T.C.
MARMARA ÜNİVERSİTESİ
TEKNİK BİLİMLER MESLEK YÜKSEKOKULU

# GERÇEK ZAMANLI GÖRÜNTÜ İŞLEME İLE NESNE TANIMA SİSTEMİ

**360523012 Emre YAĞCI**

**360523033 Sabri SEVİNÇLİ**

**360523046 Bünyamin EREN**

**360523021 Tayfun KARATAŞ**

**BİTİRME PROJESİ**

**ELEKTRONİK VE OTOMASYON BÖLÜMÜ**

**ELEKTRONİK TEKNOLOJİSİ PROGRAMI**

**DANIŞMAN**

**Öğr. Gör. MERVE AYDIN**

**İSTANBUL 2025**

**T.C.**
**MARMARA ÜNİVERSİTESİ**
**TEKNİK BİLİMLER MESLEK YÜKSEKOKULU**

# GERÇEK ZAMANLI GÖRÜNTÜ İŞLEME İLE NESNE TANIMA SİSTEMİ

**360523012 Emre YAĞCI**

**360523033 Sabri SEVİNÇLİ**

**360523046 Bünyamin EREN**

**360523021 Tayfun KARATAŞ**

**BİTİRME PROJESİ**

**ELEKTRONİK VE OTOMASYON BÖLÜMÜ**

**ELEKTRONİK TEKNOLOJİSİ PROGRAMI**

**DANIŞMAN**

**Öğr. Gör. MERVE AYDIN**

**İSTANBUL 2025**

# ÖNSÖZ

Bu proje kapsamında Türkiye'de otonom araç sistemleri, endüstriyel üretim hattı izleme, güvenlik ve devriye robotları, depo ve lojistik otomasyon sistemleri, tarımsal otomasyon ve arama-kurtarma gibi birçok farklı alanda uygulanabilecek bir sistem tasarımı gerçekleştirilmiştir. Projemizin her aşamasında elde ettiğimiz bilgi ve deneyimler, mesleki gelişimimize önemli katkılar sağlamıştır.

Projeyi belirleme, yürütme ve sorunlarla başa çıkma sürecinde değerli danışmanlıklarıyla bizlere rehberlik eden Sayın Öğr. Gör. Merve AYDIN'a en içten teşekkürlerimizi sunarız. Ayrıca Marmara Üniversitesi'nin sağladığı laboratuvar, donanım ve teknik altyapı imkânları sayesinde proje sürecini verimli bir şekilde yürütebilme fırsatı bulduk.

Çalışmayı dört kişilik bir ekip olarak büyük bir uyum ve iş birliği içinde gerçekleştirdik. Projede yer alan Bünyamin EREN, Emre YAĞCI, Sabri SEVİNÇLİ, Tayfun KARATAŞ olarak, sürecin her aşamasında ortaya koyduğumuz ekip çalışması ve ortak emeğin, projemizi başarıyla tamamlamamızda en önemli etken olduğunu belirtmek isteriz.

Bünyamin EREN

Emre YAĞCI

Sabri SEVİNÇLİ

Tayfun KARATAŞ

23/06/2025

# İÇİNDEKİLER

**ÖNSÖZ** ... 2
**0.1. SEMBOLLER LİSTESİ** ... 8
**0.2. KISALTMALAR LİSTESİ** ... 9
**0.3 ŞEKİLLER LİSTESİ** ... 10
**BÖLÜM I: GİRİŞ VE AMAÇ** ... 11
**BÖLÜM II: GENEL BİLGİLER** ... 12
II.1. GENEL BİLGİLER ... **12**
II.2. TEORİK YAKLAŞIMLAR ... **13**
II.2.1. Yüz Tanıma Sistemi Tasarım Yaklaşımı ... 13
II.2.2. Renk Tabanlı Nesne Takip Yaklaşımı ... 14
II.2.3. Temel Yüz Takibi (Tracking) ... 14
II.2.4. YOLO Tabanlı Nesne Algılama ve Takip Yaklaşımı ... 14
II.2.5. Mod Geçişlerinde Stabilite Sağlama Yaklaşımı ... 14
**II.3. KAYNAK BİLGİLERİN İRDELENMESİ** ... **15**
II.3.1. Derin Öğrenme Tabanlı Nesne Takibi Üzerine İncelemeler ... 15
II.3.2. Geleneksel Yüz Tanıma Algoritmaları Üzerine İncelemeler ... 15
II.3.3. Mediapipe Tabanlı Görüntü İşleme Çalışmalarının İncelenmesi ... 16
II.3.4. Modüler Sistem Yaklaşımının Literatürdeki Yeri ... 16
II.3.5. Projenin Literatürdeki Boşlukları Doldurma Hedefi ... 17
**BÖLÜM III: TEZ ÇALIŞMALARI** ... 17
**III.1. ARAŞTIRMA ARAÇLARI** ... **17**
**III.2. YAPILAN ÇALIŞMALAR** ... **17**
III.2.1. Yüz Tanıma Sisteminin Geliştirilmesi ... 17
III.2.1.1. Veri Toplama (Collect Modülü) ... 17
III.2.1.2. Model Eğitimi (Train Modülü) ... 18
III.2.1.3. Yüz Tanıma ve Takip (Detect Modülü) ... 19
III.2.1.4. Servo ve Motor Kontrolünün Entegrasyonu ... 19
III.2.1.5. Çoklu İş Parçacığı (Thread) Yönetimi ve Donma Problemleri ... 19
III.2.2. Renk Bazlı Takip (Color Tracking Modülü) ... 20
III.2.3. Klasik Yüz Takibi (Tracking Modülü) ... 20
III.2.4. Sistemin Yeniden Eğitimi ve Reset Modülü ... 21
III.2.5. Derin Öğrenme Tabanlı YOLOv8 Algoritması ile Nesne Takibi ... 21
III.2.5.1. YOLO Nesne Takibi (Servo Modülü) ... 22
III.2.5.2. YOLO Full Detection (Sadece Görüntü Modülü) ... 22

**III.3. Proje Süreçleri** ..... **22**
**III.3.1. Yazılım Geliştirme Süreci** ..... 22
**III.3.1.1. Arduino Kod Açıklamaları** ..... 23
**III.3.1.2. Python Kod Açıklamaları: Collect.py** ..... 30
*Kod Bloğu 1:* ..... 30
*Kod Bloğu 2:* ..... 31
*Kod Bloğu 4:* ..... 31
*Kod Bloğu 5:* ..... 31
*Kod Bloğu 6:* ..... 32
*Kod Bloğu 8:* ..... 32
*Kod Bloğu 9:* ..... 32
*Kod Bloğu 10:* ..... 33
*Kod Bloğu 11:* ..... 33
*Kod Bloğu 12:* ..... 33
**III.3.1.3. Python Kod Açıklamaları: Train.py** ..... 35
*Kod Bloğu 1:* ..... 35
*Kod Bloğu 2:* ..... 35
*Kod Bloğu 3:* ..... 35
*Kod Bloğu 4:* ..... 35
*Kod Bloğu 5:* ..... 36
*Kod Bloğu 6:* ..... 38
*Kod Bloğu 7:* ..... 38
*Kod Bloğu 8:* ..... 38
*Kod Bloğu 9:* ..... 38
**III.3.1.4. Python Kod Açıklamaları: Detect.py** (Python.org, 2020) (OpenCV.org, 2020)..... 39
*Kod Bloğu 1:* ..... 39
*Kod Bloğu 2:* ..... 39
*Kod Bloğu 3:* ..... 39
*Kod Bloğu 4:* ..... 40
*Kod Bloğu 5:* ..... 40
*Kod Bloğu 6:* ..... 40
*Kod Bloğu 7:* ..... 40
*Kod Bloğu 8: (Arayüz için ayrı thread)* ..... 41
*Kod Bloğu 9: (Arayüz başlatma)* ..... 41
*Kod Bloğu 10:* ..... 41
*Kod Bloğu 11:* ..... 42
*Kod Bloğu 12: (Ana döngü)* ..... 42

*Kod Bloğu 13: (Geçiş Koruması)* ..... 42
*Kod Bloğu 14: (Yüz Algılama ve Tanıma)* ..... 43
*Kod Bloğu 15:* ..... 43
*Kod Bloğu 16:* ..... 43
*Kod Bloğu 17: (Kişi bulunamazsa)* ..... 44
*Kod Bloğu 18: (Araç Kontrolü)* ..... 44
*Kod Bloğu 19:* ..... 44
*Kod Bloğu 20:* ..... 44
**III.3.1.5. Python Kod Açıklamaları: reset.py** ..... 47
*Kod Bloğu 1: Gerekli Kütüphaneler ve Dizin Tanımı* ..... 47
*Kod Bloğu 2: Dataset ve Model Dosyalarının Temizlenmesi* ..... 47
**III.3.1.6. Python Kod Açıklamaları: Color_Tracking.py** (Python.org, 2020) ..... 48
*Kod Bloğu 1:* ..... 48
*Kod Bloğu 2:* ..... 49
*Kod Bloğu 3:* ..... 49
*Kod Bloğu 4:* ..... 49
*Kod Bloğu 5:* ..... 49
*Kod Bloğu 6: (Renk Seçimi Arayüzü)* ..... 50
*Kod Bloğu 7: (Arayüzün Oluşturulması)* ..... 50
*Kod Bloğu 8: (Arayüz Thread'i Başlatma)* ..... 51
*Kod Bloğu 9: (Kamera Başlatma)* ..... 51
*Kod Bloğu 10: (Ana Döngü)* ..... 51
*Kod Bloğu 11: (Renk Maskesi Oluşturma)* ..... 51
*Kod Bloğu 13: (Nesne Takibi ve Arduino'ya Veri Gönderme)* ..... 52
*Kod Bloğu 14: (Çizim ve Görüntüleme)* ..... 52
*Kod Bloğu 15: (Renk Bulunamazsa)* ..... 52
*Kod Bloğu 16:* ..... 53
*Kod Bloğu 17:* ..... 53
**III.3.1.7. Python Kod Açıklamaları: Tracking.py** ..... 56
*Kod Bloğu 1:* ..... 56
*Kod Bloğu 2:* ..... 56
*Kod Bloğu 3:* ..... 56
*Kod Bloğu 4:* ..... 57
*Kod Bloğu 5 (Ana Döngü Başlangıcı):* ..... 57
*Kod Bloğu 6 (Yüz Tespiti ve Koordinat Hesaplama):* ..... 57
*Kod Bloğu 7 (Görsel Çizimler):* ..... 58
*Kod Bloğu 8 (Ekranda Gösterim ve Çıkış Kontrolü):* ..... 58

*Kod Bloğu 9 (Kaynakları Serbest Bırakma):* ..... 58
**III.3.1.8. Python Kod Açıklamaları: yolo_object_tracking.py** (THU-MIG, 2024) (Ultralytics, 2023)..... 60
*Kod Bloğu 1:* ..... 60
*Kod Bloğu 2:* ..... 60
*Kod Bloğu 3:* ..... 60
*Kod Bloğu 4: GUI Arayüzü (Thread ile çalışıyor)* ..... 61
*Kod Bloğu 5: GUI Yapısı* ..... 61
*Kod Bloğu 6: Tüm sınıflara buton ekleme* ..... 61
*Kod Bloğu 7:* ..... 61
*Kod Bloğu 8: Arayüzü ayrı thread'de başlatma* ..... 62
*Kod Bloğu 9: Kamera ve Haar Cascade başlatma* ..... 62
*Kod Bloğu 10: Ana Döngü Başlıyor* ..... 62
*Kod Bloğu 11: Sonuçları İşleme* ..... 62
*Kod Bloğu 12: Sadece Seçili Nesne Takibi* ..... 63
*Kod Bloğu 13: Görüntü Üzerine Çizimler* ..... 63
*Kod Bloğu 14:* ..... 63
*Kod Bloğu 15: Görüntüyü Gösterme ve Çıkış Kontrolü* ..... 64
*Kod Bloğu 16: Kaynakları Serbest Bırakma* ..... 64
**III.3.1.9. Python Kod Açıklamaları: yolo_full_detection.py** (Ultralytics, 2023) (THU-MIG, 2024)..... 67
*Kod Bloğu 1:* ..... 67
*Kod Bloğu 2:* ..... 67
*Kod Bloğu 3: YOLOv8 Modelinin Yüklenmesi* ..... 67
*Kod Bloğu 4: Kamera Başlatma* ..... 67
*Kod Bloğu 5: Ana Döngü Başlıyor* ..... 68
*Kod Bloğu 6: Görüntüyü Aynalamak* ..... 68
*Kod Bloğu 7: YOLO Modeline Görüntü Gönderme* ..... 68
*Kod Bloğu 9: Görüntü Üzerine Çizim* ..... 69
*Kod Bloğu 10: Görüntüyü Gösterme* ..... 69
*Kod Bloğu 11: Kaynakların Serbest Bırakılması* ..... 69
**III.3.1.10. Python Kod Açıklamaları: main.py (Ana Kontrol Paneli)** (Python.org, 2020) ..... 71
*Kod Bloğu 1: Gerekli Kütüphaneler* ..... 71
*Kod Bloğu 2: Proje Dizini ve Global Değişken* ..... 71
*Kod Bloğu 3: Çalışan Modu Durdurma Fonksiyonu* ..... 71
*Kod Bloğu 4:* ..... 72
*Collect Modülünü Çalıştırma* ..... 72
*Reset Fonksiyonu:* ..... 74

*Kod Bloğu 6: Ana Arayüz Oluşturma* ..... 74
*Kod Bloğu 8: Sonsuz Döngü (Ana GUI Çalıştırma)* ..... 75
**III.3.2. Donanım Geliştirme Süreci** ..... 79
III.3.2.1.Kullanaılan Araçların Görselleri ..... 81
III.3.2.2.Devre Şeması ..... 82
III.3.3. Arayüz ..... 83
**BÖLÜM IV: SONUÇLAR VE DEĞERLENDİRME** ..... 88
**IV.1. Genel ..... 88**
IV.1.1. Sistem Başarım ve Stabilitesi ..... 88
IV.1.2.Yüz Tanıma Modülü ..... 88
IV.1.3.Renk Takibi Modülü ..... 88
IV.1.4. YOLOv8 ile Nesne Algılama ve Takip ..... 88
IV.1.5. Motor ve Araç Kontrol Entegrasyonu ..... 89
IV.1.6. Donanımın Tasarımı ve Taşınabilirliği ..... 89
IV.1.7. Sistem Geliştirilebilirlik Potansiyeli ..... 89
**IV.2.Sonuç ..... 89**
**Kaynakça ..... 90**

## 0.1. SEMBOLLER LİSTESİ

| Sembol | Açıklama | Birim |
|---|---|---|
| **X** | Görüntü üzerindeki X koordinatı | piksel |
| **Y** | Görüntü üzerindeki Y koordinatı | piksel |
| **cx** | Tespit edilen nesnenin merkez X koordinatı | piksel |
| **cy** | Tespit edilen nesnenin merkez Y koordinatı | piksel |
| **W** | İleri hareket komutu | **-** |
| **A** | Sol dönüş komutu | **-** |
| **S** | Geri hareket komutu | **-** |
| **D** | Sağ dönüş komutu | **-** |
| **PWM** | Motor hız sinyali | % |
| **θ (theta)** | Servo motor açısı | derece |

## 0.2. KISALTMALAR LİSTESİ

| Kısaltma | Açılımı |
|---|---|
| **GUI** | Graphical User Interface (Grafik Kullanıcı Arayüzü) |
| **FPS** | Frame Per Second (Saniyedeki Kare Sayısı) |
| **YOLO** | You Only Look Once |
| **HSV** | Hue-Saturation-Value (Renk Doygunluğu Modeli) |
| **GIL** | Global Interpreter Lock |
| **LBPH** | Local Binary Patterns Histograms |
| **USB** | Universal Serial Bus |
| **DC** | Direct Current (Doğru Akım) |
| **PWM** | Pulse Width Modulation (Darbe Genişlik Modülasyonu) |
| **UART** | Universal Asynchronous Receiver Transmitter (Seri Haberleşme Protokolü) |

# 0.3 ŞEKİLLER LİSTESİ

Şekil 1 ..... 28
Şekil 2 ..... 29
Şekil 3 ..... 30
Şekil 4 ..... 34
Şekil 5 ..... 37
Şekil 6 ..... 45
Şekil 7 ..... 46
Şekil 8 ..... 48
Şekil 9 ..... 54
Şekil 10 ..... 55
Şekil 11 ..... 59
Şekil 12 ..... 65
Şekil 13 ..... 66
Şekil 14 ..... 70
Şekil 15 ..... 77
Şekil 16 ..... 77
Şekil 17 ..... 78
Şekil 18 ..... 81
Şekil 19 ..... 81
Şekil 20 ..... 81
Şekil 21 ..... 81
Şekil 22 ..... 81
Şekil 23 ..... 81
Şekil 24 ..... 81
Şekil 25 ..... 82
Şekil 26 ..... 83
Şekil 27 ..... 83
Şekil 28 ..... 84
Şekil 29 ..... 85

# BÖLÜM I: GİRİŞ VE AMAÇ

Teknolojinin hızla gelişmesiyle birlikte görüntü işleme ve yapay zeka alanları hayatımızın birçok alanında kullanılmaya başlamıştır. Özellikle gerçek zamanlı nesne algılama sistemleri, günümüzde oldukça önemli bir araştırma konusu haline gelmiştir. Bu sistemler sayesinde kameradan alınan görüntüler işlenerek, ortamda bulunan nesneler anlık olarak tespit edilebilmekte ve takip edilebilmektedir. Böyle bir sistemin geliştirilmesi; hem akademik açıdan bilgi ve tecrübe kazandırmakta, hem de günlük yaşamda karşılaşılan birçok problemin çözümüne katkı sağlamaktadır.

Bu proje kapsamında, görüntü işleme tekniklerini kullanarak gerçek zamanlı nesne algılayabilen ve bu nesnelerin hareketlerini takip edebilen bir sistem geliştirilmesi hedeflenmiştir. Sistem, kamera aracılığıyla sürekli görüntü almakta ve bu görüntüleri anlık olarak işleyerek sahnedeki nesneleri tespit etmektedir. Algılanan nesnelerden elde edilen bilgiler, sistemin farklı uygulamalarda kullanılmasına olanak sağlamaktadır.

Projenin amacı yalnızca bir görüntü işleme sistemi kurmak değildir. Aynı zamanda ülkemizde farklı sektörlerde kullanılabilecek, yerli ve özgün bir sistemin temelini oluşturmak da amaçlanmıştır. Türkiye'de bu tür sistemler; savunma sanayi, trafik kontrol sistemleri, güvenlik kameraları, fabrika otomasyonları, tarım uygulamaları ve hatta sağlık sektöründe bile kullanılabilir. Bu açıdan bakıldığında, geliştirilecek sistemin kullanım alanları oldukça geniştir.

Projenin önemli bir diğer amacı da, düşük maliyetli ve taşınabilir bir sistem oluşturmaktır. Kullanılan donanım ve yazılımlar sayesinde sistemin kurulumu ve çalıştırılması kolay olacak, farklı senaryolara da uyarlanabilecektir. Özellikle öğrenciler, araştırmacılar ve girişimciler için erişilebilir bir altyapı sağlamayı da hedeflemektedir.

Sonuç olarak bu proje, gerçek zamanlı görüntü işleme ve nesne algılama alanında hem teorik bilgiyi pratiğe dönüştürmek hem de ülkemizde bu alanda yapılabilecek çalışmalara katkı sağlamak amacıyla hazırlanmıştır.

# BÖLÜM II: GENEL BİLGİLER

## II.1. GENEL BİLGİLER

Bu çalışmada görüntü işleme ve nesne algılama teknikleri kullanılarak gerçek zamanlı bir nesne tespit ve takip sistemi geliştirilmiştir. Projede hem yazılım hem de donanım bileşenleri bir arada kullanılmıştır. Sistem temel olarak kameradan alınan görüntülerin anlık olarak işlenmesini ve elde edilen verilerin donanım bileşenlerini kontrol etmek için kullanılmasını sağlamaktadır.

Görüntü işleme kısmında Python programlama dili kullanılmıştır. Python dili esnek yapısı ve zengin kütüphane desteği sayesinde görüntü işleme ve yapay zeka uygulamalarında sıkça tercih edilmektedir. Bu projede görüntülerin alınması ve işlenmesi için OpenCV kütüphanesi, vücut ve yüz hareketlerinin algılanması için MediaPipe kütüphanesi, grafik arayüz tasarımı için Tkinter kütüphanesi, nesne tespiti için ise YOLO (You Only Look Once) algoritması kullanılmıştır. Ayrıca işletim sistemi işlemleri ve dosya yönetimi için Python'un os modülünden de faydalanılmıştır.

Donanım tarafında ise, görüntü işleme sonucunda elde edilen bilgilerin fiziksel harekete dönüştürülmesi amacıyla Arduino mikrodenetleyici platformu kullanılmıştır. Arduino üzerinden servo motorlar ve DC motorlar kontrol edilmiştir. DC motorların kontrolünü sağlamak için L298N motor sürücü kartı tercih edilmiştir. Sistem enerji ihtiyacını karşılamak amacıyla lityum-iyon piller kullanılmış ve gerilim regülasyonu için delikli pertinaks plaka üzerine kurulu regülatör devresi tasarlanmıştır. Bu sayede sistem taşınabilir ve bağımsız bir şekilde çalışabilir hale getirilmiştir.

Geliştirilen bu sistem, düşük maliyetli ve modüler yapısıyla farklı uygulamalara kolayca uyarlanabilir. Hareketli nesnelerin takibi, güvenlik uygulamaları, robotik sistemler, otonom araç prototipleri ve endüstriyel otomasyon gibi birçok farklı alanda kullanılabilecek potansiyele sahiptir.

## II.2. TEORİK YAKLAŞIMLAR

Bu proje kapsamında geliştirmeyi planladığımız sistemde temel hedef, gerçek zamanlı görüntü işleme teknikleri kullanılarak hem yüz hem de farklı nesneleri algılayabilen ve bunları fiziksel hareketle takip edebilen esnek bir platform oluşturmaktır. Projeyi modüler ve genişletilebilir şekilde kurgulamak, sistemin farklı uygulama alanlarında kullanılabilirliğini artıracaktır.

### II.2.1. Yüz Tanıma Sistemi Tasarım Yaklaşımı

Sistemi güvenilir ve kişiye özgü çalışabilir hale getirmek için öncelikle yüz verilerinin toplanması gerekecektir. Bunun için, projede bir **veri toplama (Collect)** modülü oluşturmamız uygun olacaktır. Bu modülde kamera aracılığıyla farklı açılardan yüz görüntüleri alınmalı ve sistemin eğitimine hazır bir veri seti oluşturulmalıdır. Farklı ışık koşullarında ve farklı pozisyonlardan alınacak yaklaşık 100 görüntü, modelin çeşitlilik kazanmasını ve gerçek ortamda daha iyi performans göstermesini sağlayacaktır.

Toplanan verilerin sistemin eğitimi için işlenmesi gerekecektir. Bu amaçla LBPH (Local Binary Pattern Histogram) gibi klasik ama küçük veri setlerinde başarılı olan bir algoritma tercih edilebilir. Böylece **eğitim (Train)** modülü ile her bireyin yüz özelliklerini içeren bir model dosyası üretebiliriz. Bu eğitim işlemini gerçekleştirdikten sonra sistemimiz daha sonraki canlı yüz tanıma işlemlerinde bu modeli referans alarak tahmin yapabilecektir.

Asıl yüz tanıma işlemi **Detect** modülünde gerçekleşecektir. Burada kameradan alınan canlı görüntüler anlık işlenerek sistem daha önce eğitilmiş yüzleri tanıyabilecektir. Yüz algılandıktan sonra dikkat edilmesi gereken önemli bir husus, yüzün tam merkez noktasının hesaplanması olacaktır. Çünkü yüz yalnızca bir nokta değil, iki boyutlu genişliği ve yüksekliği olan bir nesnedir. Merkez noktanın tespitiyle servo motorları tam odak noktasına yönlendirebiliriz. Bu sayede servo sisteminin stabil ve estetik hareket etmesini sağlayabiliriz.

Çoklu yüz algılamada sistemin kararsız kalması olasıdır. Bu sorunu engellemek adına, kullanıcıya arayüz üzerinden kişi seçimi imkânı sunmayı planlıyoruz. Kullanıcı aktif hedef kişiyi belirlediğinde sistem sadece o yüzü takip edecektir. Bu yaklaşımla birden fazla yüz olduğunda dahi sistem stabil çalışabilecektir. Ayrıca kişi değişiminde geçici bir belirsizlik yaşanmaması için küçük bir gecikme (örneğin 100ms gibi) eklenirse geçişlerin daha sağlıklı olacağını öngörüyoruz.

Tüm bu işlemler sonucunda elde edilecek X ve Y koordinat verilerini, Arduino ile haberleşmek amacıyla seri port protokolüne uygun formatta paketlememiz gerekecektir. Arduino bu verileri aldığında, servo motorlara uygun açı pozisyonlarını uygulayacaktır.

### II.2.2. Renk Tabanlı Nesne Takip Yaklaşımı

Yüz dışında renk bazlı nesne takibi de sistemin önemli bir modülü olabilir. Renk seçimine dayalı esnek bir takip sistemi geliştirmek için kullanıcıya bir renk seçim arayüzü sunulabilir. Burada örneğin kırmızı, yeşil ve mavi renk seçenekleri bulunabilir.

Seçilen renge göre HSV (Hue-Saturation-Value) renk uzayında belirli eşik değerleri uygulanarak görüntü işleme gerçekleştirilecektir. Belirlenen renk maskeleriyle, görüntüdeki ilgili renkler izole edilebilir. Elde edilen renkli alanların merkezleri tespit edilerek yine koordinat bilgisi halinde Arduino'ya iletilebilir. Bu sayede kullanıcı kolayca istediği renk cismi takip ettirebilir. Böyle bir yaklaşım, sistemin uygulama esnekliğini artıracaktır.

### II.2.3. Temel Yüz Takibi (Tracking)

Sistemimizde bir diğer mod olarak herhangi bir yüzü kimlik doğrulaması yapmaksızın sadece tespit ve takip etmeye odaklanan **Tracking** modunu da geliştirmemiz uygun olacaktır. Burada kamera görüntüsünden yüz algılanacak, merkezi hesaplanacak ve servo sistemine aktarılacaktır.

Ayrıca bu modda sistemi biraz daha işlevsel hale getirmek için araç kontrolünü de entegre etmek faydalı olacaktır. Klavye üzerinden W-A-S-D tuşlarıyla hareket komutları alınabilir. Servo hareketleri ve araç hareket komutlarının aynı anda iletilebilmesi için verilerin aynı anda işlenmesini sağlayacak bir protokol ve zamanlama geliştirmemiz gerekecektir.

Çoklu veri gönderimi sırasında zamanlama problemleri yaşanmaması adına burada **thread tabanlı** bir yapı tasarlamak önemlidir. Servo ve motor komutları aynı anda ve senkronize bir şekilde Arduino'ya seri port üzerinden paket halinde gönderilebilir. Bu sayede servo hareketleri ve araç hareketleri birbirinden bağımsız ve sorunsuz çalışabilir.

### II.2.4. YOLO Tabanlı Nesne Algılama ve Takip Yaklaşımı

Projenin en ileri düzey modülünü oluşturacak olan **YOLO (You Only Look Once)** tabanlı sistem, daha genel nesne algılama görevleri için düşünülmüştür. YOLO'nun derin öğrenmeye dayalı yapısı sayesinde görüntüyü bütün olarak işleyerek aynı anda farklı nesneleri algılayabilmesi hedeflenmektedir.

Burada iki farklı yaklaşım planlayabiliriz:

- **YOLO Object Tracking:** Kullanıcı, mevcut nesne sınıflarından (örneğin şişe, telefon, insan vb.) birini arayüzden seçtiğinde sistem yalnızca o nesneyi takip eder ve servo sistemini hedefe kilitler.
- **YOLO Full Detection:** Tüm tespit edilen nesneler yalnızca görüntü üzerinde gösterilir, ancak servo hareketi yapılmaz. Bu mod tamamen analiz ve görsel test amacıyla kullanılabilir.

Bu yaklaşım, klasik renk ve yüz takip modlarına kıyasla çok daha geniş kullanım senaryoları oluşturabilir. YOLO sayesinde ortamda farklı nesneler anlık olarak tespit edilip etiketlenebilir.

### II.2.5. Mod Geçişlerinde Stabilite Sağlama Yaklaşımı

Çoklu mod yapısı düşünüldüğünde sistem kaynaklarının aşırı tüketimi, gecikmeler ve FPS düşüşleri kaçınılmaz olabilir. Bu sorunları önlemek amacıyla sistemin çalışma mantığını **"her zaman yalnızca tek aktif mod"** prensibi üzerine kurgulamamız daha stabil bir yapı sağlayacaktır.

Her buton yalnızca ilgili modülün çalışmasını başlatacak, diğer tüm modları otomatik olarak devre dışı bırakacaktır. Bu yaklaşım, kaynak yönetimini optimize edecek ve sistemin uzun süreli stabil çalışmasına katkı sağlayacaktır.

## II.3. KAYNAK BİLGİLERİN İRDELENMESİ

Projenin planlanma ve geliştirme aşamasında, özellikle görüntü işleme ve nesne takip sistemleri konusunda literatürdeki birçok güncel çalışma ve açık kaynak projeleri incelenmiştir. Bu bölümde, projede yararlanılması planlanan temel kaynaklar ve bu kaynakların proje tasarımındaki etkileri detaylı şekilde irdelenecektir.

### II.3.1. Derin Öğrenme Tabanlı Nesne Takibi Üzerine İncelemeler

Günümüzde gerçek zamanlı nesne algılama ve takip sistemleri denildiğinde en sık karşılaşılan ve başarıyla kullanılan modellerden biri YOLO (You Only Look Once) algoritmasıdır. YOLO'nun farklı versiyonları birçok araştırmacının ve endüstri profesyonelinin ilgisini çekmiştir. Özellikle:

- YOLOv8-DeepSORT-Object-Tracking (MuhammadMoinFaisal, GitHub) açık kaynak projesinde hem YOLOv8 algoritması hem de DeepSORT algoritması birleştirilerek çok güçlü bir nesne takip altyapısı geliştirilmiştir. (Faisal, 2023) Bu projeyi incelediğimizde, görüntüden hem nesne tespiti hem de sürekli kimlik takibi (ID Tracking) yapabilmenin sistemimize nasıl entegre edilebileceği hakkında fikir sahibi olduk.
- (THU-MIG, 2024). Bu algoritmaların ileri seviye GPU tabanlı donanımlar gerektirdiği, ancak biz proje kapsamında daha basit ve düşük sistem gereksinimiyle de çalışabilecek YOLOv8'in küçük modellerini (örneğin yolov8n.pt) tercih etmeyi planladık.

Bu kaynaklar yoluyla YOLO tabanlı nesne tespiti ve takip sistemlerinde kullanılan mimari yapıları ve parametre optimizasyonlarını anlamaya çalıştık. Biz kendi projemizde YOLOv8'in temel nesne tespit yeteneklerinden yararlanarak modlar tasarlamayı planladık.

### II.3.2. Geleneksel Yüz Tanıma Algoritmaları Üzerine İncelemeler

Derin öğrenme tabanlı yüz tanıma sistemleri oldukça güçlü çözümler sunsa da, projede donanım kaynaklarının kısıtlı olabileceğini göz önünde bulundurarak daha basit ve optimize edilebilir geleneksel algoritmalar üzerine de araştırmalar yapılmıştır.

- Ageitgey Face Recognition (GitHub) projesi, derin öğrenme tabanlı dlib ve CNN tabanlı modellerle yüksek doğrulukta yüz tanıma işlemleri gerçekleştiren açık kaynaklı bir projedir. (Geitgey, 2017) Bu sistemin yüksek doğruluğuna rağmen donanım ihtiyacının daha fazla olması nedeniyle projemize doğrudan entegre edilmesinin **uygun olmayacağı** sonucuna vardık. Ancak çalışma mantığı, yüzün encoding (özellik vektörü) yöntemiyle temsil edilmesi fikrini anlamamıza katkı sağladı.

- Buna ek olarak, biz kendi sistemimizde LBPH (Local Binary Patterns Histogram) yöntemini tercih etmeyi düşündük. Çünkü:
    - Eğitim süresi hızlıdır.
    - Küçük veri setlerinde etkilidir.
    - Düşük işlem gücüyle çalışabilir.

Bu yaklaşımla yüz tanıma sistemimizi üç temel faza ayırmayı planlıyoruz: veri toplama (collect), model eğitimi (train) ve tanıma (detect). Bu mimari yaklaşım, literatürde sıkça karşılaşılan klasik yüz tanıma mimarisine uygun olacaktır.

### II.3.3. Mediapipe Tabanlı Görüntü İşleme Çalışmalarının İncelenmesi

Ayrıca Google tarafından geliştirilen Mediapipe kütüphanesinin farklı uygulamalarını da inceledik. Mediapipe ile vücut, el, yüz ve poz algılama konusunda yapılan birçok proje üzerinden çeşitli senaryolar gözlemlendi. Örneğin:

- Mediapipe Face Mesh projelerinde, yalnızca yüz değil, yüzün 468 farklı noktası üzerinden detaylı konum bilgisinin hesaplanabildiğini gördük.
- Bazı kaynak projelerde (örneğin: "Mediapipe Hand Tracking with Arduino" projeleri) Mediapipe kütüphanesi ile elde edilen el ve yüz koordinatlarının doğrudan Arduino'ya seri port üzerinden iletilerek servo motorlarla hareketli sistemlerin kontrol edilebildiğini gözlemledik.

Biz Mediapipe'i doğrudan projemize entegre etmek yerine, görüntü işleme mantığını basitleştirmek ve sistem yükünü azaltmak amacıyla yalnızca Haar Cascade gibi geleneksel yöntemlerle temel yüz tespiti yapmayı uygun gördük. Böylelikle sistemin FPS kaybı yaşamadan daha stabil çalışabileceğini öngördük.

### II.3.4. Modüler Sistem Yaklaşımının Literatürdeki Yeri

Literatürde çoğu proje ya yalnızca yüz takibi, ya sadece renk takibi, ya da sadece nesne tespiti üzerine yoğunlaşmaktadır. Bizim projemizde ise;

- Yüz Takibi (Collect - Train - Detect),
- Klasik Yüz Tracking (servo ile),
- Renk Bazlı Takip (Color Tracking),
- YOLO Nesne Takibi ve
- YOLO Nesne Tespiti (servo kontrolü olmadan)

gibi çok sayıda farklı mod aynı sistem içinde tasarlanmak istenmektedir.

Bu tür çoklu mod sistemlerinde kaynak paylaşımı (kamera, işlemci, GUI yönetimi, seri port yönetimi) literatürde yaygın bir problem olarak tanımlanmıştır. Bu nedenle projede "her mod aktif çalıştığında diğer modların kapatıldığı" bir sistem mimarisi tercih edilmesinin daha stabil ve uygulanabilir bir çözüm sağlayacağı sonucuna vardık.

### II.3.5. Projenin Literatürdeki Boşlukları Doldurma Hedefi

İncelenen çalışmalar genel olarak güçlü ama tek modlu sistemler tasarlamışken, biz daha modüler, düşük maliyetli ve kolay uygulanabilir çok modlu bir demo platformu tasarlamayı hedefliyoruz. Özellikle eğitim ortamları, laboratuvar denemeleri, öğrenci projeleri ve prototipleme çalışmalarında kullanılabilir bir sistem ortaya çıkarmak projenin temel amacını oluşturmaktadır.

# BÖLÜM III: TEZ ÇALIŞMALARI

## III.1. ARAŞTIRMA ARAÇLARI

Bu tez çalışmasında kullanılan yazılım ve donanım araçları aşağıda listelenmiştir:

- **Python Programlama Dili:** Yazılım geliştirme sürecinde ana programlama dili olarak kullanılmıştır.
- **OpenCV Kütüphanesi:** Görüntü işleme işlemlerinde kullanılmıştır.
- **MediaPipe Kütüphanesi:** İnsan vücudu ve yüz tespitinde kullanılmıştır.
- **YOLO Algoritması:** Nesne algılama amacıyla uygulanmıştır.
- **Tkinter:** Kullanıcı arayüzü tasarımı için tercih edilmiştir.
- **Arduino Uno:** Donanım kontrolü ve motor sürüş işlemleri için kullanılmıştır.
- **L298N Motor Sürücü Kartı:** DC motorların kontrolünü sağlamak için kullanılmıştır.
- **HC-05 Bluetooth Modülü:** Kablosuz haberleşme için kullanılmıştır.
- **Delikli Pertinaks ve Regülatör Devresi:** Sistemin enerji yönetimi için kullanılmıştır.
- **Lityum İyon Piller:** Sistemin taşınabilirliğini sağlamak için güç kaynağı olarak kullanılmıştır.

## III.2. YAPILAN ÇALIŞMALAR

### III.2.1. Yüz Tanıma Sisteminin Geliştirilmesi

Bu proje kapsamında yüz tanıma işlemi, hazır bulut tabanlı veya derin öğrenme temelli sistemler yerine, tamamen kendi eğitim verimizi toplayarak ve kendi modelimizi oluşturarak gerçekleştirilmiştir. Böylece düşük işlem gücü gerektiren, basit ve bağımsız çalışabilen bir sistem tasarlanması hedeflenmiştir. Ayrıca hazır eğitilmiş model olan YOLO da projemize dahil edilmiştir.

#### III.2.1.1. Veri Toplama (Collect Modülü)

İlk aşamada yüz verisinin oluşturulması gerektiği için sistemde *Collect* adı verilen bir modül planlandı. Kullanıcı arayüzünden yeni kullanıcı eklemek için isim girilerek kişiye özel klasörler açıldı. Yaklaşık 100'er adet yüz görüntüsü alınarak veri seti oluşturuldu. Bu veriş etleri dataset klasöründe depolandı.

**Karşılaşılan sorunlar ve çözüm yaklaşımları:**

- **Tekrarlı veri sorunu:** Sürekli aynı açılardan alınan görüntüler, eğitim setinde çeşitlilik oluşturmayabiliyordu. Bu yüzden veri toplarken kişiye çeşitli yüz ifadeleri ve açılardan (sağa bakma, sola bakma, yukarı-aşağı bakma vb.) pozisyonlar verdirdik. Böylelikle farklı varyasyonlar sağladık.
- **Veri kalitesi:** Bazen ışık yetersizliği veya odaklanma hatası nedeniyle bulanık veriler oluşuyordu. Bu görüntülerden kurtulmak için yüz tespit edilen kareleri 200x200 boyutuna normalize ettik ve görüntü ön işleme adımları ekledik.
- **Yanlış Konum:** Dataset klassöründen alınan veriler eğitilmek için kullanıldığında yanlış konuma kaydeedeilebiliyordu bunu önlemek amacıyla konumu tam olarak "project_folder" dosyasının uzantısını tam olarak belirttik.

### III.2.1.2. Model Eğitimi (Train Modülü)

Toplanan veriler üzerinde eğitim yapmak için **OpenCV'nin LBPHFaceRecognizer** sınıfını kullandık. LBPH algoritması, düşük sistem kaynaklarıyla çalışabilen ve sınırlı veri setlerinde dahi iyi sonuçlar verebilen bir yöntemdir.

**Karşılaşılan sorunlar ve çözüm yaklaşımları:**

- **Overfitting (Aşırı Ezberleme):** Çok benzer görüntüler modelin ezberleme riskini artırıyordu. Bu sorunu çeşitlilik sağlamak için veri toplama aşamasında farklı pozisyonlardan veri almakla dengeledik.
- **Hatalı etiket sorunu:** Eğitim sırasında bazı klasör isimlerinin yanlış yazılması (boşluk, Türkçe karakter vb.) etiket karmaşası oluşturuyordu. İsim girişlerini arayüz üzerinden kontrollü alarak bu sorunun önüne geçtik.
- **Yanlış Konum:** Dataset klassöründen alınan veriler eğitilmek için kullanıldığında yanlış konuma kaydeedeilebiliyordu bunu önlemek amacıyla konumu tam olarak "project_folder" dosyasının uzantısını tam olarak belirttik.

### III.2.1.3. Yüz Tanıma ve Takip (Detect Modülü)

Detect modülü çalıştırıldığında, kameradan alınan her karede öncelikle **Haar Cascade** ile yüzler tespit edildi. Daha sonra tespit edilen yüzler üzerinde LBPH modeli çalıştırılarak kime ait olduğu tahmin edildi.

**Karşılaşılan sorunlar ve çözüm yaklaşımları:**

- **Birden fazla yüz problemi:** Aynı karede birden fazla yüz görünüyorsa servo kararsız çalışıyordu. Kullanıcının arayüzden aktif kişiyi seçmesi sağlandı. Böylece sistem sadece seçilen kişiyi tanıyıp servoya veri gönderdi. Diğer kişilerin bilgileri seçinceye kadar hesaplanmadı ve sisteme binen yük azaltıldı.
- **Geçiş çakışmaları:** Kullanıcı kişiyi değiştirdiğinde, anlık olarak her iki yüz aynı karede bulunabiliyordu. Bu çakışmada servo kararsız hareket ediyordu. Bunu çözmek için seçim değişiminde 100 ms'lik bir geçici yüz tespiti duraklaması (delay) ekledik. Böylece sistem çakışmadan yeni kişiyi tanımaya başladı.
- **Sistem Over Load:** Aynı karade birden fazla kişi olduğu zaman sistem her kişinin yüz verisini hesaplayıp sadece seçili kişinin yüz versini servoya gönderiyordu bu da fazla kişi olduğunda sistemde fps düşüşü yaşanmasına neden olmaktaydı bunu sadece seçili kişinin yüz veririsi hesaplanmasını sağlayarak çözmüş olduk.

### III.2.1.4. Servo ve Motor Kontrolünün Entegrasyonu

Yüz tanıma sistemi servo motorlara bağlanarak yatay ve dikey yönlendirme sağlandı. Ancak projenin ilerleyen aşamasında servo takibine ek olarak araç hareketleri de yapılmak istendi.

- Klavyeden **W, A, S, D** tuşlarıyla hareket komutları alınarak DC motorlar kontrol edildi. Böylece servo ve motor sistemleri birbirinden bağımsız çalıştı.
- Servo koordinatları ve motor komutları **tek bir veri paketi** içinde Arduino'ya gönderildi. (örn: 320,240,W)

### III.2.1.5. Çoklu İş Parçacığı (Thread) Yönetimi ve Donma Problemleri

İlk denemelerde tüm sistem aynı thread (iş parçacığı) üzerinde çalıştırıldığında donma ve FPS kaybı yaşandı. Görüntü işleme, seri haberleşme ve arayüz yoğun işlem yükü oluşturuyordu.

- **Multithreading yaklaşımıyla**: Arayüz ana thread üzerinde çalışırken, kamera görüntüsü ve işleme ayrı thread'e ayrıldı. Ancak yine de OpenCV ve Tkinter GUI birlikte çalışırken GIL (Global Interpreter Lock) kaynaklı zamanlama hataları oluşabiliyordu.
- **PyQt5 yerine Tkinter:** PyQt5 ile ilk başta denemeler yapılmış ancak performans ve uyum sorunları nedeniyle vazgeçilmiş, yerine Python'un iç yapısıyla daha iyi uyum sağlayan Tkinter arayüz kütüphanesi tercih edilmiştir. Böylelikle sistem çok daha stabil ve akıcı hale getirilmiştir.

### III.2.2. Renk Bazlı Takip (Color Tracking Modülü)

Renk takibinde temel amaç belirlenen renk aralıkları içerisindeki objenin konumunu bulup servoya iletmekti. Kullanıcı arayüzünde **Kırmızı, Yeşil, Mavi** butonlarıyla renk seçimi yapıldı.

- HSV renk uzayında belirlenen aralıklarla renk maskesi oluşturuldu.
- Maskeden çıkan konturların merkez koordinatları hesaplanıp Arduino'ya gönderildi.
- Anlık renk değişimi yapabilmek için Tkinter arayüzü ile dinamik renk seçimi sağlandı.

**Karşılaşılan sorunlar ve çözüm yaklaşımları:**

- **Işık etkisiyle renk sapması:** Işık koşulları değiştikçe renk algılaması sapıyordu. Bunu minimize etmek için renk maskelerinde geniş aralıklar belirlenerek tolerans artırıldı.
- **Küçük gürültü noktaları:** Ufak renk lekeleri büyük alanmış gibi algılanıyordu. Bunu önlemek için kontur alanına (area > 500 px) filtreleme uyguladık.

### III.2.3. Klasik Yüz Takibi (Tracking Modülü)

Tracking modunda sistem, eğitimli model kullanmadan yalnızca gerçek zamanlı olarak yüz tespiti yaparak yüzün merkez koordinatını bulur ve servo motorları bu koordinatlara göre hareket ettirir. Bu modda herhangi bir kişiye özel tanıma yapılmamaktadır. Sistemin amacı sadece algılanan yüz merkezine kitlenmektir.

**Karşılaşılan sorunlar ve çözüm yaklaşımları:**

- **Kamera Merkezleme Problemi:** Yüz algılandıktan sonra koordinatları servoya gönderiliyordu; ancak yüz bir dikdörtgen alan değil, genişlik ve yüksekliği olan bir objedir. Eğer kenar koordinatları servo referansı yapılırsa, servo tam merkeze kilitlenemiyor ve salınım yapıyordu. Bu nedenle her yüz algılamasında dikdörtgenin merkez noktası (x + w/2, y + h/2) hesaplanarak servo kontrolü yapıldı ve yüzün tam ortasına kitlenmesi sağlandı.
- **Anlık Kayıplar ve Titreme:** Yüz tespiti sırasında bazen ani görüntü kayıpları veya yanlış pozisyon hesaplamaları oluyordu. Özellikle hızlı hareketlerde servo sürekli pozisyon değiştirerek salınıma giriyordu. Bu sorunu azaltmak için kontur kayıplarında servo pozisyonunu koruma (en son bilinen konumu tutma) mantığı geliştirildi.
- **Veri İletiminde Donma Problemi:** Servo hareket bilgileri gönderilirken aynı anda araç hareket komutları da gönderilmekteydi. İki farklı veri paketinin sürekli ardışık gönderimi sırasında buffer sorunları oluşabiliyordu. Bunun için servo pozisyonları ve araç komutları **tek paket** halinde (örn: 320,240,W) gönderildi. Böylece veri bütünlüğü sağlandı.
- **Servo ve Araç Kontrolünün Bağımsızlığı:** Servo yüz pozisyonuna göre hareket ederken araç kontrolleri de kullanıcı tarafından klavyeden (W, A, S, D) bağımsız şekilde yapılmak istenmişti. Bu nedenle araç komutları, arayüzden alınan ekstra tuş kontrolüyle servo komutlarının yanına eklendi. Böylece servo takibi ve araç sürüşü eş zamanlı olarak gerçekleşti.

- **Servo Kuyruğa Veri Birikmesi Problemi:** Kamera görüntüsü sürekli yenilendiği için saniyede çok sayıda yeni koordinat servoya gönderiliyordu. Arduino tarafında ise seri porttan gelen bu veriler işlenmeden bekleyen veri yığılmaları oluşuyordu. Bu durum servoların, o anda değil, birkaç saniye önce gelen eski koordinatlara göre hareket etmesine neden oluyordu. Servonun gecikmeli hareket etmesi sistemin stabilitesini bozuyordu. Bu sorunu çözebilmek için Arduino yazılımında sadece en son alınan veri işlenmeye başlandı. Seri portta birden fazla veri biriktiğinde, önce tüm tampon (buffer) temizleniyor ve sadece en güncel komut işleme alınıyordu. Böylece servo motor her zaman o anki görüntünün gerçek zamanlı pozisyonuna göre hareket ediyordu. Kuyruk sorunu tamamen ortadan kaldırılmış oldu.

### III.2.4. Sistemin Yeniden Eğitimi ve Reset Modülü

Projenin en önemli özelliklerinden biri sistemin sadece bir defaya mahsus değil, istenildiği kadar geliştirilebilir ve güncellenebilir olmasıdır. Yüz tanıma sisteminde eğitimli modelin başarımı, kullanılan verilerin kalitesine doğrudan bağlıdır. Kullanıcıların yüz pozları, ışık şartları, saç-sakal değişimi gibi etkenler zamanla değişebileceği için eğitim verilerini periyodik olarak güncellemek gerekebilir.

Bu noktada **Reset Modülü** devreye girmektedir.

**Reset Modülü'nün Amacı ve Gerekliliği:**

- **Eski ve yetersiz veri setlerinden kurtulma:** Zamanla düşük kaliteli veya yeterince çeşitli olmayan veri setleri modelin doğruluğunu düşürebilir. Reset modülü ile tüm eski veriler ve model dosyaları kolayca silinip sistem sıfırdan veri toplamaya ve eğitime hazırlanabilir.
- **Sistemi sıfırdan yeniden kurabilme esnekliği:** Projenin ilerleyen aşamalarında yeni kullanıcılar ekleneceğinde veya algoritmalar güncelleneceğinde sistemin tamamen temizlenerek yeniden eğitim sürecine başlanabilmesi gereklidir.
- **Uzun vadeli bakım ve sürdürülebilirlik:** Proje ilerledikçe sistemin modüler ve geliştirilebilir yapısını koruyabilmek için kullanıcıya sistemin her an yeniden başlatılabilme esnekliği verilmiştir.
- **Test ve Deneme Kolaylığı:** Geliştirme aşamasında sistem farklı senaryolarda sık sık denenmiştir. Test sırasında oluşabilecek kirli verileri temizlemek için reset seçeneği büyük kolaylık sağlamıştır.

**Reset Modülü Nasıl Çalışır?**

- Dataset klasöründeki tüm yüz veri klasörlerini siler.
- Trainer.yml ve labels.txt dosyaları gibi model ve etiketleme dosyalarını temizler.
- Sistem yeniden Collect, Train ve Detect aşamalarıyla yeni eğitim sürecine hazır hale gelir.

### III.2.5. Derin Öğrenme Tabanlı YOLOv8 Algoritması ile Nesne Takibi

Sisteme ek olarak ileri seviye nesne tespiti için YOLOv8 (You Only Look Once v8) modeli eklendi. **Ultralytics kütüphanesi** kullanılarak eğitimli modelden faydalanıldı.

#### III.2.5.1. YOLO Nesne Takibi (Servo Modülü)

- Kullanıcı arayüzünde YOLO sınıfları arasında seçim yapıldı.
- Seçilen nesne algılandığında yalnızca bu nesneye servo kitlenerek koordinatları Arduino'ya gönderildi.
- Aynı anda diğer nesneler algılansa bile sistem yalnızca seçili sınıfı dikkate aldı.

**Karşılaşılan sorunlar ve çözüm yaklaşımları:**

- **Modelin sistem kaynağı kullanımı:** YOLOv8 büyük modellerde donanım yükü oluşturuyordu. Bu yüzden YOLOv8n (nano) küçük versiyonu tercih edilerek FPS kaybı minimize edildi.
- **Anlık sınıf değişimi:** Kullanıcı istediği an başka nesne sınıfını seçebiliyor ve servo hemen yeni hedefe yöneliyordu.

#### III.2.5.2. YOLO Full Detection (Sadece Görüntü Modülü)

- Bu modda servo kontrolü yapılmadan sadece tüm algılanan nesneler ekran üzerinde işaretlenip isimleri gösterildi. Böylece sistemin çoklu nesne algılama yeteneği test edildi.

## III.3. Proje Süreçleri

Bu bölümde proje yapım ve geliştirme sürecine ait kodları ve donanımı anlattık.

### III.3.1. Yazılım Geliştirme Süreci

Tez çalışmasına başlarken öncelikle **Python programlama dili** ve **OpenCV kütüphanesi** ile görüntü işleme mantığı öğrenilmiştir. Görüntülerin nasıl işleneceği, piksel seviyesinde analizlerin nasıl yapılacağı ve gerçek zamanlı görüntüden nasıl veri elde edileceği üzerinde çalışmalar yapılmıştır.

Bu temel bilgiler öğrenildikten sonra sistemin yazılım kısmı için kodlama çalışmalarına başlanmıştır. İlk denemelerde projeyi **ESP32** mikrodenetleyici platformu ile gerçekleştirmek planlanmıştır. Ancak, sistem gerçek zamanlı görüntü işleme ve nesne algılama işlemlerinde yüksek işlem gücü gerektirdiği için ESP32'nin işlem kapasitesi yetersiz kalmış ve sistem sürekli donmalar yaşamıştır. Bu sebeple, ESP32 ile yapılan denemeler sonlandırılarak farklı bir çözüm yoluna gidilmiştir.

Sonrasında proje, **Arduino platformu** ve **Bluetooth modülü (HC-05)** üzerinden yeniden tasarlanmıştır. Kablosuz haberleşmenin yanı sıra, kablolu (modülsüz) çalışma için de sistemin yazılımı uygun şekilde geliştirilmiştir.

Öncelikle Python'da işlediğimiz veriyi Arduino'da kontrol edebilmemiz için Arduino kodunu hazırladık.

### III.3.1.1. Arduino Kod Açıklamaları

---

*Kod Bloğu 1:*

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

**Açıklama:**
Arduino'nun standart servo motor kütüphanesi projeye dahil edilmiştir. Bu kütüphane sayesinde servo motorların konum kontrolü kolayca yapılabilir.

---

*Kod Bloğu 2:*

```
Servo servoX;
Servo servoY;
```

**Açıklama:**
İki adet servo motor için `Servo` sınıfından nesneler oluşturulmuştur. `servoX` yatay eksende (X), `servoY` ise düşey eksende (Y) hareket sağlayacaktır.

---

*Kod Bloğu 3:*

```
String inputString = "";
```

**Açıklama:**
Seri port üzerinden alınan verilerin geçici olarak saklanacağı karakter dizisi tanımlanmıştır. Bu değişken, gelen verinin tam paket olarak alınmasını sağlar.

---

*Kod Bloğu 4:*

```
const int frameWidth = 640;
const int frameHeight = 480;
```

**Açıklama:**
Kamera görüntüsünün çözünürlüğü tanımlanmıştır. Gelen X ve Y koordinatları bu boyutlara göre ölçeklendirilecektir. Bu değerler görüntü işleme tarafındaki yazılımla senkron çalışır.

---

*Kod Bloğu 5:*

```
const int IN1 = 2;
const int IN2 = 3;
const int IN3 = 4;
const int IN4 = 5;
const int ENA = 6;
const int ENB = 7;
```

**Açıklama:**
DC motorları kontrol eden L298N motor sürücü kartına bağlı Arduino pinleri tanımlanmıştır. IN1-IN4 yön kontrolünü sağlarken, ENA ve ENB motorların hızını (PWM ile) kontrol eder.

*Kod Bloğu 6:*

```
void setup() {
  Serial.begin(9600);
  servoX.attach(9);
  servoY.attach(10);
```

**Açıklama:**
Setup fonksiyonunun başlangıcıdır. Arduino ile bilgisayar arasında seri haberleşme başlatılmıştır (9600 baudrate). Servo motorlar dijital 9 ve 10 numaralı pinlere bağlanarak kontrol için hazır hale getirilmiştir.

---

*Kod Bloğu 7:*

```
  servoX.write(90);
  servoY.write(90);
```

**Açıklama:**
Her iki servo motor başlangıçta merkez pozisyon olan 90 dereceye konumlandırılır. Böylece sistem başlarken her zaman nötr pozisyonda başlar.

---

*Kod Bloğu 8:*

```
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
```

**Açıklama:**
Motor sürücü kartına bağlı pinlerin hepsi çıkış (OUTPUT) olarak ayarlanır. Böylece Arduino bu pinler üzerinden motorları sürebilecektir.

---

*Kod Bloğu 9:*

```
  Serial.println("Servo ve motor sistemi başlatıldı.");
}
```

**Açıklama:**
Sistem başlatıldığında seri monitöre bilgi amaçlı mesaj gönderilir. Bu, sistemin sağlıklı başladığını kontrol etmek için kullanılır.

---

*Kod Bloğu 10:*

```
void loop() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    if (inChar == '\n') {
      processInput(inputString);
```

```
      inputString = "";
    } else {
      inputString += inChar;
    }
  }
}
```

**Açıklama:**
Ana döngü sürekli olarak seri portu kontrol eder. Gelen veriler `inputString` değişkeninde biriktirilir. Gelen veri tamamlandığında (yeni satır karakteri `\n` alındığında), `processInput()` fonksiyonu çağrılarak işleme alınır.

---

*Kod Bloğu 11:*

```
void processInput(String data) {
  data.trim();
  int firstComma = data.indexOf(',');
  int secondComma = data.indexOf(',', firstComma + 1);
```

**Açıklama:**
Veri işleme fonksiyonunun başlangıcıdır. Gelen verideki boşluklar temizlenir. Ardından veri içerisindeki ilk ve ikinci virgülün (`,`) konumları tespit edilir. Bu sayede veri paketindeki X, Y ve motor komutu ayrıştırılacaktır.

---

*Kod Bloğu 12:*

```
  if (firstComma > 0 && secondComma > firstComma) {
    int posX = data.substring(0, firstComma).toInt();
    int posY = data.substring(firstComma + 1, secondComma).toInt();
    char motorCommand = data.charAt(secondComma + 1);
```

**Açıklama:**
Gelen veri doğru formatta ise X ve Y pozisyonları ile motor komutu alınır. `substring()` fonksiyonu ile X ve Y değerleri ayrıştırılır ve tam sayıya (`int`) çevrilir. Motor komutu karakter olarak elde edilir.

---

*Kod Bloğu 13:*

```
    int servoPosX = map(posX, 0, frameWidth, 0, 180);
    int servoPosY = map(posY, 0, frameHeight, 0, 180);
    servoX.write(servoPosX);
    servoY.write(servoPosY);
```

**Açıklama:**
Kamera çözünürlüğü ile servo motorların hareket sınırları farklıdır. Bu yüzden gelen pozisyon değerleri `map()` fonksiyonu ile 0-180 derece aralığına dönüştürülür ve servo motorlara uygulanır.

---

*Kod Bloğu 14:*

```
    Serial.print("ServoX: ");
    Serial.print(servoPosX);
    Serial.print(" | ServoY: ");
    Serial.print(servoPosY);
    Serial.print(" | Motor: ");
    Serial.println(motorCommand);
```

**Açıklama:**
İzleme ve hata ayıklama (debug) amacıyla servo açıları ve motor komutu seri monitöre yazdırılır.

---

*Kod Bloğu 15:*

```
    controlMotor(motorCommand);
  }
}
```

**Açıklama:**
Motor komutunu alıp ilgili fonksiyona (`controlMotor`) gönderir. Bu sayede gelen komuta göre DC motor hareket ettirilir.

---

*Kod Bloğu 16:*

```
void controlMotor(char cmd) {
  if (cmd == 'W') {
    // İleri
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    analogWrite(ENA, 180);
    analogWrite(ENB, 180);
  }
```

**Açıklama:**
`W` komutu alındığında her iki motor ileri hareket edecek şekilde sürülür. Yön pinleri ayarlanır ve PWM sinyali 180 seviyesinde uygulanır.

---

*Kod Bloğu 17:*

```
  else if (cmd == 'S') {
    // Geri
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    analogWrite(ENA, 180);
    analogWrite(ENB, 180);
  }
```

**Açıklama:**
`S` komutu ile geri hareket sağlanır. Yön pinleri ters çevrilir.

---

*Kod Bloğu 18:*

```
  else if (cmd == 'A') {
    // Sol
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    analogWrite(ENA, 180);
    analogWrite(ENB, 180);
  }
```

**Açıklama:**
`A` komutunda sola dönüş yapılır. Motorlardan biri ileri diğeri geri çalıştırılarak dönüş sağlanır.

---

*Kod Bloğu 19:*

```
  else if (cmd == 'D') {
    // Sağ
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    analogWrite(ENA, 180);
    analogWrite(ENB, 180);
  }
```

**Açıklama:**
`D` komutunda sağa dönüş sağlanır.

---

*Kod Bloğu 20:*

```
  else {
    // Dur
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
  }
}
```

**Açıklama:**
Hiçbir geçerli komut gelmediğinde motorlar durdurulur. Tüm yön pinleri sıfırlanır.

arduino_code.ino

```ino
#include <Servo.h>

Servo servoX;
Servo servoY;

String inputString = "";

const int frameWidth = 640;
const int frameHeight = 480;

// Motor pinleri
const int IN1 = 2;
const int IN2 = 3;
const int IN3 = 4;
const int IN4 = 5;
const int ENA = 6;
const int ENB = 7;

void setup() {
  Serial.begin(9600);
  servoX.attach(9);
  servoY.attach(10);

  // Servolar başlangıçta ortaya
  servoX.write(90);
  servoY.write(90);

  // Motor pin çıkış
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);

  Serial.println("Servo ve motor sistemi başlatıldı.");
}

void loop() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    if (inChar == '\n') {
      processInput(inputString);
      inputString = "";
    } else {
      inputString += inChar;
    }
  }
}
```

*Şekil 1*

```
50
51  void processInput(String data) {
52    data.trim();
53    int firstComma = data.indexOf(',');
54    int secondComma = data.indexOf(',', firstComma + 1);
55
56    if (firstComma > 0 && secondComma > firstComma) {
57      int posX = data.substring(0, firstComma).toInt();
58      int posY = data.substring(firstComma + 1, secondComma).toInt();
59      char motorCommand = data.charAt(secondComma + 1);
60
61      int servoPosX = map(posX, 0, frameWidth, 0, 180);
62      int servoPosY = map(posY, 0, frameHeight, 0, 180);
63      servoX.write(servoPosX);
64      servoY.write(servoPosY);
65
66      Serial.print("ServoX: ");
67      Serial.print(servoPosX);
68      Serial.print(" | ServoY: ");
69      Serial.print(servoPosY);
70      Serial.print(" | Motor: ");
71      Serial.println(motorCommand);
72
73      controlMotor(motorCommand);
74    }
75  }
76
77  void controlMotor(char cmd) {
78    if (cmd == 'W') {
79      // İleri
80      digitalWrite(IN1, HIGH);
81      digitalWrite(IN2, LOW);
82      digitalWrite(IN3, HIGH);
83      digitalWrite(IN4, LOW);
84      analogWrite(ENA, 180);
85      analogWrite(ENB, 180);
86    } else if (cmd == 'S') {
87      // Geri
88      digitalWrite(IN1, LOW);
89      digitalWrite(IN2, HIGH);
90      digitalWrite(IN3, LOW);
91      digitalWrite(IN4, HIGH);
92      analogWrite(ENA, 180);
93      analogWrite(ENB, 180);
```

*Şekil 2*

```
94    } else if (cmd == 'A') {
95      // Sol
96      digitalWrite(IN1, LOW);
97      digitalWrite(IN2, HIGH);
98      digitalWrite(IN3, HIGH);
99      digitalWrite(IN4, LOW);
100     analogWrite(ENA, 180);
101     analogWrite(ENB, 180);
102   } else if (cmd == 'D') {
103     // Sağ
104     digitalWrite(IN1, HIGH);
105     digitalWrite(IN2, LOW);
106     digitalWrite(IN3, LOW);
107     digitalWrite(IN4, HIGH);
108     analogWrite(ENA, 180);
109     analogWrite(ENB, 180);
110   } else {
111     // Dur
112     digitalWrite(IN1, LOW);
113     digitalWrite(IN2, LOW);
114     digitalWrite(IN3, LOW);
115     digitalWrite(IN4, LOW);
116   }
117 }
118
```

*Şekil 3*

---

### III.3.1.2. Python Kod Açıklamaları: Collect.py

*Kod Bloğu 1:*

```
import cv2
import os
import sys
```

**Açıklama:**
Proje için gerekli kütüphaneler eklenmiştir.

- **cv2:** OpenCV kütüphanesi, görüntü işleme işlemlerinde kullanılacak.
- **os:** Dosya ve klasör işlemleri için kullanılacak.
- **sys:** Komut satırından kullanıcıdan alınan ismi almak için kullanılacak.

---

*Kod Bloğu 2:*

```
base_path = r"C:\Users\smfse\OneDrive\Masaüstü\project_folder"
os.chdir(base_path)
```

**Açıklama:**
Çalışma dizini `project_folder` klasörü olarak ayarlanmıştır. Tüm dosya işlemleri bu dizin altında yapılacaktır.

---

*Kod Bloğu 3:*

```
if len(sys.argv) < 2:
    print("İsim parametresi verilmedi!")
    exit()
```

**Açıklama:**
Komut satırından isim parametresi kontrol edilir. Eğer isim girilmemişse program sonlandırılır. Collect modülü kişiye özel veri topladığı için isim zorunludur.

---

*Kod Bloğu 4:*

```
name = sys.argv[1]
path = os.path.join("dataset", name)
os.makedirs(path, exist_ok=True)
```

**Açıklama:**
Girilen isimle birlikte `dataset` klasörü altında o kişiye özel bir klasör oluşturulur. Böylece her kişiye ait görüntüler ayrı klasörlerde tutulur.

---

*Kod Bloğu 5:*

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
count = 0
```

**Açıklama:**

- OpenCV'nin hazır Haar Cascade yüz algılama modeli yüklenir.
- Kamera açılır (0 numaralı kamera aygıtı).
- `count` değişkeni ile kaç fotoğraf kaydedildiği tutulur.

---

*Kod Bloğu 6:*

```
while count < 100:
```

**Açıklama:**
Toplamda 100 adet yüz resmi toplanması hedeflenir. Eğitim için yeterli veri çeşitliliği sağlanması amaçlanır.

---

*Kod Bloğu 7:*

```
ret, frame = cap.read()
if not ret:
    continue
frame = cv2.flip(frame, 1)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

**Açıklama:**

- Kameradan görüntü alınır.
- Görüntü aynalanır (flip yapılır), böylece kamera hareketi doğal olur.
- Görüntü gri tonlamaya çevrilir çünkü yüz algılama gri görüntü üzerinde yapılır (işlem yükü düşürülür).

---

*Kod Bloğu 8:*

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
```

**Açıklama:**
Haar Cascade modeli ile yüzde yüz tespit edilir. Parametreler (ölçek ve komşuluk sayısı) yüz hassasiyeti için ayarlanmıştır.

---

*Kod Bloğu 9:*

```
for (x, y, w, h) in faces:
    count += 1
    roi = gray[y:y+h, x:x+w]
    roi = cv2.resize(roi, (200, 200))
    cv2.imwrite(f"{path}/{count}.jpg", roi)
```

**Açıklama:**

- Algılanan yüz bölgesi `roi` (Region of Interest) olarak alınır.
- 200x200 piksele normalize edilir (tüm eğitim görüntüleri aynı boyutta olmalı).
- Kişinin klasörüne ilgili görüntü `.jpg` olarak kaydedilir.

---

*Kod Bloğu 10:*

```
cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
cv2.putText(frame, f"{count}/100", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,
255, 0), 2)
break
```

**Açıklama:**

- Ekrana yüzün etrafına dikdörtgen çizilir ve kaçıncı fotoğraf olduğu gösterilir.
- `break` komutuyla aynı kareden sadece bir yüz alınması sağlanır. (Tek yüz toplanıyor.)

---

*Kod Bloğu 11:*

```
cv2.imshow("Collecting Faces", frame)
if cv2.waitKey(1) & 0xFF == 27:
    break
```

**Açıklama:**

- Toplama işlemi sırasında ekranda anlık görüntü gösterilir.
- `Esc` tuşuna basıldığında veri toplama işlemi manuel olarak sonlandırılabilir.

---

*Kod Bloğu 12:*

```
cap.release()
cv2.destroyAllWindows()
```

**Açıklama:**
Kamera ve pencere kapatılır, kaynaklar serbest bırakılır.

collect.py

C: > Users > smfse > OneDrive > Masaüstü > project_folder > collect.py > ...

```python
    print("İsim parametresi verilmedi!")
    exit()

name = sys.argv[1]
path = os.path.join("dataset", name)
os.makedirs(path, exist_ok=True)

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
count = 0

while count < 100:
    ret, frame = cap.read()
    if not ret:
        continue
    frame = cv2.flip(frame, 1)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    for (x, y, w, h) in faces:
        count += 1
        roi = gray[y:y+h, x:x+w]
        roi = cv2.resize(roi, (200, 200))
        cv2.imwrite(f"{path}/{count}.jpg", roi)
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
        cv2.putText(frame, f"{count}/100", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        break

    cv2.imshow("Collecting Faces", frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

*Şekil 4*

### III.3.1.3. Python Kod Açıklamaları: Train.py

*Kod Bloğu 1:*

```
import cv2
import os
import numpy as np
```

**Açıklama:**
Projede kullanılan temel kütüphaneler yüklenmiştir:

- **cv2:** OpenCV görüntü işleme işlemleri için,
- **os:** Dosya ve klasör işlemleri için,
- **numpy:** Dizi işlemleri (etiketleri liste haline getirmek ve eğitim yapmak için).

---

*Kod Bloğu 2:*

```
base_path = r"C:\Users\smfse\OneDrive\Masaüstü\project_folder"
os.chdir(base_path)
```

**Açıklama:**
Projenin temel çalışma klasörü ayarlanır. Tüm veri ve model kayıtları bu dizinde yapılacaktır.

---

*Kod Bloğu 3:*

```
dataset_path = "dataset"
if not os.path.exists(dataset_path):
    print("Dataset klasörü bulunamadı!")
    exit()
```

**Açıklama:**
Dataset klasörü kontrol edilir. Eğer henüz Collect modülüyle hiç veri toplanmamışsa, eğitim yapamayacağı için program sonlandırılır. Eğitim yapabilmek için öncelikle dataset klasörünün dolu olması gerekir.

---

*Kod Bloğu 4:*

```
faces = []
labels = []
label_map = {}
```

**Açıklama:**

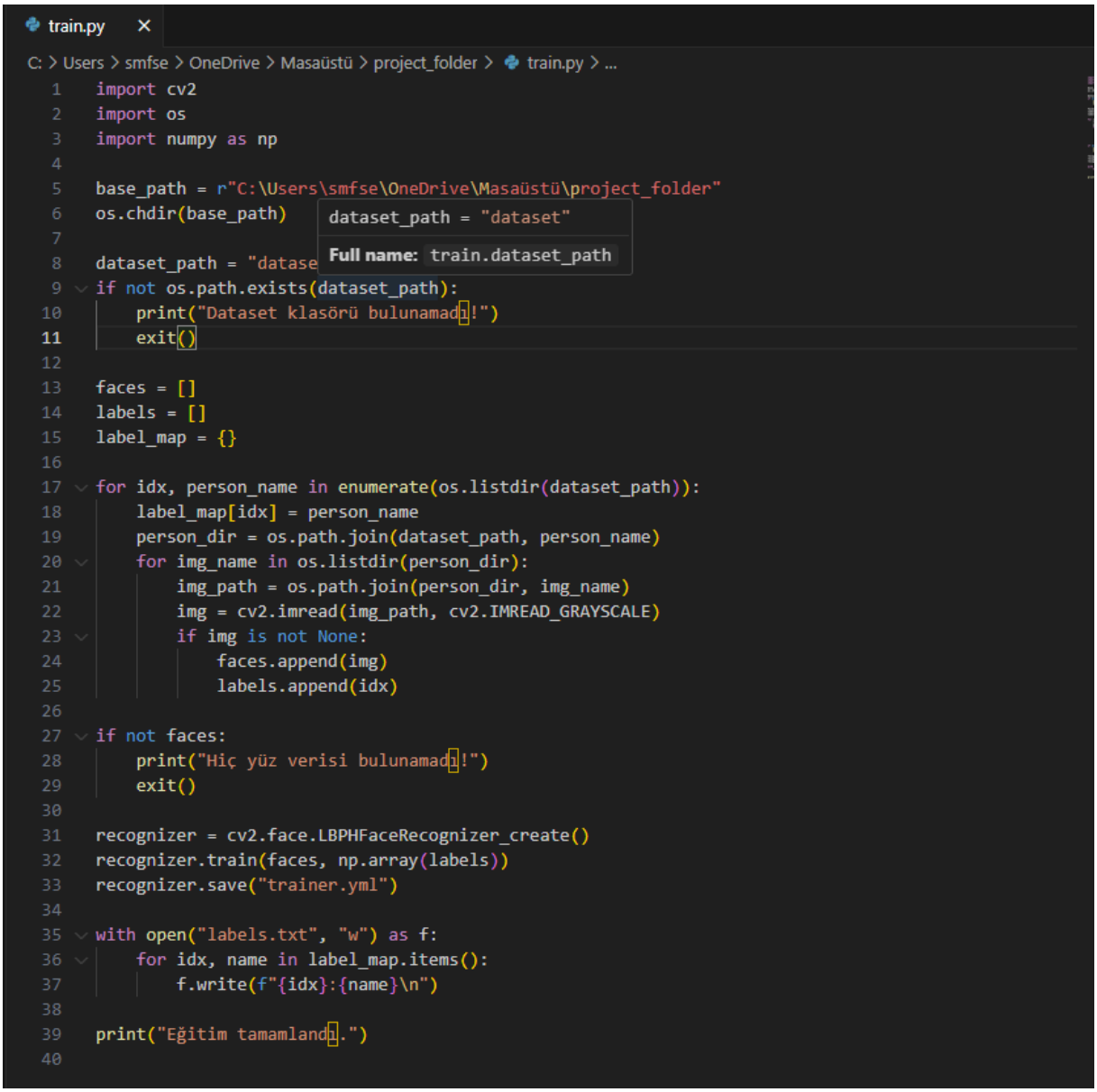
- `faces`: Görüntü verilerini tutacak liste.
- `labels`: İlgili kişi etiketlerini tutacak liste.
- `label_map`: Kişi isimlerini etiket numaralarıyla eşleştiren sözlük (mapping).

*Kod Bloğu 5:*

```
for idx, person_name in enumerate(os.listdir(dataset_path)):
    label_map[idx] = person_name
    person_dir = os.path.join(dataset_path, person_name)
    for img_name in os.listdir(person_dir):
        img_path = os.path.join(person_dir, img_name)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        if img is not None:
            faces.append(img)
            labels.append(idx)
```

**Açıklama:**

- Dataset klasörü içindeki her kişi klasörüne tek tek girilir.
- Klasördeki tüm yüz görüntüleri okunur.
- Görüntüler gri formatta (eğitim modeli gri görüntü ile çalışıyor) okunur ve eğitim verisi `faces` listesine eklenir.
- Hangi kişiye ait olduğu `labels` listesine eklenir.
- Her kişiye otomatik bir `idx` etiketi atanır. (Labeling yapılır.)




```python
import cv2
import os
import numpy as np

base_path = r"C:\Users\smfse\OneDrive\Masaüstü\project_folder"
os.chdir(base_path)

dataset_path = "dataset"
if not os.path.exists(dataset_path):
    print("Dataset klasörü bulunamadı!")
    exit()

faces = []
labels = []
label_map = {}

for idx, person_name in enumerate(os.listdir(dataset_path)):
    label_map[idx] = person_name
    person_dir = os.path.join(dataset_path, person_name)
    for img_name in os.listdir(person_dir):
        img_path = os.path.join(person_dir, img_name)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        if img is not None:
            faces.append(img)
            labels.append(idx)

if not faces:
    print("Hiç yüz verisi bulunamadı!")
    exit()

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.train(faces, np.array(labels))
recognizer.save("trainer.yml")

with open("labels.txt", "w") as f:
    for idx, name in label_map.items():
        f.write(f"{idx}:{name}\n")

print("Eğitim tamamlandı.")

```

*Şekil 5*

*Kod Bloğu 6:*

```
if not faces:
    print("Hiç yüz verisi bulunamadı!")
    exit()
```

**Açıklama:**
Hiç yüz görüntüsü bulunamadıysa eğitim yapılamaz. Bu kontrolle veri eksikliği engellenir.

---

*Kod Bloğu 7:*

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.train(faces, np.array(labels))
recognizer.save("trainer.yml")
```

**Açıklama:**

- **LBPHFaceRecognizer:** Local Binary Patterns Histogram yüz tanıma algoritması OpenCV üzerinden başlatılır.
- Toplanan `faces` ve `labels` ile model eğitimi yapılır.
- Eğitim tamamlandıktan sonra model `trainer.yml` dosyasına kaydedilir. Bu dosya Detect modülü tarafından kullanılacaktır.

---

*Kod Bloğu 8:*

```
with open("labels.txt", "w") as f:
    for idx, name in label_map.items():
        f.write(f"{idx}:{name}\n")
```

**Açıklama:**
Hangi etiketin hangi kişiye ait olduğunu tutan `labels.txt` dosyası oluşturulur. Böylece Detect modülü kime ait olduğunu anlayabilir.

---

*Kod Bloğu 9:*

```
print("Eğitim tamamlandı.")
```

**Açıklama:**
Eğitim süreci başarıyla tamamlandığında bilgi mesajı yazdırılır.

### III.3.1.4. Python Kod Açıklamaları: Detect.py (Python.org, 2020) (OpenCV.org, 2020)

*Kod Bloğu 1:*

```
import cv2
import serial
import time
import numpy as np
import os
import tkinter as tk
from threading import Thread
import keyboard
```

**Açıklama:**

- **cv2:** OpenCV — görüntü işleme için
- **serial:** Arduino ile seri haberleşme için
- **time:** Zaman gecikmeleri ve geçiş kontrolü için
- **numpy:** Sayısal işlemler için (aslında burada çok kullanılmasa da)
- **os:** Dosya ve klasör yönetimi için
- **tkinter:** Arayüz için
- **threading:** Arayüz ve görüntü işleme işlemlerini ayrı iş parçacıklarında çalıştırmak için
- **keyboard:** Klavye üzerinden araç kontrol komutlarını almak için

---

*Kod Bloğu 2:*

```
ser = serial.Serial('COM7', 9600, timeout=1)
time.sleep(2)
```

**Açıklama:**

- Arduino ile COM7 portu üzerinden 9600 baudrate hızında seri haberleşme başlatılır.
- **time.sleep(2):** Arduino'nun açılış sonrası hazır olması için 2 saniyelik bekleme süresi eklenir.

---

*Kod Bloğu 3:*

```
base_path = r"C:\Users\smfse\OneDrive\Masaüstü\project_folder"
os.chdir(base_path)
```

**Açıklama:**
Projeye ait tüm dosyaların bulunduğu klasör dizinine geçilir.

---

*Kod Bloğu 4:*

```
if not os.path.exists("trainer.yml") or not os.path.exists("labels.txt"):
    print("Model veya label dosyası eksik!")
    exit()
```

**Açıklama:**

- Eğitim yapılmamışsa (train.py çalıştırılmamışsa), model ve etiket dosyası bulunamayacağı için program sonlandırılır.

---

*Kod Bloğu 5:*

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("trainer.yml")
```

**Açıklama:**

- LBPH yüz tanıma modeli yüklenir ve daha önce eğitilmiş model dosyası `trainer.yml` dosyasından okunur.

---

*Kod Bloğu 6:*

```
label_map = {}
with open("labels.txt", "r") as f:
    for line in f:
        idx, name = line.strip().split(":")
        label_map[int(idx)] = name
```

**Açıklama:**

- Kişi isimleri ve etiket numaraları `labels.txt` dosyasından okunur ve `label_map` sözlüğüne yüklenir.
- Artık sistem her etiketi karşılık gelen isimle eşleştirebilir.

---

*Kod Bloğu 7:*

```
selected_person = [list(label_map.values())[0]]
switching = [False]
last_switch_time = [0]
```

**Açıklama:**

- Program başlangıcında aktif kişi ilk kişi olarak ayarlanır.

- **switching** ve **last_switch_time** değişkenleri kişi değiştirme sırasında çakışmayı engellemek için kullanılacak.

---

*Kod Bloğu 8: (Arayüz için ayrı thread)*

```
def gui_thread():
    def change_target(name):
        if selected_person[0] != name:
            selected_person[0] = name
            switching[0] = True
            last_switch_time[0] = time.time()
            print(f"Kişi değişti: {name}")
```

**Açıklama:**

- Arayüzden kullanıcı istediği kişiyi seçtiğinde `selected_person` güncellenir.
- Kişi değişimi anında sistem 100 ms'lik geçiş koruması uygulayacak.

---

*Kod Bloğu 9: (Arayüz başlatma)*

```
    gui = tk.Tk()
    gui.title("Kişi Seçimi")
    for name in label_map.values():
        btn = tk.Button(gui, text=name, width=20, height=2, command=lambda
n=name: change_target(n))
        btn.pack(pady=5)
    gui.mainloop()
```

**Açıklama:**

- Arayüz oluşturulur.
- Arayüzde her kişi için bir buton yer alır.
- Butona tıklandığında `change_target()` fonksiyonu çağrılır.

---

*Kod Bloğu 10:*

```
t = Thread(target=gui_thread, daemon=True)
t.start()
```

**Açıklama:**

- Arayüz ayrı bir iş parçacığında çalıştırılır.
- Böylece arayüz donmadan kamera görüntüsü akmaya devam eder.

---

*Kod Bloğu 11:*

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
```

**Açıklama:**

- Haar Cascade yüz tespit sınıflandırıcısı yüklenir.
- Kameradan canlı görüntü akışı başlatılır.

---

*Kod Bloğu 12: (Ana döngü)*

```
while True:
    ret, frame = cap.read()
    if not ret:
        continue
    frame = cv2.flip(frame, 1)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

**Açıklama:**

- Kameradan alınan görüntü yatay çevrilir (ayna etkisi verir)
- Gri tonlamaya çevrilir. Haar Cascade gri görüntü ile çalışır.

---

*Kod Bloğu 13: (Geçiş Koruması)*

```
    if switching[0]:
        if time.time() - last_switch_time[0] < 0.2:
            cv2.imshow("Detect", frame)
            if cv2.waitKey(1) & 0xFF == 27:
                break
            continue
        else:
            switching[0] = False
```

**Açıklama:**

- Kişi değişiminde 100-200 ms geçiş koruması uygulanır.
- Geçici yüz tespiti devre dışı bırakılır. Böylece eski ve yeni kişi çakışmaları engellenir.

---

*Kod Bloğu 14: (Yüz Algılama ve Tanıma)*

```
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    target_found = False
```

**Açıklama:**

- Görüntüdeki tüm yüzler tespit edilir.
- `target_found`: Seçilen kişi bulunup bulunmadığını kontrol eder.

---

*Kod Bloğu 15:*

```
    for (x, y, w, h) in faces:
        roi = cv2.resize(gray[y:y+h, x:x+w], (200, 200))
        id_, conf = recognizer.predict(roi)
        name = label_map[id_] if conf < 80 else "Unknown"

        if name == selected_person[0]:
            target_found = True
            cx = x + w//2
            cy = y + h//2
            data = f"{cx},{cy}\n"
            ser.write(data.encode('utf-8'))
            print(f"GÖNDERİLEN: {data.strip()}")
```

**Açıklama:**

- Her yüz için LBPH modeli ile kime ait olduğu tahmin edilir.
- Eğer tahmin edilen kişi seçili kişiyle eşleşiyorsa:
    - Yüzün merkez koordinatı hesaplanır.
    - Seri port üzerinden Arduino'ya `cx,cy` formatında veri gönderilir.
    - Servo motor bu koordinata kitlenecektir.
- Eğer kişi eşleşmiyorsa servo hareket etmez.

---

*Kod Bloğu 16:*

```
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
            cv2.putText(frame, name, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX,
0.8, (0, 255, 0), 2)
            break
```

**Açıklama:**

- Seçilen kişi bulunduğunda yeşil dikdörtgen çizilir.
- Kişi ismi ekrana yazdırılır.

---

*Kod Bloğu 17: (Kişi bulunamazsa)*

```
    if not target_found:
        print("Seçilen kişi bulunamadı, servo komut gönderilmiyor.")
```

**Açıklama:**

- Seçilen kişi karede değilse hiçbir servo komutu gönderilmez.

---

*Kod Bloğu 18: (Araç Kontrolü)*

```
    if keyboard.is_pressed('w'):
        ser.write(b'W')
    elif keyboard.is_pressed('s'):
        ser.write(b'S')
    elif keyboard.is_pressed('a'):
        ser.write(b'A')
    elif keyboard.is_pressed('d'):
        ser.write(b'D')
    else:
        ser.write(b'Z')
```

**Açıklama:**

- Klavye üzerinden araç kontrolü sağlanır (WASD).
- Servo komutlarından bağımsız olarak araç motor komutları Arduino'ya gönderilir.

---

*Kod Bloğu 19:*

```
    cv2.imshow("Detect", frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break
```

**Açıklama:**

- Görüntü ekranda gösterilir.
- ESC tuşuna basıldığında çıkış yapılır.

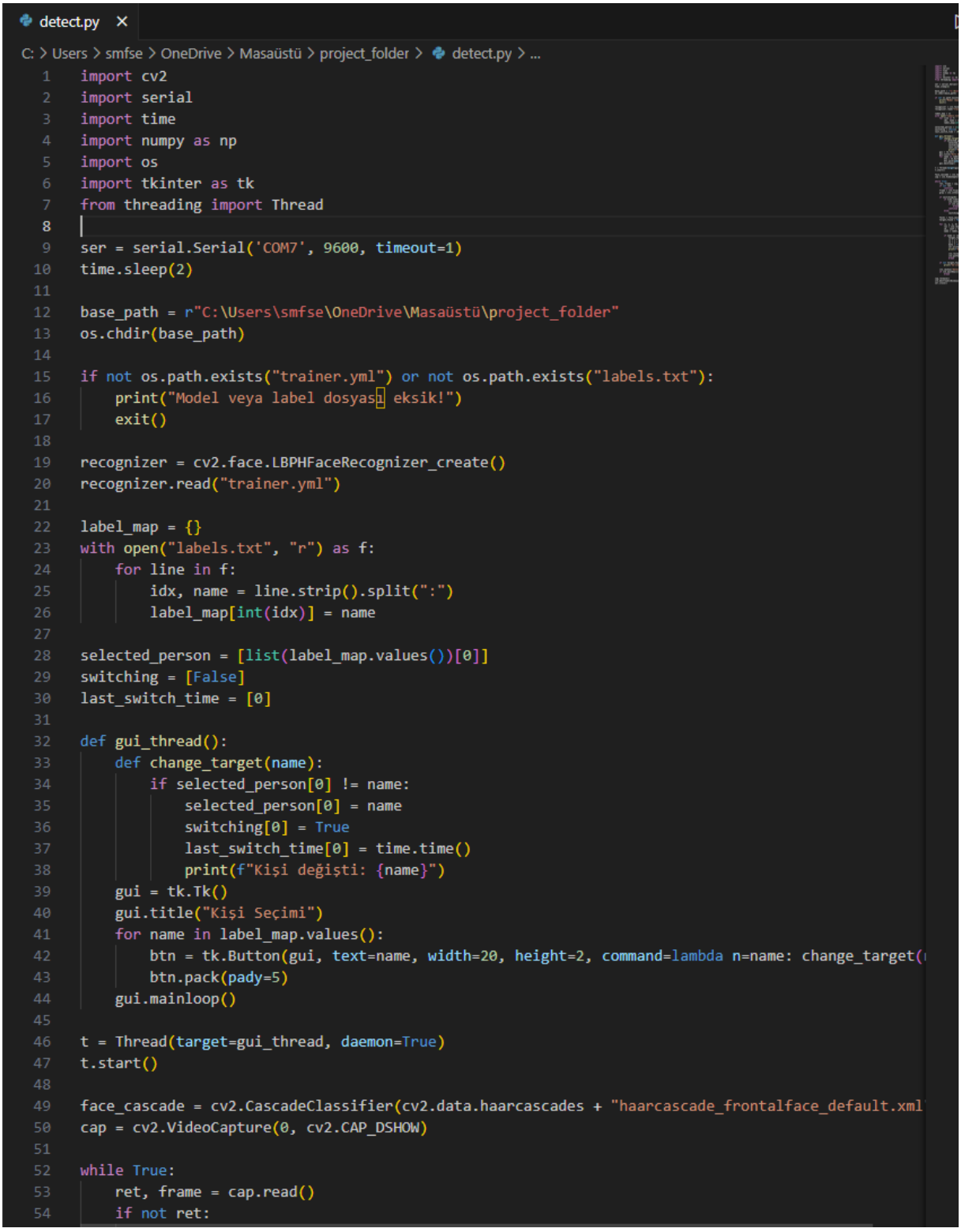
---

*Kod Bloğu 20:*

```
cap.release()
cv2.destroyAllWindows()
ser.close()
```

**Açıklama:**

- Kamera ve seri port düzgün kapatılır, kaynaklar serbest bırakılır.




*Şekil 6*

```python
while True:
    ret, frame = cap.read()
    if not ret:
        continue
    frame = cv2.flip(frame, 1)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    if switching[0]:
        if time.time() - last_switch_time[0] < 0.2:
            cv2.imshow("Detect", frame)
            if cv2.waitKey(1) & 0xFF == 27:
                break
            continue
        else:
            switching[0] = False

    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    target_found = False

    for (x, y, w, h) in faces:
        roi = cv2.resize(gray[y:y+h, x:x+w], (200, 200))
        id_, conf = recognizer.predict(roi)
        name = label_map[id_] if conf < 80 else "Unknown"

        if name == selected_person[0]:
            target_found = True
            cx = x + w//2
            cy = y + h//2
            data = f"{cx},{cy}\n"
            ser.write(data.encode('utf-8'))
            print(f"GÖNDERİLEN: {data.strip()}")

            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
            cv2.putText(frame, name, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 2)
            break

    if not target_found:
        print("Seçilen kişi bulunamadı, servo komut gönderilmiyor.")

    cv2.imshow("Detect", frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break

cap.release()
cv2.destroyAllWindows()
ser.close()
```

*Şekil 7*

### III.3.1.5. Python Kod Açıklamaları: reset.py

---

*Kod Bloğu 1: Gerekli Kütüphaneler ve Dizin Tanımı*

```
import shutil
import os

base_path = r"C:\Users\smfse\OneDrive\Masaüstü\project_folder"
os.chdir(base_path)
```

**Açıklama:**

- **shutil:** Klasör ve dosya işlemleri için kullanılır (silme işlemlerinde aktif).
- **os:** İşletim sistemi tabanlı yol yönetimi ve çalışma dizini ayarlaması yapılır.
- **base_path:** Projenin çalıştığı kök klasör tanımlanmıştır.
- **os.chdir():** Çalışma dizini proje klasörüne çekilmiştir.

---

*Kod Bloğu 2: Dataset ve Model Dosyalarının Temizlenmesi*

```
if os.path.exists("dataset"):
    shutil.rmtree("dataset")
if os.path.exists("trainer.yml"):
    os.remove("trainer.yml")
if os.path.exists("labels.txt"):
    os.remove("labels.txt")
os.makedirs("dataset", exist_ok=True)
print("Sistem tamamen sıfırlandı.")
```

**Açıklama:**

- **dataset klasörü:**
  Toplanan yüz görüntülerini içerir. Bu klasör tamamen silinir.
- **trainer.yml dosyası:**
  Eğitilmiş yüz tanıma model dosyasıdır. Silinerek yeni eğitim için temizlenir.
- **labels.txt dosyası:**
  Eğitime alınan kişilerin etiket isimlerini tutar. Silinerek sıfırlanır.
- **os.makedirs("dataset", exist_ok=True):**
  Silinen dataset klasörünü yeniden oluşturur, böylece yeni veri toplayabilmek için hazır olur.
- **print():**
  Kullanıcıya temizleme işleminin tamamlandığı bilgisini verir.

reset.py

C: > Users > smfse > OneDrive > Masaüstü > project_folder > reset.py > base_path

```python
import shutil
import os

base_path = r"C:\Users\smfse\OneDrive\Masaüstü\project_folder"
os.chdir(base_path)

if os.path.exists("dataset"):
    shutil.rmtree("dataset")
if os.path.exists("trainer.yml"):
    os.remove("trainer.yml")
if os.path.exists("labels.txt"):
    os.remove("labels.txt")

os.makedirs("dataset", exist_ok=True)
print("Sistem tamamen sıfırlandı.")

```

*Şekil 8*

### III.3.1.6. Python Kod Açıklamaları: Color_Tracking.py (Python.org, 2020)

---

*Kod Bloğu 1:*

```
import cv2
import serial
import time
import numpy as np
import os
import tkinter as tk
from threading import Thread
```

**Açıklama:**

- **cv2:** Görüntü işleme (OpenCV)
- **serial:** Arduino ile seri port haberleşmesi
- **time:** Zamanlama için
- **numpy:** Görüntü üzerinde matematiksel işlemler ve maskeleme için
- **os:** Klasör yönetimi için
- **tkinter:** Renk seçimi arayüzünü oluşturmak için
- **threading:** Renk seçim arayüzünü görüntü işleme döngüsünden ayırmak için

---

*Kod Bloğu 2:*

```
ser = serial.Serial('COM7', 9600, timeout=1)
time.sleep(2)
```

**Açıklama:**

- Arduino ile COM7 üzerinden haberleşme başlatılır.
- Arduino'nun açılış sonrası hazır hale gelmesi için 2 saniyelik bekleme uygulanır.

---

*Kod Bloğu 3:*

```
base_path = r"C:\Users\smfse\OneDrive\Masaüstü\project_folder"
os.chdir(base_path)
```

**Açıklama:**

- Çalışma klasörü olarak proje dizinine geçilir.

---

*Kod Bloğu 4:*

```
selected_color = ["red"]
```

**Açıklama:**

- Başlangıçta seçili renk "red" (kırmızı) olarak belirlenir.
- Renk seçim arayüzünde bu değişken güncellenerek sistem hangi rengi izleyeceğini bilecek.

---

*Kod Bloğu 5:*

```
color_ranges = {
    "red": [
        (np.array([0, 100, 100]), np.array([10, 255, 255])),
        (np.array([160, 100, 100]), np.array([179, 255, 255]))
    ],
    "green": [
        (np.array([40, 50, 50]), np.array([80, 255, 255]))
    ],
    "blue": [
        (np.array([100, 150, 0]), np.array([140, 255, 255]))
    ]
}
```

**Açıklama:**

- HSV renk uzayında her renk için alt ve üst sınırlar tanımlanır.
- Kırmızı iki ayrı aralıkla alınır (0-10 ve 160-179), çünkü kırmızı renk HSV dairesinin başında ve sonunda yer alır.
- Bu sayede geniş renk toleransı sağlanır.

---

*Kod Bloğu 6: (Renk Seçimi Arayüzü)*

```
def color_gui():
    def set_color(c):
        selected_color[0] = c
        print(f"Renk değiştirildi: {c}")
```

**Açıklama:**

- Kullanıcı butonlardan birine bastığında seçili renk güncellenir.
- Tek elemanlı liste (mutable list) kullanılarak thread'ler arası canlı değişim sağlanır.

---

*Kod Bloğu 7: (Arayüzün Oluşturulması)*

```
    gui = tk.Tk()
    gui.title("Renk Seçimi (Canlı)")
    gui.geometry("300x300")

    tk.Button(gui, text="Kırmızı", font=("Arial", 14), width=20, height=2,
              command=lambda: set_color("red")).pack(pady=10)
    tk.Button(gui, text="Yeşil", font=("Arial", 14), width=20, height=2,
              command=lambda: set_color("green")).pack(pady=10)
    tk.Button(gui, text="Mavi", font=("Arial", 14), width=20, height=2,
              command=lambda: set_color("blue")).pack(pady=10)

    gui.mainloop()
```

**Açıklama:**

- Tkinter arayüz penceresi açılır.
- 3 renk için ayrı butonlar tanımlanır.
- Butonlara basıldığında seçili renk güncellenir ve ekrana yazılır.

---

*Kod Bloğu 8: (Arayüz Thread'i Başlatma)*

```
t = Thread(target=color_gui, daemon=True)
t.start()
```

**Açıklama:**

- Renk seçim arayüzü ayrı thread üzerinde başlatılır.
- Böylece kamera görüntü akışını etkilemeden arayüz açık kalır.

---

*Kod Bloğu 9: (Kamera Başlatma)*

```
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
```

**Açıklama:**

- Bilgisayara bağlı kameradan canlı görüntü alınır.

---

*Kod Bloğu 10: (Ana Döngü)*

```
while True:
    ret, frame = cap.read()
    if not ret:
        continue
    frame = cv2.flip(frame, 1)
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

**Açıklama:**

- Kameradan görüntü alınır, yatay çevrilir.
- Renk analizinde daha doğru sonuç almak için BGR formatındaki görüntü HSV formatına çevrilir.

---

*Kod Bloğu 11: (Renk Maskesi Oluşturma)*

```
    mask = None
    for lower, upper in color_ranges[selected_color[0]]:
        current_mask = cv2.inRange(hsv, lower, upper)
        mask = current_mask if mask is None else cv2.bitwise_or(mask,
current_mask)
```

**Açıklama:**

- Seçilen renge göre maskeleme yapılır.
- Renk sınırlarının belirttiği piksel aralığı tespit edilir.
- Eğer renk birden fazla aralık içeriyorsa (örneğin kırmızı), maskeler üst üste bindirilir.

---

*Kod Bloğu 12: (Kontur Tespiti)*

```
    contours,          _          =          cv2.findContours(mask,          cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    largest_contour = max(contours, key=cv2.contourArea, default=None)
```

**Açıklama:**

- Renk maskesi üzerinde konturlar (renkli alan sınırları) bulunur.
- En büyük kontur seçilir (büyük nesneyi takip etmek için).

---

*Kod Bloğu 13: (Nesne Takibi ve Arduino'ya Veri Gönderme)*

```
    if largest_contour is not None and cv2.contourArea(largest_contour) >
500:
        (x, y, w, h) = cv2.boundingRect(largest_contour)
        cx = x + w // 2
        cy = y + h // 2
        data = f"{cx},{cy}\n"
        ser.write(data.encode('utf-8'))
        print(f"{selected_color[0].upper()}          TAKİP          GÖNDERİLDİ:
{data.strip()}")
```

**Açıklama:**

- Eğer kontur alanı belirli bir eşikten büyükse (küçük gürültüleri ayıklamak için 500 px),
- Konturun merkez koordinatları hesaplanır.
- Koordinatlar seri port üzerinden Arduino'ya gönderilir.
- Servo motor bu merkeze kitlenir.

---

*Kod Bloğu 14: (Çizim ve Görüntüleme)*

```
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 255), 2)
        cv2.circle(frame, (cx, cy), 5, (255, 0, 0), -1)
```

**Açıklama:**

- Tespit edilen renkli alan etrafına dikdörtgen çizilir.
- Orta noktasına mavi daire çizilerek takip edildiği görsel olarak gösterilir.

---

*Kod Bloğu 15: (Renk Bulunamazsa)*

```
    else:
        print(f"{selected_color[0].upper()} nesne bulunamadı.")
```

**Açıklama:**

- Eğer istenen renk görüntüde bulunamazsa, Arduino'ya veri gönderilmez.
- Konsolda bilgi yazdırılır.

*Kod Bloğu 16:*

```
    cv2.imshow("Color Tracking", frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break
```

**Açıklama:**

- Görüntü ekranda gösterilir.
- ESC tuşuna basıldığında program sonlandırılır.

---

*Kod Bloğu 17:*

```
cap.release()
cv2.destroyAllWindows()
ser.close()
```

**Açıklama:**

- Kamera ve seri port düzgün şekilde kapatılarak kaynaklar serbest bırakılır.

color_tracking.py

C: > Users > smfse > OneDrive > Masaüstü > project_folder > color_tracking.py > ...

```python
import cv2
import serial
import time
import numpy as np
import os
import tkinter as tk
from threading import Thread

ser = serial.Serial('COM7', 9600, timeout=1)
time.sleep(2)

base_path = r"C:\Users\smfse\OneDrive\Masaüstü\project_folder"
os.chdir(base_path)

selected_color = ["red"]

color_ranges = {
    "red": [
        (np.array([0, 100, 100]), np.array([10, 255, 255])),
        (np.array([160, 100, 100]), np.array([179, 255, 255]))
    ],
    "green": [
        (np.array([40, 50, 50]), np.array([80, 255, 255]))
    ],
    "blue": [
        (np.array([100, 150, 0]), np.array([140, 255, 255]))
    ]
}

def color_gui():
    def set_color(c):
        selected_color[0] = c
        print(f"Renk değiştirildi: {c}")

    gui = tk.Tk()
    gui.title("Renk Seçimi (Canlı)")
    gui.geometry("300x300")

    tk.Button(gui, text="Kırmızı", font=("Arial", 14), width=20, height=2,
              command=lambda: set_color("red")).pack(pady=10)
    tk.Button(gui, text="Yeşil", font=("Arial", 14), width=20, height=2,
              command=lambda: set_color("green")).pack(pady=10)
    tk.Button(gui, text="Mavi", font=("Arial", 14), width=20, height=2,
              command=lambda: set_color("blue")).pack(pady=10)

    gui.mainloop()

t = Thread(target=color_gui, daemon=True)
t.start()

cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)

```

*Şekil 9*

```python
while True:
    ret, frame = cap.read()
    if not ret:
        continue

    frame = cv2.flip(frame, 1)
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    mask = None
    for lower, upper in color_ranges[selected_color[0]]:
        current_mask = cv2.inRange(hsv, lower, upper)
        mask = current_mask if mask is None else cv2.bitwise_or(mask, current_mask)

    contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    largest_contour = max(contours, key=cv2.contourArea, default=None)

    if largest_contour is not None and cv2.contourArea(largest_contour) > 500:
        (x, y, w, h) = cv2.boundingRect(largest_contour)
        cx = x + w // 2
        cy = y + h // 2
        data = f"{cx},{cy}\n"
        ser.write(data.encode('utf-8'))
        print(f"{selected_color[0].upper()} TAKİP GÖNDERİLDİ: {data.strip()}")
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 255), 2)
        cv2.circle(frame, (cx, cy), 5, (255, 0, 0), -1)
    else:
        print(f"{selected_color[0].upper()} nesne bulunamadı.")

    cv2.imshow("Color Tracking", frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break

cap.release()
cv2.destroyAllWindows()
ser.close()

```

*Şekil 10*

### III.3.1.7. Python Kod Açıklamaları: Tracking.py

---

*Kod Bloğu 1:*

```
import cv2
import serial
import time
import os
```

**Açıklama:**

- **cv2:** OpenCV görüntü işleme kütüphanesi.
- **serial:** Arduino ile haberleşme sağlamak için seri port.
- **time:** Zamanlama ve bekleme işlemleri için.
- **os:** Dosya ve klasör işlemleri için.

---

*Kod Bloğu 2:*

```
base_path = r"C:\Users\smfse\OneDrive\Masaüstü\project_folder"
os.chdir(base_path)
```

**Açıklama:**

- Kodun çalıştığı dizin proje klasörüne ayarlanır.
- Eğitim verileri, model dosyaları ve etiketler burada tutulur.

---

*Kod Bloğu 3:*

```
ser = serial.Serial('COM7', 9600, timeout=1)
time.sleep(2)
```

**Açıklama:**

- Arduino ile COM7 portu üzerinden 9600 baud hızında seri haberleşme başlatılır.
- Arduino'nun hazır hale gelmesi için 2 saniye beklenir.

---

*Kod Bloğu 4:*

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
```

**Açıklama:**

- Haar Cascade yüz sınıflandırıcısı yüklenir.
- Kameradan canlı görüntü almak için `VideoCapture` başlatılır.

---

*Kod Bloğu 5 (Ana Döngü Başlangıcı):*

```
while True:
    ret, frame = cap.read()
    if not ret:
        continue
    frame = cv2.flip(frame, 1)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

**Açıklama:**

- Kameradan görüntü alınır.
- Görüntü ayna efekti (flip) ile ters çevrilir (kullanıcı açısından doğal görüntü için).
- Renkli görüntü griye dönüştürülür (yüz algılamada gri görüntü kullanılır).

---

*Kod Bloğu 6 (Yüz Tespiti ve Koordinat Hesaplama):*

```
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x, y, w, h) in faces:
        cx = x + w//2
        cy = y + h//2
        data = f"{cx},{cy}\n"
        ser.write(data.encode('utf-8'))
        print(f"GÖNDERİLEN: {data.strip()}")
```

**Açıklama:**

- `detectMultiScale` fonksiyonu ile yüzler algılanır.
- Yüzün tam merkez koordinatları hesaplanır.
- Hesaplanan X ve Y koordinatları seri port üzerinden Arduino'ya gönderilir.
- Servo motorlar bu koordinatlara göre hareket eder.

*Kod Bloğu 7 (Görsel Çizimler):*

```
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 255, 0), 2)
        cv2.circle(frame, (cx, cy), 5, (0, 0, 255), -1)
        break
```

**Açıklama:**

- Algılanan yüz etrafına dikdörtgen çizilir.
- Yüzün merkezine küçük bir daire çizilir.
- `break` ifadesiyle yalnızca ilk algılanan yüz işlenir (birden fazla yüz algılansa bile sadece ilk yüz servoya gönderilir).
- Bu sayede kararsızlık ve servo çakışmaları önlenmiş olur.

---

*Kod Bloğu 8 (Ekranda Gösterim ve Çıkış Kontrolü):*

```
    cv2.imshow("Tracking", frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break
```

**Açıklama:**

- Görüntü ekranda gösterilir.
- ESC tuşuna basıldığında döngüden çıkılır.

---

*Kod Bloğu 9 (Kaynakları Serbest Bırakma):*

```
cap.release()
cv2.destroyAllWindows()
ser.close()
```

**Açıklama:**

- Kamera ve pencere kaynakları serbest bırakılır.
- Seri port kapatılır.

tracking.py

C: > Users > smfse > OneDrive > Masaüstü > project_folder > tracking.py > face_cascade

```python
import cv2
import serial
import time
import os
import keyboard

# Seri port ayarı
ser = serial.Serial('COM7', 9600, timeout=1)
time.sleep(2)

# Yüz tanıma
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)

while True:
    ret, frame = cap.read()
    if not ret:
        continue

    frame = cv2.flip(frame, 1)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    motor_command = "Z"  # Default: stop

    if keyboard.is_pressed('w'):
        motor_command = "W"
    elif keyboard.is_pressed('s'):
        motor_command = "S"
    elif keyboard.is_pressed('a'):
        motor_command = "A"
    elif keyboard.is_pressed('d'):
        motor_command = "D"

    for (x, y, w, h) in faces:
        cx = x + w//2
        cy = y + h//2

        data = f"{cx},{cy},{motor_command}\n"
        ser.write(data.encode('utf-8'))
        print(f"GÖNDERİLEN: {data.strip()}")

        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 255, 0), 2)
        cv2.circle(frame, (cx, cy), 5, (0, 0, 255), -1)
        break

    cv2.imshow("Tracking + Motor Kontrol", frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break

cap.release()
cv2.destroyAllWindows()
ser.close()
```

*Şekil 11*

### III.3.1.8. Python Kod Açıklamaları: yolo_object_tracking.py (THU-MIG, 2024) (Ultralytics, 2023)

---

*Kod Bloğu 1:*

```
import cv2
import serial
import time
import os
import tkinter as tk
from threading import Thread
from ultralytics import YOLO
```

**Açıklama:**

- **cv2:** Görüntü işleme için OpenCV.
- **serial:** Arduino ile seri haberleşme için.
- **time:** Bekleme ve zaman kontrolü.
- **os:** Dizin işlemleri.
- **tkinter:** Kullanıcı arayüzü oluşturmak için.
- **threading:** Arayüzü ve görüntü işleme işlemini paralel çalıştırmak için.
- **ultralytics:** YOLOv8 derin öğrenme modeli için kütüphane.

---

*Kod Bloğu 2:*

```
ser = serial.Serial('COM7', 9600, timeout=1)
time.sleep(2)
```

**Açıklama:**

- Arduino ile COM7 üzerinden seri port açılıyor.
- 2 saniye bekleyerek bağlantının tam açılması sağlanıyor.

---

*Kod Bloğu 3:*

```
model = YOLO("yolov8n.pt")
class_names = model.names
selected_class = [list(class_names.values())[0]]
```

**Açıklama:**

- YOLOv8 (nano modeli) yükleniyor.
- Modelin tanıyabileceği sınıf isimleri (object class) alınıyor.
- Başlangıçta ilk sınıf otomatik olarak seçiliyor.

---

*Kod Bloğu 4: GUI Arayüzü (Thread ile çalışıyor)*

```
def gui_thread():
    def set_target(name):
        selected_class[0] = name
        print(f"Seçilen sınıf: {name}")
```

**Açıklama:**

- Kullanıcının hangi nesneyi takip etmek istediğini seçmesi için arayüz oluşturuluyor.
- Seçim değiştiğinde `selected_class` güncelleniyor.

---

*Kod Bloğu 5: GUI Yapısı*

```
    gui = tk.Tk()
    gui.title("YOLOv8 Nesne Seçimi")
    gui.geometry("300x500")

    canvas = tk.Canvas(gui)
    scrollbar = tk.Scrollbar(gui, orient="vertical", command=canvas.yview)
    frame = tk.Frame(canvas)
```

**Açıklama:**

- Tkinter ile kaydırmalı bir arayüz oluşturuluyor.
- Çok sayıda sınıf olabileceği için scroll bar ekleniyor.

---

*Kod Bloğu 6: Tüm sınıflara buton ekleme*

```
    for idx in class_names:
        btn = tk.Button(frame, text=class_names[idx], width=30, height=1,
                        command=lambda n=class_names[idx]: set_target(n))
        btn.pack(pady=2)
```

**Açıklama:**

- YOLO modelinde tanımlı tüm nesneler için butonlar dinamik olarak oluşturuluyor.
- Kullanıcı hangi nesneye tıklarsa o sınıf hedef olarak ayarlanıyor.

---

*Kod Bloğu 7:*

```
    gui.mainloop()
```

**Açıklama:**

- Arayüz başlatılıyor.

---

*Kod Bloğu 8: Arayüzü ayrı thread'de başlatma*

```
t = Thread(target=gui_thread, daemon=True)
t.start()
```

**Açıklama:**

- Arayüz ve görüntü işleme birbirini engellemesin diye GUI ayrı thread'de çalıştırılıyor.

---

*Kod Bloğu 9: Kamera ve Haar Cascade başlatma*

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 
"haarcascade_frontalface_default.xml")
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
```

**Açıklama:**

- Kamera başlatılır.
- (Burada `face_cascade` yedekte kalmış, aslen yüz tanıma kullanılmıyor, sadece nesne takibi yapıyoruz.)

---

*Kod Bloğu 10: Ana Döngü Başlıyor*

```
while True:
    ret, frame = cap.read()
    if not ret:
        continue
    frame = cv2.flip(frame, 1)
    results = model(frame)
```

**Açıklama:**

- Kameradan görüntü alınır ve ters çevrilir (ayna etkisi).
- YOLOv8 modeline görüntü gönderilir ve algılama yapılır.

---

*Kod Bloğu 11: Sonuçları İşleme*

```
    found = False
    for r in results:
        for box in r.boxes:
            cls_id = int(box.cls[0].item())
            cls_name = class_names[cls_id]
```

**Açıklama:**

- Algılanan nesneler taranır.
- Her kutu (bounding box) için sınıf ID'si ve isim alınır.

---

*Kod Bloğu 12: Sadece Seçili Nesne Takibi*

```
            if cls_name == selected_class[0]:
                found = True
                x1, y1, x2, y2 = box.xyxy[0].cpu().numpy().astype(int)
                cx = (x1 + x2) // 2
                cy = (y1 + y2) // 2
                data = f"{cx},{cy}\n"
                ser.write(data.encode('utf-8'))
                print(f"{cls_name.upper()} GÖNDERİLDİ: {data.strip()}")
```

**Açıklama:**

- Sadece seçilen sınıfa ait nesne algılanırsa koordinatlar hesaplanır.
- Nesnenin merkez noktası bulunur ve bu koordinatlar seri port üzerinden Arduino'ya gönderilir.
- Servo motorlar buna göre hareket eder.

---

*Kod Bloğu 13: Görüntü Üzerine Çizimler*

```
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
                cv2.putText(frame,       cls_name,       (x1,       y1-10),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
                break
```

**Açıklama:**

- Algılanan nesnenin etrafına dikdörtgen çizilir.
- Sınıf ismi görüntü üzerine yazılır.

---

*Kod Bloğu 14:*

```
    if not found:
        print("Seçilen nesne bulunamadı, servo komut gönderilmiyor.")
```

**Açıklama:**

- Seçilen sınıftan herhangi bir nesne bulunamazsa bilgi mesajı verilir.

---

*Kod Bloğu 15: Görüntüyü Gösterme ve Çıkış Kontrolü*

```
    cv2.imshow("YOLOv8 Object Tracking", frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break
```

**Açıklama:**

- Görüntü ekranda gösterilir.
- ESC tuşuna basıldığında çıkış yapılır.

---

*Kod Bloğu 16: Kaynakları Serbest Bırakma*

```
cap.release()
cv2.destroyAllWindows()
ser.close()
```

**Açıklama:**

- Kamera, pencere ve seri port kapatılır.

yolo_object_tracking.py

C: > Users > smfse > OneDrive > Masaüstü > project_folder > yolo_object_tracking.py > ...

```python
import cv2
import serial
import time
import os
import tkinter as tk
from threading import Thread
from ultralytics import YOLO

# Seri portu ayarla
ser = serial.Serial('COM7', 9600, timeout=1)
time.sleep(2)

# YOLO modelini yükle
model = YOLO("yolov8n.pt")
class_names = model.names
selected_class = [list(class_names.values())[0]]  # Başlangıçta ilk sınıf seçili

# Tkinter GUI + SCROLL menü
def gui_thread():
    def set_target(name):
        selected_class[0] = name
        print(f"Seçilen sınıf: {name}")

    gui = tk.Tk()
    gui.title("YOLO Nesne Seçimi")
    gui.geometry("300x500")

    canvas = tk.Canvas(gui)
    scrollbar = tk.Scrollbar(gui, orient="vertical", command=canvas.yview)
    frame = tk.Frame(canvas)

    frame.bind(
        "<Configure>",
        lambda e: canvas.configure(scrollregion=canvas.bbox("all"))
    )

    canvas.create_window((0, 0), window=frame, anchor="nw")
    canvas.configure(yscrollcommand=scrollbar.set)
    canvas.pack(side="left", fill="both", expand=True)
    scrollbar.pack(side="right", fill="y")

    for idx in class_names:
        btn = tk.Button(frame, text=class_names[idx], width=30, height=1,
                        command=lambda n=class_names[idx]: set_target(n))
        btn.pack(pady=2)

    gui.mainloop()
```

*Şekil 12*

```python
t = Thread(target=gui_thread, daemon=True)
t.start()

cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)

while True:
    ret, frame = cap.read()
    if not ret:
        continue

    frame = cv2.flip(frame, 1)
    results = model(frame)

    found = False

    for r in results:
        for box in r.boxes:
            cls_id = int(box.cls[0].item())
            cls_name = class_names[cls_id]

            if cls_name == selected_class[0]:
                found = True
                x1, y1, x2, y2 = box.xyxy[0].cpu().numpy().astype(int)
                cx = (x1 + x2) // 2
                cy = (y1 + y2) // 2

                data = f"{cx},{cy}\n"
                ser.write(data.encode('utf-8'))
                print(f"{cls_name.upper()} GÖNDERİLDİ: {data.strip()}")

                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
                cv2.putText(frame, cls_name, (x1, y1-10), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255,
                break

    if not found:
        print("Seçilen nesne bulunamadı, servo komut gönderilmiyor.")

    cv2.imshow("YOLOv8 Object Tracking (Servo)", frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break

cap.release()
cv2.destroyAllWindows()
ser.close()
```

*Şekil 13*

### III.3.1.9. Python Kod Açıklamaları: yolo_full_detection.py (Ultralytics, 2023) (THU-MIG, 2024)

---

*Kod Bloğu 1:*

```
import cv2
import os
from ultralytics import YOLO
```

**Açıklama:**

- **cv2:** Görüntü işleme için OpenCV.
- **os:** Dizin ve klasör işlemleri için.
- **ultralytics:** YOLOv8 modelini çağırmak için.

---

*Kod Bloğu 2:*

```
base_path = r"C:\Users\smfse\OneDrive\Masaüstü\project_folder"
os.chdir(base_path)
```

**Açıklama:**

- Çalışma dizini proje klasörüne ayarlanıyor.
- Böylece model dosyasına ve diğer kaynaklara kolay erişim sağlanıyor.

---

*Kod Bloğu 3: YOLOv8 Modelinin Yüklenmesi*

```
model = YOLO("yolov8n.pt")
class_names = model.names
```

**Açıklama:**

- YOLOv8 nano (yolov8n.pt) modeli yükleniyor.
- Modelin tanıyabildiği tüm sınıf isimleri (class_names) alınıyor.

---

*Kod Bloğu 4: Kamera Başlatma*

```
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
```

**Açıklama:**

- Bilgisayara bağlı kameradan video akışı başlatılıyor.

*Kod Bloğu 5: Ana Döngü Başlıyor*

```
while True:
    ret, frame = cap.read()
    if not ret:
        continue
```

**Açıklama:**

- Kameradan yeni kare alınır.
- Eğer kamera görüntüsü alınamazsa döngü devam eder.

---

*Kod Bloğu 6: Görüntüyü Aynalamak*

```
    frame = cv2.flip(frame, 1)
```

**Açıklama:**

- Görüntü yatay olarak çevrilir (ayna görüntüsü efekti) — kullanıcı kendisini doğal bir şekilde ekranda görür.

---

*Kod Bloğu 7: YOLO Modeline Görüntü Gönderme*

```
    results = model(frame)
```

**Açıklama:**

- Kare görüntü YOLO modeline verilir.
- Model, nesne tespiti yapar ve sonuçları döndürür.

---

*Kod Bloğu 8: Algılanan Tüm Nesnelerin İşlenmesi*

```
    for r in results:
        for box in r.boxes:
            cls_id = int(box.cls[0].item())
            cls_name = class_names[cls_id]
            x1, y1, x2, y2 = box.xyxy[0].cpu().numpy().astype(int)
```

**Açıklama:**

- Her nesne kutusu için sınıf ID'si alınır ve isimle eşleştirilir.
- Nesnenin konum koordinatları (x1, y1, x2, y2) elde edilir.

---

*Kod Bloğu 9: Görüntü Üzerine Çizim*

```
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 255), 2)
            cv2.putText(frame, cls_name, (x1, y1 - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 255), 2)
```

**Açıklama:**

- Algılanan her nesne etrafına dikdörtgen çizilir.
- Sınıf adı (örneğin "person", "bottle", "cell phone" gibi) yazılır.

---

*Kod Bloğu 10: Görüntüyü Gösterme*

```
    cv2.imshow("YOLOv8 Full Detection (Sadece Görüntü)", frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break
```

**Açıklama:**

- Son işlenmiş görüntü ekranda gösterilir.
- ESC tuşuna basıldığında döngü sonlandırılır.

---

*Kod Bloğu 11: Kaynakların Serbest Bırakılması*

```
cap.release()
cv2.destroyAllWindows()
```

**Açıklama:**

- Kamera kapatılır ve tüm açık pencereler kapatılır.

yolo_full_detection.py

C: > Users > smfse > OneDrive > Masaüstü > project_folder > yolo_full_detection.py > YOLO

```python
import cv2
import os
from ultralytics import YOLO

base_path = r"C:\Users\smfse\OneDrive\Masaüstü\project_folder"
os.chdir(base_path)

model = YOLO("yolov8n.pt")
class_names = model.names

cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)

while True:
    ret, frame = cap.read()
    if not ret:
        continue

    frame = cv2.flip(frame, 1)
    results = model(frame)

    for r in results:
        for box in r.boxes:
            cls_id = int(box.cls[0].item())
            cls_name = class_names[cls_id]
            x1, y1, x2, y2 = box.xyxy[0].cpu().numpy().astype(int)

            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 255), 2)
            cv2.putText(frame, cls_name, (x1, y1 - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 255), 2)

    cv2.imshow("YOLOv8 Full Detection (Görüntü)", frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break

cap.release()
cv2.destroyAllWindows()

```

*Şekil 14*

### III.3.1.10. Python Kod Açıklamaları: main.py (Ana Kontrol Paneli) (Python.org, 2020)

---

*Kod Bloğu 1: Gerekli Kütüphaneler*

```
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import subprocess
import os
import shutil
```

**Açıklama:**

- **tkinter:** Grafik kullanıcı arayüzünü (GUI) oluşturmak için.
- **subprocess:** Python'dan harici .py dosyalarını çalıştırmak için.
- **os ve shutil:** Dosya sistemi ve klasör işlemleri yapmak için.

---

*Kod Bloğu 2: Proje Dizini ve Global Değişken*

```
base_path = r"C:\Users\smfse\OneDrive\Masaüstü\project_folder"
current_process = None
```

**Açıklama:**

- Projenin çalıştığı dizin belirleniyor.
- **current_process:** O anda çalışan modun bilgisini tutuyor. Modlar arası geçiş yaparken eski çalışan işlemi durdurmak için kullanılıyor.

---

*Kod Bloğu 3: Çalışan Modu Durdurma Fonksiyonu*

```
def stop_current():
    global current_process
    if current_process and current_process.poll() is None:
        current_process.terminate()
        current_process = None
```

**Açıklama:**

- Yeni bir mod başlatmadan önce aktif çalışan eski modu güvenli şekilde sonlandırır.

---

*Kod Bloğu 4:*

*Collect Modülünü Çalıştırma*

```
def run_collect():
    stop_current()
    popup = tk.Toplevel(root)
    popup.title("Collect")
    popup.geometry("300x200")

    tk.Label(popup, text="İsim giriniz:").pack(pady=10)
    entry = tk.Entry(popup)
    entry.pack(pady=10)

    def start():
        name = entry.get().strip()
        if not name:
            messagebox.showwarning("Uyarı", "İsim giriniz.")
            return
        global current_process
        current_process = subprocess.Popen(
            ["python", os.path.join(base_path, "collect.py"), name],
            cwd=base_path
        )
        popup.destroy()

    tk.Button(popup, text="Başla", command=start).pack(pady=10)
```

**Açıklama:**

- Kullanıcıdan yeni kişi için isim alınıyor.
- Girilen isim ile veri toplama (Collect) modülü başlatılıyor.
- Bu kısım kullanıcı etkileşimi gerektiriyor.

---

*Train Modu:*

```
def run_train():
    stop_current()
    global current_process
    current_process = subprocess.Popen(
        ["python", os.path.join(base_path, "train.py")],
        cwd=base_path
    )
```

*Detect Modu:*

```
def run_detect():
    stop_current()
    global current_process
    current_process = subprocess.Popen(
        ["python", os.path.join(base_path, "detect.py")],
        cwd=base_path
    )
```

*Klasik Tracking (Servo Yüz Takip):*

```
def run_tracking():
    stop_current()
    global current_process
    current_process = subprocess.Popen(
        ["python", os.path.join(base_path, "tracking.py")],
        cwd=base_path
    )
```

*Color Tracking (Renk Takip):*

```
def run_color_tracking():
    stop_current()
    global current_process
    current_process = subprocess.Popen(
        ["python", os.path.join(base_path, "color_tracking.py")],
        cwd=base_path
    )
```

*YOLOv8 Object Tracking (Servo + Nesne Takip):*

```
def run_yolo_object_tracking():
    stop_current()
    global current_process
    current_process = subprocess.Popen(
        ["python", os.path.join(base_path, "yolo_object_tracking.py")],
        cwd=base_path
    )
```

*YOLOv8 Full Detection (Sadece Algılama - Servo Yok):*

```
-def run_yolo_full_detection():
    stop_current()
    global current_process
    current_process = subprocess.Popen(
        ["python", os.path.join(base_path, "yolo_full_detection.py")],
        cwd=base_path
    )
```

**Açıklama:**

- Her mod, kendi dosyasını başlatır.
- Mod değiştirmek için eski çalışan mod durdurulur.
- Bu yapı sayesinde sistem stabil çalışır.

---

*Reset Fonksiyonu:*

```
def run_reset():
    stop_current()
    dataset_path = os.path.join(base_path, "dataset")
    trainer_path = os.path.join(base_path, "trainer.yml")
    label_path = os.path.join(base_path, "labels.txt")
    if os.path.exists(dataset_path):
        shutil.rmtree(dataset_path)
    if os.path.exists(trainer_path):
        os.remove(trainer_path)
    if os.path.exists(label_path):
        os.remove(label_path)
    os.makedirs(dataset_path, exist_ok=True)
    messagebox.showinfo("Reset", "Tüm veriler sıfırlandı.")
```

**Açıklama:**

- Sistemdeki tüm eğitim verileri ve modelleri temizler.
- Kullanıcının sistem üzerinde yeni bir eğitim sürecine başlamasını sağlar.
- Projeyi her zaman temiz ve sıfırdan tekrar geliştirebilme şansı verir.

---

*Kod Bloğu 6: Ana Arayüz Oluşturma*

```
root = tk.Tk()
root.title("Yüz & Nesne Takip Sistemi")
root.geometry("400x800")
```

**Açıklama:**

- Tkinter ana pencere oluşturuluyor.
- Başlık ve pencere boyutu ayarlanıyor.

---

*Kod Bloğu 7: Butonların Tanımlanması*

```
tk.Label(root, text="Kontrol Paneli", font=("Arial", 16, "bold")).pack(pady=20)

tk.Button(root, text="Collect", width=25, height=2, command=run_collect).pack(pady=5)
tk.Button(root, text="Train", width=25, height=2, command=run_train).pack(pady=5)
tk.Button(root, text="Detect (Yüz)", width=25, height=2, command=run_detect).pack(pady=5)
tk.Button(root, text="Tracking (Klasik)", width=25, height=2, command=run_tracking).pack(pady=5)
tk.Button(root, text="Color Tracking (Renk)", width=25, height=2, command=run_color_tracking).pack(pady=5)
tk.Button(root, text="YOLOv8 Object Tracking (Servo)", width=25, height=2, command=run_yolo_object_tracking).pack(pady=5)
tk.Button(root, text="YOLOv8 Full Detection (Görüntü)", width=25, height=2, command=run_yolo_full_detection).pack(pady=5)
tk.Button(root, text="Reset", width=25, height=2, command=run_reset).pack(pady=5)
```

**Açıklama:**

- Kullanıcının tüm modlara kolay erişimi için 8 adet büyük buton oluşturulmuştur.
- Tüm sistem tek arayüzden kontrol edilebilir.

---

*Kod Bloğu 8: Sonsuz Döngü (Ana GUI Çalıştırma)*

```
root.mainloop()
```

**Açıklama:**

- Tkinter GUI'nin sonsuz döngüsünü başlatır ve pencereyi sürekli açık tutar.

main.py

C: > Users > smfse > OneDrive > Masaüstü > project_folder > main.py

```python
import tkinter as tk
from tkinter import messagebox
import subprocess
import os
import shutil

base_path = r"C:\Users\smfse\OneDrive\Masaüstü\project_folder"
current_process = None

def stop_current():
    global current_process
    if current_process and current_process.poll() is None:
        current_process.terminate()
        current_process = None

def run_collect():
    stop_current()
    popup = tk.Toplevel(root)
    popup.title("Collect")
    popup.geometry("300x200")

    tk.Label(popup, text="İsim giriniz:").pack(pady=10)
    entry = tk.Entry(popup)
    entry.pack(pady=10)

    def start():
        name = entry.get().strip()
        if not name:
            messagebox.showwarning("Uyarı", "İsim giriniz.")
            return
        global current_process
        current_process = subprocess.Popen(
            ["python", os.path.join(base_path, "collect.py"), name],
            cwd=base_path
        )
        popup.destroy()

    tk.Button(popup, text="Başla", command=start).pack(pady=10)

def run_train():
    stop_current()
    global current_process
    current_process = subprocess.Popen(
        ["python", os.path.join(base_path, "train.py")],
        cwd=base_path
    )

def run_detect():
    stop_current()
    global current_process
    current_process = subprocess.Popen(
        ["python", os.path.join(base_path, "detect.py")],
        cwd=base_path
    )
```

*Şekil 15*

```
55
56  def run_tracking():
57      stop_current()
58      global current_process
59      current_process = subprocess.Popen(
60          ["python", os.path.join(base_path, "tracking.py")],
61          cwd=base_path
62      )
63
64  def run_color_tracking():
65      stop_current()
66      global current_process
67      current_process = subprocess.Popen(
68          ["python", os.path.join(base_path, "color_tracking.py")],
69          cwd=base_path
70      )
71
72  def run_yolo_object_tracking():
73      stop_current()
74      global current_process
75      current_process = subprocess.Popen(
76          ["python", os.path.join(base_path, "yolo_object_tracking.py")],
77          cwd=base_path
78      )
79
80  def run_yolo_full_detection():
81      stop_current()
82      global current_process
83      current_process = subprocess.Popen(
84          ["python", os.path.join(base_path, "yolo_full_detection.py")],
85          cwd=base_path
86      )
87
88  def run_reset():
89      stop_current()
90      dataset_path = os.path.join(base_path, "dataset")
91      trainer_path = os.path.join(base_path, "trainer.yml")
92      label_path = os.path.join(base_path, "labels.txt")
93      if os.path.exists(dataset_path):
94          shutil.rmtree(dataset_path)
95      if os.path.exists(trainer_path):
96          os.remove(trainer_path)
97      if os.path.exists(label_path):
98          os.remove(label_path)
99      os.makedirs(dataset_path, exist_ok=True)
100     messagebox.showinfo("Reset", "Tüm veriler s□f□rland□.")
101
```

*Şekil 16*

```
101
102 root = tk.Tk()
103 root.title("Yüz & Nesne Takip Sistemi")
104 root.geometry("400x800")
105
106 tk.Label(root, text="Kontrol Paneli", font=("Arial", 16, "bold")).pack(pady=20)
107
108 tk.Button(root, text="Collect", width=25, height=2, command=run_collect).pack(pady=5)
109 tk.Button(root, text="Train", width=25, height=2, command=run_train).pack(pady=5)
110 tk.Button(root, text="Detect (Yüz)", width=25, height=2, command=run_detect).pack(pady=5)
111 tk.Button(root, text="Tracking (Klasik)", width=25, height=2, command=run_tracking).pack(pady=5)
112 tk.Button(root, text="Color Tracking (Renk)", width=25, height=2, command=run_color_tracking).pack(pady=5)
113 tk.Button(root, text="YOLOv8 Object Tracking (Servo)", width=25, height=2, command=run_yolo_object_tracking).pack(pady=5)
114 tk.Button(root, text="YOLOv8 Full Detection (Görüntü)", width=25, height=2, command=run_yolo_full_detection).pack(pady=5)
115 tk.Button(root, text="Reset", width=25, height=2, command=run_reset).pack(pady=5)
116
117 root.mainloop()
118
```

*Şekil 17*

## III.3.2. Donanım Geliştirme Süreci

Donanım tasarımı yapılırken sistemin hem işlevsel hem de taşınabilir olması temel hedefler arasında belirlenmiştir. Projenin tüm hareketli aksamı ve kontrol sistemi mobil bir platform üzerinde çalıştırılabilir şekilde oluşturulmuştur.

İlk aşamada, sistemin hareket kabiliyetini sağlamak için şasi üzerine yerleştirilen **2 adet redüktörlü DC motor** kullanılmıştır. Bu motorlar hem düşük devirde yüksek tork sağlamış hem de sabit ve kararlı hareket sağlamıştır. Motorların sürülmesi için **L298N motor sürücü kartı** devreye alınmıştır. Arduino UNO'dan gelen düşük akımlı kontrol sinyalleri L298N sayesinde motorların ihtiyacı olan yüksek akım ve gerilime çevrilmiştir.

Yüz ve nesne takibi sırasında kameranın açısal hareketini sağlayabilmek amacıyla, üzerine **MG90S modelinden 2 adet servo motor** monte edilmiştir. Bu servo motorlar, X ve Y ekseninde hareket ettirilen özel olarak tasarlanmış servo standı üzerine yerleştirilmiştir. Bu sayede hem yatay hem de dikey eksende hassas şekilde kameranın konumu ayarlanmıştır.

Kamera sistemi olarak, projede düşük gecikmeli ve yüksek çözünürlüklü görüntü elde edebilmek amacıyla mobil cihaz kamerası tercih edilmiştir. Bilgisayara görüntünün aktarılması için kablosuz bağlantı üzerinden **Iriun Webcam uygulaması** kullanılmıştır. Iriun uygulaması, Wi-Fi üzerinden mobil cihaz kamerasını doğrudan bilgisayara aktarmakta ve OpenCV üzerinden doğrudan erişim sağlamaktadır. Mobil cihaz kamerasının tercih edilmesinin sebebi, özellikle düşük gecikmeli ve yüksek kare hızında görüntü sağlayabilmesidir. Telefon, servolar üzerine kurulan mekanizmanın üst kısmına sabitlenerek kameranın hareketli servo sistemiyle entegre edilmiştir.

Enerji kaynağı olarak sistemin taşınabilirliğini sağlamak amacıyla **4 adet 18650 lityum iyon batarya** kullanılmıştır. Bataryalar 2 seri ve 2 paralel bağlantı yapılarak toplamda **7.4V 2P2S konfigürasyonu** elde edilmiştir. Bu yapı hem gerilim hem de akım kapasitesi bakımından sistemi besleyebilecek kapasite sağlamıştır. Batarya gerilimleri sistem bileşenleri için uygun gerilim seviyelerine indirilebilmesi amacıyla **7805 voltaj regülatörü** devreye alınmıştır. Bu sayede Arduino ve diğer 5V çalışan bileşenler güvenli şekilde beslenmiştir.

Sistem içerisindeki tüm kablo karmaşasını ve devreyi daha kompakt hale getirebilmek için **delikli pertinaks plaka** üzerinde özel olarak tasarlanmış bir güç dağıtım ve regülasyon devresi oluşturulmuştur. Bu pertinaks kart üzerine tüm gerilim regülatörleri, motor sürücü bağlantıları, seri port ve servo çıkışları düzenli şekilde yerleştirilmiş ve lehimleme işlemleri gerçekleştirilmiştir. Böylece devre karmaşası büyük oranda azaltılmış, müdahale ve hata oranı minimize edilmiştir.

Bluetooth üzerinden haberleşme için sistemin ilerleyen aşamalarında denemeler yapılmış ve kablosuz senaryolar için **HC-05 Bluetooth modülü** ile Arduino arasında seri port haberleşmesi sağlanmıştır. Ancak kablolu bağlantının

stabilitesi daha yüksek olduğu için son versiyonda doğrudan USB üzerinden kablolu haberleşme tercih edilmiştir.

### Kullanılan Malzemeler Listesi

#### Elektronik Bileşenler:

1. Arduino UNO (1 adet)
2. L298N Motor Sürücü Modülü (1 adet)
3. HC-05 Bluetooth Modülü (1 adet) *(alternatif kablosuz haberleşme için)*
4. MG90S Servo Motor (2 adet)
5. Redüktörlü DC Motor (2 adet)
6. 7805 Voltaj Regülatörü (1 adet)
7. 18650 Li-ion Batarya (4 adet) (2 seri - 2 paralel bağlantılı)
8. Delikli Pertinaks Plaka (1 adet)
9. Çeşitli erkek-dişi jumper kablolar
10. Anahtar, sigorta ve butonlar (güç yönetimi için)

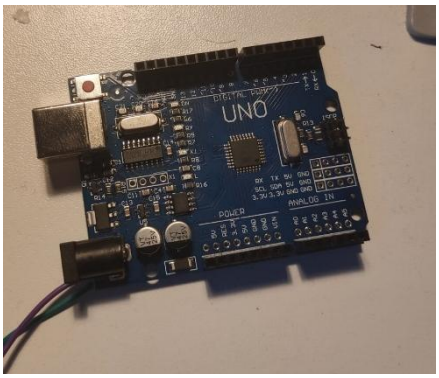
#### Mekanik Bileşenler:

1. Servo Hareketli Kamera Standı (X-Y eksen hareketi sağlayan özel mekanizma)
2. Araç Şasisi (mobil platform gövdesi)
3. Sarhoş (serbest) tekerlek (1 adet)
4. Dişli sistem ve motor bağlantı aparatları

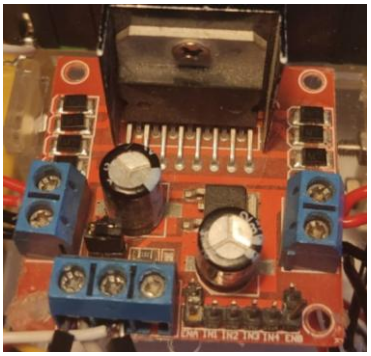
#### Kamera ve Görüntüleme:

1. Mobil Telefon Kamerası (Iriun Webcam uygulaması ile bilgisayara bağlanan)
2. Bilgisayar (Görüntü işleme ve arayüz yazılımını çalıştırmak için)

### III.3.2.1.Kullanılan Araçların Görselleri




Şekil 21

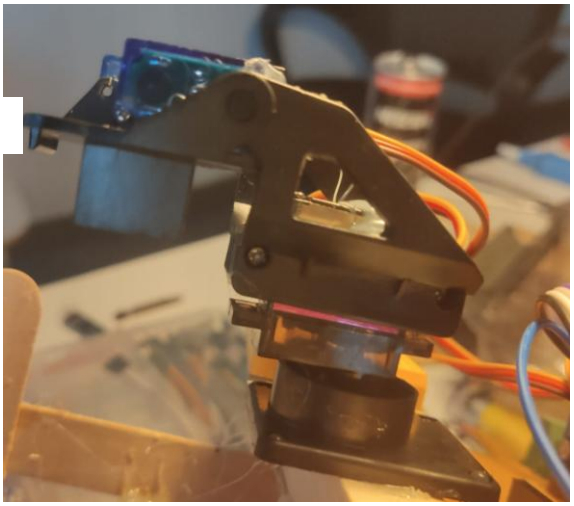


Şekil 18

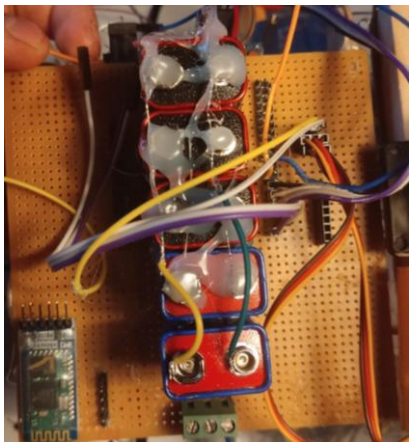
Şekil 22

Şekil 20

Şekil 19

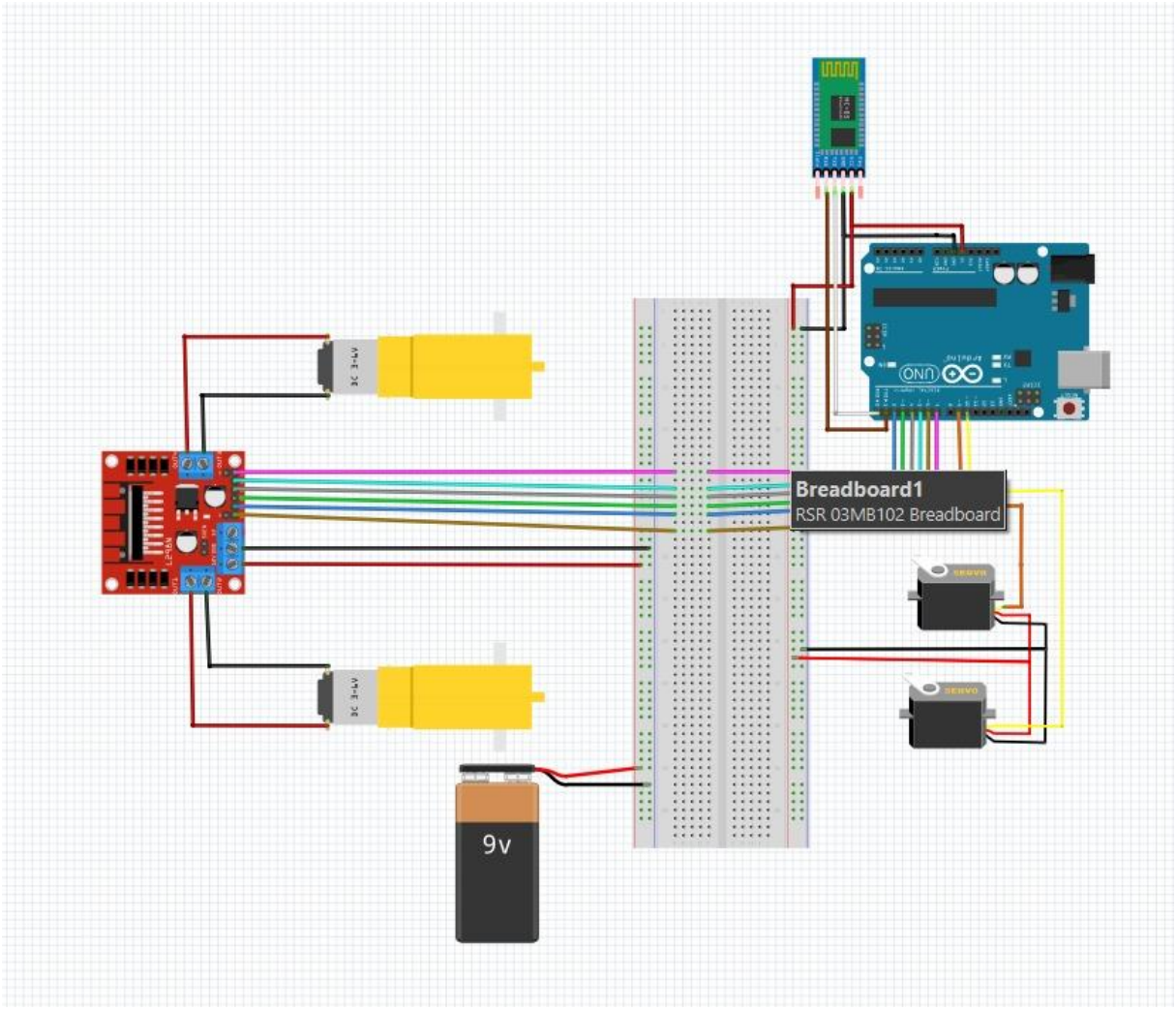


Şekil 23



Şekil 24

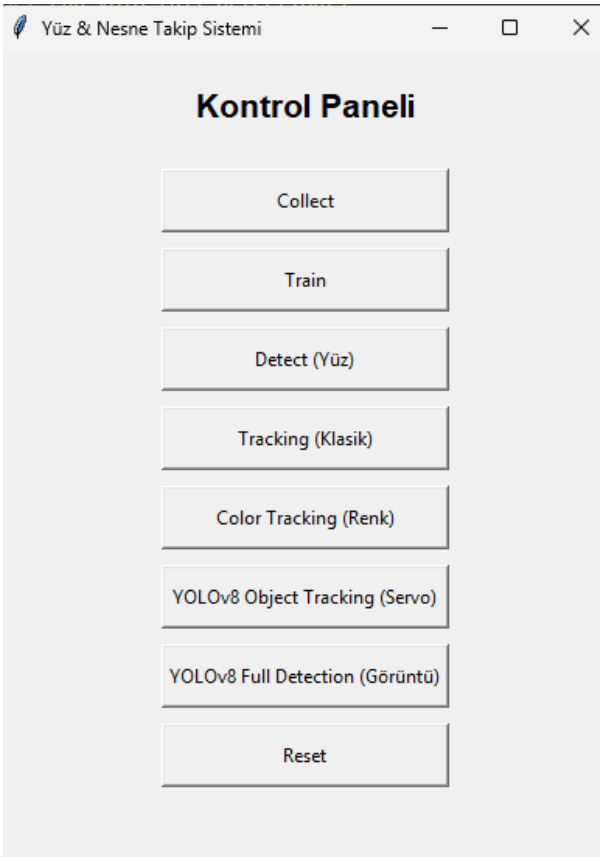
### III.3.2.2.Devre Şeması




*Şekil 25*

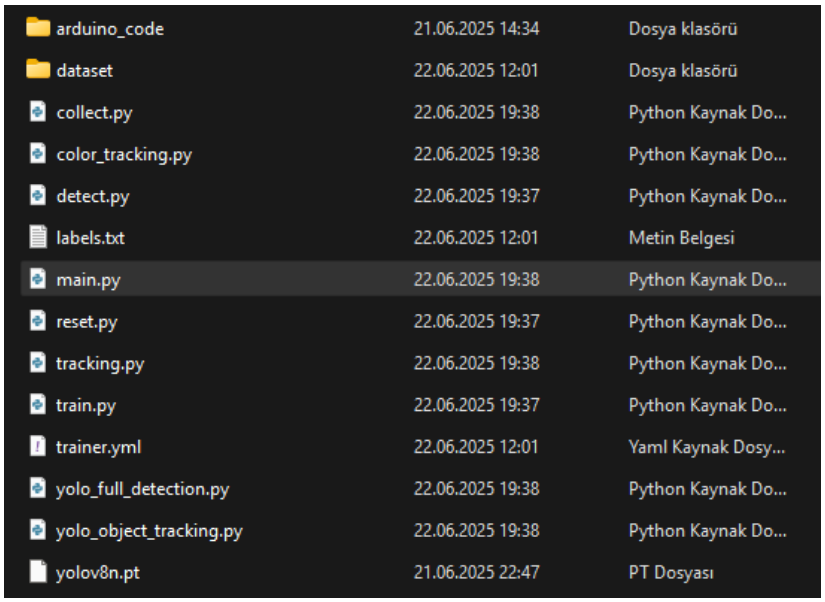
## III.3.3. Arayüz

*"Main.py arayüzü"*



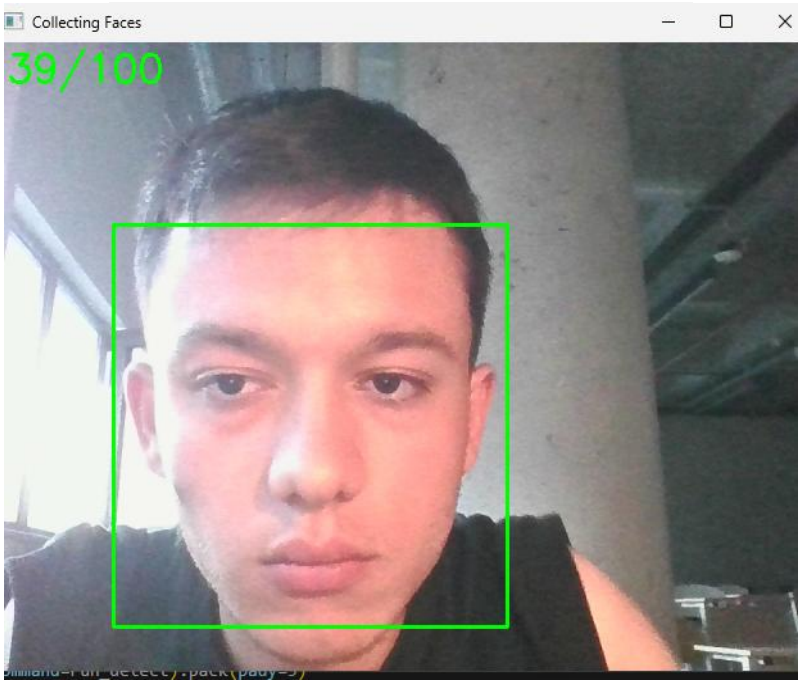

*Şekil 26*

*"project_folder"*



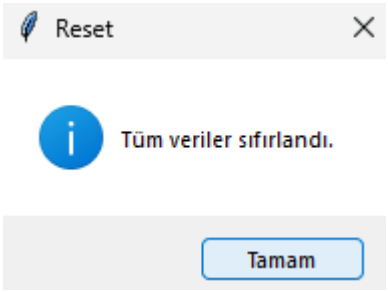

*Şekil 27*

*“ collect”*



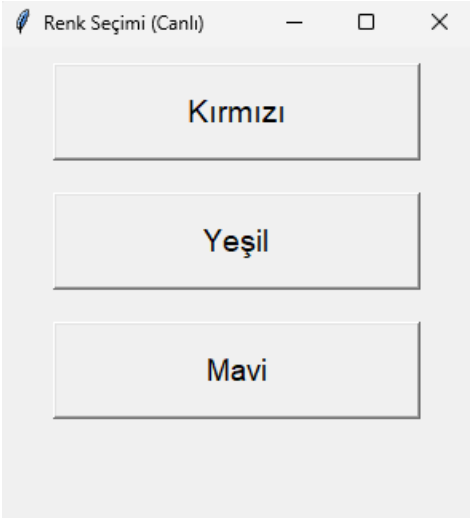

*“collect veri alma işlemi sırasından bir görüntü”*

“ reset”




*Şekil 28*


*Şekil 29*




*Şekil 30*

*"yolo"*

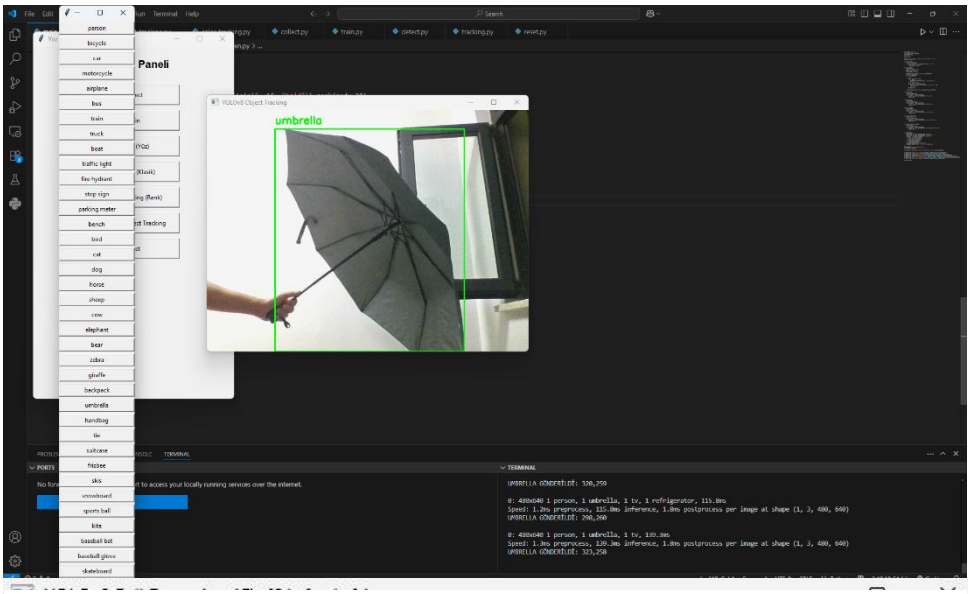

*Şekil 32*

*Şekil 33*

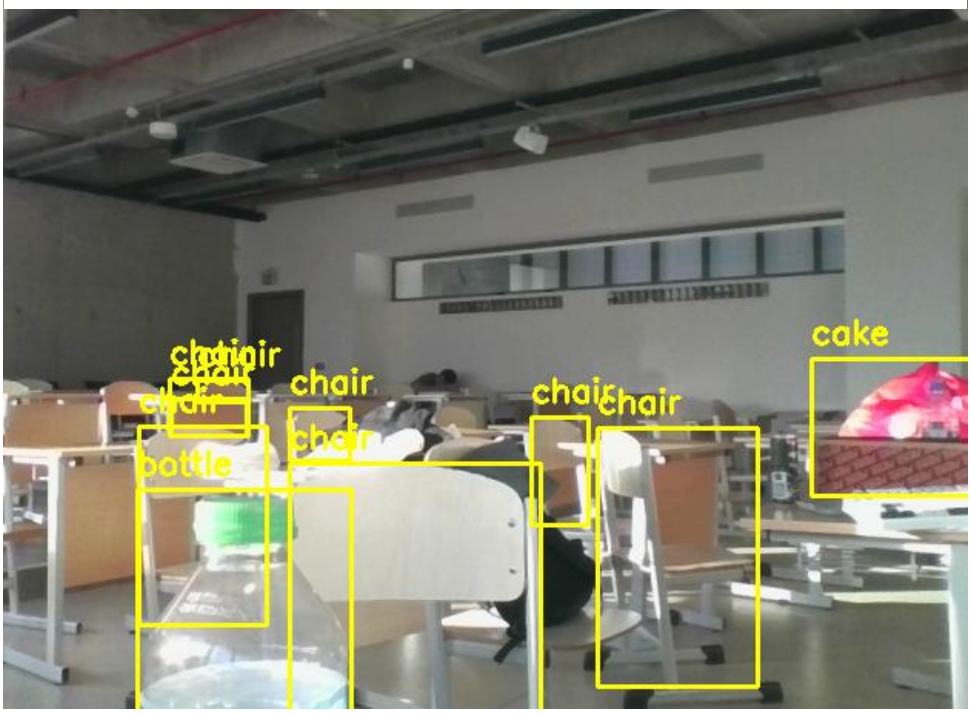
*"servo veri değerleri"*


*Şekil 34*

“yolo ile nesne tanımalar”




*Şekil 35*




*Şekil 36*

# BÖLÜM IV: SONUÇLAR VE DEĞERLENDİRME

## IV.1. Genel

Bu tez çalışmasında, görüntü işleme tabanlı, çok modlu (yüz tanıma, renk takibi, nesne tanıma ve takip, araç kontrolü) bir nesne takip sistemi başarıyla tasarlanmış ve uygulanabilirliği gösterilmiştir. Yapılan çalışmalar sonucunda elde edilen bulgular ve değerlendirmeler aşağıda sıralanmıştır:

### IV.1.1. Sistem Başarım ve Stabilitesi

- Gerçek zamanlı görüntü işleme temelinde geliştirilen sistem, farklı modlar arasında geçiş yapabilen esnek bir yapıya sahiptir. Özellikle seçilen kişi ya da nesneye kitlenip sadece o koordinatları takip etmesi sayesinde servo kontrolünde yüksek kararlılık sağlanmıştır.
- Çoklu thread yönetimi ve sistem kaynaklarının kontrollü kullanımıyla birlikte, arayüz donmaları ve FPS düşüşleri başarılı şekilde minimize edilmiştir. Sistemin stabil çalışabilmesi için arayüz ve görüntü işleme süreçlerinin dikkatlice ayrılması önemli rol oynamıştır.

### IV.1.2.Yüz Tanıma Modülü

- Klasik hazır derin öğrenme ağı kullanmak yerine, kendi oluşturduğumuz eğitim veri setleriyle LBPH algoritması üzerinden kişiye özgü yüz tanıma sistemi geliştirilmiştir.
- Eğitim sırasında veri çeşitliliği ve pozisyon farklılıkları dikkate alınarak overfitting problemleri önlenmiş ve modelin doğruluk oranı artırılmıştır.
- Seçim geçişlerinde çakışma yaşanmaması için 100 ms'lik geçici bekleme süresi çözüm getirmiş, sistemin kararlı çalışması sağlanmıştır.

### IV.1.3.Renk Takibi Modülü

- Renk takibinde, farklı ışık koşullarında dahi stabil çalışabilmesi için HSV renk aralığında toleranslı maskeleme uygulanmıştır.
- Renk seçimini kullanıcıya anlık değiştirme imkanı sağlayan dinamik arayüz ile pratik kullanım imkanı sunulmuştur.
- Renk bazlı takipte küçük lekeler ve parazitler kontur filtreleme yöntemiyle engellenmiştir.

### IV.1.4. YOLOv8 ile Nesne Algılama ve Takip

- YOLOv8 algoritmasının entegrasyonu sayesinde sistem sadece insan yüzü değil; nesne sınıflarını da tanıyabilme ve servo ile kilitlenebilme yeteneği kazanmıştır.
- Derin öğrenme tabanlı modelin hız ve doğruluk açısından oldukça başarılı sonuçlar verdiği gözlemlenmiştir.
- Ayrıca full detection modunda tüm sınıflar belirlenerek görsel testler gerçekleştirilmiştir.

### IV.1.5. Motor ve Araç Kontrol Entegrasyonu

- Servo hareketleri ile araç motor hareketlerinin birbirinden bağımsız çalıştırılması sistemin esnekliğini artırmıştır.
- Klavye kontrolü ile yönlendirme sağlanarak kullanıcı kontrollü sürüş testleri başarıyla uygulanmıştır.
- Servo ve motor komutlarının tek bir veri paketi içinde Arduino'ya iletilmesi sayesinde veri bütünlüğü korunmuştur.

### IV.1.6. Donanımın Tasarımı ve Taşınabilirliği

- Servo standının X ve Y eksen hareket kabiliyeti, mobil platform üzerine yerleştirilen kamera ile birlikte sistemin farklı açılardan çalışabilmesine olanak sağlamıştır.
- 4 adet 18650 pilin 2S2P bağlantısıyla sağlanan güç, sistemin uzun süre bağımsız ve taşınabilir şekilde çalışmasını sağlamıştır.
- Tüm devre elemanları için delikli pertinaks üzerinde özel bir dağıtım devresi hazırlanarak kablo karmaşası önlenmiştir.
- Telefon kamerası (Iriun uygulaması ile) bilgisayara bağlanarak hızlı ve pratik görüntü transferi sağlanmıştır.

### IV.1.7. Sistem Geliştirilebilirlik Potansiyeli

- Sistemde farklı modlar arasında geçiş yapabilmek için geliştirdiğimiz esnek modüler yapı, ileride yapılabilecek yeni algoritma entegrasyonlarına imkan sağlamaktadır.
- Sisteme eklenebilecek yeni derin öğrenme modelleri, MediaPipe destekli el-yüz hareket tanıma, sesli komut entegrasyonu gibi geliştirme potansiyelleri mevcuttur.
- Ayrıca sistemin baştan eğitim verilerinin toplanması ve yeniden eğitilebilmesi amacıyla "Reset" modülü eklenmiştir. Bu mod sayesinde istenirse tüm kayıtlı veriler temizlenerek sistem sıfırdan yeni kullanıcılarla yeniden eğitilebilir hale getirilmiştir.

## IV.2.Sonuç

Bu çalışma sonucunda; düşük maliyetli, taşınabilir, modüler ve çok amaçlı bir görüntü işleme tabanlı nesne ve yüz takip sistemi başarıyla gerçekleştirilmiş ve donanım-yazılım entegrasyonunun güçlü bir örneği ortaya konmuştur. Proje sürecinde karşılaşılan her teknik problem çözülerek sistem kararlı, güvenilir ve pratik bir yapıya kavuşturulmuştur.

# Kaynakça


AI, G. (2021). *MediaPipe*. mediapipe.dev: https://mediapipe.dev/ adresinden alındı

arduino.cc. (2020). *Arduino Uno Documentation*. arduino.cc: https://www.arduino.cc/ adresinden alındı

Faisal, M. M. (2023). *YOLOv8-DeepSORT-Object-Tracking*. GitHub: https://github.com/MuhammadMoinFaisal/YOLOv8-DeepSORT-Object-Tracking adresinden alındı

Geitgey, A. (2017). *face_recognition*. GitHub: https://github.com/ageitgey/face_recognition adresinden alındı

OpenCV.org. (2020). *OpenCV*. opencv.org: https://opencv.org/ adresinden alındı

Python.org. (2020). *Tkinter*. python.org: https://docs.python.org/3/library/tkinter.html adresinden alındı

THU-MIG. (2024). *YOLOv10*. GitHub: https://github.com/THU-MIG/yolov10 adresinden alındı

Ultralytics. (2023). *Ultralytics YOLOv8*. GitHub: https://github.com/ultralytics/ultralytics adresinden alındı