TEHDİT DEĞERLENDİREN OTONOM ARAÇ ENTEGRESİ

Muhammet ÖZMEN

212523019

4. Sınıf

Bilgisayar Mühendisliği

İSKENDERUN TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ

ŞUBAT 2025

TEHDİT DEĞERLENDİREN OTONOM ARAÇ ENTEGRESİ
(Lisans Bitirme Projesi Ön Raporu)

Muhammet ÖZMEN

İSKENDERUN TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ FAKÜLTESİ

EKİM 2024

ÖZET

Teknolojinin gelişmesi ile ortaya çıkan daha güçlü mini bilgisayarlar ve daha optimize algoritmalar ile yazılımın kompleks yapıları gittikçe robotik alanında yaygınlaşmaktadır. Bu sayede insanlık olarak bilgisayarların kompleks iş yapabilme yetisini hayatın her alanında kullanabilecek hale geldik.

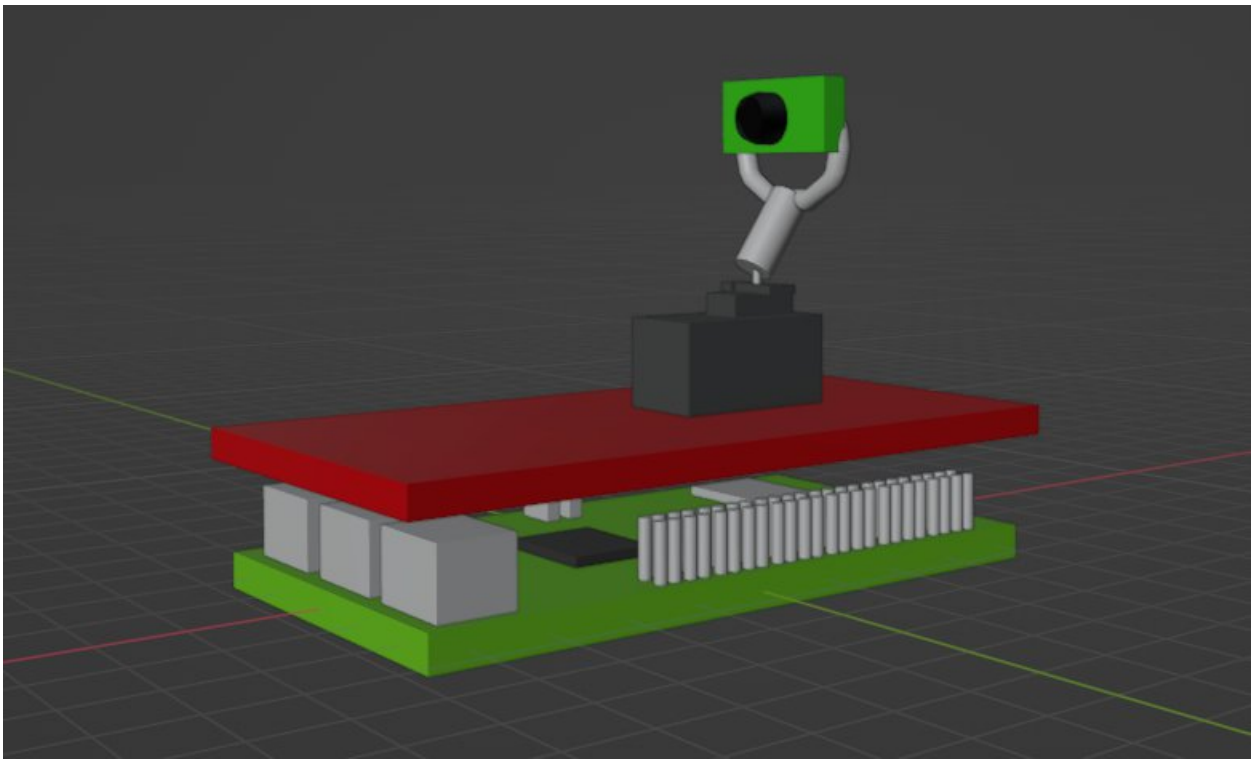
Bu proje dahilinde hareket halinde olanlar odak noktamız olmak üzere dronelar, İHA'lar gibi çeşitli otonom cihazların kullanabileceği bir gömülü (embedded) entegre sistemi hazırlanacaktır. Bu gömülü sistem sayesinde insan odaklı tehditler değerlendirilip takılı olduğu otonom cihazın erken tepki verilmesi sağlanabilecek ve ana bilgisayara bilgilendirme verilip kontrolü sağlanılabilecektir.

Proje için bir arduino kullanılmış olup bilgisayarla elektronik iletişim port üzerinden sağlanmıştır. Bunun yanında bazı devre elemanlarıyla çevresel etkileşim gücü ve okunabilirlik arttırılmıştır..

Anahtar Kelimeler : Entegre devre, gömülü sistem, görüntü işleme, Arduino
Sayfa Adedi : 25
Ders Veren : DOÇ.DR. Gökhan ALTAN

İÇİNDEKİLER

I. GENEL ÇALIŞMA PRENSİBİ

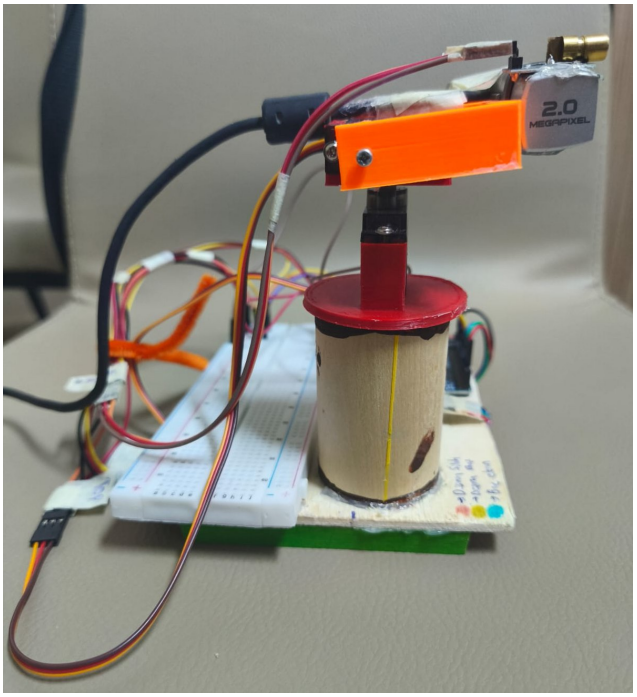


Görsel 1.1: Projenin ilk Blender ile konsept 3D çizimi

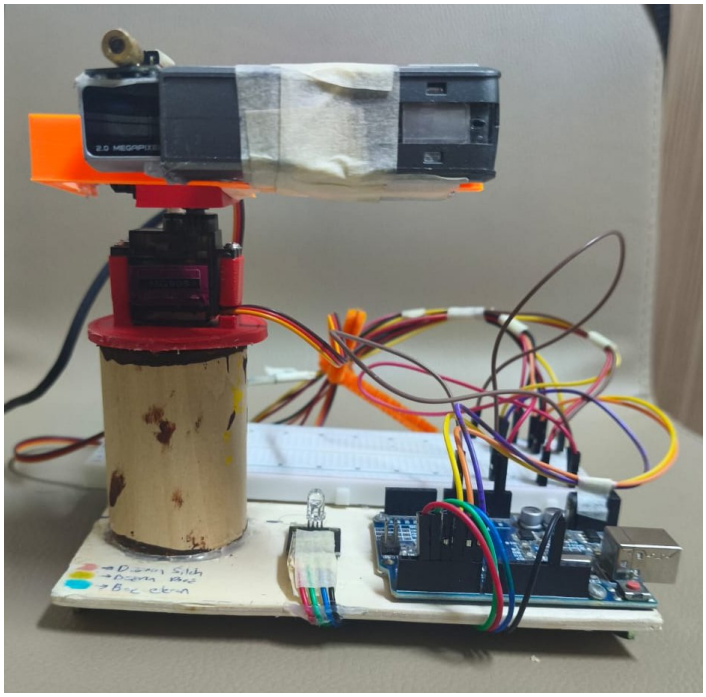
Proje bir entegre gömülü sistemdir. Şehir içinde kullanım amacıyla tasarlanan bu cihaz çeşitli robotik cihazlara takılması amacıyla üretilmiştir. Kamera modülünün gimbal ile dengelenmesi sayesinde drone gibi hava araçlarında, tekerlekli ve tekerleksiz kara araçlarında çalışabilmeye uygun tasarlanmıştır..

Sistem içerisindeki yazılımın çalışma mantığı, görüntü işleme ile algılanan şahsın elindeki silahın tehdit seviyesini değerlendirip bunu ana bilgisayara göndermektir. Bunun yanında hedef belli bir tehdit seviyesinde ise bu hedefi işaretleyecek ve servo motoru yardımıyla kamerasını hedefe dönecektir. Böylece veri besleme konusunda bir kopukluk olmayacaktır. Arayüz üzerinden çeşitli kontroller sağlanabilecektir. Bunlar da dahil diğer tüm detaylar sonraki alt başlıklarda detaylandırılmıştır.

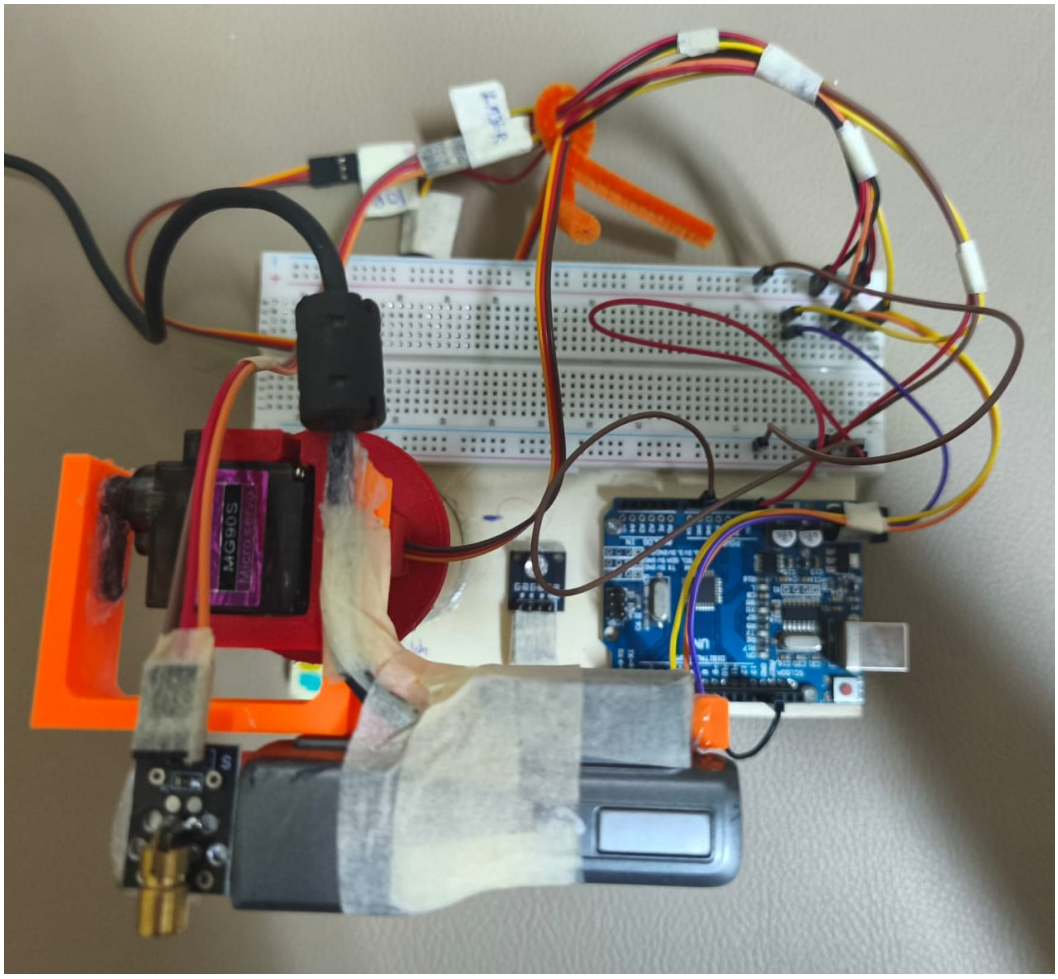
Bu entegre sistemi takıldığı cihaza tehdit seviyelerini bildirdiğinden dolayı şehir içi güvenlik drone'larında kullanılabilir ve polis drone'u olarak görev yapabilir. Ya da çeşitli şehir içinde gezen robotların hareket halindeyken anlık tehdidi kullanıcıya bildirmesini, böylece robotun kontrolü sırasında kamera kontrolünü otonomlaştırmayı sağlayabilir.



Görsel 1.2: Projenin yandan çekilmiş görseli



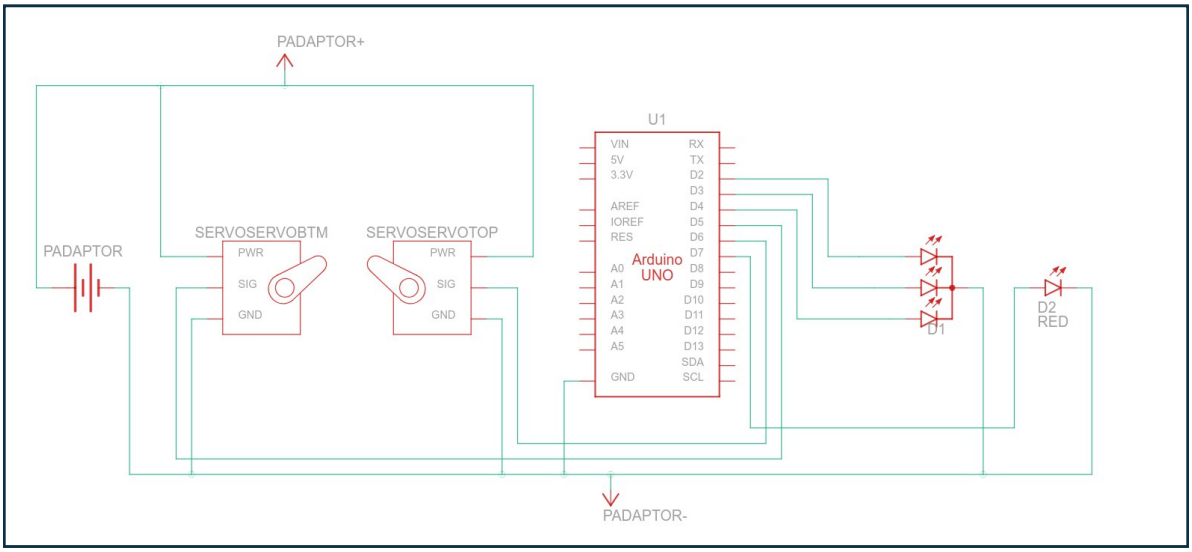
Görsel 1.3: Projenin önden çekilmiş görseli



Görsel 1.4: Projenin üstten çekilmiş görseli

I.a. DONANIMSAL DETAYLAR,

Projede donanımsal olarak kullanılacak elemanlar aşağıda detaylandırılmakla beraber proje halihazırda bir devre şemasına sahiptir. Devre elemanları belge içinde sonrasında detaylandırılmıştır.



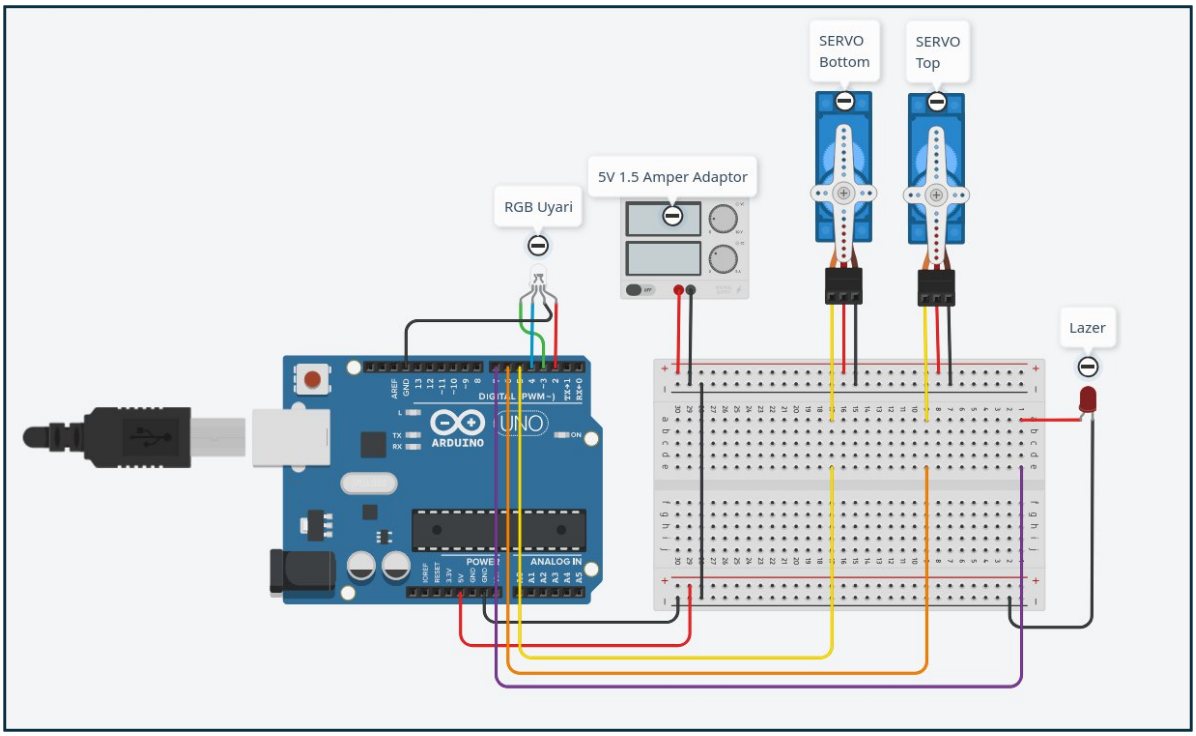
Görsel 1.5: Projenin devre şematiği.

Devre şematiği hakkında:

Proje devresi için aşağıdaki elemanlar kullanılmıştır. Tüm elemanların tercih sebebi detaylandırılmıştır.

- Arduino UNO:
  - Pin sayısının fazlalığı sayesinde geliştirme kiti olarak uygun bir mikrodenetleyicidir. İşlemler bilgisayar üzerinden olduğundan dolayı kendi işlemcisi içerisinde dönen algoritma için yeterli işlemci gücüne sahiptir.
- 2 tane MG90S :
  - 1.5 kg torka kadar ağırlık taşıyor olması, düşük enerji tüketimi ve diğer servolardan farklı olarak çelik dişlilere sahip olması sebebiyle ideal bir servodur.
- Lazer Modülü:
  - 5V ile çalışan güçlü bir lazer modülü tercih edilmiştir.
- RGB Led Modülü:
  - Düşük enerji tüketen ve hazır devre üzerinde bulunan bir RGB led seçilmiştir.
- 5V Adaptör Besleme:
  - Devredeki lazer ve led modülleri arduino tarafından besleniyor olsa da stabilite için iki servo ayrı olarak 5V 1.5 Amper bir adaptörden beslenmektedir. Aksi durumda devre düzgün çalışmayabilir.

Devreden bağımsız bir şekilde bilgisayar USB ile bağlı bir 2 MP webcam kamera mevcuttur. Bu kameranın kullandığı enerji veya verdiği verinin doğrudan servoyla bir ilişiği olmamakla beraber iki durum için de bilgisayarı kullanır.



Görsel 1.6: Proje devresi tinkercad üzeri tasarlanmış görseli

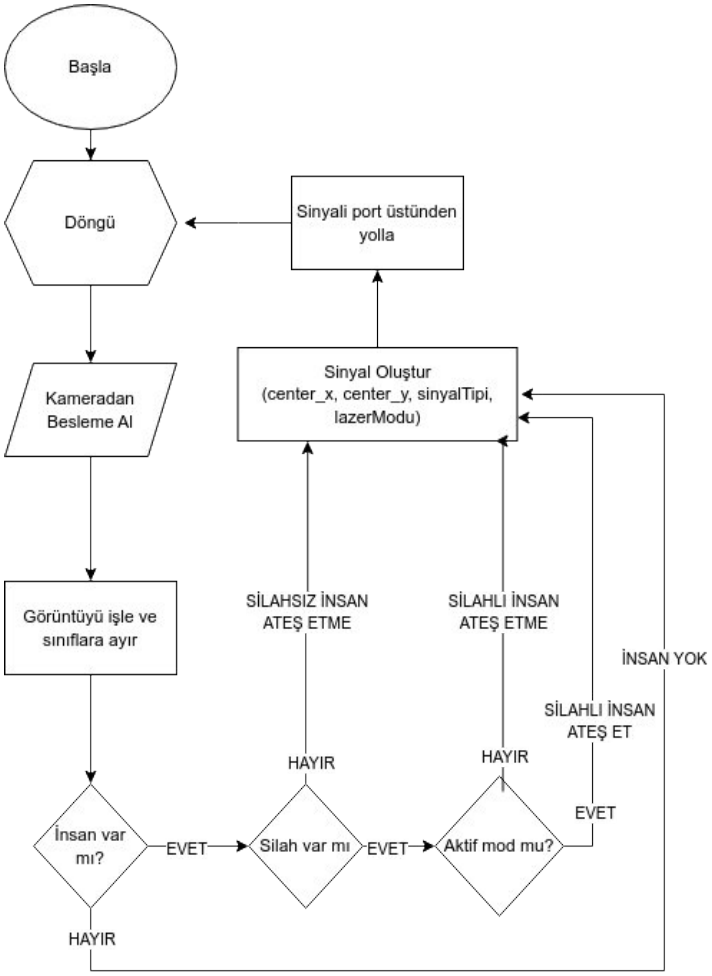
Turet gimball için üç farklı parça basılmış olup, ilk iki parça (kırmızı parça) sağlamlık ihtiyacından ötürü ABS ile basılmış olup, son parça (turuncu parça) hafiflik gözetildiğinden PLA ile basılmıştır. Parça kamerayla uyumlu olacak şekilde güçlendirilmiş tasarlanmış ve 3D yazıcı kullanılarak basılmıştır. Devrenin diğer iskelet parçaları (zemin, yükseltme silindiri vs) için ahşap kullanılmıştır.

I.b. YAZILIMSAL DETAYLAR

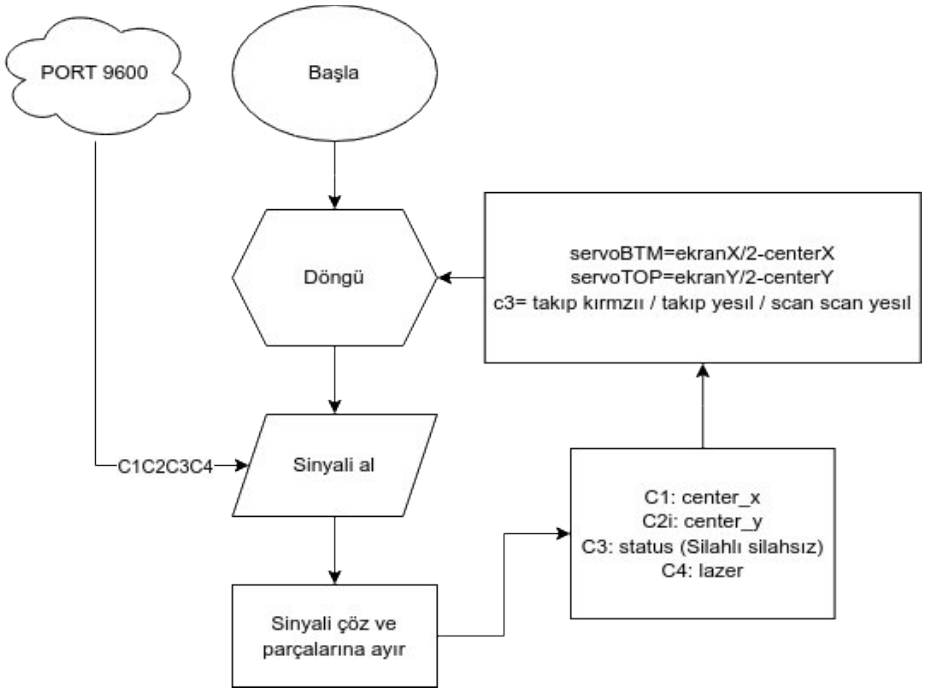
Proje genelinde iki yazılım çalışacaktır. Birinci yazılım bilgisayar içerisinde çalışan ve kullanıcı tarafından kontrolü sağlayan "Application" adlı yazılım; ikinci yazılım ise bilgisayar içerisinde çalışacak "Turet" adlı yazılımdır. Bu isimler karışıklık olmaması ve konunun daha rahat anlaşılabilmesi için açıklanmıştır.

Model Yazılımı:

Model yazılımı Python'la yazılmış bir yazılımdır. Webcamden görüntüyü alır ve YOLOV8 ile işler. Proje başında YOLO11 ile çalışmaya başlanmış ama verimli sonuçlar alınmadığından dolayı YOLOv8 modeline geçmiştir.
Model insan ve silahları tanıyacak şekilde eğitilmiştir. Böylece kişinin silahlı olup olmadığını tespit edebilmektedir.



Görsel 1.7: Proje içindeki bilgisayarda gerçekleşen Application yazılımı akış şeması



Görsel 1.8: Proje içindeki denetleyici içinde gerçekleşen Turet yazılımı akış şeması

Application Kod Blokları ve Açıklamalar:

Arduino ile bağlantı kurulması ve veri iletiminin sağlanması için geliştirilen mesajlaşma fonksiyonu.

```
# Arduino Bağlantısı
arduino_port = "/dev/ttyUSB0"
baud_rate = 9600
try:
    arduino = serial.Serial(arduino_port, baud_rate)
except serial.SerialException as e:
    print(f"Arduino bağlantı hatası: {e}")
    sys.exit(1)

def send_command(center_x, center_y, c3, c4):  # Arduino için mesaj gönderimi
    # Komut c1c2c3c4
    command = f"{center_x:03}{center_y:03}{c3}{c4}\n"
    print(command)
    arduino.write(command.encode())
```

YOLOv8 modelinde kullanılan parametreler ve değişkenler.

```
# YOLO değişken konfigleri
PATH_TO_MODEL = "silahli_yakin.pt"  # model path
DEVICE = "cuda"  # ekran kartı
CONFIDENCE_THRESHOLD = 0.6  # confidence toleransı
IOU_THRESHOLD = 0  # kutu iç içe girme toleransı
DISTANCE_THRESHOLD = 100  # uzaklık toleransı
X_MINUTES = 0.5  # bekleme toleransı
MAX_OBJECTS_PER_CLASS = {0: 1, 1: 1}  # sınıf başı obje sayısı
```

Silah ve insan tespitinde olası gizlenme veya görüntüleme sorunlarına karşı zamanlayıcı ve tolerans ayarı mekanizması.

```
# Silahı unutmak için
GUN_WAIT = 10  # Silah görünce ne kadar beklemesi gerektiği
MAN_WAIT = 1 # Adam gorunce ne kadar beklemesi gerektigi
def myTimer(seconds):
    return getTime() + seconds

def getTime():
    return int(time.time())
```

PySide6 ile geliştirilen uygulama arayüzü.

```
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.useLaser = 'N' # Lazer başta kapalı
        self.setWindowTitle("YOLOv8 İnsan Tespiti")
        self.setGeometry(100, 100, 500, 300)
        # Widgets
        self.graphics_view = QGraphicsView(self)
        self.scene = QGraphicsScene(self)
        self.graphics_view.setScene(self.scene)
        self.graphics_view.setFixedSize(640, 480)
        self.label_header1 = QLabel("Kişinin Merkez X : Y Konumu:", self)
        self.label_xy = QLabel("X : Y", self)
        self.label_header2 = QLabel("Kişinin Tehlike Durumu:", self)
        self.label_status = QLabel("Belirsiz", self)
        # Bu kısımdan sonrası layout ile alakalı olmakla beraber dökümanlanmamıştır.
```

Aktif ve pasif arama butonlarının işlevleri.

```
    def pasif_arama(self):
        self.useLaser = 'N'
        self.btn_passive.setStyleSheet("color: white; background-color: #F44336; padding: 10px; border-
radius: 5px;")
        self.btn_active.setStyleSheet("color: white; background-color: gray; padding: 10px; border-radius:
5px; opacity: 0.5;")


    def aktif_arama(self):
        self.useLaser = 'Y'
        self.btn_passive.setStyleSheet("color: white; background-color: gray; padding: 10px; border-
radius: 5px; opacity: 0.5;")
        self.btn_active.setStyleSheet("color: white; background-color: #4CAF50; padding: 10px; border-
radius: 5px;")
```

Model, verilen girdilere göre ayarlanarak sonuçlar için koordinatların elde edilmesi sağlanır.

```
def update_frame(self):
        ret, frame = self.cap.read()
        if ret:
            # YOLOv8 ile insan tespiti
            results = self.model(frame, conf=CONFIDENCE_THRESHOLD, iou=IOU_THRESHOLD,
device=DEVICE)


            # Sınıfların x ve y konumları
            self.box_man = None
            self.box_gun = None
            # Sonuçları ayarlama
            for result in results:
                for box in result.boxes:
                    x1, y1, x2, y2 = map(int, box.xyxy[0])  # Koordinatları al
                    class_id = int(box.cls[0])  # Sınıf kimliği al
                    conf = box.conf[0]  # Güven skoru
```

Merkez noktaları, ilgili sınıfa göre kaydedilir. Bu sayede, kişi silahlı olsa bile, kişinin merkez noktaları gönderilir. Ardından, her sınıf için ilgili kutuya alma işlemi gerçekleştirilir.

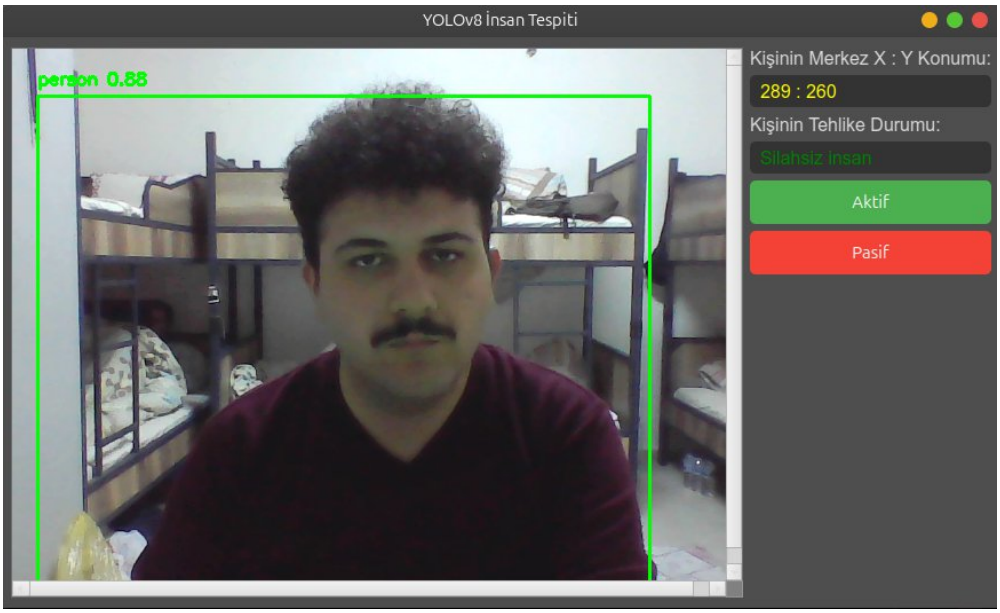
```
# Merkezleri sınıfa göre kaydetme
                    if self.box_man is not None: # son gozuken insan
                        self.last_box_man= self.box_man
                    if class_id == 0:
                        # Arduino ile iletişim için merkezin hesaplanması
                        self.center_x = (x1 + x2) // 2
                        self.center_y = (y1 + y2) // 2
                        self.box_man = (self.center_x, self.center_y)
                        self.label_xy.setText(f"{self.center_x} : {self.center_y}")  # Label güncelle
                    elif class_id == 1:
                        # Arduino ile iletişim için merkezin hesaplanması
                        self.center_x_gun = (x1 + x2) // 2
                        self.center_y_gun = (y1 + y2) // 2
                        self.box_gun = (self.center_x_gun, self.center_y_gun)

                    label = f"{self.model.names[class_id]} {conf:.2f}"

# İnsan için yeşil, silah için kırmızı kutu çizimi
                    if class_id == 0:
                        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
                        cv2.putText(frame, label, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX,
                                    0.5, (0, 255, 0), 2)
                    else:
                        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 2)
                        cv2.putText(frame, label, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX,
                                    0.5, (0, 0, 255), 2)
```

Kamerada sadece kişi tespit edilirse "silahsız insan", kişi ve silah tespit edilirse "silahlı insan", hiçbir şey tespit edilmezse "hiç kimse yok" sinyali oluşturulur. Silahlı insan tespiti durumunda, belirli bir süre boyunca (bu durumda 10 saniye) kişinin silahlı olduğu varsayılır. Benzer şekilde, silahsız insan tespitinde de yine 10 saniye süreyle kameranın tepkisiz kalması sağlanarak insan tespiti doğrulanır. Bu yöntem, silahın ve insanın saklanma durumunun önüne geçilmesine yardımcı olur.

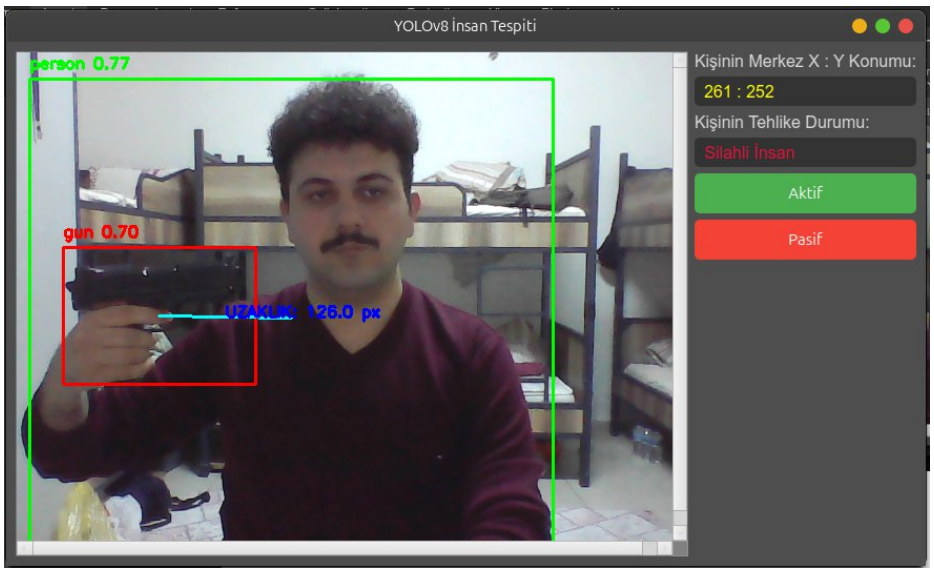
```
currTime = getTime()  # Zaman kontrolü
        if currTime >= self.timer_gun: # Silahli gozuken kisi uzerinden x saniye gecti mi
            if self.box_man and not self.box_gun:
                print("Silahsiz insan")
                self.label_status.setText("Silahsiz insan")
                self.label_status.setStyleSheet("color: green; background-color: #333; padding: 5px;
border-radius: 5px;")
                send_command(self.center_x, self.center_y, 'A', 'N')
                self.timer_man = myTimer(MAN_WAIT)
            elif self.box_gun and self.box_man:
                print("Silahli insan")
                self.label_status.setText("Silahli insan")
                self.label_status.setStyleSheet("color: red; background-color: #333; padding: 5px; border-
radius: 5px;")
                send_command(self.center_x, self.center_y, 'B', self.useLaser)
                self.timer_man = myTimer(MAN_WAIT)
                self.timer_gun = myTimer(GUN_WAIT)
            else:
                if currTime>= self.timer_man:
                    print("Hiçbir şey yok")
                    self.label_status.setText("Bir şey bulunamadı")
                    self.label_xy.setText(" ? : ?")
                    self.label_status.setStyleSheet("color: blue; background-color: #333; padding: 5px;
border-radius: 5px;")
                    send_command(0, 0, 'C', 'N')
                else:
                    print("Insan olabilir bekleyelim")
                    self.label_status.setText("Insan gorulmus olabilir!")
                    self.label_status.setStyleSheet("color: Orange; background-color: #333; padding: 5px;
border-radius: 5px;")
                    send_command(self.center_x, self.center_y, 'C', 'N')
        else: # Gecmediyse insan hala silahli varsay
            self.label_status.setText("Silahli İnsan")
            self.label_status.setStyleSheet("color: crimson; background-color: #333; padding: 5px;
border-radius: 5px;")
            send_command(self.last_box_man[0], self.last_box_man[1], 'B', self.useLaser)
```

Uzaklık hesaplama ve PySide6 `QGraphicsView` üzerinde OpenCV frame'inin dinamik olarak yansıtılması işlemi.

```
# Uzaklık hesaplama
        if self.box_man and self.box_gun:
            distance = math.hypot(self.box_man[0] - self.box_gun[0], self.box_man[1] - self.box_gun[1])
            cv2.line(frame, self.box_man, self.box_gun, (255, 255, 0), 2)
            mid_point = ((self.box_man[0] + self.box_gun[0]) // 2, (self.box_man[1] + self.box_gun[1]) //2)
            cv2.putText(frame, f"UZAKLIK: {distance:.1f} px", mid_point,
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)

        # OpenCV görüntüsünü QImage'e çevirme
        rgb_image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        h, w, ch = rgb_image.shape
        bytes_per_line = ch * w
        convert_to_Qt_format = QImage(rgb_image.data, w, h, bytes_per_line,
QImage.Format_RGB888)


        # QPixmap oluşturup QGraphicsScene'e ekleme
        pixmap = QPixmap.fromImage(convert_to_Qt_format)
        self.scene.clear()
        self.scene.addPixmap(pixmap)
```



Görsel 1.9: Proje application arayüzü silahsız insan tespiti



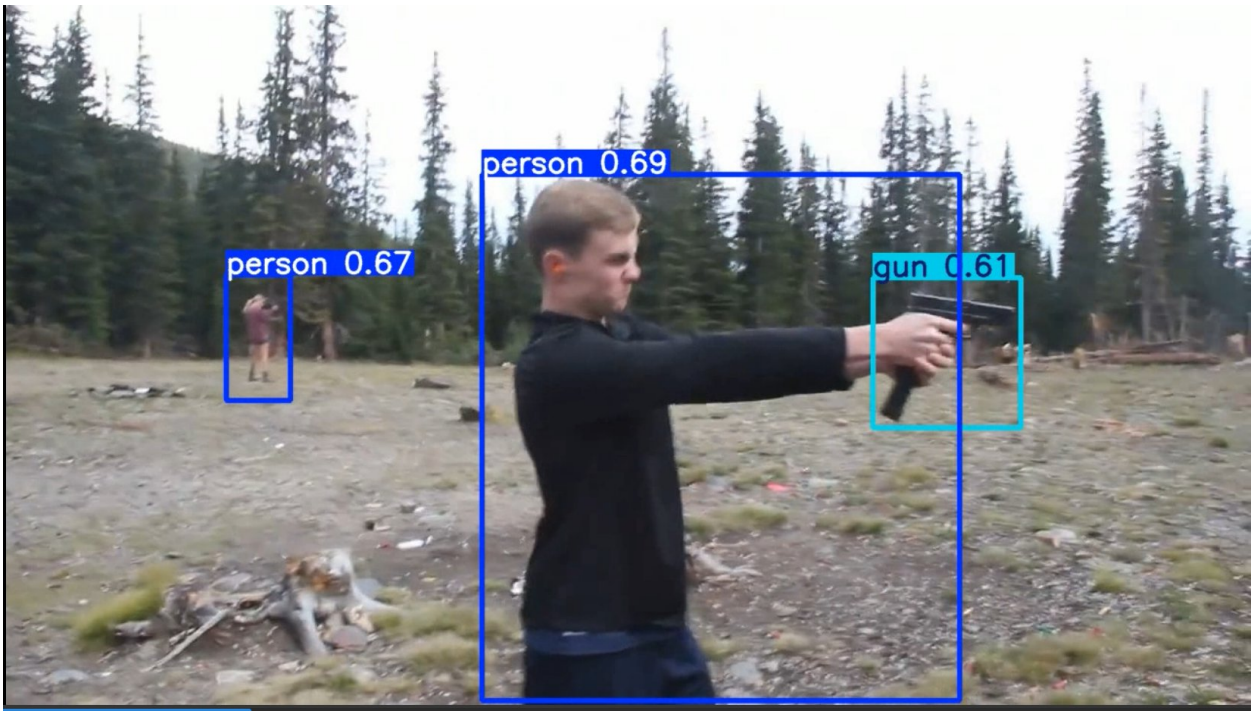
Görsel 1.10: Proje application arayüzü silahlı insan tespiti

YOLOv8 Modeli ve Açıklamaları:

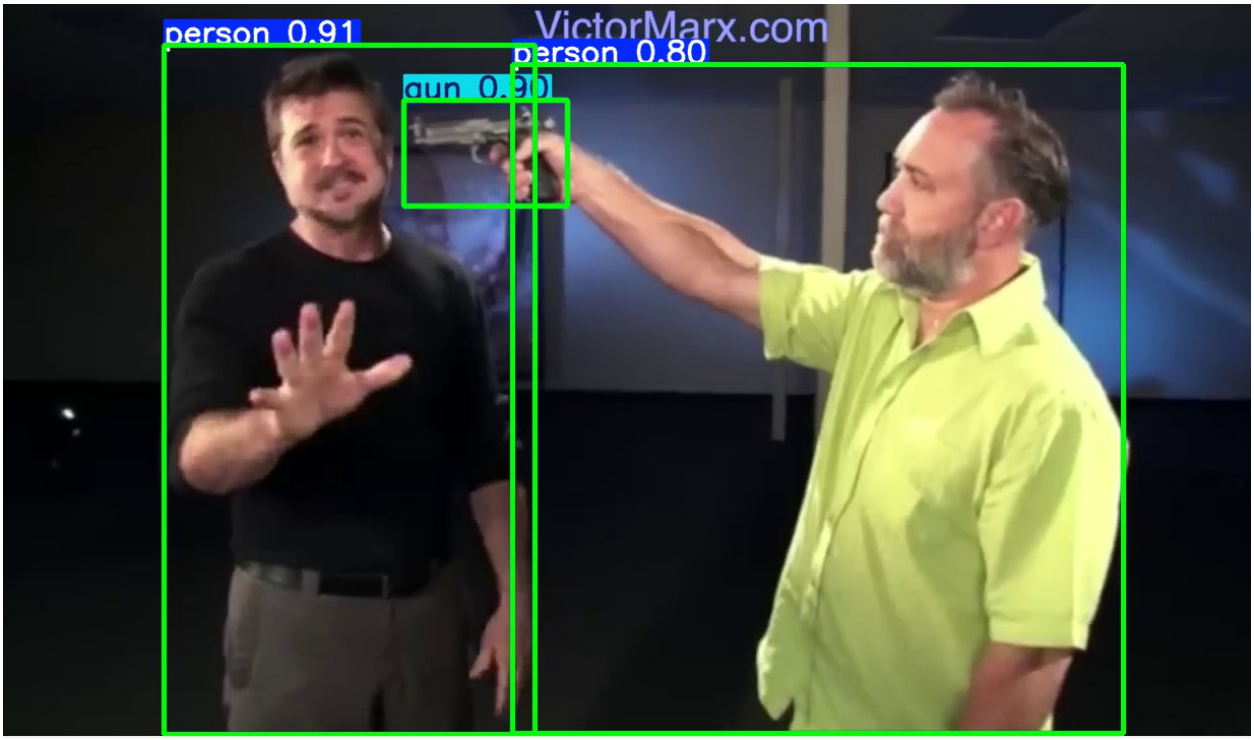
Model, halihazırda bulunan YOLOv8 modeli üzerine eğitim alarak geliştirilmiştir. Bu modelin amacı, 1-2 metreden daha kısa uzaklık ve belirli yüksekliklerde irtifa tespiti yapabilmektir. Eğitim süreci, özellikle yakın çekim görüntüleri kullanılarak gerçekleştirilmiştir. Parametreler şu şekildedir: 64 Epoch, üstüne train edilen model yolov8l.pt, batch 8.

Görsel 1.11: Batch 2 için etiketlemeler

Görsel 1.12: Batch 2 için predictionlar



Görsel 1.13: Modelin test edildiği bir videodan ekran görüntüsü



Görsel 1.14: Modelin test edildiği bir videodan ekran görüntüsü

Görsel 1.15: Modelin test edildiği bir videodan ekran görüntüsü

Turet Kod Blokları ve Açıklamalar:

Pin ve değişken tanımlamaları ile birlikte setup kısmı belirlenir.

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

// Devre pin girisleri
int ledR = 2;   // Kırmızı LED pini
int ledG = 3;   // Yeşil LED pini
int ledB = 4;   // Mavi LED pini
int pinBTM = 5; // Yatay servo pini
int pinTOP = 6; // Dikey servo pini
int lazer = 7;  // Lazer pini

Servo servoBTM;
Servo servoTOP;

// Sifreleme degiskenleri
int c1;
int c2;
char c3;
char c4='N';

// Hareket degiskenleri
int displayCenterX= 320;
int displayCenterY= 240;

double tolerans = 5;
double hareket = 2;
bool isToLeft=true; // Sag sol donmesi, basta sag

// Servo son konumlari
int sonBTM=90; //Son TOP servo konumu
int sonTOP=45; //Son BTM servo konumu

void setup() {
   servoBTM.attach(pinBTM);
   servoBTM.write(sonBTM); /// Baslangic BTM konumum
   servoTOP.attach(pinTOP);
   servoTOP.write(sonTOP); // Baslangic TOP konumum


   pinMode(ledR, OUTPUT);
   pinMode(ledG, OUTPUT);
   pinMode(ledB, OUTPUT);
   pinMode(lazer, OUTPUT);


   Serial.begin(9600); // Iletisim
}
```

Pin ve değişken tanımlamaları ile birlikte setup kısmı belirlenir. Döngü işlemleri ve sinyallere dayalı hareket komutları verilir. Arayüzden alınan pasif ve aktif komutlara göre lazer sinyali yönetilir, model çıktısındaki silah verilerine göre ise LED ve turet hareketi kararları alınır.

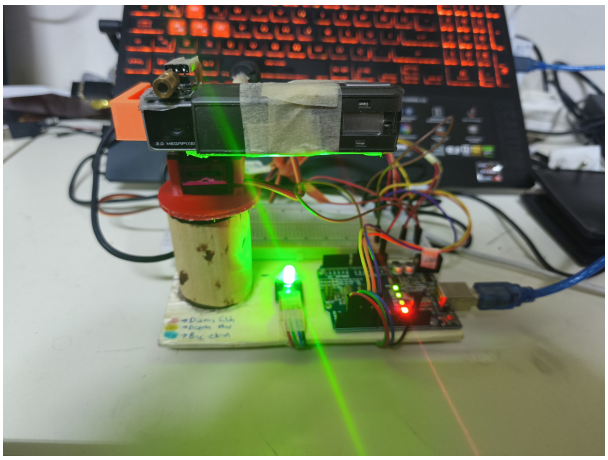
```
void loop() {

  // Seri porttan veri bekleniyor
  if (Serial.available() > 0) {
    String data = Serial.readStringUntil('\n');  // Satır sonuna kadar oku
    parseData(data);  // Gelen veriyi işle

  }

  if(c4=='Y'){
    digitalWrite(lazer, HIGH);
  }else if(c4=='N'){
    digitalWrite(lazer, LOW);
  }

  if(c3=='A'){
      digitalWrite(ledR, LOW);
      digitalWrite(ledG, HIGH);
      digitalWrite(ledB, LOW);
      trackObjectX();
      trackObjectY();
    }else if(c3=='B'){
      digitalWrite(ledR, HIGH);
      digitalWrite(ledG, LOW);
      digitalWrite(ledB, LOW);
      trackObjectX();
      trackObjectY();
    }else if(c3=='C'){
      digitalWrite(ledR, LOW);
      digitalWrite(ledG, LOW);
      digitalWrite(ledB, HIGH);
      areaScan();
    }else{
      digitalWrite(ledR, LOW);
      digitalWrite(ledG, LOW);
      digitalWrite(ledB, LOW);
    }
}
```

Sinyalin ayrıştırılması, alınan verilerin anlamlı bilgilere dönüştürülmesi ve ilgili işlevlerin tetiklenmesi süreci.

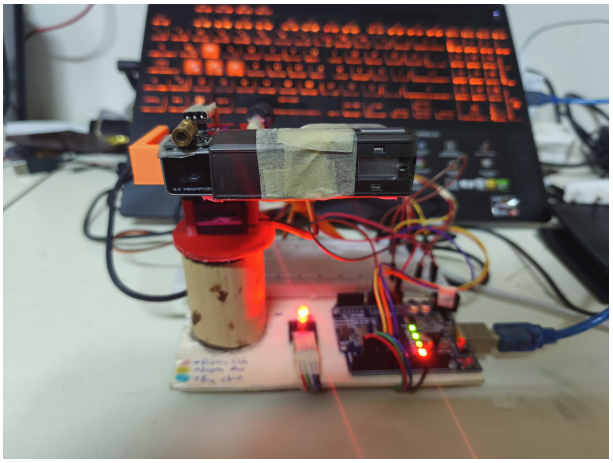
```
void parseData(String input) {
  input.trim();  // Başındaki ve sonundaki boşlukları temizle
  if (input.length() < 8) { // Veri formatı kontrolü
    Serial.println("VERI HATALI!");
    return; }
  c1 = input.substring(0, 3).toInt();  // İlk 3 hane (X koordinatı merkezi)
  c2 = input.substring(3, 6).toInt();  // Sonraki 3 hane (Y koordinatı merkezi)
  c3 = input.charAt(6);  // 7. karakter (Silahlı/Silahsız durumu)
  c4 = input.charAt(7);  // 8. karakter (Lazer durumu)
```

Verilen merkez noktalarından merkez nokta farkını kullanarak adım adım sağ veya sol yapma

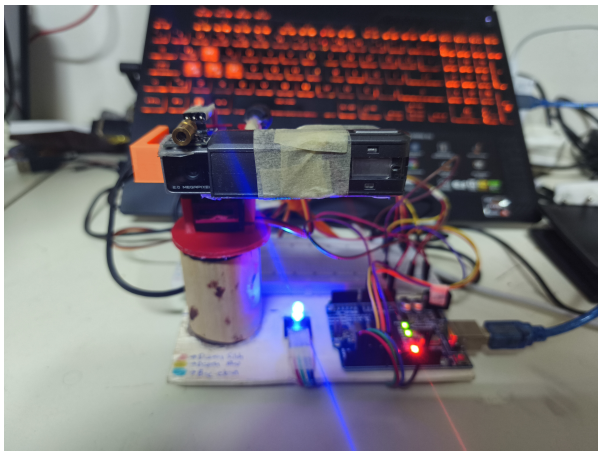
```
void areaScan(){
    if(isToLeft){
        sonBTM+=hareket;
    }else{
        sonBTM-=hareket;
    } if(sonBTM>=180 || sonBTM<=0){
        isToLeft=!isToLeft; }
    sonBTM = constrain(sonBTM, 0, 180);
    sonTOP = 45;
    servoTOP.write(sonTOP);
    servoBTM.write(sonBTM);
    delay(50);
}
void trackObjectX() {
  // X eksenindeki hareketi kontrol et
  if (abs(displayCenterX - c1) > tolerans) {
    if (displayCenterX > c1) {
      sonBTM += hareket;
    } else {
      sonBTM -= hareket;
    }
    sonBTM = constrain(sonBTM, 0, 180);
    servoBTM.write(sonBTM);
    delay(50);
  }}
void trackObjectY() {
  // X eksenindeki hareketi kontrol et
  if (abs(displayCenterY - c2) > 2*tolerans) {
    if (displayCenterY > c2 || sonTOP>45) {
      sonTOP -= hareket;
    } else {
      sonTOP += hareket;
    }
    sonTOP = constrain(sonBTM, 0, 90);
    servoTOP.write(sonBTM);
    delay(50);
  }}
```



Görsel 1.16: Turet silahsız insan görme durumuna ait görsel



*Görsel 1.17: Turet silahlı insan görme durumu*na ait görsel



Görsel 1.18: Turet insan görmeme durumuna ait görsel

II. DİĞER SİSTEMLERDEN FARKLARI

Benzer bir mantıkla çalışan SafePointe [2], Xtract One [3] gibi farklı projeler vardır. Bunlar kar amacı güden şirket ürünleridir ve bu projenin bahsi geçen projelerden kritik farkları bulunmaktadır.

Lokal Görüntü İşleme:

Yukarıda verilen modellerin aksine bu modelde işlem tamamen lokal yapılmaktadır. Görüntü işlemenin işleme sürecinden, değerlendirme sürecine kadar tüm adımlar bilgisayar içerisinde oluşturulmaktadır. Bu hem çevrimdışı çalışma özelliği, hem de veri gizliliği konusunda oldukça fayda sağlayacaktır.

Taşınabilirlik:

Yukarıdaki modellerin hepsi bir yere monte olacak şekilde bir kameraya bağlı çalışır. Bu sistemde ise tüm gerekli elemanlar sadece bağlanması gereken bir gömülü yazılım entegre sistemidir. Ayrıca boyut ve gimbal sebebiyle çeşitli araçlara takılabilir hafifliktedir.

Takipli Kamera:

Yukarıdaki modellerin hepsi belli güvenlik kameralarına bağlı olması durumu var ve bu güvenlik kameralarının hareket yeteneği sınırlıdır. Bu projede ise dikey ve yatay 180 derece dönme yetisine sahip bir servo motor ile çalışıldığından kamera modülü rahatça hedef takibi yapabilecek şekilde dönebilir.

Amaç:

Yukarıda sayılan modeller, genellikle yapıların güvenliğinin sağlanması için kullanılırken bu projedeki sistem otonom araçların ve robotların etrafıyla farkındalığını arttırarak pilotun çevreyle olan etkileşimini arttırır.

III. KARŞILAŞILAN PROBLEMLER VE ÇÖZÜM YÖNTEMLERİ

Proje doğrultusunda bazı problemlerle karşılaştım. Bu problemlerin bazıları ile çözümleri aşağıda verilmiştir.

Servo Motorlara Yetersiz Akım Sağlanması:

- Servo motorlara sağlanan akım yetersiz olduğu için sadece bir servo çalışıyor veya yavaş hareket ediyordu.Çözüm olarak, USB kablosunun veri hatları iptal edilerek yalnızca enerji kısmı kullanıldı. Multimetre ile yapılan ölçümde 5V, 1.5A değerinde güç elde edildi ve devreye ayrı bir güç kaynağı olarak bağlandı. GND hatları seri bağlanarak parazit oluşumu önlendi.

Kamera Görüş Açısının Kısıtlı Olması

- İlk tasarımda kamera doğrudan dizüstü bilgisayarın web kamerası kullanılarak görüntü alıyordu, bu da görüş açısını ciddi şekilde kısıtlıyordu. Güncellenen sistemde, kamera sağa ve sola dönebilir hale getirildi, böylece çok daha geniş bir görüş açısı sağlandı.

Hareket Algoritmasının Eski Tasarımla Uyum Sorunu

- Eski tasarımda hareket algoritması doğrudan web kameradan alınan verilere bağlıydı. Harici bir kamera kullanımı, bu hareket algoritmasını bozuyordu. Bu sorunu aşmak için koordinat temelli bir hareket modeli yerine uzaklık temelli bir model geliştirildi.

Görüntü Kalitesinin Düşüklüğü ve Modelin Accurity Yetersizliği:

- İnsan modeli her ne kadar stabil çalışsa da silah modeli problem yaşayabilmekte ve zaman zaman gözükmemekte. Bunun çeşitli sebebi olsa da ana sebepleri görüntü aygıtının düşük kaliteli olması ve modelin yeterliği tutarlılık düzeyine sahip olmamasıdır. Aynı zamanda saldırganın silahı saklama durumu da mevcut. Bunu önlemek için silahın tespiti durumunda 10 saniye boyunca silahın varlığı hakkında uyarı verilir. Sonrasında tekrar saldırgan kontrolü yapar

IV. KAYNAKLAR

1. Raspberry Pi with Ultralytics YOLO11 [https://docs.ultralytics.com/guides/raspberry-pi/#reproduce-our-results] [Çevrimiçi] (27.10.2024)

2. Stealth Weapons Detection [https://www.soundthinking.com/security/weapons-detection/] [Çevirmiçi] (27.10.2024)

3. AI Powered Weapon Detection System [https://xtractone.com/] [Çevirmiçi] (27.10.2024)

TEKNOVERSİTE