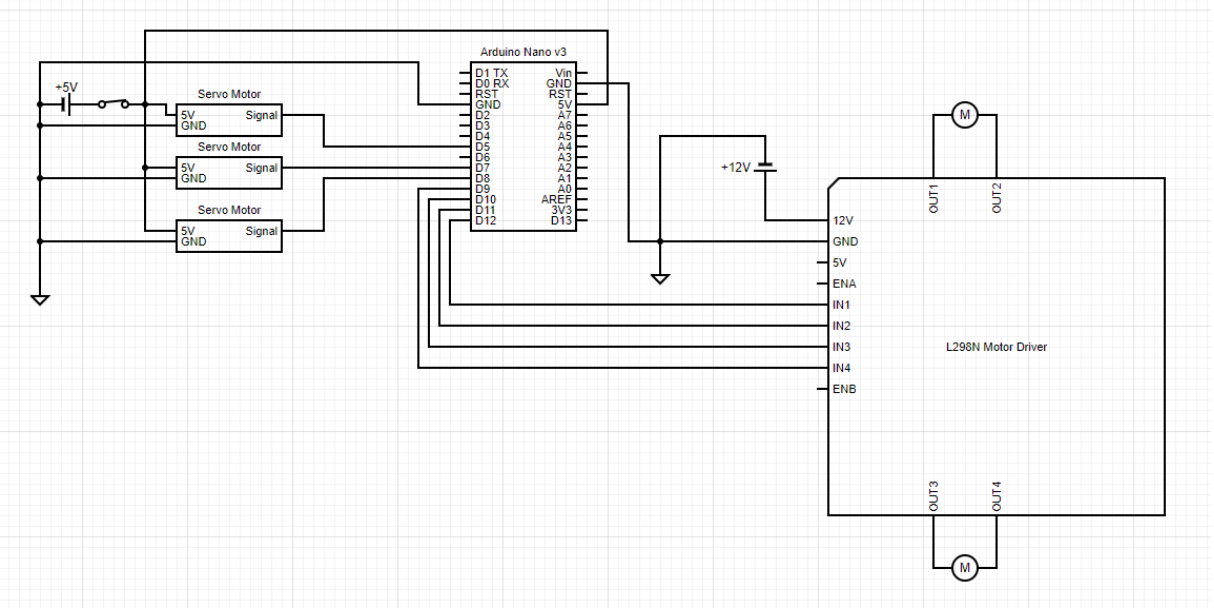
Taret sisteminin elektronik devresindeki 5V adaptör ‘ün erkek jak ucuna bağlıdır. Bu sayede, adaptör ucu kesilmeden kolaylıkla bağlanabilmekte ve çıkarılabilmektedir. Diğer ucu ise elektronik devreyi beslemek amacıyla pertinaks ‘a lehimlenmiştir.

Şekil 3.8 Kablolu dişi jak

3.2 Sistemin Elektronik Donanım Şeması

Sistemin elektronik donanımına bakıldığında, güç kaynağı olarak 5V 3A ve 12V 1A olmak üzere 2 adet adaptör kullanılmaktadır. 3 adet servo motor, 2 adet DC motor, 1 adet L298N motor sürücü kartı, 1 adet Arduino Nano, 2 adet ON / OFF 3P switch ve 1 adet 2.1 mm dişi jak kullanılmaktadır.

- Servo motorların beslemesi 5V 3A adaptör tarafından gerçekleştirilmektedir. Servo motorların sinyal kabloları ise sırasıyla Arduino Nano ‘da bulunan D5, D7 ve D8 dijital pinlerine bağlıdır.
- L298N motor sürücü kartı 12V 1A adaptör tarafından beslenmektedir. 12V 1A adaptörün pozitif (PWR) hattı L298N ‘in 12V pinine bağlıdır, adaptörün negatif (GND) hattı ise L298N ‘in GND pinine bağlıdır. 2 adet DC motor L298N ‘in OUT1, OUT2, OUT3 ve OUT4 pinlerine bağlıdır. DC motorların beslenmesi L298N üzerinden yapılmaktadır. L298N, girişine (12V pinine) uygulanan 12V ‘u içerisinde bulunan 78M05 Voltaj regülatörü vasıtasıyla 5V ‘a düşürerek DC motorlara 5V ‘luk bir çıkış (output) sağlayacaktır. Motorların sürülmesi için ise L298N ‘den Arduino Nano ‘ya bağlı 4 adet hat bulunmaktadır. Bunlar IN1, IN2, IN3 ve IN4 hattıdır. Bu pinler, motorların yönünü ve hızını kontrol etmek için kullanılırlar. Arduino Nano ‘dan L298N ‘in bu pinlerine belirli sinyaller gönderme koşuluyla, L298N içerisindeki H-Köprü devrelerini kullanarak motorun dönüş yönü ve hızı kontrol edilebilmektedir . DC motor 1 ‘in kontrolu için IN1 ve IN2 pinleri kullanılır, DC motor 2 ‘nin kontrolü için IN3 ve IN4 pinleri kullanılır. Ayrıca, L298N ile Arduino Nano ‘nun GND hattı birleştirilmesi gerekir.
- Arduino Nano ‘nun 5V pinine 5V adaptörün pozitif (PWR) hattı, Arduino Nano ‘nun GND pinine ise 5V adaptörün negatif (GND) hattı bağlıdır. Bu sayede, Nano harici bir güç kaynağından beslenir.



Şekil 3.9 Sistemin elektronik donanım şemasının temsili gösterimi

3.3 Sistemde Kullanılan Yazılım Platformları

3.3.1 Spyder IDE

Spyder, Python ‘da bilimsel hesaplama, veri analizi ve sayısal hesaplama görevleri için kullanılan bir IDE ‘dir. Spyder IDE ‘nin genel özellikleri ise sırasıyla:

- Etkileşimli Geliştirme Ortamı: Spyder, IPython konsolu, değişken gezgini (variable explorer) ve entegre hata ayıklayıcı (integrated debugger) gibi özelliklere sahip etkileşimli bir ortam sağlayarak kullanıcıların sorunsuz bir şekilde kod yazmasına, yürütmesine ve hatta ayıklamasına olanak tanır.
- Bilimsel Araç Entegrasyonu: NumPy, SciPy, Matplotlib, Pandas ve daha fazlası gibi veri analizinde yaygın olarak kullanılan çok sayıda bilimsel kütüphane ve araçlar ile önceden yüklenmiş olarak gelir. Bu entegrasyon, kullanıcıların kütüphanelere ve araçlara kolay erişim sağlamasına olanak tanıyarak bilimsel hesaplama görevlerini kolaylaştırmaktadır.
- Zengin Özelliklere Sahip Kod Düzenleyicisi: Sözdizimi (syntax) vurgulama, kod tamamlama, kodun iç gözlemi ve renklendirme gibi özelliklere sahip güçlü bir kod editörü sunar. Bu sayede kodun okunabilirliğini ve verimliliğini arttırır.
- Değişken Gezgini: Kullanıcılar değişken gezginini kullanarak Spyder ‘ın arayüzündeki değişkenleri ve veri yapılarını inceleyebilir, değiştirebilir ve görselleştirebilir. Veri analizi ve manipülasyonunu daha uygun hale getirebilir.
- Grafikler ve Görselleştirme: IDE, veri görselleştirmesi için Matplotlib ‘i entegre ederek, kullanıcıların veri görselleştirmek için doğrudan IDE içinde çeşitli grafik türleri oluşturmasına olanak tanımaktadır.
- Özelleştirme ve Genişletilebilirlik: Spyder kullanıcı ihtiyacına veya zevkine göre özelleştirilebilirdir. Kullanıcılar ortamdaki iş gereksinimlerine göre Spyder ‘ı yapılandırabilmektedir.
- Jupyter Notebook’larla Entegrasyon: Spyder, Jupyter notebook’ları destekler ve kullanıcıların IDE içinde Jupyter notebook’ları oluşturup çalıştırmasını sağlar. Bu sayede okuryazar bir programlama ortamı sağlar.
- Çoklu İşletim Sistemleri Desteği: Farklı platformlarda esneklik ve erişilebilirlik sağlar. Bu özelliği neticesinde, Windows, macOS ve Linux ile uyumlu bir şekilde çalışır.
- Topluluk ve Destek: Spyder, kullanıcıların IDE ile çalışırken kaynakları, öğreticileri (tutorials) ve desteği (support) bulmalarını kolaylaştıran aktif bir topluluğa ve kapsamlı belgelere sahiptir.

Yukarıda belirtilen bu özellikler Spyder IDE ‘yi bilimsel hesaplama, veri analizi, Python programlama görevleri, sayısal analiz, veri görselleştirme ve araştırma gibi üzerinde çalışılması gereken alanlarda uygun bir seçim haline getirmektedir.

3.3.2 Arduino IDE

Arduino IDE, Arduino mikrodenetleyici tabanlı kartların programlanması için özel olarak tasarlanmış bir yazılım platformudur. Arduino kartlarının ihtiyaçlarına göre uyarlanmış bir programlama ortamı olarak hizmet eder ve Arduino mikrodenetleyicilerine kod yazmak, derlemek ve yüklemek için kullanıcı dostu bir arayüz sağlar. Ayrıca, içerisinde bulunan veya internet üzerinde sahip olduğu geniş kitle vasıtasıyla elde edilebilecek temel kütüphaneler ve araçlar ile kullanıcıların yapması gereken projelerinde başarıya ulaşmalarını sağlar.

Sözdizimi vurgulama ve kod tamamlama gibi özelliklere sahip basit bir kod editörü sunarak kod oluşturma ve yazma işlemini basitleştirir. Ek olarak, yazılan kodların derlenmesini ve Arduino kartlarına basit bir arayüz aracılığıyla yüklenmesini kolaylaştırmaktadır.

Arduino IDE ‘nin genel özellikleri:

- Basit ve Kullanıcı Dostu Arayüz: Arduino IDE, basit ve kullanımı kolay bir arayüz sunarak programlama ve elektroniğe yeni başlayanlar için gerekli kolaylıkları sağlar.
- Platformlar Arası Uyumluluk: Windows, macOS ve Linux işletim sistemleriyle uyumludur ve çeşitli platformlarda çalışabilmektedir.
- Arduino Kartları için Destek: IDE, çok çeşitli Arduino kartlarını destekleyerek, Arduino Uno gibi giriş seviyesi kartlardan Arduino Mega veya Arduino Nano gibi daha gelişmiş olan birçok Arduino kartlarına destek sağlar.
- Entegre Kod Düzenleyici: IDE, söz dizimi vurgulama, otomatik girdi ve kod tamamlama gibi özelliklere sahip bir kod düzenleyici sağlayarak Arduino programlarının yazılmasını ve düzenlenmesini basitleştirir.
- Kütüphane Yöneticisi: Arduino IDE, kullanıcıların kütüphaneleri kolay bir şekilde kurmasına, yönetmesine ve güncellemesine olanak tanıyan, ek kod ve işlevleri entegre ederek Arduino kartlarının işlevlerini ve yeteneklerini genişleten bir kütüphane yöneticisi içerir.
- Yerleşik Örnekler: Arduino IDE, içerisinde barındırdığı yerleşik örnekler ile çeşitli sensörler, aktüatörler, iletişim protokolleri ve diğer bileşenler için şablon görevi görür. Bu sayede, kullanıcıların Arduino kartlarının farklı işlevlerini anlamalarına ve kullanmalarına yardımcı olur.
- Seri Monitör: IDE, Arduino kartı ile bilgisayar arasında hata ayıklama ve iletişim için bir seri monitör içerir. Bu monitör, kullanıcıların veri göndermesine / almasına ve programlarındaki hatayı ayıklamalarına olanak tanır.
- Yükleme ve Derleme: Kullanıcılar programlarını derleyebilir ve bunları USB kablo vasıtasıyla Arduino kartlarına yükleyebilir. Bu durum, donanım üzerinde kod programlama ve test etme süreçlerini kolaylaştırır.

3.3.3 EpocCam Uygulaması

EpocCam, kesintisiz bir video akışı sunmak için telefon cihazlarıyla entegre olan çok yönlü ve yenilikçi bir uygulamadır. Kinoni tarafından geliştirilen EpocCam, akıllı telefonları yüksek çözünürlüklü web kameralarına dönüştürerek, çeşitli uygulamaların yapılmasında bir dizi işlevsellikler sunmaktadır. EpocCam, kablosuz (WiFi) bağlantıdan yararlanarak, kullanıcının telefon kamerası üzerinden sağlanan video akışını bilgisayar ortamındaki uygulamasına yansıtır. EpocCam 'in kullanıcı dostu arayüzü ve birden fazla platform ile uyumluluğu bulunmaktadır. Bilgisayar ve telefon arasında sağladığı entegrasyon ile görüntülü aramalar, video akışı sağlama, kişisel ve profesyonel kullanım gibi birçok ihtiyaçlara hizmet edebilme yeteneğini bünyesinde bulundurmaktadır. Ayrıca, kullanıcıların yüksek kalitede video akışı almalarını sağlamaktadır.

EpocCam, bilgisayarla görme uygulamaları ve gerçek zamanlı izlemeler için kesintisiz video akışının çok önemli olduğu Nerf Taret gibi projelerde önemli bir bileşen haline gelmektedir. Bu çok yönlü uygulama, kullanım kolaylığı ve uyarlanabilirliği ile birleştiğinde, çeşitli alanları önemli ölçüde etkileyerek günlük ihtiyaçlarımız ve yenilikçi projelerimiz için teknolojiden yararlanma şeklimizi farklı boyutlara şekillendirmektedir.

3.3.4 Ultralytics YOLOv8

YOLOv8 veya You Only Look Once version 8, Görüntü İşleme uygulamalarında gerçek zamanlı nesne algılama teknolojisidir. YOLOv8, önceki sürümlerine göre doğruluk ve hız açısından iyileştirmeler sunarak çeşitli projeler için çok yönlü kullanılabilme imkanı sağlamaktadır. YOLOv8 kullanılan sistemler, nesneleri gerçek zamanlı olarak algılama ve tanıma yeteneğini kazanmaktadırlar.

Tek bir sinir ağından (a single neural network) yararlanan YOLOv8, bir görüntüdeki veya video akışı içerisindeki birden fazla hedefi tanımak için gerekli sınırlayıcı kutuları ve sınıf olasılıklarını aynı anda tahmin etmek üzere tasarlanmıştır. Ayrıca, model bilgisini hızlı bir şekilde işleme yeteneği ve dinamik ortamlarda hızlı karar verme talebiyle uyumlu olarak çalışmaktadır. Bu verimlilik, uğraşılan Taret Robot projesindeki EpocCam ve OBS Studio aracılığıyla elde edilen kamera akışında yeşil elma (veya sarımsı yeşil elma) modeli gibi hedeflerin hızlı ve doğru bir şekilde tespit edilebilmesini sağlamasından dolayı genel proje kapsamına bakıldığında sistem için fazlasıyla önem arz etmektedir.

YOLOv8 'in Python ile uyumluluğu, genel sistem mimarisine sorunsuz olarak entegre edilebilmesini kolaylaştırır. Bu durum, Görüntü İşleme, Arduino ve projede kullanılan diğer bileşenlerin birbirleriyle uyumlu bir şekilde çalışmasına olanak tanımaktadır.

3.3.5 YOLOv8 ile Eğitim Yapılması

YOLOv8 ile eğitim aşaması aşağıdaki işlemler yapılarak başlatılır:

1. Yeşil elma modelinin veya tespit edilmesi istenen nesnenin resimlerini içeren özel bir veri seti oluşturulur. Veri setini oluşturmak için "Open Images Dataset V7" sitesinden ilgili veriler toplanabilir veya kullanıcı kendi kamerası ile çektiği görüntüleri veri setine ekleyebilir.
   https://storage.googleapis.com/openimages/web/index.html

2. Toplanan resimler "Open Data Annotation Platform (CVAT)" gibi siteler üzerinden gerekli etiketleme işlemlerine tabi tutulur.
   https://www.cvat.ai/

3. Elde edilen veriler Şekil 3.10 'da gösterildiği gibi gerekli parametrelere atanır.

```yaml
path: C:\Users\y4sef\OneDrive\Masaüstü\Taret_Code\Dataset
train: images/images1 # train 'lenecek dosyalarin klasor lokasyonu
val: images/images1 # val: images eklememizin olayi ise train yapildiktan sonra
# dipnot (cerceveye) aldigimiz dosyalar ile karsilastirip dogruluk hassasiyetini olcmek icindir.

#Class tanimlamak gerekiyor
# 0 --> birinci index, apple = isim
names:
  0: apple
```

Şekil 3.10 YOLOv8 için yaml dosyasının oluşturulması

path = Bütün dosyaların lokasyonunu ifade eder.
train = Eğitilecek dosyaların lokasyonunu belirtir.
val = Mevcut veri seti ile eğitim tamamlandıktan sonra, veri setinin doğruluk hassasiyetini karşılaştırmak amacıyla başka bir veri seti ile kıyaslama yapılır.

4. Gerekli işlemler tamamlandıktan sonra Şekil 3.11 'de gösterildiği gibi Python içerisinde eğitim işlemine başlanır.

```python
from ultralytics import YOLO

# Load a model
model = YOLO("yolov8n.yaml")  # build a new model from scratch
model = YOLO("yolov8n.pt")  # load a pretrained model (recommended for training)

# Use the model
model.train(data="apple.yaml", epochs=3)  # train the model
```

Şekil 3.11 YOLOv8 ile eğitim yapılması

from ultralytics import YOLO = Ultralytics 'ten YOLO nesne algılama framework 'ünü dahil eder.
model = YOLO("apple.yaml") = Oluşturulan model yüklenir.
model = YOLO("yolov8n.pt") = Önceden eğitilmiş bir model yüklenir (Eğitim için gerekli değildir).
model.train(data="apple.yaml", epochs=3) = Veri seti ve kaç tur eğitim yapılacağı belirtilir.

3.4 Sistemdeki Python Yazılımı ve Kod Dizinleri

Ekler bölümünde Python kodunun tam hali bulunmaktadır.

3.4.1 Gerekli Kütüphanelerin Eklenmesi

Spyder IDE ‘de yazılan Python koduna sırasıyla cv2, pygetwindow as gw, pyautogui as pag, numpy as np, serial, time ve ultralytics ‘deki YOLO kütüphaneleri dahil edilmiştir. Bu kütüphanelerin işlevlerine bakıldığında:

- cv2 (OpenCV): OpenCV, görüntü ve video işleme için kullanılan güçlü bilgisayarlı görme (veya görüntü işleme) kütüphanesidir. Üzerinde çalışılan projede, EpocCam aracılığıyla iPhone 11 ‘den video akışını yakalamaya ve işlemeye olanak sağlar. OpenCv, elde edilen video akışı ile YOLOv8 kullanılarak nesne algılama ve izleme gibi çeşitli görüntü işleme görevlerini gerçekleştirmeye olanak tanımaktadır.
- pygetwindow as gw: Bu kütüphane, ekrandaki pencereleri tanımlamaya ve bunlarla etkileşime girmeye yardımcı olur. Projedeki EpocCam ‘in video akışını görünteleyen pencereyi tanımlamak ve yönetmek için kullanılır.
- pyautogui as pag: PyAutoGUI kütüphanesi, bilgisayar ekranındaki belirli bölgelerin ekran görüntülerini yakalamada çok önemli bir rol oynamaktadır. Örneğin “screenshot = pag.screenshot(region=(left, top, width, height))” kodu, ekranın sol, üst, genişlik ve yükseklik parametrelerine bakarak, ekranın belirlenmiş bölgelerindeki görüntüleri almak için kullanılır. Nerf Taret projesindeki EpocCam ve OBS Studio aracılığıyla alınan video akışı, bilgisayar ekranında görüntülenir ve bu ekran görünümünü yakalamak için PyAutoGUI kütüphanesi kullanılır.
- numpy as np: NumPy, Python ‘da sayısal hesaplamalar için kullanılan temel bir kütüphanedir. İçerisinde güçlü dizi işlemleri ve matematiksel fonksiyonlar barındırır. Projede, görüntü işleme veya nesne algılama ile ilgili verilerin verimli bir şekilde hesaplanmasında kullanılır.
- serial: Serial kütüphanesi, Spyder IDE ve Arduino kartı arasındaki seri iletişimi kolaylaştıran “ser_arduino = serial.Serial('COM3', 115200, timeout=1)” kodu yazmamızı sağlar. Bu Projede, Serial kütüphanesi kullanılarak “COM3” portu vasıtasıyla 115200 saniye başına veri hızında (baud rate) veri iletimi sağlanır.
- time: Time (zaman) kütüphanesi, zamanla ilgili görevler için işlevler ve gerekli metotların kullanılmasını sağlar. Kodun farklı bölümleri arasında zamanlama, gecikmeler (delay) veya senkronizasyon için kullanılmaktadır.
- from ultralytics import YOLO: Bu kütüphane, Ultralytics ‘ten YOLO nesne algılama framework ‘ünü ifade eder. YOLOv8, derin öğrenme (deep learning) tabanlı gerçek zamanlı nesne algılama modelidir. Nesne algılama görevlerini gerçekleştirmek, OpenCV tarafından yakalanan video akışı içindeki hedefleri (bu projede yeşil elma) tespit etmek ve tanımlamak için kullanılır.

Kısaca özetlemek gerekirse, bu kütüphaneler toplu olarak iPhone cihazından gelen video akışını işlemeye, OBS Studio aracılığıyla video akışının Python ‘a aktarılmasını ve YOLOv8 kullanılarak eğitilen “Yeşil Elma” modelinin kullanılması ile hedefin tespiti sağlanır. Tespit edilen hedefin X ve Y koordinat bilgileri gerekli kütüphane ve metotlar ile elde edildikten sonra “COM3” portu üzerinden Arduino ‘ya aktarılır. X ve Y koordinatlarını alan Arduino

mikrodenetleyicisi, bu parametre değerlerini kullanarak Taret Robot ‘un servo motorlarını gerekli açı değerlerine döndürecek ve Robot ‘un hareketini sağlayacaktır.

```
import cv2
import pygetwindow as gw
import pyautogui as pag
import numpy as np
import serial
import time
from ultralytics import YOLO
```

Şekil 3.12 Gerekli Kütüphanelerin Python koduna eklenmesi

3.4.2 Gerekli Parametrelerin Oluşturulması

1. Arduino ‘ya Seri Bağlantı Yapılması

Arduino kartı ile Python arasındaki seri iletişimi sağlamak için “ser_arduino = serial.Serial('COM3', 115200, timeout=1)” kodu kullanılır.

- COM3, Arduino ‘nun bağlı olduğu portu belirtir.
- 115200, iletişim hızını belirten veri iletim (baud rate) hızıdır.
- timeout = 1, okuma işlemleri için saniye cinsinden bir zaman aşımı değeri belirler ve komut dosyasının zaman aşımına uğramadan önce gelen verileri 1 saniyeye kadar bekletmesine olanak tanır.

```
ser_arduino = serial.Serial('COM3', 115200, timeout=1)
```

Şekil 3.13 Arduino seri bağlantı parametreleri

2. OBS Pencere Bilgisinin Alınması

OBS pencere bilgisinin alınması için “obs_window = gw.getWindowsWithTitle('Camera Hub')[0]” ve “left, top, width, height = obs_window.left, obs_window.top, obs_window.width, obs_window.height” satırları kullanılmaktadır.

Burada, “Camera Hub” başlıklı OBS Studio penceresi hakkında bilgi almak için pygetwindow ‘u (gw olarak içe aktarılır) kullanır. getWindowsWithTitle parametresi, belirtilen başlığa sahip pencereleri bulur. “[0]” ifadesi, bulunan ilk pencereye eriş anlamına gelmektedir.

“left, top, width, height” değişkenleri, tanımlanan pencerenin konumunu ve boyutlarını öğrenerek ekrandaki video akışının konumunu ve boyutunu yakalar (capturing).

```
obs_window = gw.getWindowsWithTitle('Camera Hub')[0]

left, top, width, height = obs_window.left, obs_window.top, obs_window.width, obs_window.height
```

Şekil 3.14 OBS pencere bilgisinin alınması

3. YOLO Modelini Başlatmak

model_path = "C:\\Users\\y4sef\\Downloads\\kod_ve_ptle\\train29\\weights\\best.pt" parametresi, eğitilen modelin bulunduğu dizini ifade eder.

“model = YOLO(model_path)” parametresi ise, belirtilen bu yolu kullanarak Ultralytics ‘ten YOLOv8 nesne algılama modelini başlatır.

Ultralytics kütüphanesinden YOLO sınıfı (class) başlatılır ve önceden eğitilmiş modeli kullanarak nesne algılama görevlerini gerçekleştirmeye olanak tanır.

```
model_path = "C:\\Users\\y4sef\\Downloads\\kod_ve_ptle\\train29\\weights\\best.pt"

model = YOLO(model_path)
```

Şekil 3.15 YOLOv8 modelinin başlatılması

4. Nesnenin Algılanması için Renk Eşikleri

Python kodunda bulunan “hue_lower” ve “hue_upper” parametreleri, HSV renk uzayındaki farklı renkler için alt ve üst eşikleri belirler. Örneğin “hue_lower_apple” ve “hue_upper_apple” parametreleri, elmaya benzer renkleri algılamak için ton aralığını tanımlar. Benzer değişkenler, yeşil nesneler ve sarı nesneler gibi diğer renler için ton aralıkları bu parametreler vasıtasıyla belirtilir.

```
hue_lower_apple = 10
hue_upper_apple = 40

hue_lower_shirt = 100
hue_upper_shirt = 130

hue_lower_green = 40
hue_upper_green = 70

hue_lower_yellow = 20
hue_upper_yellow = 30
```

Şekil 3.16 Renk parametrelerinin ayarlanması

5. Eşik ve Boyut Parametreleri

Python kodunda bulunan “threshold” parametresi, bazı tespit veya karşılaştırma işlemlerinde kullanılan hassasiyet eşiğini temsil etmektedir.

“min_size_threshold” ve “max_size_shreshold” parametresi ise algılanan nesneler için minimum ve maksimum eşikleri belirtir. Bu eşik değerleri, nesneleri boyutlarına göre filtrelemek ve çok küçük veya büyük algılamaları ortadan kaldırmak için kullanılır.

```
threshold = 0.01

min_size_threshold = 5
max_size_threshold = 18000
```

Şekil 3.17 Eşik ve boyut parametrelerinin ayarlanması

6. Maksimum Güven Düzeyi ve Koordinatlar

Python kodunda bulunan "max_confidence" parametresi, algılamalar için maksimum güven düzeyini ayarlamaktadır. Nesne algılama sürecindeki güven eşikleri için referans olarak kullanılır.

"coordinates_to_send" parametresi, komut dosyasının başka bir bölümünde gönderilecek veya kullanılacak nesnelerin koordinat bilgilerini depolar. Başlangıç değeri "None" olarak ayarlanmıştır.

```
max_confidence = 0

coordinates_to_send = None
```

Şekil 3.18 Güven düzeyi ayarlanması ve koordinat bilgilerinin depolanması

7. Ekran Görüntüsü Yakalama ve Renk Filtreleme

Python kodunda bulunan "screenshot = pag.screenshot(region=(left, top, width, height))" satır ile sol, üst, genişlik ve yükseklik olarak tanımlanan parametreleri kullanarak ekranda tanımlı bir bölgenin sürekli olarak ekran görüntülerini yakalar.

Yakalanan ekran görüntüleri "cv2.cvtColor" metotuyla OpenCV formatına dönüştürülür ve "hsv_frame = cv2.cvtColor(obs_capture, cv2.COLOR_BGR2HSV)" satırıyla HSV renk uzayına dönüştürülür.

"cv2.inRange" metotu, yakalanan HSV çerçevelerini kullanır ve bu çerçeveleri tanımlanmış renk aralıklarına (hue_lower and hue_upper) göre belirli renkleri (elma rengi, yeşil renk ve sarı gibi) izole etmek için maskeler oluşturur.

```
while True:
    screenshot = pag.screenshot(region=(left, top, width, height))
    obs_capture = cv2.cvtColor(np.array(screenshot), cv2.COLOR_RGB2BGR)

    hsv_frame = cv2.cvtColor(obs_capture, cv2.COLOR_BGR2HSV)

    mask_apple = cv2.inRange(hsv_frame, (hue_lower_apple, 50, 50), (hue_upper_apple, 255, 255))
    mask_shirt = cv2.inRange(hsv_frame, (hue_lower_shirt, 50, 50), (hue_upper_shirt, 255, 255))
    mask_green = cv2.inRange(hsv_frame, (hue_lower_green, 50, 50), (hue_upper_green, 255, 255))
    mask_yellow = cv2.inRange(hsv_frame, (hue_lower_yellow, 50, 50), (hue_upper_yellow, 255, 255))
```

Şekil 3.19 Ekran görüntüsünün yakalanması ve renk filtrelemeleri

8. YOLOv8 ile Nesne Tespiti

"combined_mask" gibi oluşturulan maskeler, aynı anda birden fazla rengin algılanması için "cv2.bitwise_or" metodunun kullanımı ile birleştirilir.

Ortaya çıkan "combined_mask" değeri, istenmeyen renkleri filtrelemek için orijinal yakalanan kareye (video akışı) uygulanır ve "filtered_frame" içerisinde yalnızca algılanan nesneler kalır.

Daha sonrasında, nesne algılamayı gerçekleştirmek için filtrelenmiş çerçeveye (filtered_frame) YOLOv8 teknikleri uygulanır ve "for result in results.boxes.data.tolist()" komutu ile tespit edilen nesnelerin koordinatları ve güven puanları bir döngüde işlenir. Çıkan sonuçlar "x1, y1, x2, y2, score, class_id" gibi değişkenler içerisinde saklanır.

```
combined_mask = cv2.bitwise_or(cv2.bitwise_or(cv2.bitwise_or(mask_apple, mask_shirt), mask_green), mask_yellow)

filtered_frame = cv2.bitwise_and(obs_capture, obs_capture, mask=combined_mask)
```

Şekil 3.20 Renklerin algılanması ve filtrelenmesi

9. Arduino İletişimi için Yönlendirme Koordinatları

Bir nesne belirlenen eşikleri aşarsa "if score > threshold and min_size_threshold < box_size < max_size_threshold" komutu ile nesnenin X ve Y koordinat değerleri hesaplanır ve önceden tanımlanmış renk aralıklarına göre kontrol edilir.

En yüksek puana (score) sahip nesnenin koordinatları "coordinates_to_send" parametresinde saklanır.

```
for result in results.boxes.data.tolist():
    x1, y1, x2, y2, score, class_id = result
    box_size = (x2 - x1) * (y2 - y1)

    if score > threshold and min_size_threshold < box_size < max_size_threshold:
        center_x = (x1 + x2) / 2
        center_y = (y1 + y2) / 2

        if hue_lower_green <= (hue_lower_apple + hue_upper_apple) / 2 <= hue_upper_green:
            score += 0.5

        if hue_lower_yellow <= (hue_lower_apple + hue_upper_apple) / 2 <= hue_lower_yellow:
            score += 0.2

        if score > max_confidence:
            max_confidence = score
            coordinates_to_send = (center_x, center_y)# X ve Y koordinat degerleri "coordinat
```

Şekil 3.21 Yönlendirme kurallarının belirlenmesi

10. Koordinatların Haritalanması ve Arduino 'ya Gönderilmesi

Eğer "coordinates_to_send" verisi mevcutsa, içerisinde barındırdığı değerler formatlanır ve seri bağlantı yoluyla "ser_arduino.write(data_to_send.encode())" komutu kullanılarak Arduino 'ya gönderilir.

"map_value" metodu ise haritalama işlevi görür. Video akışı 1920 x 1080p formatında olduğundan dolayı, Arduino 'da bulunan servo motorların anlayabileceği aralığa (0 – 180 dereceye) dönüştürmek için kullanılır.

```
# Algılanan nesnenin koordinatları arduinoya yollanır.
if coordinates_to_send is not None:
    data_to_send = f"{int(coordinates_to_send[0])},{int(coordinates_to_send[1])}\n"
    #Arduino 'ya gidecek koordinatların terminalden okunmasi
    # X 1920 'den --> 0 - 180 araligina "mapped_center_x"
    # Y 1080 'den --> 0 - 180 araligina "mapped_center_y"
    mapped_center_x = map_value(center_x, 0, 1920, 0, 180)
    mapped_center_y = map_value(center_y, 0, 1080, 0, 180)
    #print("X= ", mapped_center_x, " angle", " Y= ", mapped_center_y, " angle") #Ori
    print(f"X= {mapped_center_x:.2f} angle, Y= {mapped_center_y:.2f} angle") # X ve
    #print(f"X= {int(mapped_center_x)} angle, Y= {int(mapped_center_y)} angle") # X
    #Terminalden X ve Y degerlerini 0.2 saniye araliklar ile gormek icin time.sleep(
    time.sleep(0.2)
    #Arduino 'ya koordinatların gonderilmesi
    ser_arduino.write(data_to_send.encode())
```

Şekil 3.22 Koordinat bilgilerinin Arduino tarafına gönderilmesi

11. Nesne Bilgilerini Görüntüleme

“cv2.rectangle” metodu, yakalanan çerçeve üzerinde algılanan nesnelerin üzerine sınırlayıcı kutular çizer.

“cv2.putText” metodu, yakalanan çerçeve üzerine nesnenin koordinatlarını ve güven puanlarını içeren metin açıklamaları ekler.

```
# Algılanan nesnenin cercevede konfigurasyonu
if coordinates_to_send is not None:
    x, y = coordinates_to_send
    cv2.rectangle(obs_capture, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 4)
    label = f"Coordinates: ({int(x)}, {int(y)}) | Confidence: {max_confidence:.2f}"
    #X ve Y piksel degerlerinin yazdirilmasi
    print(f"X coordinate: {int(x)}, Y coordinate: {int(y)}")
    cv2.putText(obs_capture, label, (int(x1), int(y1) - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 3, cv2.LINE_AA)
```

Şekil 3.23 Algılanan hedefin bilgilerini görüntülemek

12. Sonuçların Görüntülenmesi ve Çıkış İşlemi

Python kodunda bulunan “cv2.imshow('Frame', obs_capture)” satırı, tespit edilen nesnelerin “cv2.rectangle” ve “cv2.putText” metotları ile gerekli bilgileri eklenmiş haliyle ekranda görüntülenmesini sağlar.

"cv2.waitKey(1) & 0xFF == ord('q')” kod bölümü ise, kullanıcı girişine (input) dayalı olarak sonsuz döngüyü (while True:) kırmanın bir koşulu olarak yazılmıştır. Kod satırında görüleceği üzere “if” koşulu ile başlayan kod satırına “q” parametresi girdi olarak verildiği koşulda “while” döngüsü sonlanacaktır.

```
# Algılanan nesnenin cercevede (Frame) gosterimi
cv2.imshow('Frame', obs_capture)

if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

Şekil 3.24 Sonuçların görüntülenmesi

3.5 Sistemdeki Arduino Yazılımı ve Kod Dizinleri

Ekler bölümünde Arduino kodunun tam hali bulunmaktadır.

1. Gerekli Kütüphanelerin Eklenmesi

Arduino IDE içerisinde servo motorların tanımlamalarının yapılması için öncelikle "<Servo.h>" kütüphanesi eklenir.

```
#include <Servo.h>
```

Şekil 3.25 Gerekli kütüphanelerin Arduino IDE tarafına dahil edilmesi

2. Gerekli Parametrelerin Belirlenmesi

Taret Robot 'un başlangıç pozisyon bilgilerini, hedef X ve Y koordinatlarını, Taret 'in mekanik kısıtlamalardan dolayı dönebileceği maksimum ve minimum konum değerlerini, mevcut pozisyon ve yeni pozisyon gibi bilgileri tutmak amacıyla gerekli parametreler oluşturulur.

```
int baslangic_taret_positionY = 85;
int baslangic_taret_positionZ = 100;
int dart_push_position = 105;

int hedefin_x_kordinat_acisi = 0;
int hedefin_y_kordinat_acisi = 0;
int hedefin_ters_X_koordinat_acisi = 0;
int hedefin_ters_Y_koordinat_acisi = 0;

int taret_Z_ekseni_ust_sinir_degeri = 180; //Z ekseninde ust sınır 180 derecedir
int taret_Y_ekseni_ust_sinir_degeri = 125; //Y ekseninde ust sınır 40 derecedir, 85 (default) + 40 = 125
int taret_Z_ekseni_alt_sinir_degeri = 0; //Z ekseninde alt sınır 0 derecedir
int taret_Y_ekseni_alt_sinir_degeri = 70; //Y ekseninde alt sınır 15 derecedir, 85 (default) - 15 = 70

int eski_Z_pozisyonu = 0;
int eski_Y_pozisyonu = 0;
int current_Z_position = 0;
int current_Y_position = 0;
int new_Z_position = 0;
int new_Y_position = 0;

int Z_ekseni_pozisyon_farki = 0;
int Y_ekseni_pozisyon_farki = 0;
```

Şekil 3.26 Taret için gerekli parametrelerin tanımlanması

3. Servo Motorlara Gerekli Atamaların Yapılması ve Seri İletişimin Başlatılması

"void setup()" fonksiyonu içerisinde servo motorların bağlı olduğu pinler tanıtılır ve Taret 'in varsayılan pozisyona gelebilmesi amacıyla "servo.write()" fonksiyonu kullanılarak servolar döndürülür.

Burada "write()" önündeki "servo" ifadesi "<Servo.h>" kütüphanesine ait bir nesnedir ve kullanıcının oluşturduğu isimlere göre bu ifade değişebilmektedir. Bu projede, servoZ, servoY ve servoD şeklinde oluşturulmuştur.

"Serial.begin(115200)" ifadesindeki Serial, Arduino kartındaki seri iletişim arayüzünü temsil eden bir nesnedir ve bu kod satırı ile Arduino 'nun USB portu üzerinden diğer bağlandığı cihazlar ile veri alıp gönderebilmesini sağlar (seri iletişimi başlatır), 115200 ifadesi ise iletişim hızını belirten veri iletim (baud rate) hızıdır.

pinMode() fonksiyonu, 2 adet motorun L298N sürücü kartına bağlı pinlerini tanımlamak için kullanılır.

```
void setup() {
  delay(2000); //Arduino 'nun kararli hale gecmesi icin 2 saniye sistemi bekletiyoruz.
  servoZ.attach(5);
  servoY.attach(7);
  servoD.attach(8);
  servoD.write(dart_back_position);
  delay(1000);
  servoZ.write(baslangic_taret_positionZ);
  delay(1000);
  servoY.write(baslangic_taret_positionY);
  delay(1000);
  Serial.begin(115200); // Seri iletisimin baslatilması
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

}
```

Şekil 3.27 Servo motor atamalarının yapılması ve seri iletişimin başlatılması

4. Seri İletişimin Kontrol Edilmesi

Arduino IDE 'de bulunan "Serial.available() > 0" ifadesi, seri bağlantı noktasından okunabilecek herhangi bir veri olup olmadığını kontrol etmek için kullanılır.

```
if (Serial.available() > 0) {
  String data = Serial.readStringUntil('\n');
  int delimiter = data.indexOf(",");
```

Şekil 3.28 Seri iletişimin kontrolü

5. Gerekli Koordinat Bilgilerinin Alınması ve Düzenlenmesi

Arduino kodunda bulunan "int centerX = data.substring(0, delimiter).toInt();" ifadesi, Python 'dan gelen X koordinat bilgisini alır ve aldığı değeri tamsayıya dönüştürür.

"int centerY = data.substring(delimiter + 1, data.lastIndexOf(',')).toInt();" ifadesi ise (X,Y) olarak gelen koordinat bilgisinin Y eksenindeki değerini alır ve aldığı değeri tamsayıya dönüştürür.

```
//Gelen koordinat degerlerinin ayristirilmasi
int centerX = data.substring(0, delimiter).toInt();
int centerY = data.substring(delimiter + 1, data.lastIndexOf(',')).toInt();

//Goruntu Python 'dan ters gelmektedir.
hedefin_ters_X_koordinat_acisi = map(centerX, 0, 1920, 0, 180);
hedefin_ters_Y_koordinat_acisi = map(centerY, 0, 1080, 0, 180);

//Goruntuyu duzeltmek icin "180 - Koordinat degeri" formulu uygulanir.
hedefin_x_kordinat_acisi = 180 - hedefin_ters_X_koordinat_acisi;
hedefin_y_kordinat_acisi = 180 - hedefin_ters_Y_koordinat_acisi;
```

Şekil 3.29 Koordinat bilgilerinin alınması ve düzenlenmesi

Gelen X ve Y koordinat bilgileri ilk olarak servo motorun dönüş açılarına uyarlanacak şekilde "map()" fonksiyonuyla 0 – 180 derece aralığına ayarlanır ve Python 'dan gelen ters görüntü "180 – koordinat açısı" olacak şekilde düzeltilir.

6. Servo Motorların Mevcut Pozisyonlarının Okunması

Taret üzerinde bulunan servolar, tasarımdan kaynaklı kısıtlamalar doğrultusunda hareket edebilmesi ve gerekli matematiksel operatörler üzerinden işlem görebilmesi amacıyla servonun konum bilgileri gerekli parametreler içerisinde depolanır.

```
current_Z_position = servoZ.read();
current_Y_position = servoY.read();
```

Şekil 3.30 Servo motorların mevcut pozisyonlarının okunması

7. Servo Motorların Açısal Kısıtlarının Belirlenmesi

Taret 'in tasarımından kaynaklı kısıtları Z ekseni için bulunmamaktadır fakat Y ekseni için orijin noktası 0 kabul edilirse;

Pozitif yönde 40 – 45 derece

Negatif yönde ise 15 – 20 derece hareket edebilmektedir.

Bu veriler doğrultusunda gerekli " if " (koşul) ifadeleri belirlenerek servoların dönüşleri belirli sınırları aşmamak kaydıyla kontrol edilir.

```
if((new_Z_position >= 0 && new_Z_position <= 180) && (new_Y_position >= 70 && new_Y_position <= 125)){
  //Orn: hedef Z = 65, Y = 70 olsun, default Z = 100, Y = 85
  Z_ekseni_pozisyon_farki = new_Z_position - current_Z_position;
  Y_ekseni_pozisyon_farki = new_Y_position - current_Y_position;
  //Y Servosunu dondurmek
  if(Y_ekseni_pozisyon_farki > 0){
    for(int i = 0; i <= Y_ekseni_pozisyon_farki; i++){
      servoY.write(current_Y_position + i);
      delay(15);
    }
```

Şekil 3.31 Servo motorların açısal kısıtlarının belirlenmesi

8. Servoların Konum Kontrolünün Sağlanması

Bu bölümde, servolara PID kontrolcü tasarlamaktan ziyade "konumsal hata" hesabı üzerinden bir düzeltme uygulanması uygun görülmüştür. Servo yeni koordinat bilgisini aldığında o konuma gidecektir. Eğer ki gitmesi gereken konuma erişemezse veya aşarsa, mevcut konum bilgisi okunacaktır ve gitmesi gereken konum ile arasındaki fark hesaplanarak şu anki konum değerine bu fark eklenecektir. Bu sayede, servo motor konumsal olarak hata yapsa dahi "hata" (error) değerine göre tekrar işlem görecektir.

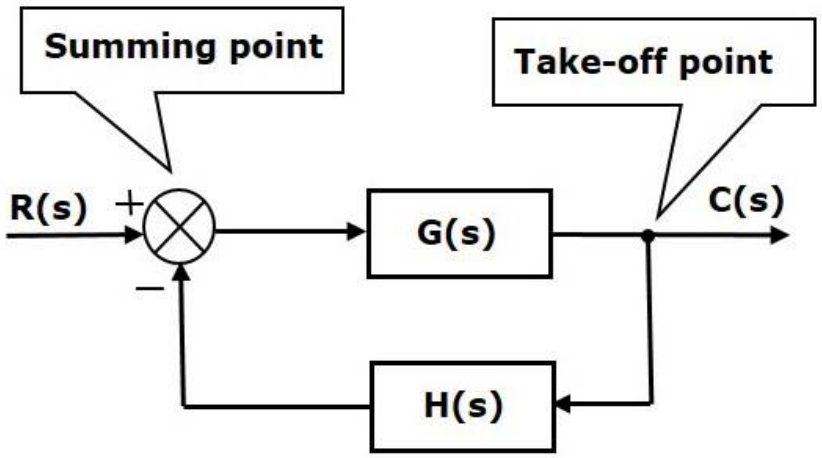
```
current_Y_position = servoY.read();
//Y servosu icin konumsal hata
Y_konumsal_hata = new_Y_position - current_Y_position;
servoY.write(current_Y_position + Y_konumsal_hata);
```

Şekil 3.32 Konumsal hataya göre konum kontrolünün sağlanması

3.6 Taret Kontrolü için Transfer Fonksiyonu Modellemesi

3.6.1 Kontrol Sistemlerinde Transfer Fonksiyonlarına Giriş

Kontrol Sistemleri, dinamik sistemlerin davranışlarını yönetmenin temelini oluşturur. Sistemin çıkış yanıtlarını (system output) analiz etmeyi ve düzenlemeyi sağlamaktadır. Bu sistemlerin davranışlarını analiz etme ve anlamada en güçlü araç transfer fonksiyonları kavramıdır. Transfer fonksiyonları, bir sistemin giriş – çıkış ilişkisinin frekans uzayında/alanında (domain) matematiksel bir gösterimini sağlar ve sistemin dinamiklerini tahmin edebilmemize olanak tanır.



Şekil 3.33 Sistemin Blok Diyagram gösterimi

3.6.2 Transfer Fonksiyonlarını Anlamak

Transfer fonksiyonları "F(s)" şeklinde gösterilir. Bu fonksiyonlar, sistem analizinde kullanılan diferansiyel denklemleri "Laplace" etki alanına çeviren önemli unsurlardır. Bir sistemin dinamiklerini kapsülleyerek (encapsulate), sistemin çeşitli girdilere ve bozukluklara karşı nasıl tepki vereceğine dair fikirler verir.

Bu fonksiyonlar Laplace değişkeni "s" formatında belirtilir. "s" parametresi, doğrusal zamanla değişmeyen (LTI) bir sistemi temsil eder.

3.6.3 Sistem Dinamiklerinin Temsil Edilişi

Genellikle "G(s)" olarak gösterilen Transfer fonksiyonu, sistemin çıktısını frekans uzayındaki girdisiyle ilişkilendirir. Hem iç dinamikleri hem de dış etkenleri içeren sistemin dinamiklerini karakterize eder.

Transfer fonksiyonları sayesinde mekanik, elektrik veya kontrol sistemleri gibi karmaşık sistemlerin analizleri yapılabilir ve bu sistemlerin performansları gerekli modellemeler doğrultusunda istenilen noktaya getirilebilir.

Temel formüller:

$$G_{(s)} = \frac{Y(s)}{X(s)}$$

$G_{(s)}$ = Transfer fonksiyonu

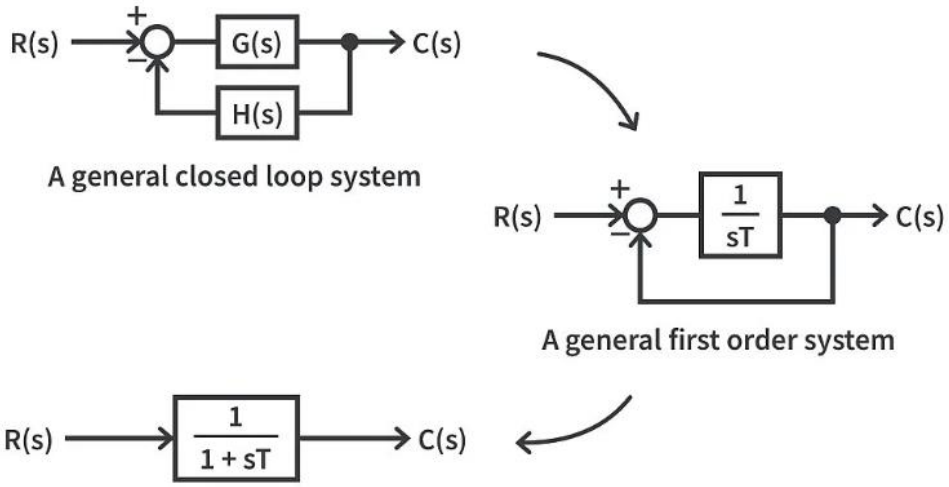
$Y_{(s)}$ = Sistem çıkışının Laplace dönüşümüdür

$X_{(s)}$ = Sistem girişinin Laplace dönüşümüdür

Bu bölümde inceleyeceğimiz Transfer fonksiyonu konusunda "Birinci Mertebe" ve "İkinci Mertebe" sistemlere odaklanacağız.

Birinci Mertebeden Sistemler

Birinci mertebeden bir transfer fonksiyonu tek bir kutupla karakterize edilir ve basit dinamiklere sahip sistemleri temsil etmek için yaygın olarak kullanılır.



Şekil 3.34 Birinci Mertebeden Sistemin Diyagram gösterimi

R(s) = Sistemin giriş parametresidir (input).

C(s) = Sistemin çıkış parametresidir (output).

H(s) = Sistemin geri besleme kazancıdır.

$G_{(s)} = \frac{1}{Ts}$ ve H(s) = 1 kabul edildiği bir sistemde Transfer fonksiyonunun ifadesi:

$$\frac{C_{(s)}}{R(s)} = \frac{G_{(s)}}{1+G_{(s)}\cdot H(s)} = \frac{\frac{1}{Ts}}{1+\frac{1}{T\cdot s}\cdot 1} = \frac{1}{Ts+1}$$

"τ" veya "T" parametresi zaman sabitini ifade eder ve $\tau = \mathrm{R.C}$ formülü ile gösterilir. Zaman sabiti, birinci mertebedeki sistemlerin dinamik davranışını belirleyen parametredir.

Birinci mertebedeki bir sistemin ideal olabilmesi için matematiksel ifadesi aşağıdaki gibi olmalıdır:

$$G_{(s)} = \frac{1}{Ts+1}$$

Birinci mertebeden ideal olmayan bir sistem için matematiksel ifade aşağıdaki gibidir:

$$G_{(s)} = \frac{K}{Ts + 1}$$

K = Sistemin kararlı hal kazancıdır.

T = τ = Zaman (time) sabitidir. Sistemin zaman tepkisini karakterize eder.

Laplace uzayından zaman uzayına geçiş yapılmak istenirse genel formül Şekil 3.35 'te gösterilmiştir.

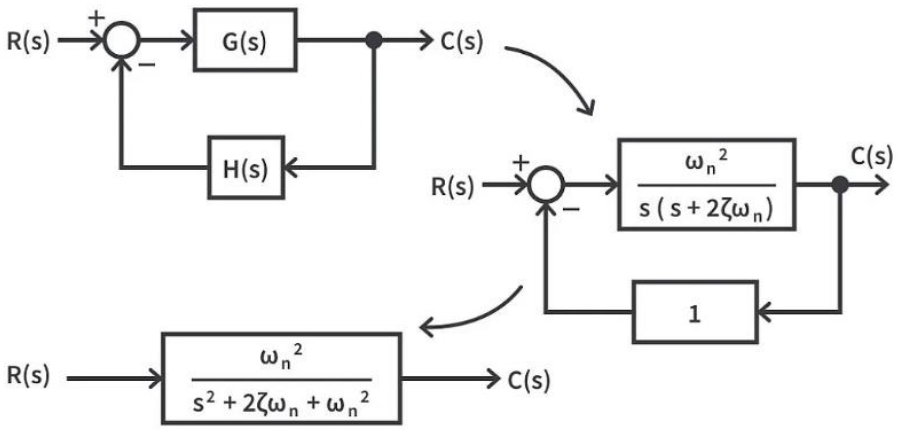
$$c(t) = L^{-1}\{C(s)\} = L^{-1}\left\{\frac{\frac{1}{T}}{s + \frac{1}{T}}\right\}$$

$$c(t) = \frac{1}{T}e^{-t/T}$$

Şekil 3.35 Birinci mertebeden sistemin uzaylar arası dönüşüm formülizasyonu

İkinci Mertebeden Sistemler

İkinci mertebeden bir Transfer fonksiyonu iki kutupla karakterize edilir ve genellikle daha karmaşık dinamikler sergileyen sistemler üzerinde kullanılır.



Şekil 3.36 İkinci Mertebeden Sistemin Diyagram gösterimi

R(s) = Sistemin giriş parametresidir (input).

C(s) = Sistemin çıkış parametresidir (output).

H(s) = Sistemin geri besleme kazancıdır.

ω_n = Doğal frekanstır.

ζ = Zeta, sönüm oranıdır.

İkinci mertebeden sistemlerin ideal formu aşağıdaki gibidir:

$G_{(s)} = \frac{\omega_n^2}{s(s+2\zeta w_n)}$ ve H(s) = 1 kabul edildiği bir sistemde Transfer fonksiyonunun ifadesi:

$$\frac{C_{(s)}}{R(s)} = \frac{G_{(s)}}{1+G_{(s)} \cdot H(s)} = \frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}$$

- $\zeta > 1$, aşırı sönümlü (over damped) sistemi ifade eder.
- $\zeta = 1$, kritik sönümlü sistemi ifade eder
- $0 < \zeta < 1$, az sönümlü sistem veya 2 imajiner kök bulunduğundan dolayı titreşimli sistem de denebilir.
- $\zeta = 0$, sönümsüz sistem veya aşırı titreşimli (sürekli dalgalı) sistem de denebilir.

3.6.4 Fiziksel Yorumlama ve Parametreler

Transfer fonksiyonları sistemin fiziksel özelliklerini ve parametrelerini kapsar. Sistemin davranışını kutuplar, sıfırlar, kazanç ve faz kayması gibi parametreler aracılığıyla belirtir. Kütle, sönümleme (damping), direnç ve endüktans gibi parametreler bu işlevler içinde temsiliyet kazanır ve bir sistemin temel dinamiklerini daha iyi anlamamızı sağlar.

3.6.5 Kontrol Sistemi Tasarımındaki Önemi

Transfer fonksiyonları, kontrol sistemi tasarımında bir sistemin kararlılığının tahmini, farklı girdilere verilecek yanıtı ve genel performansı kolaylaştıracak şekilde tasarımda önemli bir rol oynamaktadır. Kontrolörleri tasarlamak, parametrelerini ayarlamak ve değişen koşullar altında sistem performansını değerlendirmek için önemli bir rol teşkil eder.

Bütün ana fikre bakıldığında, Transfer fonksiyonunun işlevlerini anlamak hayati bir öneme sahiptir. Sistemlerin kontrol edilmesi gereken durumlarda sistemin dinamik davranışını anlamak ve ihtiyaca göre şekillendirmek için önemlidir. Yapılacak uygulamalar, basit mekanik sistemlerin modellenmesinden karmaşık sistemler için kontrol algoritmaları tasarlamaya kadar uzanır. Mühendislere gereken sistem analizi ve tasarımı için güçlü bir araç seti (toolset) sağlar.

3.6.6 Taret Robotun Transfer Fonksiyon Modellemesi

Robot ‘un kontrol sistemini ifade edebilmek için kullanılabilecek ve Laplace uzayında bulunan 2 adet dinamik denklem bulunmaktadır:

- Mekanik Sistem Modeli: Mekanik bileşenlerin (motorlar, dişliler vb.) dinamikleri aşağıda verilen formüller ile genel olarak karakterize edilmektedir.
  Motor Dinamiği = $G_{(s)} = \frac{1}{Ls+R}$ (motor endüktans ve direnci temsil eden formül)
  G(s) = Motorun transfer fonksiyonu ifade eder.

L = Motor indüktansını temsil eder
R = Motor direncini temsil eder
s = Laplace değişkenini temsil eder

Mekanik Yük Dinamiği = H(s) = $\frac{1}{ms+B}$ (doğrusal hareket için mekanik atalet ve sönümlemenin modellenmesi)

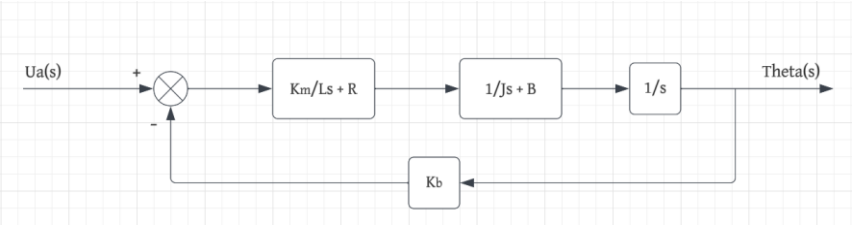
Mekanik Yük Dinamiği = H(s) = $\frac{1}{Js+B}$ (rotasyonel hareket için mekanik atalet ve sönümlemenin modellenmesi)

m = Sistemin kütlesini temsil eder
J = Sistemin eylemsizlik momentini temsil eder.
B = Rotasyonel (dönme) hareketi yapan sistemdeki sönümleme katsayısını veya sürtünme direncini temsil eder.

- Elektrik Sistem Modeli: Transfer fonksiyonu kullanılarak elektrik bileşenlerinin dinamiklerini karakterize etmek için genel olarak aşağıda verilen formül kullanılır.

  Sensör veya Aktüatör Dinamiği = F(s) = $\frac{1}{Js+e}$ (sensör veya aktüatör dinamiklerini tanımlar)

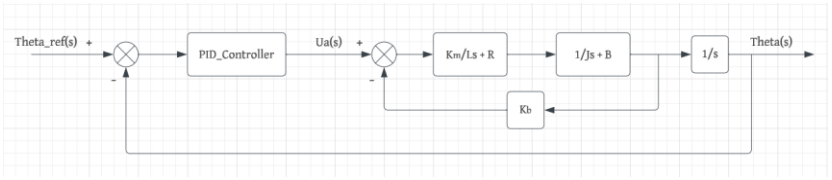
Buradaki amaç, Robot sistemindeki PID kontrolcüsünün nasıl tasarlanacağına dair sembolik bir gösterimi ele almaktır. Bundan dolayı, gerekli donanım ekipmanları bulunmadığından dolayı Robot 'un bütün parametreleri **örnek değerler** üzerinden hesaplanacaktır. Sistemin genel gösterimi aşağıda verilmiştir:



Şekil 3.37 Sistemin Genel Blok Diyagramı

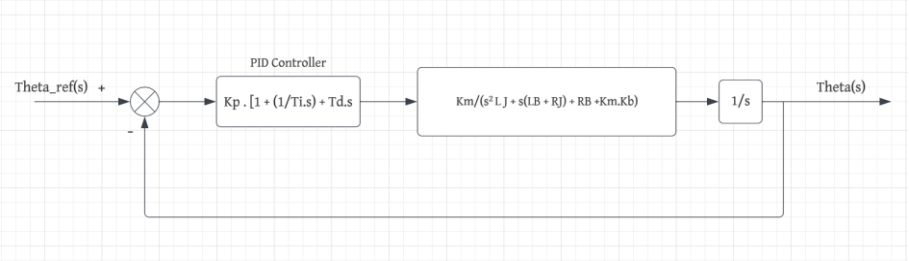
(R = 30 Ω, L = 0.75 H, J = 0,015 kg.m^2, B = 0,008 N.m.s/rad, Km = Kb = 0.22)

PID Tasarımı için Blok Diyagramının Oluşturulması ve Gerekli Hesaplamaların yapılması:



Şekil 3.38 Sistemin PID Kontrolcü Blok Diyagramı 1

$$T.F = \frac{\frac{K_m}{(Ls+R)(Js+B)}}{1+\frac{K_m K_B}{(Ls+R)(Js+B)}} = \frac{K_m}{s^2LJ+s(LB+RJ)+RB+K_m K_B} \qquad , PID = K_p = \left[1 + \frac{1}{T_i \cdot s} + T_d \cdot s\right]$$



Şekil 3.39 Sistemin PID Kontrolcü Blok Diyagramı 2

Ti $\longrightarrow \infty$, Td = 0 koşullarına göre;

$$T.F = \frac{\frac{K_m K_p}{s^3 LJ + s^2(LB+RJ) + s(RB+K_m K_B)}}{1+\frac{K_m K_p}{s^3 LJ + s^2(LB+RJ) + s(RB+K_m K_B)}} = \frac{K_m K_p}{s^3 LJ + s^2(LB+RJ) + s(RB+K_m K_B) + K_m K_p}$$

- Sistemin tip numarası 1 ve mertebesi ise 3 ‘tür.
- Tip 1 sistemde Ziegler Nichols yöntemi kullanılarak hesaplama yapılması uygun görülmüştür.

$$T.F = \frac{220K_p}{11{,}25s^3 + 456s^2 + 288{,}4s + 220Kp}$$ K.D = Karakteristik Denklem

$$K.D = 11{,}25s^3 + 456s^2 + 288{,}4s + 220Kp$$

s^3	11,25	288,4	$b_1 = \frac{-\begin{vmatrix} 11{,}25 & 288{,}4 \\ 456 & K \end{vmatrix}}{456} = \frac{-[(11{,}25 \cdot K) - (456.288{,}4)]}{456} = \frac{131510{,}4 - 11{,}25K}{456}$
s^2	456	K	
s^1	b1	0	$c_1 = \frac{-\begin{vmatrix} 456 & 220 \\ b1 & 0 \end{vmatrix}}{b1} = \frac{-[(456\,.\,0) - (b1\,.\,K)]}{b1} = K$
s^0	c1	0	

11,25 ⟳ + (131510,4 – 11,25K) / 456 > 0 ve K > 0 için:

456 ⟳ + 131510,4 > 11,25K $\longrightarrow$ K < 11689,813

(131510,4 – 11,25K) / 456 ⟳ +

K ⟳

$$\boxed{0 < K < 11689{,}813}$$

$11{,}25s^3 + 456s^2 + 288{,}4s + 11689{,}7 = 0$, $s \longrightarrow jw$ $\quad K_{cr} = 11689{,}813$

$11{,}25(jw)^3 + 456(jw)^2 + 288{,}4(jw) + 11689{,}7 = 0$ $\quad 220\ Kp_{cr} = K_{cr}$

$-11{,}25jw^3 - 456w^2 + 288{,}4\ jw + 11689{,}7 = 0$ $\quad Kp_{cr} = 53{,}135$

$jw(288{,}4 - 11{,}25w^2) + 456(25{,}6353 - w^2) = 0$

$288{,}4 - 11{,}25w^2 = 0$ $\quad 25{,}6353 - w^2 = 0$

$w \cong 5{,}063$ rad/s $\quad w \cong 5{,}063$ rad/s

$w_{cr} = 2\pi / P_{cr} \longrightarrow P_{cr} = 2\pi / (5{,}063) = 1{,}241$

PID için Katsayıların Bulunması:

$P = 0{,}6 \cdot K_{cr} = (0{,}6) \cdot (53{,}135) = 31{,}881$

$T_i = 0{,}5 \cdot P_{cr} = (0.5) \cdot (1{,}241) = 0{,}6205$

$T_d = 0.125 \cdot P_{cr} = (0.125) \cdot (1{,}241) \cong 0{,}155$

3.7 Sistemin Yatay Atış Hareketi

3.7.1 Yatay Atış Hareketine Giriş

Yatay atış hareketi, fizik biliminin önemli bir alanını oluşturmaktadır. Bu alanda yapılan çalışmalar, nesnelerin atmosferik koşullar altındaki hareketlerini anlamamıza ve füze/mermi gibi nesnelerin yatay atışlarını daha iyi bir şekilde modellememize olanak tanımaktadır.

Bu bölüm, mermilerin yatay atış hareketlerinin incelenmesi ve bu atışların matematiksel olarak modellenmesi üzerine odaklanmaktadır. Yatay atış hareketini tanımlamak için temel fizik prensipleri ve matematiksel denklemler kullanılacaktır.

Çalışmanın amacı, Taret Robot 'da bulunan dart mermisi üzerinde yatay atış hareketi prensiplerini uygulayarak merminin atış açısı, atılma hızı, uçuş süresi ve yer çekiminin mermi atışının sonucuna etkisini incelemektir.

Bu giriş bölümü, çalışmanın genel amaçlarını ve kapsamını belirtmektedir. İlerleyen bölümlerde, mermi hareketinin matematiksel modellenmesi ve teorik deneylerin sonuçları gözler önüne sunularak, bu sonuçların analizi üzerine odaklanılacaktır. Bu çalışma, mermi fiziği ve yatay atış hareketinin anlaşılmasına katkıda bulunmayı amaçlamaktadır.

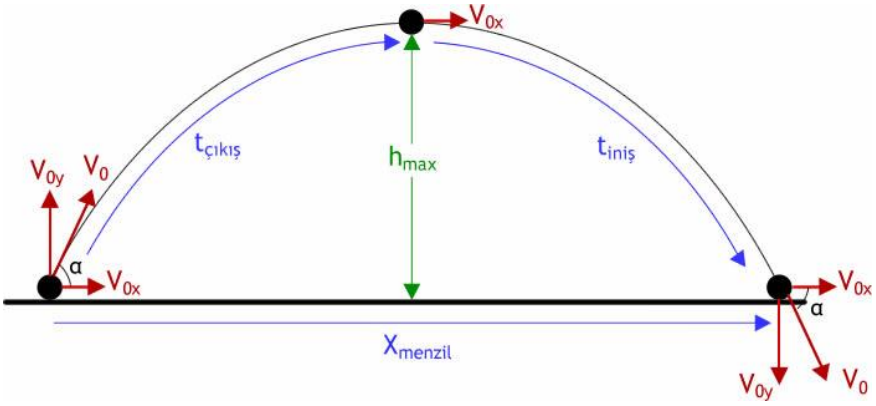
3.7.2 Yatay Atış Hareketinin Matematiksel Modellenmesi

Yatay atış hareketi, temelde nesnenin yatay ve dikey bileşenlerinin ayrıldığı bir harekettir. Yatay atış hareketinde, yerçekimi ivmesinin (g) sabit kabul edildiği ve hava direnci ihmal edildiği bir durumda temel formüller şunlardır:

Merminin yatayda alacağı mesafe toplam mesafe "R" (menzil uzaklığı) ve dikeyde alacağı mesafe "h" (tepe noktası) olarak kabul edilmektedir.

Merminin atılacağı yönü "+y" koordinatı kabul edersek, yer çekimi ivmesi "-y" koordinatı yönündedir.

Havanın direnci ihmal edildiğinden dolayı merminin X ekseni yönündeki ivmesi a(x) = 0 kabul edilir.



Şekil 3.40 Yatay Atış Hareketi

Bir cismin ivmesi 0 kabul edilirse, cismin hızı sabit kalır ve zamanla değişmez (sürtünme ve direnç etkilerinin ihmal edildiği bir ortamda). Bundan dolayı, merminin X eksenindeki ivmesi 0 olduğundan dolayı, mermi hızının X bileşeni zamanla değişmez (sabit kalır).

Merminin gideceği yön ile yer çekimi ivmesinin vektörel yönü terstir. Bu yüzden, merminin Y eksenindeki ivmesi a(y) = - g = - 9,81 m/s^2 'dir.

Merminin atılacağı koordinatı (0,0) orijin noktası olarak kabul edersek;

Vx = Vo.cosθ

Vy = Voy + a(y).t = Vo.sinθ – g.t

Tepe noktasında Vy = 0 'dır.

0 = Vo.sinθ – g.t

t_iniş = t_çıkış = $\frac{V_0 \sin \theta}{g}$

t_uçuş (merminin havada kalma süresi) = $\frac{2 . V_0 \sin \theta}{g}$

Merminin herhangi bir anda Y eksenindeki koordinatı için;

Y = Yo + Voy . t + $\frac{1}{2} a_y t^2$

Eğer t = t_çıkış ise;

h = Vo.sinθ . $\frac{V_0 \sin \theta}{g}$ - $\frac{1}{2}$ g $\frac{V_0^2 \sin^2 \theta}{g^2}$

h = $\frac{V_0^2 \cdot \sin^2 \theta}{2g}$

Merminin herhangi bir anda X eksenindeki koordinatı için;

X = Xo + Vox . t + $\frac{1}{2} a_x t^2$

Eğer t = t_çıkış ise X = R/2 'dir.

Eğer t = t_uçuş ise X = R 'dir.

R = (Vo . cosθ . 2Vo . sinθ) / g

R = $\frac{V_0^2 \sin(2\vartheta)}{g}$

Herhangi bir sinüs açısının maksimum değeri 1 olduğu durumda ve mermiye verilen ilk hız değerinin değişmediği bir koşulda, R (menzil uzaklığı) maksimum değerine ulaşacaktır. Böylece, $\sin(2\vartheta)$ = 90 derece , θ = 45 derece olacaktır. θ = 45 derece olduğunda mermi X ekseninde maksimum uzaklığa gidecektir.

3.7.3 Robotun Mermi Dinamiği ve Atış Hesaplamaları:

Dart mermisi ilk olarak servo motorun ittirme kuvveti ile hareketine başlar ve 2 DC motor tekerleğinin arasına ulaşır. Dart mermisi DC motor tekerleklerinin arasına ulaştıktan sonra, 2 DC motor tekerleğinin yanal yüzeylerine temas ederek dışarı fırlatılır ve yatay atış hareketine başlar.

Buradaki amaç, Robot sistemindeki dart mermisinin nasıl yatay atış hareketi yapacağına dair sembolik bir gösterimi ele almaktır. Bundan dolayı, gerekli donanım ekipmanları bulunmadığından dolayı Robot ‘un bütün parametreleri hesaplanabilen değerler veya örnek değerler üzerinden ele alınacaktır.

1. Dart Mermisinin Alacağı Yol

78 – 1.014 = 76,986 mm (merminin hazne uzunluğu)

76,986 mm – 72 mm (merminin uzunluğu) = 4.986 mm , 4.986 mm + 22,5 mm = 27,486 mm

27,486 mm = 0,027486 m

2. Servo Motorun İttiği Başlangıç Hızı

0,027486 m yolu 0.1 saniyede aldığını ve sürtünmeleri ihmal ettiğimizi kabul edersek:

$x = v \cdot t$

$v = \frac{x}{t}$ = (0,027486) / 0.1 = 0,27486 m/s ‘dir

3. DC Motor Tekerleklerinin Dönüş Hızı

DC motorlar 5V besleme ile 7000 rpm (devir / dakika) hızında dönecek şekilde kabul ediyoruz.

DC motor tekerleklerinin çapı 28.5 mm ‘dir ve DC motor hızını tekerleklere kayıp olmadan aktardığını (ikisinin bir döndüğünü) kabul ediyoruz. Bu koşullar altında;

RPM değerinin saniye başına gösterimi = 7000 / 60 saniye ≅ 116,67 devir/saniye

Açısal hız (rad/s) = 116.67 devir/saniye x 2π ≅ 733,059 rad/s

r = 14.25 mm = 0.01425 m (DC motor tekerleğinin yarı çapı)

v = w . r = (733,059 rad/s) . (0.01425) ≅ 10,446 m/s ‘dir.

4. Yatay Atış Hesaplamaları

Dart mermisinin tekerleklere temas ettiği noktadaki lineer hızı 10,446 m/s ‘dir. Bu değer, dart yatay atış hareketindeki başlangıç hızı (Vo) olarak kabul edilir.

Dart mermisinin Vo = 10,446 m/s , θ = 30 derece ile atılacağı ve yerden yüksekliği h = 1 m olan bir ortamda: a(y) = - 9,81 m/s^2 , a(x) = 0

a) Mermi ne kadar süre sonra yere çarpacaktır?

$Y = Yo + Voy \cdot t - \frac{1}{2}gt^2$

Merminin bulunduğu konum orijin noktası kabul edilirse (0,0) Yo = 0 olacaktır.

$-1 = 0 + (10,446) \cdot \sin 30 \cdot t - 4.9t^2$

$4.9t^2 - 6,5975.t - 1 = 0$

$\Delta = b^2 - 4ac = (-5,223)^2 - 4.(4.9).(-1) = (27,2797) + 19,646 = 46,8797$

$\sqrt{\Delta} \cong 6,8469$

$t_1 = \frac{-b+\sqrt{\Delta}}{2a} = [(5,223) + (6,8469)] / (9,8) = 1,2316$ saniye

b) Merminin yere çarptığı andaki hızı (V)

$\vec{V} = V_x i + V_y j$

Vx = Vox + a(x).t = Vox

Vx = Vox = Vo . cos30 = (10,446) . cos30 ≅ 9,0465 m/s

Voy = Vo . sin30 = (10,446) . sin30 ≅ 5,223 m/s

Vy = Voy + a(y).t = Vo.sinθ – g.t = (5,223) – 9.81 . (1,2316) ≅ - 6,859 m/s

$\vec{V}$ = (9,0465i – 6,859j) m/s

$|\vec{v}| = \sqrt{(9,0465)^2 + (-6,859)^2}$

$|\vec{v}| \cong 11.3527$ m/s

Merminin Y eksenindeki yere çarpma hızının negatif olmasının sebebi vektörel gösterimden kaynaklıdır. Mermi yere çarptığında X eksenindeki hızı (Vx) pozitif eksene doğru iken, Y eksenindeki hızı (Vy) ise orijin noktasına göre -Y koordinatına doğru hareket etmektedir.

BÖLÜM 4

4.1 Sonuç

Görüntü İşleme ve Arduino mikrodenetleyicisi kullanılan Nerf Taret uygulamasında, eğitilen model üzerinde uygulanan Görüntü İşleme kalibrasyonları ile hedefin tespiti yüksek oranlarla sağlanmıştır. Başarılı bir şekilde tespit edilen nesnenin X ve Y koordinatları Arduino ‘ya gönderilerek Taret ‘in başarılı bir şekilde nesneyi izlemesi ve takip etmesi sağlanmıştır.

Nesnenin algılanması ve tespit edilebilmesindeki başarı oranını etkileyen faktörlerin ortamın ışık faktörüne, nesnenin rengine, kablosuz bağlantı kalitesine, kameranın piksel kalitesine bağlı olduğu sonucuna varılmıştır.

Video akışının EpocCam gibi bir mobil uygulama üzerinden alınması ve telefon kamerasının kullanılması, kamera kalitesinin arttırılmasından dolayı video akışının yüksek piksel kaliteleriyle elde edilmesini sağlamıştır ve gerçek zamanlı nesne tespitini kolaylaştırarak taretin belirlenen hedefi daha hızlı bir şekilde algılayarak takip etmesini ve taretin uygun pozisyonu alarak ateş etmesini sağlamıştır. Böylece bu proje, işlevsel bir otomatik taret sistemi oluşturma hedefini başarıyla tamamlamıştır.

KAYNAKÇA

Redmon, J., & Farhadi, A. (2018). "YOLOv3: An Incremental Improvement." arXiv preprint arXiv:1804.02767.

Bojarski, M., et al. (2016). "End to End Learning for Self-Driving Cars." arXiv preprint arXiv:1604.07316.

Liu, W., et al. (2016). "SSD: Single Shot MultiBox Detector." ECCV 2016: 9905, 21-37.

Simonyan, K., & Zisserman, A. (2014). "Very Deep Convolutional Networks for Large-Scale Image Recognition." arXiv preprint arXiv:1409.1556.

Ren, S., et al. (2015). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." NIPS 2015: 91-99.

Bradski, G., & Kaehler, A. (2008). Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. O'Reilly Media.

Szeliski, R. (2010). Computer Vision: Algorithms and Applications. Springer.

Pratt, W. K. (2007). Digital Image Processing: PIKS Inside. John Wiley & Sons.

Jani, S., & Shah, R. (2019). Hands-On Machine Learning with Arduino. Packt Publishing.

Deitel, P., & Deitel, H. (2017). Internet & World Wide Web: How to Program. Pearson.

He, K., et al. (2017). "Mask R-CNN." IEEE International Conference on Computer Vision (ICCV), 2961-2969.

Girshick, R. (2015). "Fast R-CNN." IEEE International Conference on Computer Vision (ICCV), 1440-1448.

Hinton, G., & Salakhutdinov, R. (2006). "Reducing the Dimensionality of Data with Neural Networks." Science, 313(5786), 504-507.

Szegedy, C., et al. (2015). "Going Deeper with Convolutions." IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1-9.

Goodfellow, I., et al. (2014). "Generative Adversarial Nets." Advances in Neural Information Processing Systems (NIPS), 2672-2680.

OpenCV. (2023). "OpenCV Documentation." Retrieved from https://docs.opencv.org/4.8.0/. (Date of Access: November 2023)

YOLOv8 GitHub Repository - https://github.com/ultralytics/ultralytics (Date of Access: November 2023)

Kazi, F. S., et al. (2021). "Robust Target Tracking and Shooting Control of a Turret Robot Using Deep Reinforcement Learning." IEEE Access, 9, 51925-51938.

Ribeiro, A. S., et al. (2018). "Design and Control of an Autonomous Sentry Turret." IFAC-PapersOnLine, 51(15), 224-229.

DeLaurentis, K. J., et al. (2016). "Optimal Target Acquisition and Engagement by Autonomous Weapon Systems." IEEE Aerospace Conference, 1-12.

Roy, K., et al. (2015). "A Novel Control System Design for a Remote Operated Turret." International Journal of Mechanical and Production Engineering Research and Development, 5(4), 1-10.

Hu, L., et al. (2014). "Modeling and Simulation of a Remote Weapon Station Turret." Proceedings of the 33rd Chinese Control Conference (CCC), 3633-3638.

EKLER

Robotun Python kod bölümü:

```
import cv2
import pygetwindow as getWindow
import pyautogui as autoGUI
import numpy as np
import serial
import time
from ultralytics import YOLO

ser_arduino = serial.Serial('COM3', 115200)

obs_window = getWindow.getWindowsWithTitle('Camera Hub')[0]

left, top, width, height = obs_window.left, obs_window.top, obs_window.width, obs_window.height

model_path = "C:\\Users\\y4sef\\Downloads\\kod_ve_ptle\\train29\\weights\\best.pt"

model = YOLO(model_path)

hue_lower_apple = 10
hue_upper_apple = 40

hue_lower_shirt = 40
hue_upper_shirt = 80

hue_lower_green = 90
hue_upper_green = 130
```

```
hue_lower_yellow = 20
hue_upper_yellow = 30

threshold = 0.01

min_size_threshold = 5
max_size_threshold = 20000

max_confidence = 0

coordinates_to_send = None

old_coordinates = None

old_centerX = None
old_centerY = None
current_coordinateX = None
current_coordinateY = None

target_width, target_height = 640, 480

while True:
    screenshot = autoGUI.screenshot(region=(left, top, width, height))
    obs_capture = cv2.cvtColor(np.array(screenshot), cv2.COLOR_RGB2BGR)

    obs_capture_resized = cv2.resize(obs_capture, (target_width, target_height))

    hsv_frame = cv2.cvtColor(obs_capture_resized, cv2.COLOR_BGR2HSV)

    mask_apple = cv2.inRange(hsv_frame, (hue_lower_apple, 50, 50), (hue_upper_apple, 255,
255))
```

```
    mask_shirt = cv2.inRange(hsv_frame, (hue_lower_shirt, 50, 50), (hue_upper_shirt, 255, 255))
    mask_green = cv2.inRange(hsv_frame, (hue_lower_green, 50, 50), (hue_upper_green, 255, 255))
    mask_yellow = cv2.inRange(hsv_frame, (hue_lower_yellow, 50, 50), (hue_upper_yellow, 255, 255))

    combined_mask = cv2.bitwise_or(cv2.bitwise_or(cv2.bitwise_or(mask_apple, mask_green), mask_shirt), mask_yellow)

    filtered_frame = cv2.bitwise_and(obs_capture_resized, obs_capture_resized, mask=combined_mask)

    results = model(filtered_frame)[0]

    max_confidence = 0

    coordinates_to_send = None

    for result in results.boxes.data.tolist():
        x1, y1, x2, y2, score, class_id = result
        box_size = (x2 - x1) * (y2 - y1)

        if score > threshold and min_size_threshold < box_size < max_size_threshold:
            center_x = (x1 + x2) / 2
            center_y = (y1 + y2) / 2

            if hue_lower_green <= (hue_lower_apple + hue_upper_apple) / 2 <= hue_upper_green:
                score += 0.2

            if hue_lower_yellow <= (hue_lower_apple + hue_upper_apple) / 2 <= hue_upper_yellow:
                score += 0.5
```

```
            if score > max_confidence:
                max_confidence = score
                coordinates_to_send = (center_x, center_y)
                current_coordinateX = int(center_x)
                current_coordinateY = int(center_y)


    def map_value(value, from_low, from_high, to_low, to_high):
        return (value - from_low) * (to_high - to_low) / (from_high - from_low) + to_low



    if coordinates_to_send is not None:
        data_to_send = f"{int(coordinates_to_send[0])},{int(coordinates_to_send[1])}\n"
        mapped_center_x = map_value(center_x, 0, 1920, 0, 180)
        mapped_center_y = map_value(center_y, 0, 1080, 0, 180)
        print(f"X= {mapped_center_x:.2f} angle, Y= {mapped_center_y:.2f} angle")
        ser_arduino.write(data_to_send.encode())
        old_centerX = int(coordinates_to_send[0])
        old_centerY = int(coordinates_to_send[1])



    if coordinates_to_send is None:
        if (((old_centerX != None) and (old_centerY != None)) and ((old_centerX ==
current_coordinateX ) and (old_centerY == current_coordinateY))):
            old_coordinates = (old_centerX, old_centerY)
            ser_arduino.write(old_coordinates.encode())


        else:
            print("Nesne Algilanmiyor")
```

```
    if coordinates_to_send is not None:
        x, y = coordinates_to_send
        cv2.rectangle(obs_capture_resized, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 4)
        label = f"Coordinates: ({int(x)}, {int(y)}) | Confidence: {max_confidence:.2f}"
        print(f"X coordinate: {int(x)}, Y coordinate: {int(y)}")
        cv2.putText(obs_capture_resized, label, (int(x1), int(y1) - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 3, cv2.LINE_AA)

    cv2.imshow('Frame', obs_capture_resized)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

ser_arduino.close()
cv2.destroyAllWindows()
```

Robotun Arduino IDE kod bölümü:

```
#include <Servo.h>

Servo servoZ; //orijin X servo
Servo servoY; //orijin Y servo
Servo servoD; //nerf bullet push (115 - 15) servo


int baslangic_taret_positionY = 85;
int baslangic_taret_positionZ = 100;
int dart_push_position = 15;
int dart_back_position = 115;

int hedefin_x_kordinat_acisi = 0;
int hedefin_y_kordinat_acisi = 0;
int hedefin_ters_X_koordinat_acisi = 0;
int hedefin_ters_Y_koordinat_acisi = 0;

int taret_Z_ekseni_ust_sinir_degeri = 180; //Z ekseninde ust sınır 180 derecedir
int taret_Z_ekseni_alt_sinir_degeri = 0; //Z ekseninde alt sınır 0 derecedir
int taret_Y_ekseni_ust_sinir_degeri = 125; //Y ekseninde ust sınır 40 derecedir,
85 (default) + 40 = 125
int taret_Y_ekseni_alt_sinir_degeri = 70; //Y ekseninde alt sınır 15 derecedir,
85 (default) - 15 = 70

int eski_Z_pozisyonu = 0;
int eski_Y_pozisyonu = 0;
int current_Z_position = 0;
int current_Y_position = 0;
int new_Z_position = 0;
int new_Y_position = 0;

int Z_ekseni_pozisyon_farki = 0;
int Y_ekseni_pozisyon_farki = 0;

int in1 = 9;
int in2 = 10;
int in3 = 11;
int in4 = 12;

int hedef_Z_koordinati_piksel_degeri = 0;
int hedef_Y_koordinati_piksel_degeri = 0;

int current_Z_pixel_coordinate = 0;
int current_Y_pixel_coordinate = 0;

int konumsal_Z_piksel_farki = 0;
```

```
int konumsal_Y_piksel_farki = 0;

int old_Z_pixel_coordinate = 0;
int old_Y_pixel_coordinate = 0;

bool moveServoD = false;

void setup() {
   delay(2000); //Arduino 'nun kararli hale gecmesi icin 2 saniye sistemi
bekletiyoruz.
  servoZ.attach(5);
  servoY.attach(7);
  servoD.attach(8);
  servoD.write(dart_back_position);
  delay(1000);
  servoZ.write(baslangic_taret_positionZ);
  delay(1000);
  servoY.write(baslangic_taret_positionY);
  delay(1000);
  Serial.begin(115200); // Seri iletisimin baslatilması
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

}

void loop() {
  //DC motorlarin H - Bridge ile surulmeye baslanmasi
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);

  servoD.write(dart_back_position);

  if (Serial.available() > 0) {
    String data = Serial.readStringUntil('\n');
    int delimiter = data.indexOf(",");

    //Asenkron iletisim
    if (delimiter != -1) {
      int Z_konumsal_hata = 0;
      int Y_konumsal_hata = 0;

      //Z için mevcut konum degerini piksel olarak algilamak
      current_Z_position = servoZ.read();
      old_Z_pixel_coordinate = map(new_Z_position, 0, 180, 0, 640);
```

```
      //Y için mevcut konum degerini piksel olarak algilamak
      current_Y_position = servoY.read();
      old_Y_pixel_coordinate = map(current_Y_position, 0, 180, 0, 480);

      //Gelen koordinat degerlerinin ayristirilmasi
      int centerX = data.substring(0, delimiter).toInt();
      int centerY = data.substring(delimiter + 1).toInt();

      //Goruntu 'nun piksel 'den aci degerine cevrilmesi
      hedefin_ters_X_koordinat_acisi = map(centerX, 0, 640, 0, 180);
      hedefin_ters_Y_koordinat_acisi = map(centerY, 0, 480, 0, 180);

     //Ters goruntuyu duzeltmek icin "180 - hedef aci degeri" formulu uygulanir.
      hedefin_x_kordinat_acisi = 180 - hedefin_ters_X_koordinat_acisi;
      hedefin_y_kordinat_acisi = 180 - hedefin_ters_Y_koordinat_acisi;

      //X ekseni icin koordinatlara oran verilmesi
      if(hedefin_x_kordinat_acisi > 160 && hedefin_x_kordinat_acisi <= 180){
        hedefin_x_kordinat_acisi = hedefin_x_kordinat_acisi - 33;
      }
       else if(hedefin_x_kordinat_acisi > 140 && hedefin_x_kordinat_acisi <=
160){
        hedefin_x_kordinat_acisi = hedefin_x_kordinat_acisi - 30;
      }
       else if(hedefin_x_kordinat_acisi > 120 && hedefin_x_kordinat_acisi <=
140){
        hedefin_x_kordinat_acisi = hedefin_x_kordinat_acisi - 15;
      }
       else if(hedefin_x_kordinat_acisi > 110 && hedefin_x_kordinat_acisi <=
120){
        hedefin_x_kordinat_acisi = hedefin_x_kordinat_acisi - 10;
      }
       else if(hedefin_x_kordinat_acisi > 105 && hedefin_x_kordinat_acisi <=
110){
        hedefin_x_kordinat_acisi = hedefin_x_kordinat_acisi - 4;
      }
       else if(hedefin_x_kordinat_acisi > 100 && hedefin_x_kordinat_acisi <=
105){
        hedefin_x_kordinat_acisi = hedefin_x_kordinat_acisi - 2;
      }
      else if(hedefin_x_kordinat_acisi >= 0  && hedefin_x_kordinat_acisi < 20){
        hedefin_x_kordinat_acisi = hedefin_x_kordinat_acisi + 40;
      }
     else if(hedefin_x_kordinat_acisi >= 20 && hedefin_x_kordinat_acisi <= 40){
        hedefin_x_kordinat_acisi = hedefin_x_kordinat_acisi + 30;
      }
      else if(hedefin_x_kordinat_acisi > 40 && hedefin_x_kordinat_acisi <= 60){
        hedefin_x_kordinat_acisi = hedefin_x_kordinat_acisi + 20;
      }
```

```
      else if(hedefin_x_kordinat_acisi > 60 && hedefin_x_kordinat_acisi <= 80){
        hedefin_x_kordinat_acisi = hedefin_x_kordinat_acisi + 15;
      }
      else if(hedefin_x_kordinat_acisi > 80 && hedefin_x_kordinat_acisi <= 90){
        hedefin_x_kordinat_acisi = hedefin_x_kordinat_acisi + 10;
      }
      else if(hedefin_x_kordinat_acisi > 90 && hedefin_x_kordinat_acisi <= 100){
        hedefin_x_kordinat_acisi = hedefin_x_kordinat_acisi + 5;
      }

      //Y ekseni icin koordinatlara oran verilmesi
      if(hedefin_y_kordinat_acisi > 120 && hedefin_y_kordinat_acisi <= 124){
        hedefin_y_kordinat_acisi = hedefin_y_kordinat_acisi + 4;
      }
       else if(hedefin_y_kordinat_acisi > 115 && hedefin_y_kordinat_acisi <=
120){
        hedefin_y_kordinat_acisi = hedefin_y_kordinat_acisi + 3;
      }
       else if(hedefin_y_kordinat_acisi > 105 && hedefin_y_kordinat_acisi <=
110){
        hedefin_y_kordinat_acisi = hedefin_y_kordinat_acisi + 6;
      }
       else if(hedefin_y_kordinat_acisi > 100 && hedefin_y_kordinat_acisi <=
105){
        hedefin_y_kordinat_acisi = hedefin_y_kordinat_acisi + 10;
      }
      else if(hedefin_y_kordinat_acisi > 90 && hedefin_y_kordinat_acisi <= 95){
        hedefin_y_kordinat_acisi = hedefin_y_kordinat_acisi + 20;
      }
      else if(hedefin_y_kordinat_acisi > 85 && hedefin_y_kordinat_acisi <= 90){
        hedefin_y_kordinat_acisi = hedefin_y_kordinat_acisi + 25;
      }
      else if(hedefin_y_kordinat_acisi > 75 && hedefin_y_kordinat_acisi <= 85){
        hedefin_y_kordinat_acisi = hedefin_y_kordinat_acisi + 20;
      }
      else if(hedefin_y_kordinat_acisi >= 70 && hedefin_y_kordinat_acisi <= 75){
        hedefin_y_kordinat_acisi = hedefin_y_kordinat_acisi + 28;
      }

      new_Z_position = hedefin_x_kordinat_acisi;
      new_Y_position = hedefin_y_kordinat_acisi;

      current_Z_position = servoZ.read();
      current_Y_position = servoY.read();

       if((new_Z_position >= 0 && new_Z_position <= 180) && (new_Y_position >=
70 && new_Y_position <= 125)){
        //Hedef X ve Y koordinat acilarinin piksel degerleri
        hedef_Z_koordinati_piksel_degeri = map(new_Z_position, 0, 180, 0, 640);
```

```
hedef_Y_koordinati_piksel_degeri = map(new_Y_position, 0, 180, 0, 480);
//default Z = 100, Y = 85
Z_ekseni_pozisyon_farki = new_Z_position - current_Z_position;
Y_ekseni_pozisyon_farki = new_Y_position - current_Y_position;

//Z Servosunu dondurmek
if(Z_ekseni_pozisyon_farki > 0){
  for(int i = 0; i <= Z_ekseni_pozisyon_farki; i++){
    servoZ.write(current_Z_position + i);
    delay(7);
  }
}
else if(Z_ekseni_pozisyon_farki < 0){
  Z_ekseni_pozisyon_farki = abs(Z_ekseni_pozisyon_farki);
  for(int i = 0; i <= Z_ekseni_pozisyon_farki; i++){
    servoZ.write(current_Z_position - i);
    delay(7);
  }
}
else if (Z_ekseni_pozisyon_farki == 0){
  servoZ.write(current_Z_position);
}

current_Z_position = servoZ.read();
//Z servosu icin konumsal hata (Kp - error)
Z_konumsal_hata = new_Z_position - current_Z_position;
servoZ.write(current_Z_position + Z_konumsal_hata);


//Y Servosunu dondurmek
if(Y_ekseni_pozisyon_farki > 0){
  for(int i = 0; i <= Y_ekseni_pozisyon_farki; i++){
    servoY.write(current_Y_position + i);
    delay(7);
  }
}

else if(Y_ekseni_pozisyon_farki < 0){
  Y_ekseni_pozisyon_farki = abs(Y_ekseni_pozisyon_farki);
  for(int i = 0; i <= Y_ekseni_pozisyon_farki; i++){
    servoY.write(current_Y_position - i);
    delay(7);
  }
}

else if (Y_ekseni_pozisyon_farki == 0){
  servoY.write(current_Y_position);
}
```

```
        current_Y_position = servoY.read();
        //Y servosu icin konumsal hata (Kp - error)
        Y_konumsal_hata = new_Y_position - current_Y_position;
        servoY.write(current_Y_position + Y_konumsal_hata);

        //Konumsal piksel farki + veya - deger olabilir
            konumsal_Z_piksel_farki  =  abs(hedef_Z_koordinati_piksel_degeri  -
old_Z_pixel_coordinate);
            konumsal_Y_piksel_farki  =  abs(hedef_Y_koordinati_piksel_degeri  -
old_Y_pixel_coordinate);

        // 640/180 = 3.555555 , Z için 1 derece = 3.5555 pikseldir, 3.5555 x 3
derece = 10,6665
        // 480/180 = 2.666666 , Y için 1 derece ) 2.6666 pikseldir, 2.6666 x 3
derece = 8
        //ServoD 'nin calisicagi zaman icin flag ayarlanmasi ve minimal piksel
oynamalarinin onune gecilmesi
        if (konumsal_Z_piksel_farki > 35 || konumsal_Y_piksel_farki > 26){
          moveServoD = true; //Servo D flag
        }

        //Nerf mermisini ileri/geri ittirmek
        if (moveServoD) {
          delay(20);
          servoD.write(dart_push_position);
          delay(350);
          servoD.write(dart_back_position);
          moveServoD = false;
        }
        else{
          servoD.write(dart_back_position);
          moveServoD = false;
        }

      }

      else if((new_Y_position < 70 || new_Y_position > 125) && (new_Z_position
<= 0 && new_Z_position >= 180)){
        hedef_Z_koordinati_piksel_degeri = map(new_Z_position, 0, 180, 0, 640);

        //Z ekseni icin hareket
        Z_ekseni_pozisyon_farki = new_Z_position - current_Z_position;

        if(Z_ekseni_pozisyon_farki > 0){
          for(int i = 0; i <= Z_ekseni_pozisyon_farki; i++){
            servoZ.write(current_Z_position + i);
            delay(7);
          }
        }
```

```
        else if(Z_ekseni_pozisyon_farki < 0){
          Z_ekseni_pozisyon_farki = abs(Z_ekseni_pozisyon_farki);
          for(int i = 0; i <= Z_ekseni_pozisyon_farki; i++){
            servoZ.write(current_Z_position - i);
            delay(7);
          }
        }

        else if(Z_ekseni_pozisyon_farki == 0){
          servoZ.write(current_Z_position);
        }

        //Z servosu icin konumsal hata (Kp - error)
        current_Z_position = servoZ.read();
        Z_konumsal_hata = new_Z_position - current_Z_position;
        servoZ.write(current_Z_position + Z_konumsal_hata);


        //Y ekseni icin hareket
        if(new_Y_position > 125){
                Y_ekseni_pozisyon_farki  =  taret_Y_ekseni_ust_sinir_degeri  -
current_Y_position;
          for(int i = 0; i <= Y_ekseni_pozisyon_farki; i++){
            servoY.write(current_Y_position + i);
            delay(7);
          }
          //Y servosu icin konumsal hata (Kp - error)
          current_Y_position = servoY.read();
          //Y servosu icin konumsal hata
                     Y_konumsal_hata   =   taret_Y_ekseni_ust_sinir_degeri   -
current_Y_position;
          servoY.write(current_Y_position + Y_konumsal_hata);

                                        hedef_Y_koordinati_piksel_degeri      =
map(taret_Y_ekseni_ust_sinir_degeri, 0, 180, 0, 480);
        }

        else if(new_Y_position < 70){
          //Min=88  , Max = 91
          if(hedefin_y_kordinat_acisi > 50 && hedefin_y_kordinat_acisi <= 69){
            hedefin_y_kordinat_acisi = hedefin_y_kordinat_acisi + 30;
          }
          else if(hedefin_y_kordinat_acisi > 40 && hedefin_y_kordinat_acisi <=
50){
            hedefin_y_kordinat_acisi = hedefin_y_kordinat_acisi + 40;
          }
          else if(hedefin_y_kordinat_acisi > 30 && hedefin_y_kordinat_acisi <=
40){
```

```
            hedefin_y_kordinat_acisi = hedefin_y_kordinat_acisi + 45;
          }
          else if(hedefin_y_kordinat_acisi > 20 && hedefin_y_kordinat_acisi <=
30){
            hedefin_y_kordinat_acisi = hedefin_y_kordinat_acisi + 55;
          }
          else if(hedefin_y_kordinat_acisi > 10 && hedefin_y_kordinat_acisi <=
20){
            hedefin_y_kordinat_acisi = hedefin_y_kordinat_acisi + 62;
          }
          else if(hedefin_y_kordinat_acisi <= 10){
            hedefin_y_kordinat_acisi = hedefin_y_kordinat_acisi + 70;
          }

          new_Y_position = hedefin_y_kordinat_acisi;

          Y_ekseni_pozisyon_farki = new_Y_position - current_Y_position;

          //Y Servosunu dondurmek
          if(Y_ekseni_pozisyon_farki > 0){
            for(int i = 0; i <= Y_ekseni_pozisyon_farki; i++){
              servoY.write(current_Y_position + i);
              delay(7);
            }
          }

          else if(Y_ekseni_pozisyon_farki < 0){
            Y_ekseni_pozisyon_farki = abs(Y_ekseni_pozisyon_farki);
            for(int i = 0; i <= Y_ekseni_pozisyon_farki; i++){
              servoY.write(current_Y_position - i);
              delay(7);
            }
          }

          else if (Y_ekseni_pozisyon_farki == 0){
            servoY.write(current_Y_position);
          }

          //Y servosu icin konumsal hata (error)
          current_Y_position = servoY.read();
          //Y servosu icin konumsal hata
          Y_konumsal_hata = new_Y_position - current_Y_position;
          servoY.write(current_Y_position + Y_konumsal_hata);

            hedef_Y_koordinati_piksel_degeri = map(new_Y_position, 0, 180, 0,
480);
        }
        //Konumsal piksel farki + veya - deger olabilir
```

```
            konumsal_Z_piksel_farki  =  abs(hedef_Z_koordinati_piksel_degeri  -
old_Z_pixel_coordinate);
            konumsal_Y_piksel_farki  =  abs(hedef_Y_koordinati_piksel_degeri  -
old_Y_pixel_coordinate);

        // 640/180 = 3.555555 , Z için 1 derece = 3.5555 pikseldir, 3.5555 x 3
derece = 10,6665
        // 480/180 = 2.666666 , Y için 1 derece ) 2.6666 pikseldir, 2.6666 x 3
derece = 8
        //ServoD 'nin calisicagi zaman icin flag ayarlanmasi ve minimal piksel
oynamalarinin onune gecilmesi
        if (konumsal_Z_piksel_farki > 35 || konumsal_Y_piksel_farki > 26){
          moveServoD = true; //Servo D flag
        }

        //Nerf mermisini ileri/geri ittirmek
        if (moveServoD) {
              delay(20);  //Taretin  hedefe  kilitlenmesi  icin  kisa  bir  sure
bekletiyoruz.
          servoD.write(dart_push_position);
          delay(350);
          servoD.write(dart_back_position);
          moveServoD = false;
        }
        else{
          servoD.write(dart_back_position);
          moveServoD = false;
        }

      }


    }
  }
}
```

ÖZGEÇMİŞ

Kimlik Bilgileri

Adı ve Soyadı: Yusuf Kini

Baba Adı: Burhan

Anne Adı: İnci

Doğum Tarihi: 1999

Doğum Yeri: Aksakal

Yusuf Kini ilk ve orta öğrenimini oturduğu ilçede bulunan Bandırma Ticaret Borsası İlköğretim Okulu'nda tamamladı. Bandırma Mesleki ve Teknik Anadolu Lisesi 'nin Çelik Gemi Yapımı bölümünden mezun olduktan sonra 2019 yılında Marmara Üniversitesi Teknoloji Fakültesi Mekatronik Mühendisliği Bölümü 'nde lisans eğitimine başladı. Lisans eğitimi sürecinde, 2022 yılında Innova Bilişim Çözümleri A.Ş çatısı altında Türk Telekom AR-GE binasında kısa dönem Siber Güvenlik stajyeri olarak, 2023 yılında Innova Bilişim Çözümleri A.Ş çatısı altında ITU Ayazağa Kampüsü Türk Telekom binasında uzun dönem Siber Güvenlik stajyeri olarak çalışmış ve tecrübe kazanmıştır.

İletişim Bilgileri

E-posta: yusufkini@outlook.com

Linkedin: www.linkedin.com/in/yusuf-kini-5b52071b2

Kimlik Bilgileri

Adı ve Soyadı: Hüseyin Özgür Ceylan

Baba Adı: Kenan

Anne Adı: Leman

Doğum Tarihi: 2001

Doğum Yeri: İstanbul

Hüseyin Özgür Ceylan ilk ve orta öğrenimini oturduğu ilçede bulunan Mustafa Mihriban Boysan İlköğretim Okulu'nda tamamladı. Liseyi Kadıköy Doğa Koleji'nde okuyup 2019 yılında mezun oldu ve ardından lisans eğitimine 2019 yılında Marmara Üniversitesi Teknoloji Fakültesi Mekatronik Mühendisliği Bölümü'nde başladı. Lisans eğitim süreci içerisinde YCS Endüstri Mamulleri Ltd. Şti. çatısı altında kısa dönem stajyerliği yapmıştır.

İletişim Bilgileri

E-posta: huozce@gmail.com

Linkedin: https://www.linkedin.com/in/hüseyin-özgür-ceylan-70007a241/