KTO KARATAY
ÜNİVERSİTESİ

MÜHENDİSLİK ve DOĞA BİLİMLERİ FAKÜLTESİ

ELEKTRİK ELEKTRONİK
MÜHENDİSLİĞİ

BİTİRME PROJESİ

OTONOM DÜŞMAN SAVAR

2022-2023

BAŞAK YALÇINER || 21803016

EMRE ÖZKUL || 21803012

İÇİNDEKİLER

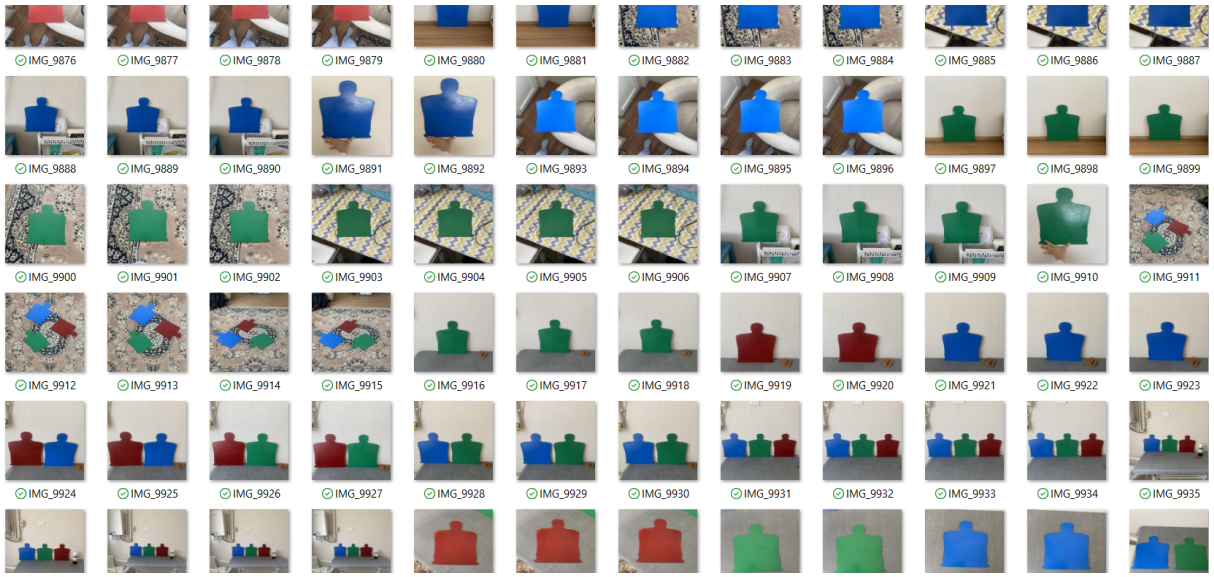
1.ÖZET

Proje görüntü işleme ile gerçek zamanlı nesne takibi yapan, mekanik ve yazılım olarak iki kısımdan oluşan bir projedir. Projede Jetson Nano işletim sistemi, usb modül kamera kullanılmıştır. YoloV4 nesne tanıma algoritması ile geliştirilen bu proje gerçek zamanlı nesne takibi yapıp, mekanizmalı döner bir sistem ile düşman olarak algıladığı nesneye odaklanıp takip eder. Sistem düşman olarak tanımladığı nesneyi sürekli ateş ederek geri püskürtmeyi amaçlamaktadır.

2. PROJE HAKKINDA

1. Veri Toplama

Projede ilk olarak telefon kamerasından nesneleri çekerek makineye öğretmek istediğimiz verileri topladık. Eğitim seti için 320 adet fotoğraf çektik. Bu fotoğrafların farklı arka planlarda ve farklı pozisyonlarda olmasına dikkat ettik. Daha sonra fotoğrafları .jpg formatına getirerek veri toplama işlemini bitirdik.



2. Veri Etiketleme

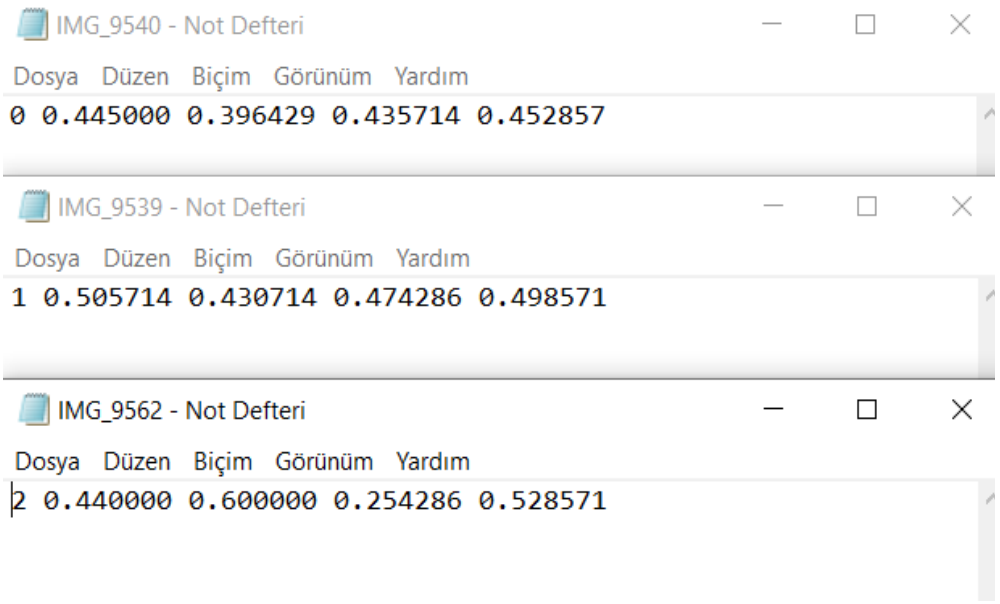
Nesne tanıma algoritmamız için gerekli olan 2. aşamada verilerimizi etiketleyerek eğitim modeline hazır hale getirdik. Burada etiketleme işlemini LabelImg uygulaması ile yaptık. Burada verilerimizi etiketlerken YOLO formatında olmasına dikkat ettik. Daha sonra nesnelerimiz etiketleme işlemine başladık. Etiket dosyalarımız yolo modeline uygu olarak .txt formatında kayıt edildi. Her resmin etiketleri içeren dosyası resimler ile aynı klasöre kaydedildi. Burada nesnelerimiz için 3 adet etiket bulunmakta bunlar;

mavi hedef tahtası= civil | 0

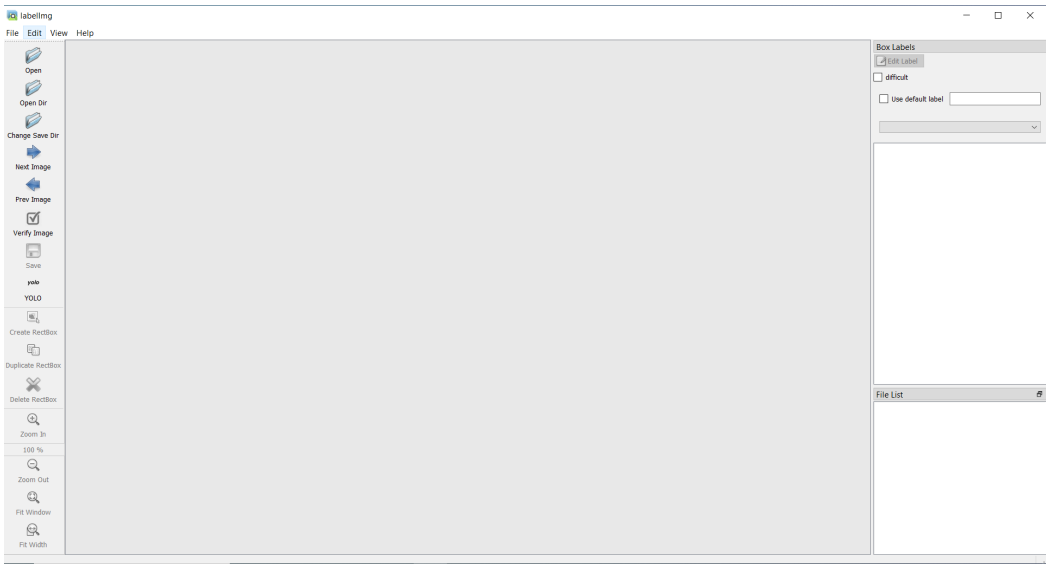
kırmızı hedef tahtası= enemy | 1

yeşil hedef tahtası= soldier | 2

Etiket Dosyaları (.txt)



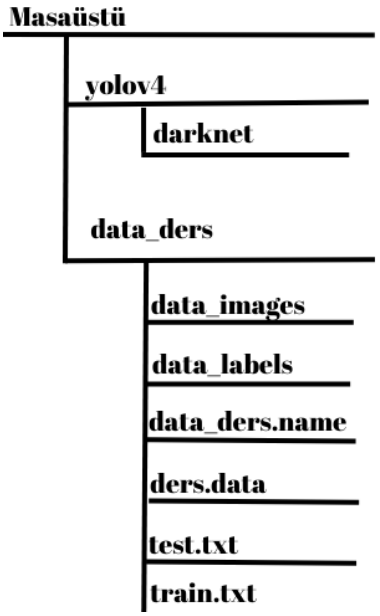
LabelImg Arayüzü:



3. Eğitim İçin Gerekli Dosyaları Hazırlama

Verilerimizin eğitim aşamasına geçmeden önce Yolov4 için gerekli dosyaları hazır hale getirdik. 2 adet dosya hazırlayacağız. İlki data_ders adlı klasörümüz olacak bu klasörde eğitim verilerimizi kullanabilmemiz için gerekli etiket dosyaları ve resimlerimizi hazır hale getireceğiz. İkinci dosyamız ise darknet yapay sinir ağı modelini içeren yolov4 dosyası olacak.

Aşağıda dosya yolunu ve içeriklerini görebilirsiniz:



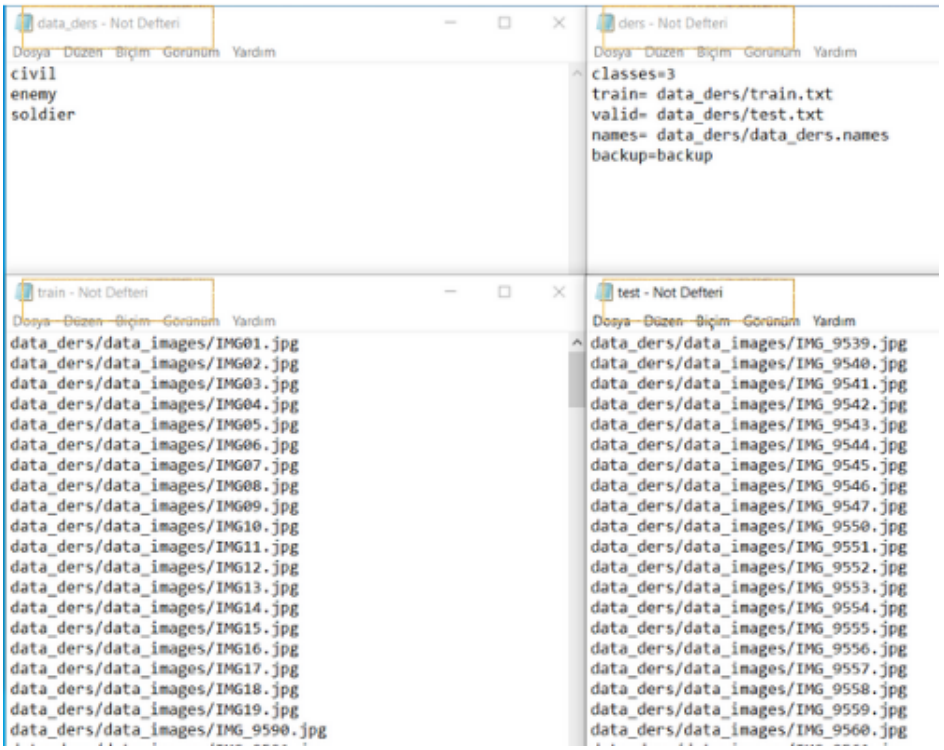
Dosya içerikleri aşağıdaki gibidir:

YoloV4 Klasörü:

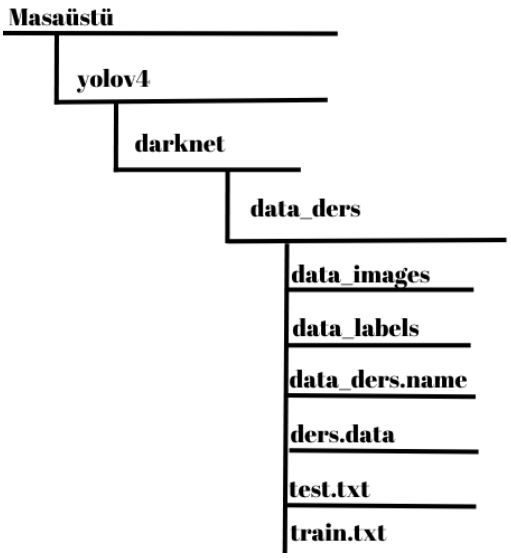
- Darknet git deposunu klonlayın
- YoloV4 modeli için yolov4.cfg ve yolov4.weights dosyalarını siteden indiriniz.
- İndirdiğiniz dosyaları darknet klasörü içine atınız.

Data_Ders Klasörü:

- Eğitim için ihtiyacınız olan ders.data, data_ders.names, data_images (data ve etiketler), data_labels (etiketler), train ve test datalarının yolları. Bu yolları dosyaya kaydederken Window formatında değil Google Colab formatına uygun olarak kayıt edilmiştir.



Tüm dosyaları oluşturduktan sonra data_ders klasörünü darknet klasörü içine atınız ve .zip dosyası haline getiriniz. Daha sonra Google Drive hesabına atınız.



4. Google Colab’te Verilerin Eğitilmesi:

Drive Bağlantısı:

```
from google.colab import drive # Drive hesabımızı ekliyoruz
drive.mount('/content/gdrive')
```

Darnet dosyasını zipten çıkarma:

```
!unzip "/content/gdrive/MyDrive/darknet.zip" -d "/content/" #
Darknet dosyamızı zipten çıkarıyoruz
```

Make:

```
# darknet'i oluşturur, böylece darknet yürütülebilir dosyasını nesne
dedektörlerini çalıştırmak veya eğitmek için kullanabilirsiniz %cd
/content/darknet !make
```

Backup Bağlantısı:

```
!rm /content/darknet/backup -r # Darknetin içindeki backup
klasörünü siliyoruz
!ln -s /content/gdrive/"My Drive"/yolo_weights/backup
/content/darknet/ # Drive içine backup adında klasör oluşturuyoruz
ve buraya bağlıyoruz %pwd
```

Eğitim:

```
!./darknet detector train data_ders/ders.data yolov4_ders.cfg
/content/darknet/yolov4.conv.137 -map -don
```

Map Test:

```
!./darknet detector map data_ders/ders.data yolov4_ders.cfg
/content/gdrive/MyDrive/yolov4_ders_lastt.we
```

Modelimizi Çalıştıralım:

```
# özel cfg'mizi test moduna ayarlamamız gerekiyor %cd
/content/darknet !sed -i 's/batch=64/batch=1/'
yolov4_ders.cfg !sed -i 's/subdivisions=64/subdivisions=1/'
yolov4_ders.cfg %cd ..
```

Test:

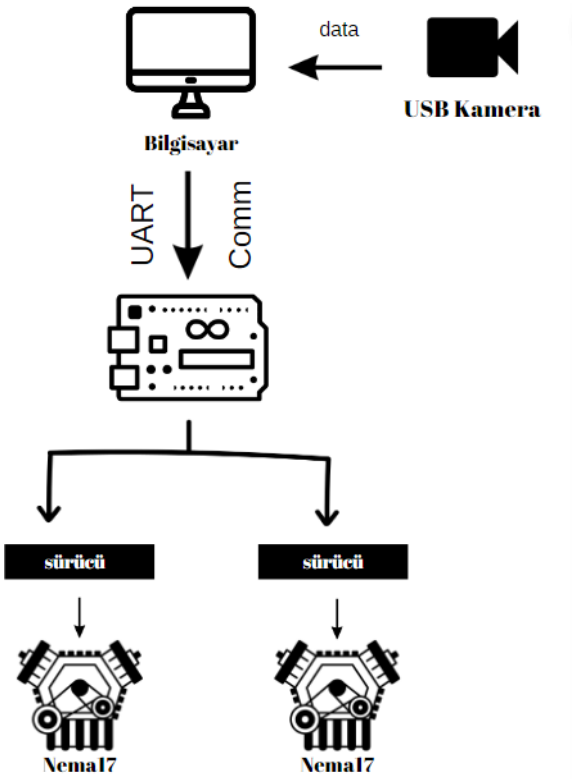
```
%cd /content/darknet !./darknet detector test
data_ders/ders.data yolov4_ders.cfg
/content/gdrive/MyDrive/yolov4_ders_lastt.weigh
```

5.USB kamera ile gerçek zamanlı nesne tespiti:

Eğittiğimiz verilerimizi test etmek için USB kamera kullandık. Burada USB kamera kullanmamızın amacı; modelimizi bilgisayar üzerinden yüksek FPS değeri ile gerçek zamanlı olarak data çıkışları elde etmektir.

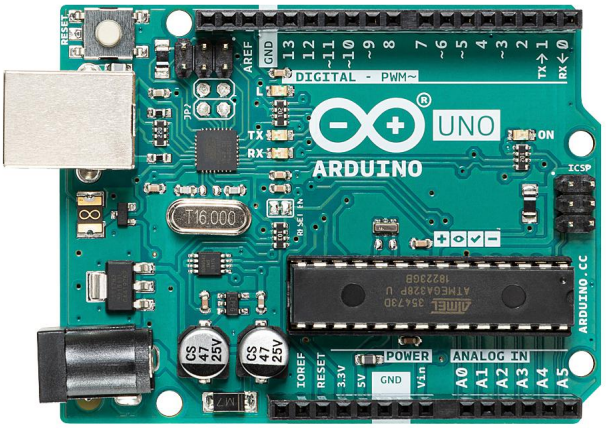
6.Python ve Arduino arasında UART Haberleşmesi ile Motor Kontrol:

Kameradan gelen veriler ile “enemy” nesnesini sistem takip etmeye başlar. Sistem görüntüde belirlediğimiz merkez noktası hedefin box değer noktaları içerisine yer alana kadar takip işlemini gerçekleştirir. Daha sonra otomatik olarak sistem ateş etmeye başlar. Aşağıda projenin çalışma şemasını görebilirsiniz.



3.KULLANILAN MALZEMELER

1.Arduino UNO



Arduino UNO’nun Özellikleri:

- Mikrodenetleyici: ATmega328P
- İşlemci Hızı: 16 MHz
- Dijital Giriş/Çıkış Pinleri: 14 (6 tanesi PWM çıkışı olarak kullanılabilir)
- Analog Giriş Pinleri: 6
- PWM Çıkış Pinleri: 6
- Seri Bağlantı Noktaları (UART): 1
- I2C ve SPI arayüzleri
- Flash Bellek: 32 KB (2 KB bootloader'e ayrılmış)
- SRAM: 2 KB
- EEPROM: 1 KB
- Giriş Gerilimi: 7-12V
- Çıkış Gerilimi: 5V

- USB Bağlantısı: Tip B

Arduino UNO’nun Tercih Edilme Sebebi:

- Seri haberleşme protokolü sayesinde OpenCV projelerinde Python ve Arduino ile bilgisayar arasında haberleşme yapılabilmektedir.
- Arayüz ve kullanım kolaylığı.
- Ulaşılabilirlik.
- Projemizde kullanacağımız kontrollerın temel görevi step motorları yapılan matematik işlemine göre kontrol edebilmesi olmuştur. Bu sebeple fiyat-performans ürünü olan Arduino’yu tercih ettik.

2.USB Kamera - Logitech C310

Logitech C310 Teknik Özellikleri

- Video Çözünürlüğü: 720p HD video kaydı
- Fotoğraf Çözünürlüğü: 5 megapiksel
- Lens: Plastik, otomatik odaklama
- Görüş Açısı: 60 derece
- Otomatik Ayarlama: Otomatik ışık düzeltme, otomatik beyaz dengesİ
- Bağlantı: USB 2.0

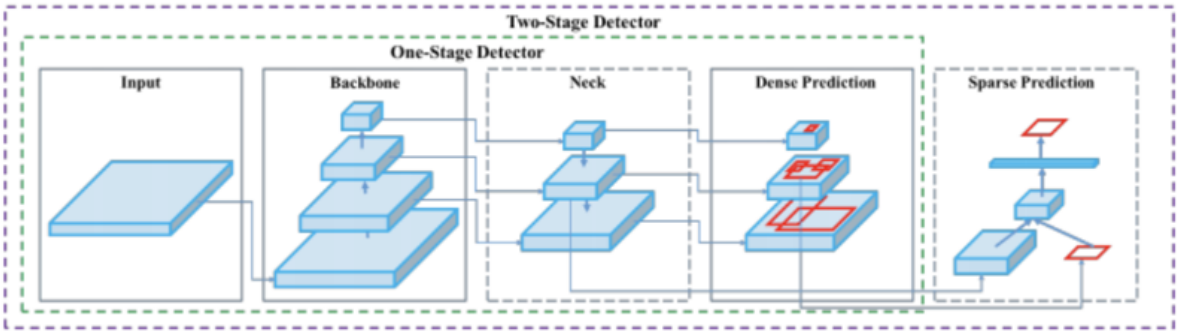
Logitech C310 Tercih Sebepleri:

Projemiz için YOLOv4 yapay zeka modelini kullandık. Projemizde gerçek zamanlı nesne tespiti temel odak noktamız olduğu için Raspberry Pi, Jetson NANO gibi ürünler hem RAM hem de hız bakımından yavaş kalması sebebiyle nesne tespiti modelimizi bilgisayar ortamında çalıştırdık. Bu yüzden Logitech C310 USB kamera bizim en büyük tercih sebeplerimizdendir.

5.ŞEMALAR

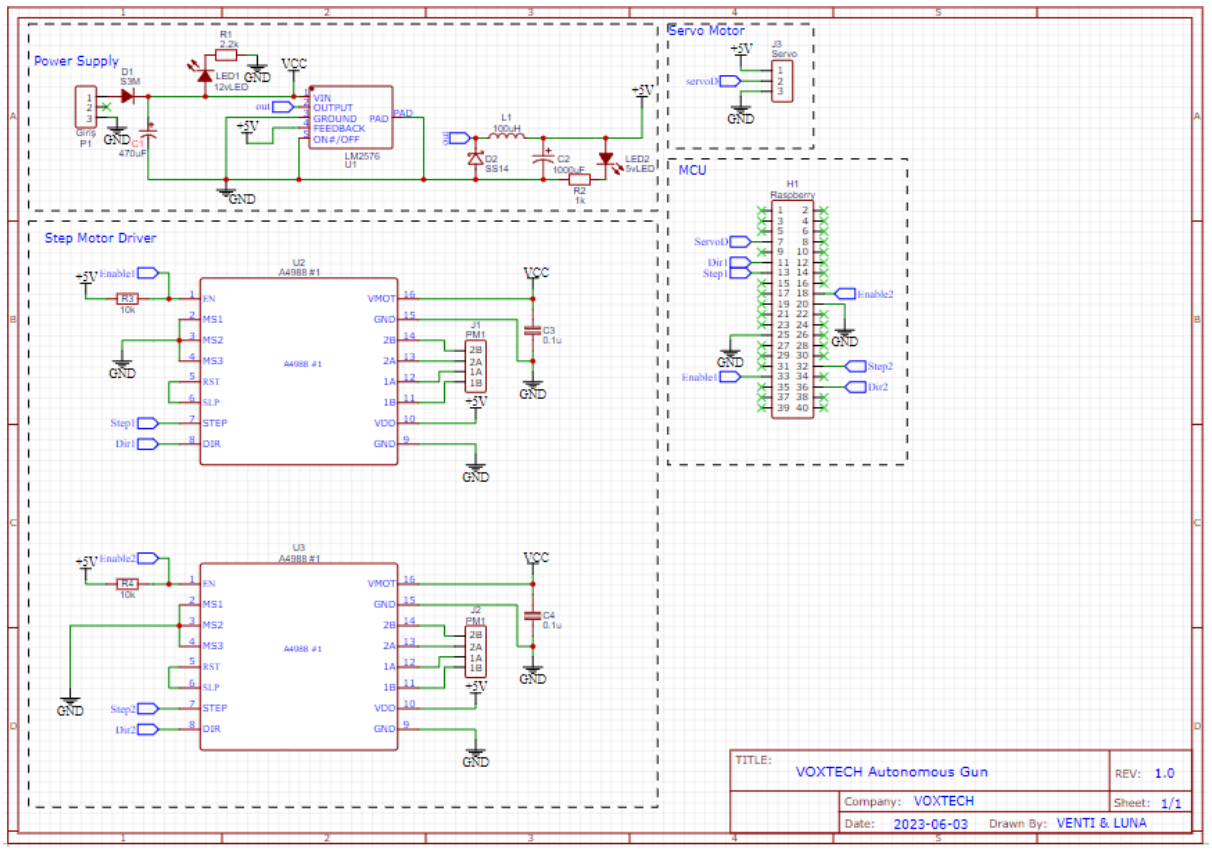
---

1. **Yolo V4 Model Şeması**

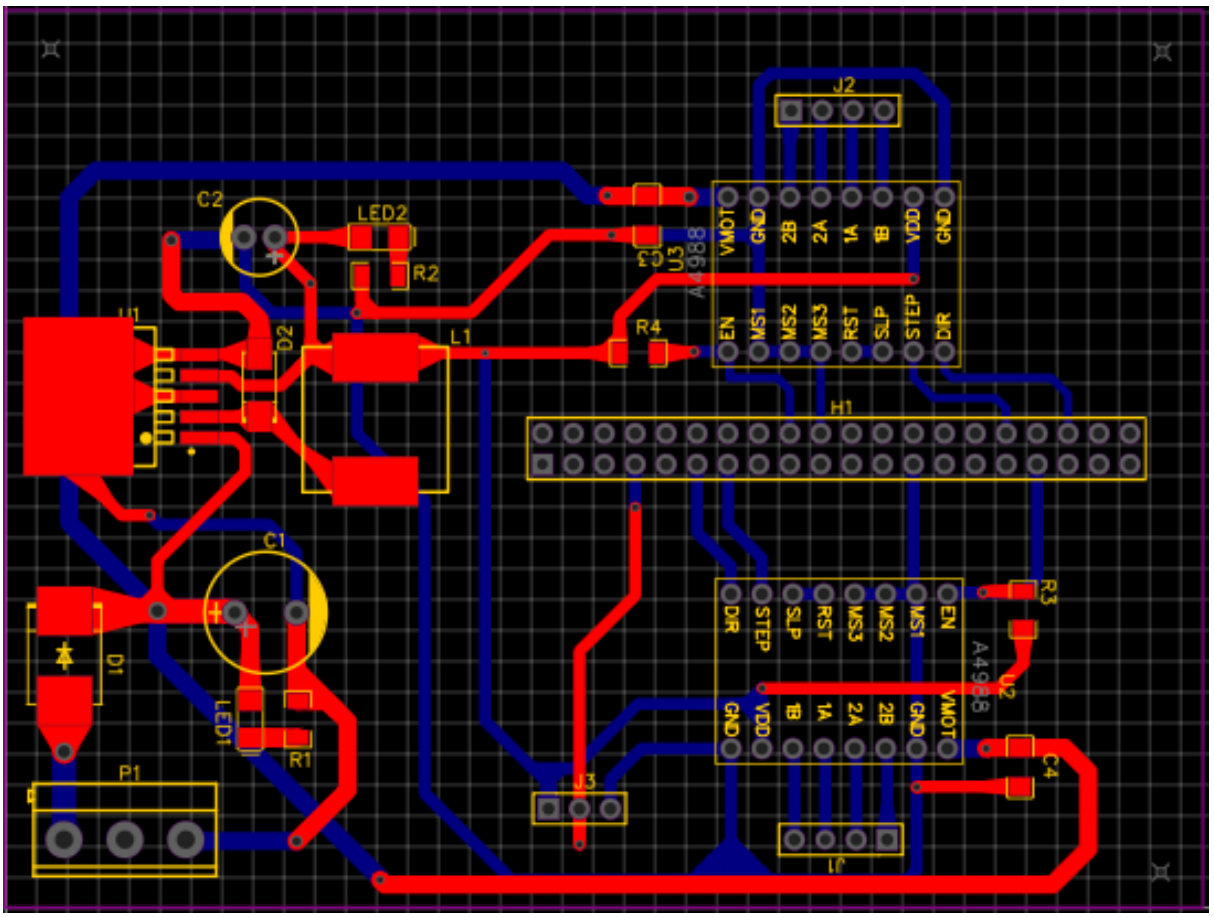


2. **Devre Şematiği**

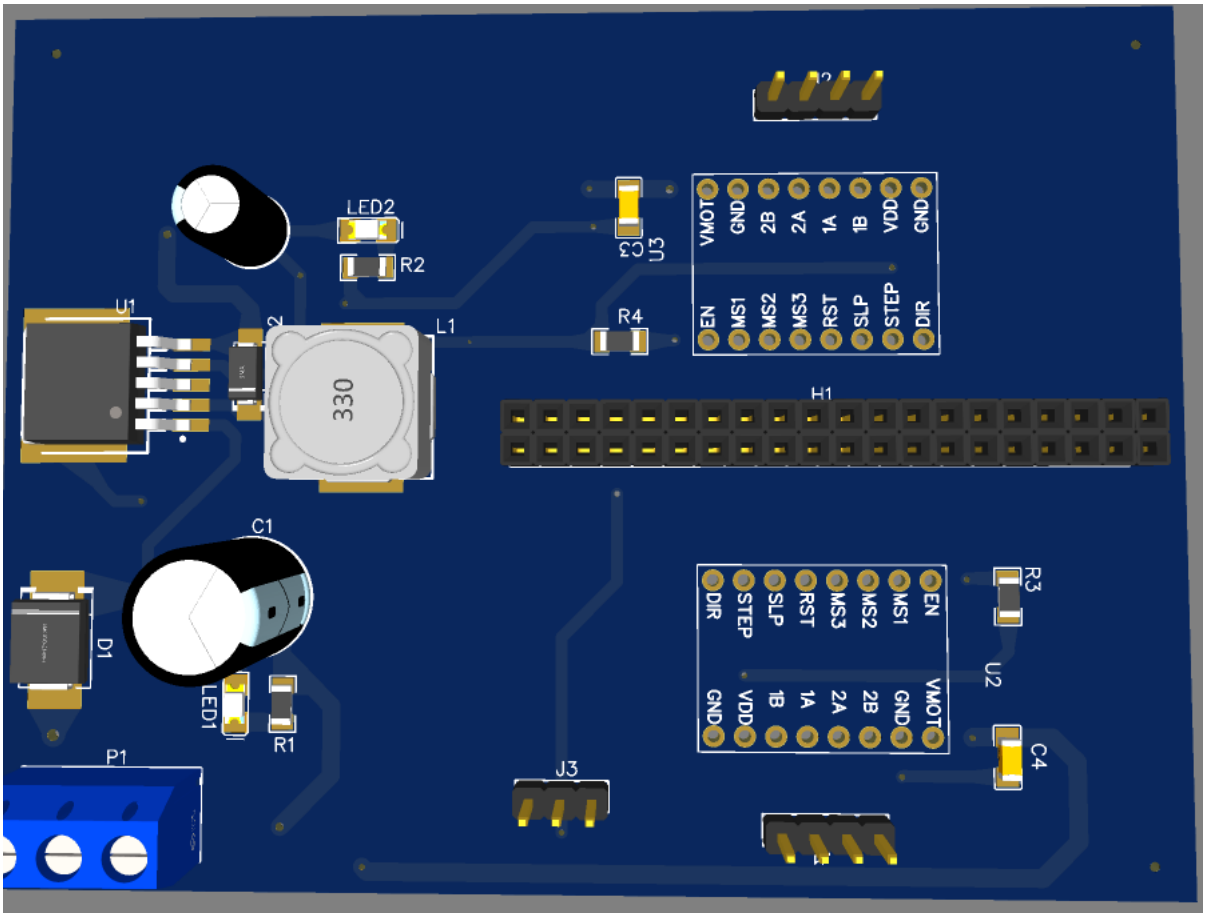
Projenin devre şematiği EasyEDA programında çizilmiştir ve PCB ortamına aktarılmıştır.



Projenin Devre Şematiği

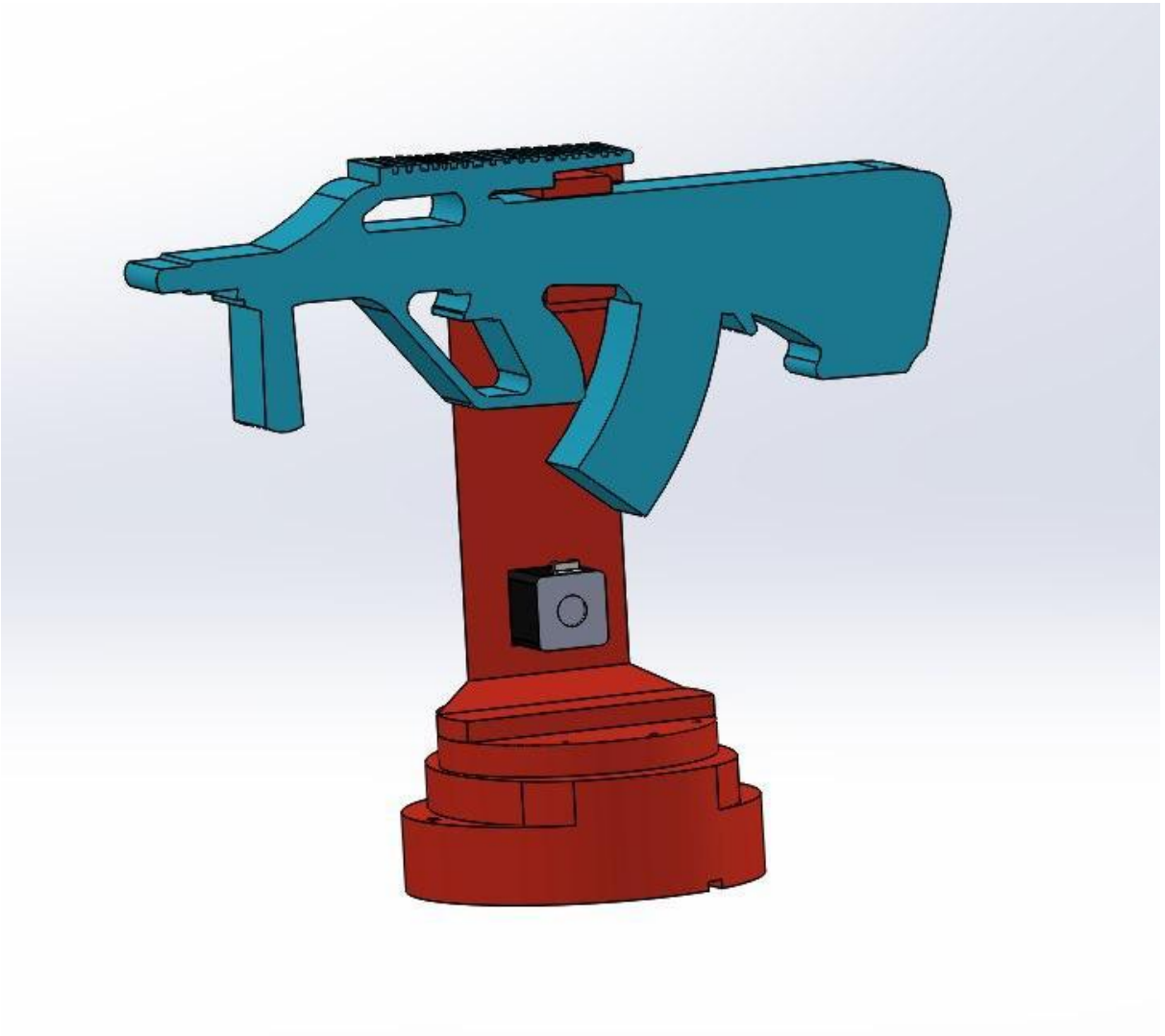


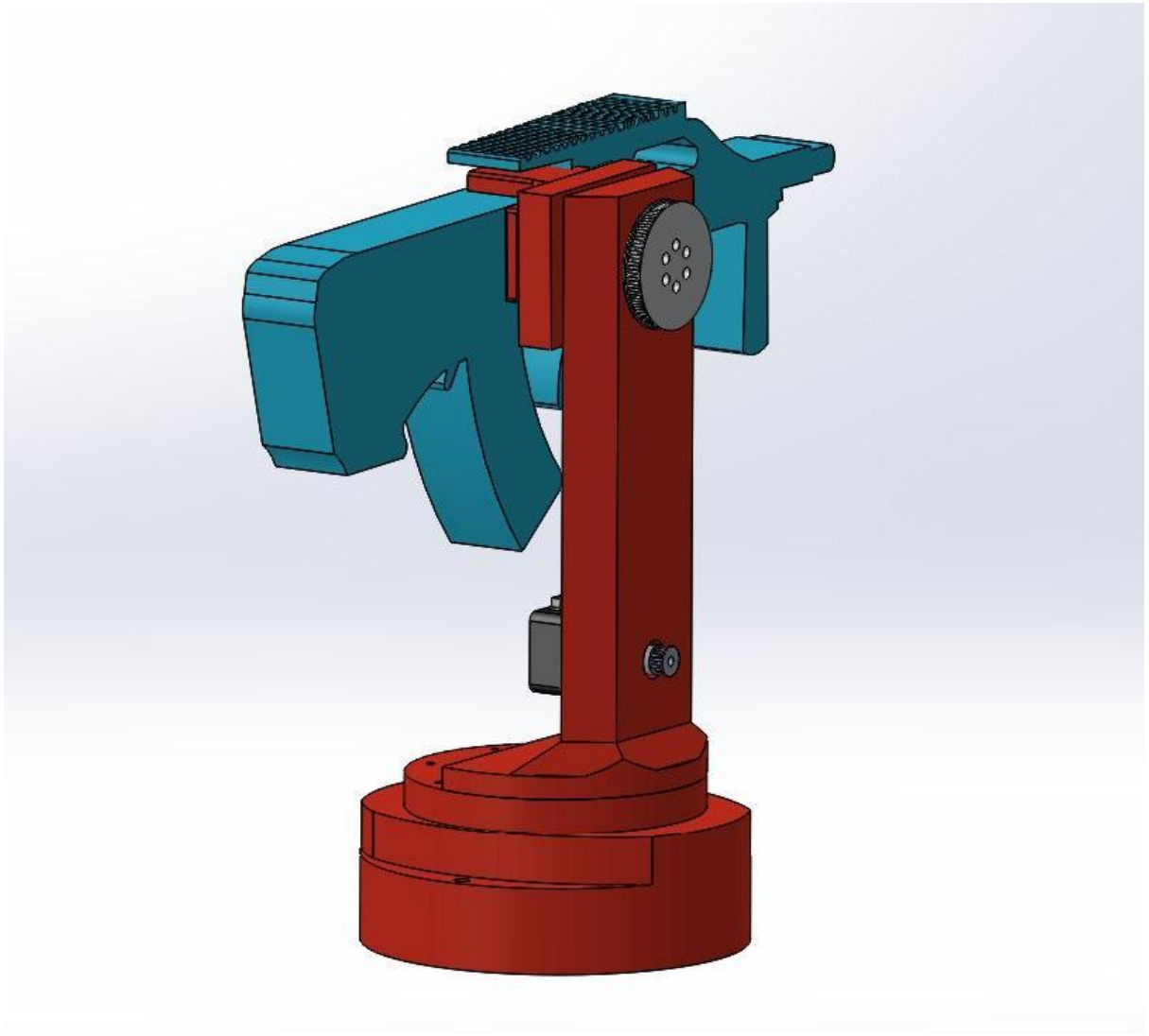
Proje PCB Baskısı

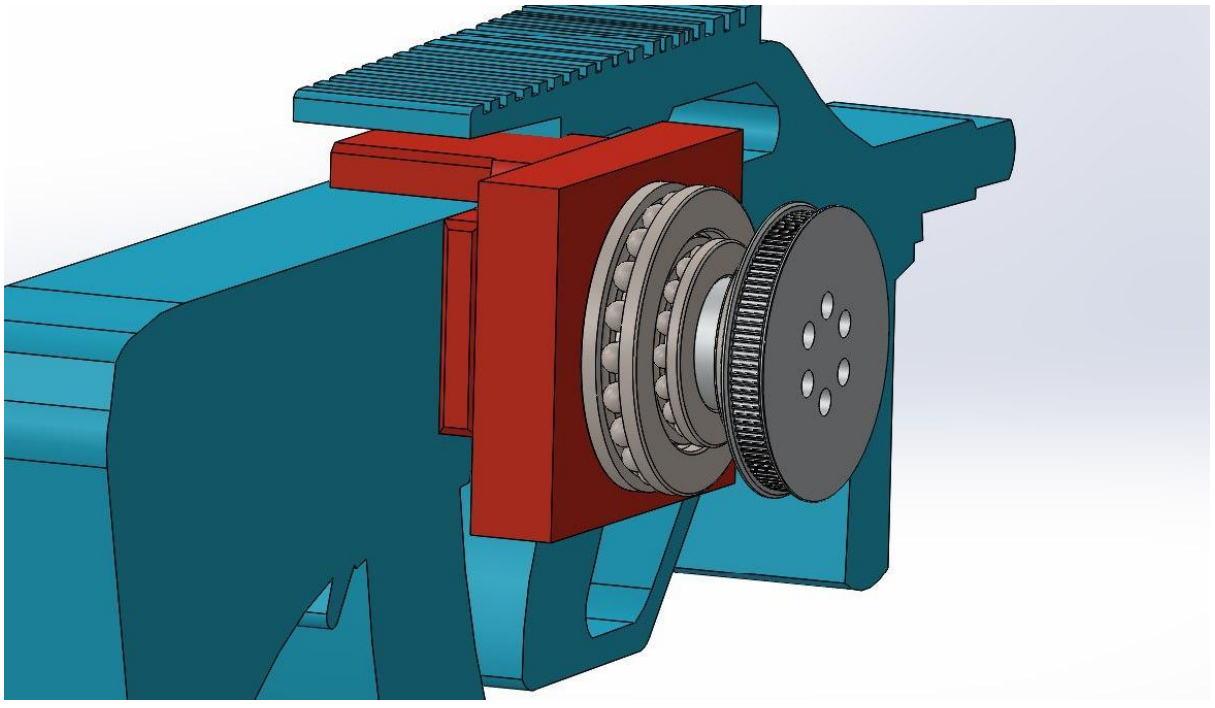


PCB Baskı 3D Görüntüsü

3. Proje 3D Tasarımı







4. Kod Bloğu

4.1 Python Görüntü İşleme ve Yapay Zeka Algoritması;

```
import cv2
import numpy as np
import serial
import time
import struct

cap = cv2.VideoCapture(1)
cv2.namedWindow("Detector", cv2.WINDOW_NORMAL)
cv2.resizeWindow("Detector", 1280, 720)

model = cv2.dnn.readNetFromDarknet(r"yolov4-tiny-ders.cfg",
r"yolov4-tiny-ders_last.weights")a
layers = model.getLayerNames()
output_layer = [layer for layer in model.getLayerNames() if "yolo"
in layer]
labels = [" ", "enemy", " "]

colors = ["0,0,255", "255,100,0", "100,0,255", "255,255,0",
"0,255,0"]
colors = [np.array(color.split(",")).astype("int") for color in
colors]
colors = np.array(colors)
colors = np.tile(colors, (18, 1))

frame_counter = 0
arduino = serial.Serial(port='COM6', baudrate=115200, timeout=.1)

while True:
    ret, frame = cap.read()
    frame = cv2.resize(frame, (416, 416))
    frame_width = frame.shape[1]
    frame_height = frame.shape[0]
    frame_blob = cv2.dnn.blobFromImage(frame, 1 / 255, (416, 416),
swapRB=True, crop=False)
    model.setInput(frame_blob)
```

```
    detection_layers = model.forward(output_layer)
    enemy_detected = False
    if frame_counter % 100 == 0:
        model.setInput(frame_blob)
        detection_layers = model.forward(output_layer)

        ids_list = []
        boxes_list = []
        confidences_list = []

        for detection_layer in detection_layers:
            for object_detection in detection_layer:
                scores = object_detection[5:]
                predicted_id = np.argmax(scores)

                confidence = scores[predicted_id]

                if confidence > 0.20:
                    label = labels[predicted_id]
                    bounding_box = object_detection[0:4] *
np.array([frame_width, frame_height, frame_width, frame_height])
                    (box_center_x, box_center_y, box_width,
box_height) = bounding_box.astype("int")
                    start_x = int(box_center_x - (box_width / 2))
                    start_y = int(box_center_y - (box_height / 2))

                    ids_list.append(predicted_id)
                    confidences_list.append(float(confidence))
                    boxes_list.append([start_x, start_y,
int(box_width), int(box_height)])

                    if label == "enemy":
                        enemy_detected = True

        max_ids = cv2.dnn.NMSBoxes(boxes_list, confidences_list,
0.5, 0.4)
        detected_objects = []
        for max_id in max_ids:
            max_class_id = int(max_id)
            predicted_id = ids_list[max_class_id]

            # Check if the predicted label is "enemy"
```

```
            if labels[predicted_id] == "enemy":
                box = boxes_list[max_class_id]
                start_x = box[0]
                start_y = box[1]
                box_width = box[2]
                box_height = box[3]
                confidence = confidences_list[max_class_id]
                end_x = start_x + box_width
                end_y = start_y + box_height

                # Print rectangle values for enemy


                rect_x = start_x + box_width // 2
                rect_y = start_y + box_height // 2
                frame_center_x = frame.shape[1] // 2
                frame_center_y = frame.shape[0] // 2
                plus_size = 10
                plus_x = frame_center_x
                plus_y = frame_center_y

                box_color = colors[predicted_id]
                box_color = [int(each) for each in box_color]
                label = "{}: {:.2f}%".format(label, confidence *
100)

                cv2.rectangle(frame, (start_x, start_y), (end_x,
end_y), box_color, 1)
                cv2.rectangle(frame, (start_x - 1, start_y),
(end_x + 1, start_y - 30), box_color, -1)
                cv2.putText(frame, label, (start_x, start_y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1)

                if enemy_detected:
                    a = plus_x - rect_x
                    b = plus_y - rect_y
                    x_fark = a
                    y_fark = b
                    veriler = [x_fark, y_fark]
                    cv2.putText(frame, "Detect Enemy!", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 2)
                    for veri in veriler:
```

```
                        arduino.write(str(veri).encode())
                        arduino.write(b'\n')  # Arduino'da yeni
satır karakterini bekleme


    # Draw a plus sign in the center of the frame
    frame_center_x = frame.shape[1] // 2
    frame_center_y = frame.shape[0] // 2
    plus_size = 10
    plus_color = (0, 0, 0)
    cv2.line(frame, (frame_center_x - plus_size, frame_center_y),
(frame_center_x + plus_size, frame_center_y), plus_color, 2)
    cv2.line(frame, (frame_center_x, frame_center_y - plus_size),
(frame_center_x, frame_center_y + plus_size), plus_color, 2)
    x1 = frame_center_x
    y1 = frame_center_y

    cv2.imshow("Detector", frame)

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

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

4.2 Arduino Uno İle Motor Kontrol Kodu

```
int datafromUser=0,motorA=0,listen_fl=0,x_fl=0,y_fl=0;
int incoming[1];
int deflectionX=20,deflectionY=20;
int deflection = 0;
int fark_x=0,fark_y=0;
int adimSayisi=400;
int transistorPin = 7;
#define dirPinX 11
#define stepPinX 12
#define dirPinY 2
#define stepPinY 3
void setup() {
  pinMode(stepPinX, OUTPUT);
  pinMode(dirPinX, OUTPUT);
  pinMode(stepPinY, OUTPUT);
```

```
  pinMode(dirPinY, OUTPUT);
  pinMode(transistorPin, OUTPUT);
  Serial.begin(115200);

}
void ListenPort(){
  if(Serial.available()>1)
    {
         fark_x = Serial.parseInt();
         fark_y = Serial.parseInt();

     }
      // farkX=Serial.readString().toInt();
      if(fark_x < 0-deflectionX || fark_x > 0+deflectionX){
        x_fl=1;
      }
      if(fark_y < 0-deflectionY || fark_y > 0+deflectionY){
        y_fl=1;
      }

      listen_fl = 1;
    }
void loop() {
  ListenPort();

  // X  EKSENİ
  if(x_fl==1 && listen_fl==1){
    if(fark_x<0){
        digitalWrite(dirPinX, LOW);
        digitalWrite(stepPinX, HIGH);
        delayMicroseconds(1300);
        digitalWrite(stepPinX, LOW);
        delayMicroseconds(1000);


    }

    if(fark_x>0){
        digitalWrite(dirPinX, HIGH);
        digitalWrite(stepPinX, HIGH);
        delayMicroseconds(1300);
        digitalWrite(stepPinX, LOW);
```

```
        delayMicroseconds(1000);

    }

    if(0-deflectionX <= fark_x && 0+deflectionX>=fark_x){
    fark_x = 0;
    x_fl = 0;
    }
}
// Y KONTROL
if(y_fl == 1 && listen_fl==1 and x_fl==0){

    if(fark_y<0){
        digitalWrite(dirPinY, LOW);
        digitalWrite(stepPinY, HIGH);
        delayMicroseconds(400);
        digitalWrite(stepPinY, LOW);
        delayMicroseconds(400);
    }
    if(fark_y>0){
        digitalWrite(dirPinY, HIGH);
        digitalWrite(stepPinY, HIGH);
        delayMicroseconds(400);
        digitalWrite(stepPinY, LOW);
        delayMicroseconds(400);

    }
    if(0-deflectionY <= fark_y && 0+deflectionY>=fark_y){
    fark_y = 0;
    y_fl = 0;
    }
    }

if(fark_x == 0 && fark_y == 0 ){
    // D7 pini üzerinden transistöre bir çıkış sinyali gönderme
    digitalWrite(transistorPin, HIGH);
    delay(3000); // 1 saniye bekleme
    digitalWrite(transistorPin, LOW);
    delay(1000); // 1 saniye bekleme
}
}
```

6.PROJE ÇIKARIMLARI

Yolov4, nesne tespiti ve sınıflandırma konusunda oldukça popüler bir derin öğrenme modelidir. İşte Yolov4'ü tercih etme sebepleri:

Yüksek Performans: Yolov4, yüksek hassasiyet ve hızlı nesne tespiti sağlayan gelişmiş bir algoritmadır. Önceki versiyonlara kıyasla daha yüksek bir doğruluk oranı sunar ve gerçek zamanlı uygulamalarda başarılı sonuçlar elde etmenizi sağlar.
Hızlı Nesne Tespiti: Yolov4, ölçeklenebilir bir ağ yapısına sahiptir ve çoklu ölçekli özellik haritaları kullanarak nesneleri tespit eder. Bu, diğer nesne tespiti modellerine kıyasla daha hızlı ve etkili bir performans sunar.
Çoklu Nesne Tespiti: Yolov4, tek bir görüntüde birden fazla nesneyi aynı anda tespit edebilme yeteneğine sahiptir. Bu, uygulamalarınızda birden çok nesne tespiti gerektiren durumlarda avantaj sağlar.
Geniş Veri Seti Desteği: Yolov4, çeşitli nesne sınıflarını tanımak için geniş bir veri setini kullanarak eğitilebilir. Bu, farklı nesne kategorilerini tanımak ve sınıflandırmak için kullanışlıdır.
Çeşitli Uygulama Alanları: Yolov4, güvenlik, trafik analizi, nesne izleme, otomatik sürüş ve diğer birçok alanda kullanılabilir. Esnek bir yapıya sahiptir ve farklı projelerde kolaylıkla uygulanabilir.
Topluluk Desteği: Yolov4, geniş bir kullanıcı topluluğuna sahiptir. Bu, yeni başlayanlar için yardım kaynaklarına ve paylaşılan kodlara kolay erişim sağlar. Ayrıca, sürekli geliştirilen ve güncellenen bir açık kaynaklı projedir.
Tabii ki, Yolov4'ün bazı dezavantajları da vardır. Örneğin, daha karmaşık bir model olduğu için eğitim ve kurulum süreci biraz daha zor olabilir. Ayrıca, donanım gereksinimleri daha yüksek olabilir. Ancak, genel olarak Yolov4, etkileyici performansı ve geniş uygulama alanlarıyla birçok araştırmacı ve geliştirici tarafından tercih edilen bir derin öğrenme modelidir.

7.REFERANSLAR

https://medium.com/analytics-vidhya/train-a-custom-yolov4-tiny-object-detector-using-google-colab-b58be08c9593

https://youtu.be/H3SJcwttTi4

https://github.com/wiryanatasunardi/YOLOv4-Custom-Object-Detection

https://www.hackster.io/ansh2919/serial-communication-between-python-and-arduino-e7cce0

https://docs.arduino.cc/learn/electronics/stepper-motors

https://www.instructables.com/Controlling-a-Stepper-Motor-with-an-Arduino/