TC.

PAMUKKALE ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

GÖRÜNTÜ İŞLEME İLE OTOPARK OTOMASYONU

Mert Hayla

18184030

Danışman

Öğr. Gör. Ercan GÖNÜLDEŞ

MEKATRONİK SİSTEM TASARIM PROJESİ

MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

DENİZLİ-HAZİRAN 2023

TEZ ONAYI

GÖRÜNTÜ İŞLEME İLE OTOPARK OTOMASYONU

Mert Hayla tarafından hazırlanan bu tez çalışması Pamukkale Üniversitesi Teknoloji Fakültesi Mekatronik Mühendisliği Bölümü'nde ders hocası Öğr. Gör. Ercan GÖNÜLDEŞ karşısında **MEKATRONİK SİSTEM TASARIM PROJESİ** olarak başarı ile savunulmuştur.

Danışman **Öğr. Gör. Ercan GÖNÜLDEŞ** .....
Pamukkale Üniversitesi

ETİK BEYANI

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.

MERT HAYLA

ÖZET

Mezuniyet Tezi

Görüntü İşleme ile Otopark Otomasyonu

Mert Hayla

Pamukkale Üniversitesi

Mekatronik Mühendisliği Bölümü

Danışman: Öğr. Gör. Ercan GÖNÜLDEŞ

Bu tez, görüntü işleme tekniklerinin kullanılarak otoparklarda plaka tanıma ve giriş-çıkış takibi yapabilen bir otomasyon sistemi geliştirilmesini amaçlamaktadır. Geliştirilen sistem, araçların plakalarını okuyarak giriş-çıkış saatlerini ve park etme sürelerini kaydederek müşteri takibi yapabilmektedir.

Sistem, Arduino geliştirme kartı ve kamera kullanılarak oluşturulmuştur. Görüntü işleme teknikleri kullanılarak plakaların tespit edilmesi ve tanınması gerçekleştirilmiştir. Plakaların tanınması için opencv ve pytesseract kütüphaneleri kullanılmıştır. Sistem, tanınan plakaları veri tabanına kaydederek giriş-çıkış saatlerini ve park etme sürelerini hesaplamaktadır.

Programlama dili olarak Python ve C dilleri kullanılmıştır. Plaka tanıma işlemi bilgisayar üzerinde yapılacak mekanik işlemler ise Arduino tarafından gerçekleştirilecektir.

Yapılan testlerde, sistemin başarılı bir şekilde çalıştığı ve plakaları doğru bir şekilde tanıdığı gözlemlenmesi hedeflenmektedir. Ayrıca, sistem tarafından toplanan verilerin doğru bir şekilde işlendiği ve müşteri takibi yapmak için yeterli olacak şekilde kurgulanmıştır.

Sonuç olarak, geliştirilen bu sistem sayesinde otopark işletmeleri müşteri takibini daha kolay ve doğru bir şekilde yapabileceklerdir. Ayrıca, bu çalışma görüntü işleme tekniklerinin farklı uygulama alanlarına da uyarlanabileceğini göstermektedir.

Anahtar Kelimeler: Görüntü işleme, opencv, otomasyon.

ABSTRACT

GRADUATION THESIS

PARKING AUTOMATION WITH IMAGE PROCESSING
Mert Hayla

Pamukkale University
Faculty of Technology
Department of Mechatronics Engineering

Supervisor: Öğr. Gör. Ercan GÖNÜLDEŞ

This thesis aims to develop an automation system that can perform license plate recognition and entry-exit tracking in car parks by using image processing techniques. The developed system can track the customers by reading the license plates of the vehicles and recording the entry-exit times and parking times.

The system is built using Arduino development board and camera. Detection and recognition of license plates were performed using image processing techniques. Opencv and pytesseract libraries were used to recognize the plates. The system calculates the entry-exit times and parking times by recording the recognized license plates in the database.

Python and C languages were used as programming languages. The plate recognition process will be performed on the computer, and the mechanical operations will be performed by Arduino.

In the tests, it is aimed to observe that the system works successfully and recognizes the plates correctly. In addition, it has been designed in such a way that the data collected by the system is processed correctly and is sufficient for customer follow-up.

As a result, thanks to this developed system, parking companies will be able to follow up customers more easily and accurately. In addition, this study shows that image processing techniques can be adapted to different application areas.

Keywords: Image processing, opencv, automation.

ÖNSÖZ

Tezimin yürütülmesinde desteğini ve emeğini hiçbir zaman esirgemeyen tez danışmanım sayın Öğr. Gör. Ercan GÖNÜLDEŞ'e çalışma süresince bana desteklerinden dolayı teşekkürlerimi sunarım.

Mert Hayla
Denizli-2023

1.GİRİŞ

Sanayi devriminden günümüzde iş hayatında insanın yerini makinaların aldığını görüyoruz. Günümüzde yazılım teknolojisinin gelişmesi ve teknolojinin öğreniminin kolaylaşması insanın hata yapma faktörünü ortadan kaldırmaktadır. Bu gelişme ışığında görüntü işleme teknolojisini kullanarak hazırladığım bu projede amaçlanan insandan kaynaklanan hataların giderilmesi ve işin makinaya yaptırılmasıdır.

Proje Python dili ve C dili kullanılarak programlanmıştır. Projede kullanılan mikrodenetleyici Arduino firmasının UNO modelidir. Proje bir prototip olarak simüle edilmiş ve yapılan testler başarıyla tamamlanmıştır.

Proje 3 Ana başlık altında yazılmıştır bu başlıklar;

- Arayüz
- Algoritma
- Elektronik Devre

Projede kullanılan araçlar, Algoritmalar, Arayüz tasarımı ve Elektronik devre bu bölümlerde tanıtılacaktır.

Projenin akış diyagramı aşağıdaki gibidir;

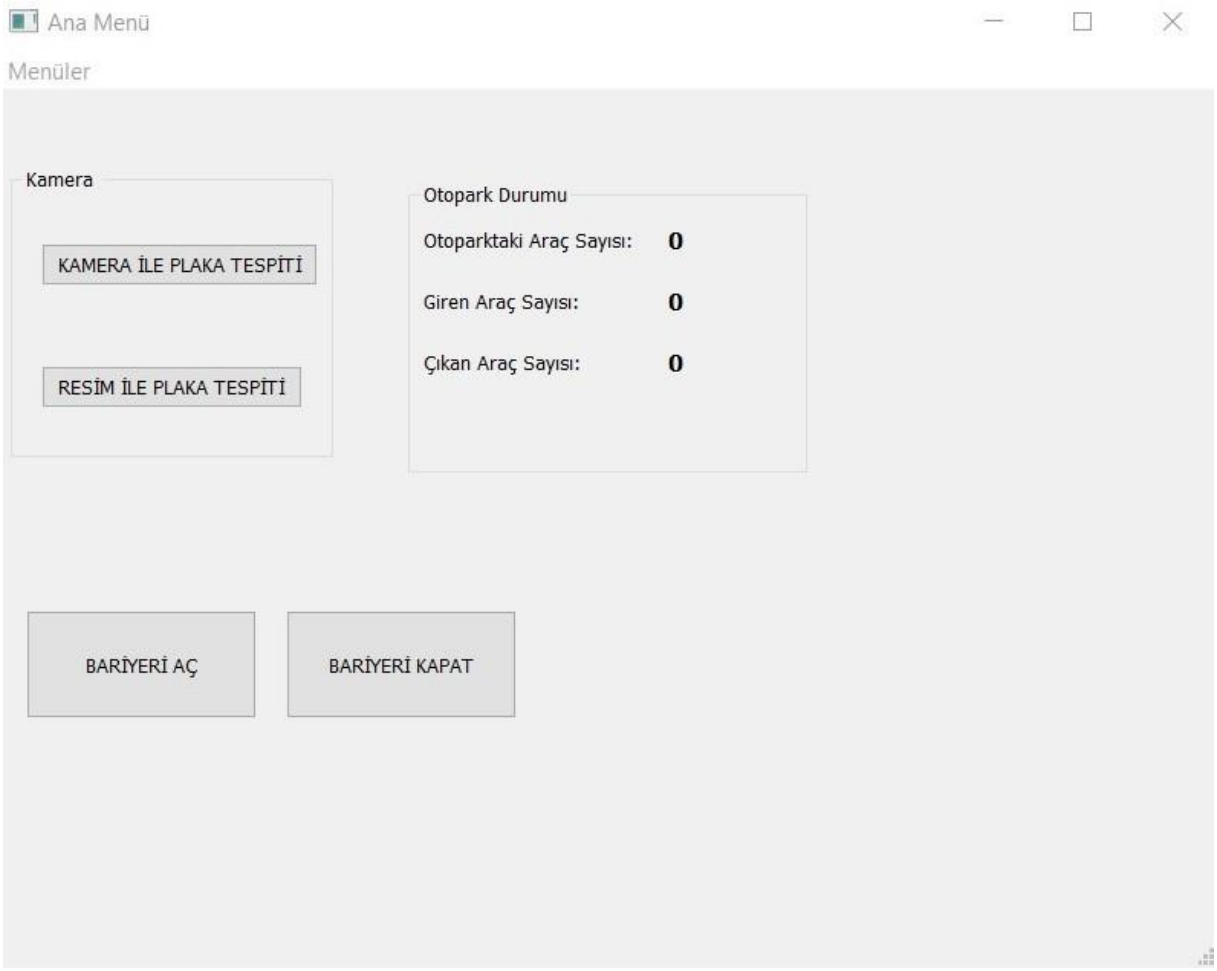
2.ARAYÜZ TASARIMI
2.1 Giriş (LOGİN) Penceresi;

Programın çalıştırıldığında bizi karşılayan ekrandır. Bu ekranın kullanılmasının amacı ticari uygulamanın temel güvenlik gereksinimlerinin karşılanmasıdır.

Giriş (Login) penceresi standart bir düzende yapılmıştır. Programda işlem yapılabilmesi için bu güvenlik aşamasının Kullanıcı Adı ve şifre büyük küçük harf düzeninde doğru girilmesi ile gerçekleşir.

Kullanıcının şifresinin uzun olması ve doğru hatırlayamaması durumu için yanında Şifreyi Göster yazan bir check box kullanılmıştır.

2.2 Ana Pencere;



Bu menü kullanıcının bilgi aldığı ve işlem yaptığı Ana penceresidir.

Bu pencerede 4 Buton ve 1 Menü bulunmaktadır. Bunun yanında Otoparktaki araç durumunun takip edildiği bir göstergede mevcuttur.

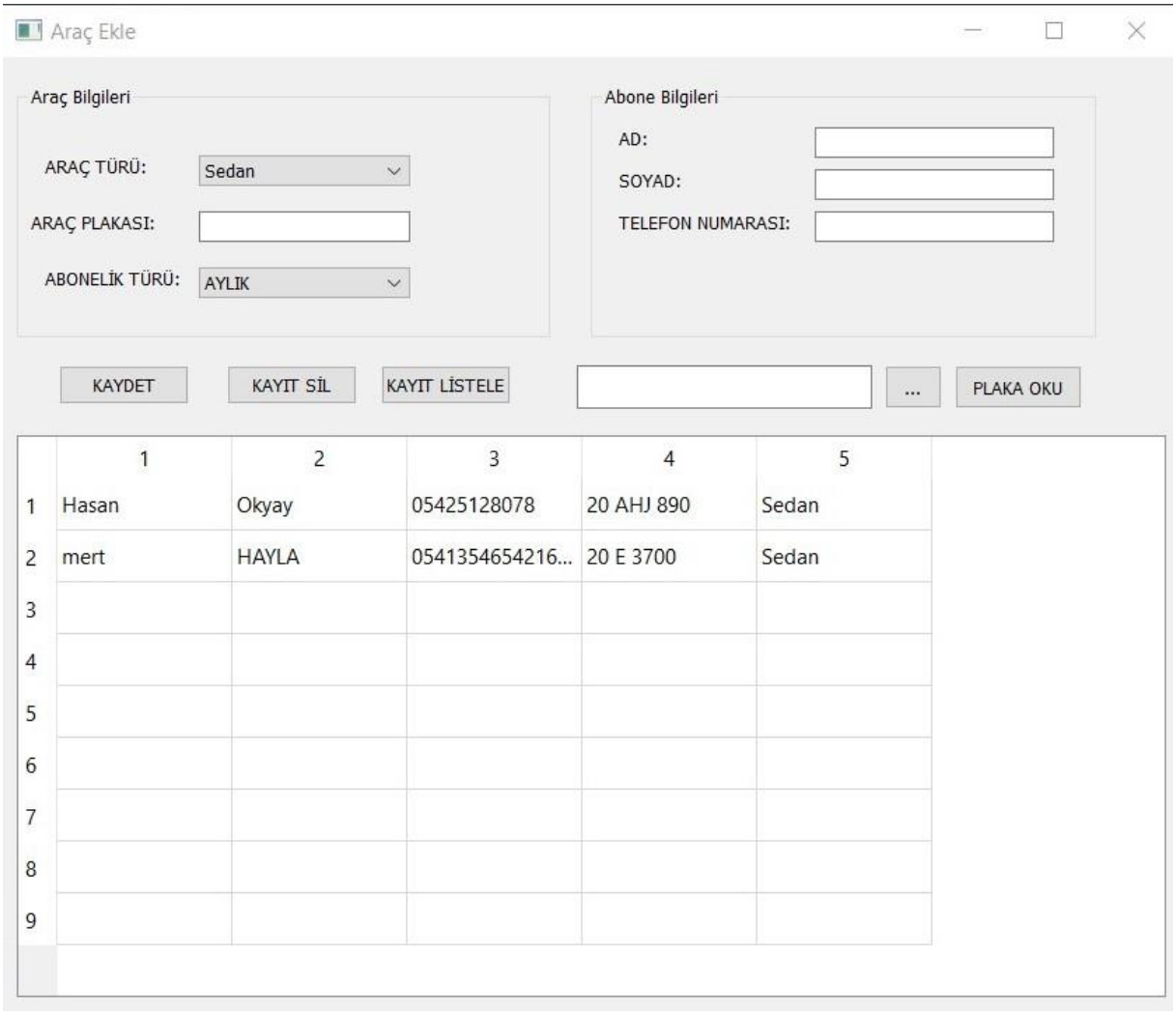
Kamera ile Plaka Tespiti: Bu buton kullanıcının sistemde kurulu olan kamera ile görüntü işleme yardımı ile aracın plakasının tespit edilmesi ve kullanıcının girişe izninin olup olmadığının kontrolü içindir.

Resim ile Plaka Tespiti: Bu buton kullanıcının araç plakasının doğru okunamaması durumda harici bir kamera ile aracın resmini çekip programa yüklemesi ile plakanın tespit edilerek işlem yapılması içindir.

Bariyeri Aç: Bu buton bariyerin manuel olarak açılması içindir.

Bariyer Kapat: Bu buton bariyerin manuel olarak kapatılması içindir.

2.3 Araç Ekleme;



Bu pencereye ana menü üzerinden menüler sekmesi ile ulaşılmaktadır.

Bu pencerede yeni bir aracın eklenmesi veya mevcut bir aracın kaydının silinmesi sağlanabilir.

Bu bölümde Aracın resmi çekilir ve sisteme kaydetmek için (…) sembollü buton ile resim seçilir. Plaka oku butonuna basıldığında görsel üzerinden görüntü işleme ile plaka okunur ve Araç Plakası bölümüne okunan veri yazdırılır.

Araç ile ilgili bilgiler girildikten sonra kaydet butonuna basılarak veriler database'e kaydedilir.

3.ALGORİTMA

3.1 Algoritmada Kullanılan Kütüphaneler;

Projede kullanılan diller Python ve C dilleridir.

Bu dillerinin kullanılmasının nedeni Python'un geliştiricisinin ve kaynaklarının bolluğu aynı zamanda görüntü işlemede kolaylık sağlayan kütüphaneleridir.

C dilinin kullanılmasının nedeni kullanılan Arduino kartının C dili ile programlanmasıdır.

Python Kütüphaneleri aşağıdaki gibidir;

- PyQt5: Arayüz tasarımında kullanılan nesne tabanlı bir Python kütüphanesidir.
- Os: Windows üzerinden görsel okumak için kullanılan bir kütüphanedir.
- OpenCV: Görüntü işleme için kullanılan temel kütüphanedir.
- Matplot.Pyplot: Görüntü işleme için kullanılan yardımcı kütüphanedir.
- Numpy: Görüntü üzerinde matrisler ve diziler ile işlem yapmaya yarayan yardımcı kütüphanedir.
- Pytesseract: Yakalanan plakadaki metni okumaya yarayan kütüphanedir.
- Sqlite3: Veritabanı işlemlerini gerçekleştiren SQL kütüphanesidir.
- Serial: Arduino ile seri haberleşmeyi sağlayan kütüphanedir.
- Time: Seri haberleşme sırasında kullanılan while döngüsünü yavaşlatmak için kullanılan sleep() fonksiyonunun bulunduğu kütüphendir.
- Enum: Veri tabanından veri çekerken gelen verilerin numaralandırılarak üzerinde işlem yapılmasını sağlayan kütüphanedir.

C tabanlı Arduino kütüphaneleri;

- SPI.h: Arduino'nun RFID modülü ile seri haberleşmesi için kullanılan kütüphanedir.
- RFID.h: RFID modülünün fonksiyonlarının kullanılmasını sağlayan kütüphanedir.
- Servo.h: Servo motor kontrolü için kullanılan kütüphanedir.

Algoritma yukarıdaki kütüphaneler kullanılarak yazılmıştır. Kullanılan Algoritmalar ilerleyen sayfalarda açıklanacaktır.

Algoritmalar sistemin işlem yükünün azaltılması için sadeleştirilmiştir.

3.2 Kullanılan Algoritmalar;

3.2.1 Bariyer Açma ve Kapatma:

```
def bariyerac(self):
    global arac_sayisi
    arac_sayisi = arac_sayisi+1
    self.anapencereform.arac_sayi_label.setText(str(arac_sayisi))
    arduino = serial.Serial('COM7',9600)
    a =0
    while a<3:
        arduino.write(b'1')
        time.sleep(90/100)
        a=a+1
        while(a == 2):
            break

def bariyerkapa(self):
    global arac_sayisi
    arac_sayisi = arac_sayisi-1
    self.anapencereform.arac_sayi_label.setText(str(arac_sayisi))
    arduino = serial.Serial('COM7',9600)
    b=0
    while b<3:
        arduino.write(b'0')
        time.sleep(90/100)
        b= b+1
        while(b==2):
            break
```

Bu bölümde Bariyer aç ve Bariyer kapa butonları üzerinden Arduino ile haberleşerek servo motorun 90° ve 0° derece konumlarına gelerek bariyerin açılıp kapanması sağlanmıştır.

Bariyeri aç butonuna basıldığında While döngüsü içerisinde görüldüğü gibi yaklaşık 2 saniye boyunca Arduino'nun serial portuna '1' değeri gönderilmiştir. Bu değer Arduino da okunarak servo motor hareket ettirilmiş ve bariyer açılmıştır.

Bariyeri kapat butonuna basıldığında ise gene While döngüsü içerisinde görüldüğü gibi 2 saniye boyunca Arduino'nun serial portuna '0' değeri gönderilmiştir. Bu değerde Arduino için servo motoru 0° konumuna getirilerek kapatılmasını ifade eder.

Kodda görülen time.sleep() fonksiyonu programın her döngüde 0,9 saniye bekletilmesi içindir. Bu kodun eklenmesinin nedeni while döngüsünün çok hızlı çalışmasından dolayıdır. While döngüsünün Arduino'nun veri okumasını gerçekleştirmeden döngüden çıkılmasını engellemek içindir bu fonksiyon kullanılmıştır.

3.2.2 Plaka Tespit Algoritması:

```
def browse(self):
    options = QFileDialog.Options()
    options |= QFileDialog.DontUseNativeDialog
    fileName, _ = QFileDialog.getOpenFileName(self,"QFileDialog.getOpenFileName()", "","All Files (*);;Python Files (*.py)", options=options)
    if fileName:
        print(fileName)
    self.resim_tespitform.lineEdit.setText(fileName)
    pixmap = QPixmap(fileName)
    self.resim_tespitform.label_resim.setPixmap(pixmap)

def plakaokuma(self):
    resim_adres_line = self.resim_tespitform.lineEdit.text()
    img = cv2.imread(resim_adres_line)
```

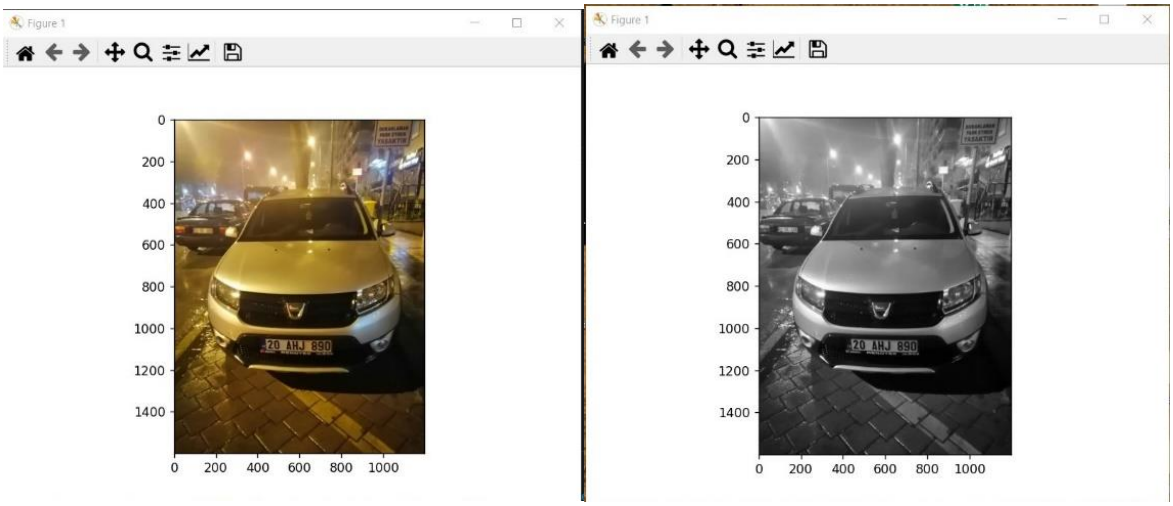
Yukarıdaki kod parçasında gösterilen iki buton araç ekle menüsündeki görseli seç (browse) ve plaka oku butonlarıdır. Görseli seç butonunda görüldüğü gibi bir diyalog penceresi açılmaktadır. Açılan bu pencereden araç görseli seçilir ve dosya yolu lineEdit aracına yazdırılır.

Plaka okuma butonu ise line editte yazılı olan dosya yolunu alarak resmi okur.

```
# Resmi Gri formata çeviriyoruz.
img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
plt.imshow(img_gray,cmap="gray")
plt.show()
```

Bu kod parçası görselin Gri formata çevrilerek daha kolay işlem yapılması içindir.

Kodun çıktısı aşağıdaki gibidir;

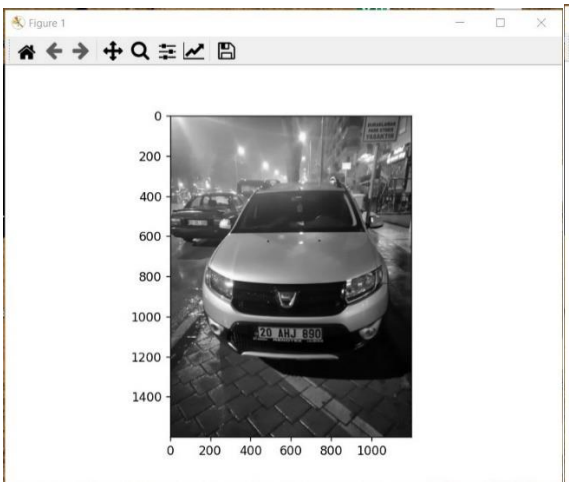


```python
#Gri Formata dönüştürdüğümüz resimden gereksiz pikselleri atmak için blur atıyoruz.
ir_img = cv2.medianBlur(img_gray,5) #5x5
ir_img = cv2.medianBlur(ir_img,5) #5x5

plt.imshow(ir_img,cmap="gray")
plt.show()
```

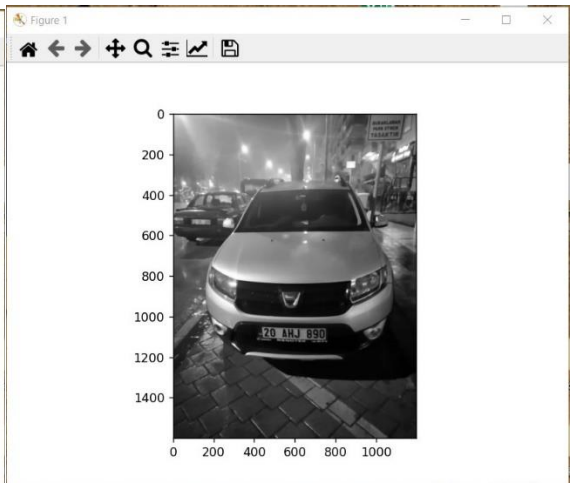
Yukarıda gösterilen kod bloğu gri formata çevrilen görseldeki istenmeyen pikselleri en aza indirmek için kullanılmıştır.

Burada 5x5 piksel hassasiyetinde görsel 2 kez bulurlanmıştır.

Kodun çıktısı aşağıdaki gibidir;



(Gri Formata Çevrilmiş)



(Medyan Bulur Uygulanmış)1

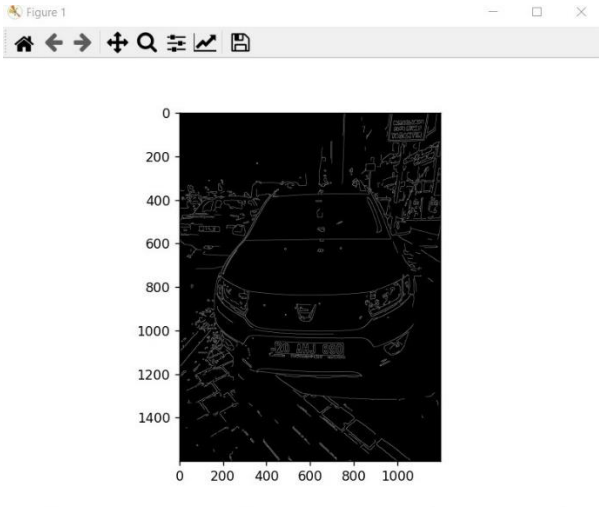
```python
#Resimden medyan değerlerini aldırıyoruz.
medyan = np.median(ir_img)
#üst ve alt medyan değerlerini tanımlıyoruz
low = 0.67*medyan
high = 1.33*medyan

#Canny komutu ile üst ve alt medyan değerlerine uyan pikselleri belirliyoruz.
kenarlik = cv2.Canny(ir_img,low,high)

plt.imshow(kenarlik,cmap="gray")
plt.show()


# np.ones((3,3),np.uint8) --> kenarlıkları 3x3 oranında büyütüyoruz.
kenarlik = cv2.dilate(kenarlik,np.ones((3,3),np.uint8),iterations=1)

plt.imshow(kenarlik,cmap="gray")
plt.show()
```

Burada görselin medyan değerleri numpy kütüphanesi kullanılarak alınmış ve en verimli oranlarda üst ve alt değerlere genişletilmiştir.

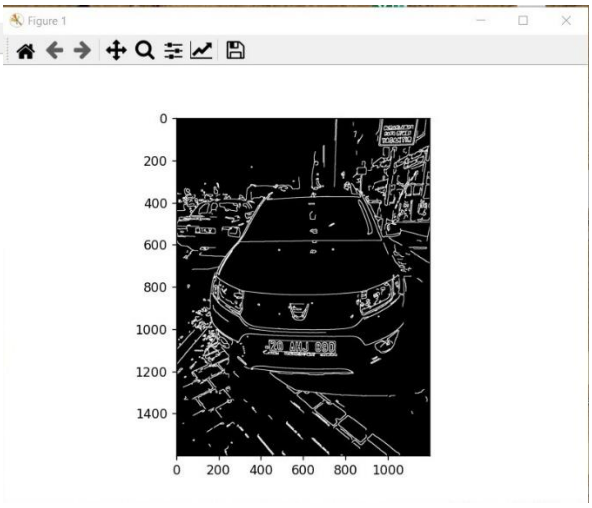
Daha sonra Canny komutu ile kenarlıklar low ve high değerleri ile tespit edilmiştir.

Son aşamada ise kenarlıklar genişletilerek belirginleşmiştir.

Kodun çıktısı aşağıdaki gibidir;



(Kenarlık Tespiti)



(Kenarlık Genişletilmiş)

```python
#cv2.RETR_TREE -> hiyerasık
#CHAIN_APPROX_SIMPLE -> kosegenleri aldık, tum pıkseller yerine zincir biçiminde
# birbirine bağlı pikselleri seçtik ve contur olarak belirledik
cnt = cv2.findContours(kenarlik,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
cnt = cnt[0]
cnt = sorted(cnt,key=cv2.contourArea,reverse=True)

H,W = 500,500
plaka = None

for c in cnt:
    rect = cv2.minAreaRect(c) #dikdortgen yapıda al (1)
    (x,y),(w,h),r = rect
    if(w>h and w>h*2) or (h>w and h>w*2):#oran en az 2 (2)
        box = cv2.boxPoints(rect) #[[12,13],[25,13],[20,13],[13,45]]
        box = np.int64(box)

        minx = np.min(box[:,0])
        miny = np.min(box[:,1])
        maxx = np.max(box[:,0])
        maxy = np.max(box[:,1])


        muh_plaka = img_gray[miny:maxy,minx:maxx].copy()
        muh_medyan = np.median(muh_plaka)


        kon1 = muh_medyan>53 and muh_medyan<200 # yogunluk kontrolu (3)
        kon2 = (w<483 and h<100)##(w<48 and h<170) or (h<50 and w<170) #sınır kontrolu (4)
        kon3 = (w<100 and h<400) #or (h<50 and w<170) #sınır kontrolu (4)

        print(f"muh_plaka medyan:{muh_medyan} genislik: {w} yukseklik:{h}")
```

- Bu aşamada cv2 kütüphanesi kullanılarak hiyerarşik yapıda tüm pikseller zincir biçiminde tespit edilmiş ve cnt dizisine eklenmiştir.
- Cnt dizisine atanan değerlerde dikdörtgen yapılar tespit edilmiş ve bunlar rect değişkenine x,y,w,h,r değerleri olarak atanmıştır.
- İf kontrolünde bir plakada ayırt edici bir faktör olan w değerinin h değerin 2 katından büyük olması özelliğiyle yakalanan dikdörtgenler incelenmiştir.
- Yakalanan dikdörtgenlerden minx, miny, maxx, maxy değişkenleri tespit edilmiş ve gri resimden bu değerler denk gelen kısımlar işaretlenerek kopyalanmıştır.
- Kon1: Yakalanan görselde medyan değeri 53 ila 200 arasında olan karelerin kontrolüdür.
- Kon2: Genişlik değeri 483 den küçük ve yükseklik değeri 100 den küçük olan karelerin kontrolü.
- Kon3: Kon2 de yazılan değerlerin tersi olması durumunun kontrolü.
- Tespit edilen değerler komut satırına yazdırılarak doğru karenin ölçüleri ile optimizasyon yapılmıştır.

Plaka yakalama algoritmasının bu kısmı görselin plaka okumasının kolaylaştırılması içindir. Bu aşamadan sonrası plakanın tespitini sağlayan algoritma olacaktır.

```
plt.figure()
kon=False
if(kon1 and (kon2 or kon3)):
    #plaka nın tespiti

    cv2.drawContours(img,[box],0,(0,255,0),2)
    plaka =[int(i) for i in [minx,miny,w,h]]#x,y,w,h


    kon=True
    if(plaka[2]<plaka[3]):
        plaka_resmi=img_gray[plaka[1]:plaka[1]+plaka[2],plaka[0]:plaka[0]+plaka[3]].copy()
    else:
        plaka_resmi=img_gray[plaka[1]:plaka[1]+plaka[3],plaka[0]:plaka[0]+plaka[2]].copy()

else:

    pass
```

- İlk satırda pyplot kütüphanesi kullanılarak okunan plaka metninin ekrana yazdırılması için bir figür oluşturulmuştur.
- İlk if bloğunda ise bir önceki bölümde belirlenen plaka olma şartlarına uygunluk kontrolü yapılmıştır
- Yakalanan dikdörtgenlerden uygun olan bölge drawContours komutu ile yeşil bir dikdörtgen içine alınmıştır.
- Yakalanan bölgenin koordinatları plaka değişkenine atanmıştır.
- Yakalanan plaka değişkeninde görselin doğru kopyalanması için ikinci if yapısı kullanımıştır. Yakalanan plaka plaka_resmi değişkenine atanmıştır.

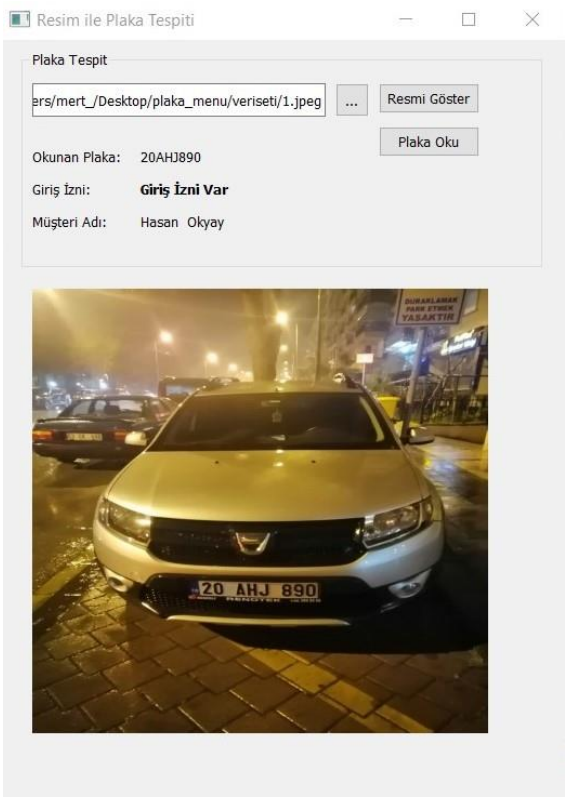
Algoritmanın kullanıldığı Resim ile Plaka tespit et menüsündeki plaka oku butonunun işlemlerinde database ile karşılaştırma yapılmaktadır. Bu bölüm ise aşağıdaki gibidir.

```
if(kon):
    metin = pytesseract.image_to_string(plaka_resmi)

    cv2.imwrite(f"{metin}.jpg",plaka_resmi)
    plaka_metni = self.resim_tespitform.okunan_plaka.text()
    baglanti= sqlite3.connect("kayit.db")
    islem = baglanti.cursor()
    baglanti.commit()
    silinmişveri = metin.strip("\n")

    islem.execute("SELECT * FROM kayit WHERE plaka = ? ",(silinmişveri,))
    data = islem.fetchall()
    print("Data Verisi:"+str(data))
    listesiz_data=str(data)
    plaka_verileri = listesiz_data.split(',')
    silinecekler1 = "'"
    silinecekler2 = " "
    yenimetin = plaka_verileri[3].replace(silinecekler1,"")
    yenimetin= yenimetin.replace(silinecekler2,"")
    plaka_yeni = silinmişveri.replace(silinecekler2,"")

    plaka_ad = plaka_verileri[0].replace(silinecekler1,"")
    plaka_ad = plaka_ad.replace("[","")
    plaka_ad =plaka_ad.replace("(","")
    plaka_soyad = plaka_verileri[1].replace(silinecekler1,"")
```

```
        if str(yenimetin) == str(plaka_yeni):#str(plaka_metni+"\n"):
            self.resim_tespitform.giris_izni.setText("Giriş İzni Var")
            self.resim_tespitform.musteriadi.setText(plaka_ad+" "+plaka_soyad)
        else:
            self.resim_tespitform.giris_izni.setText("Giriş İzni Yok")




        self.resim_tespitform.okunan_plaka.setText(plaka_yeni)
        self.resim_tespitform.giris_izni.setStyleSheet("color: green")
        self.resim_tespitform.giris_izni.setStyleSheet("font-weight: bold")


        return metin
        metin = None

        break
```

- Bu bölümde ise yakalanan plaka koordinatlarındaki resim pytesseract kütüphanesi ile okunmuş ve metin okunan plaka label öğresine yazdırılmıştır.
- Sonrasında SQL bağlantısı ile kayit.db isimli database e bağlanılmış ve yakalanan metindeki boş (Null), / (bölme işareti) gibi plaka verisine uygun olmayan veriler silinerek plaka metni veri tabanındaki metinle karşılaştırılmıştır.
- Karşılaştırma sonucu veriler aynı ise plakanın kayıtlı olduğu kullanıcının Adı ve Soy adı ilgili label'lara girilmiş ve giriş izni var ibaresi label a yazdırılmıştır.

Bu işlemin arayüzdeki görünümü aşağıdaki gibidir;



3.2.3. Veri Tabanı İşlemleri

Kayıt Listeleme;

```
def kayitlistele(self):
    baglanti= sqlite3.connect("kayit.db")
    islem = baglanti.cursor()
    baglanti.commit()
    sorgu = "select * from kayit"
    self.aracekleform.tablo_arac.clear()
    islem.execute(sorgu)
    for indexSatir,kayitNumarasi in enumerate(islem):
        for IndexSutun,kayitSutun in enumerate(kayitNumarasi):
            self.aracekleform.tablo_arac.setItem(indexSatir,IndexSutun,QTableWidgetItem(str(kayitSutun)))
```

- Bu kod parçasında araç ekle penceresindeki **KAYIT LİSTELE** butonunun işlevi görülmektedir.
- Kayit.db isimli veri tabanına bağlanılmıştır. Bu veri tabanından veriler getirilerek tablo araç isimli TableWidget öğesine sıra ile yazdırılmıştır.

Kayıt Silme;

```
def kayitsil(self):

    baglanti= sqlite3.connect("kayit.db")
    islem = baglanti.cursor()
    baglanti.commit()
    sil_mesaj = QMessageBox.question(pencere,"Silme Onayi","Silmek istediğinize eminmisiniz ?")
    QMessageBox.Yes | QMessageBox.No
    if sil_mesaj == QMessageBox.Yes:
        secilen_kayit = self.aracekleform.tablo_arac.selectedItems()#self.arackleform.tablo_arac.selectedItems()

        silinecek_kayit = secilen_kayit[0].text()
        sorgu = "delete from kayit where ad = ?"

        try:
            islem.execute(sorgu,(str(silinecek_kayit),))
            baglanti.commit()


            #kayitlistele()
        except:
            pass
    else:
        ui.statusbar.showMessage("Silme işlemi İptal Edildi")
```

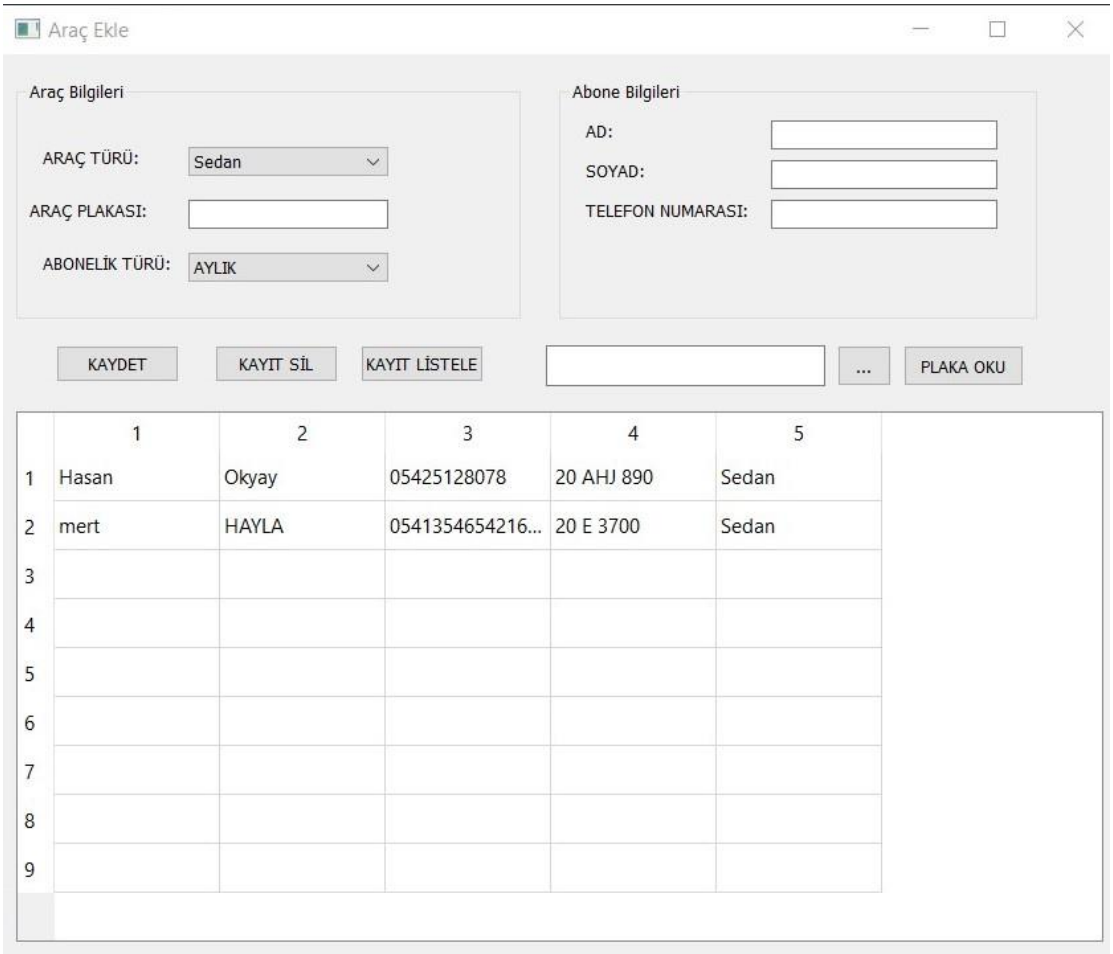
- Kod parçasında araç ekle penceresindeki **KAYIT SİL** butonunun işlevi görülmektedir.
- Butona basıldığında bir MessageBox açılmakta ve Kaydı silmek istediğinize eminimsiniz? yazısıyla evet ve hayır seçeneği bulunan bir pencere açılmaktadır.
- Evet'e basılırsa seçili olan kayıt databaseden silinerek işlem sonlandırılır.
- Hayır'a basılırsa Silme işlemi iptal edildi yazarak işlem sonlandırılır.

Kayıt Ekleme;

```
def arackaydet(self):
    baglanti= sqlite3.connect("kayit.db")
    islem = baglanti.cursor()
    baglanti.commit()
    #id = None
    ad =self.aracekleform.ad_linedit.text()
    soyad = self.aracekleform.soy_ad_linedit.text()
    tel_no = self.aracekleform.tel_lineedit.text()
    plaka = self.aracekleform.arac_plaka.text()
    arac_tur = self.aracekleform.arac_tur_combobox.currentText()
    try:

        ekle = ("INSERT INTO kayit (ad,soyad,tel,plaka,araba_tur) values (?,?,?,?,?)")
        islem.execute(ekle,(ad,soyad,tel_no,plaka,arac_tur))
        baglanti.commit()


    except:
        pass
```

- Bu bölüm araç ekle penceresinin **KAYDET** butonunun fonksiyonunu göstermektedir.
- Burada lineEditlere girilen bilgiler değişkenlere atandıktan sonra try işlemi içerisinde görüldüğü gibi veriler aynı sırayla veri tabanına eklenmektedir.

Bu bölümde anlatılan Araç ekleme arayüzü aşağıdaki gibidir;



3.2.4 Arduino Algoritması:

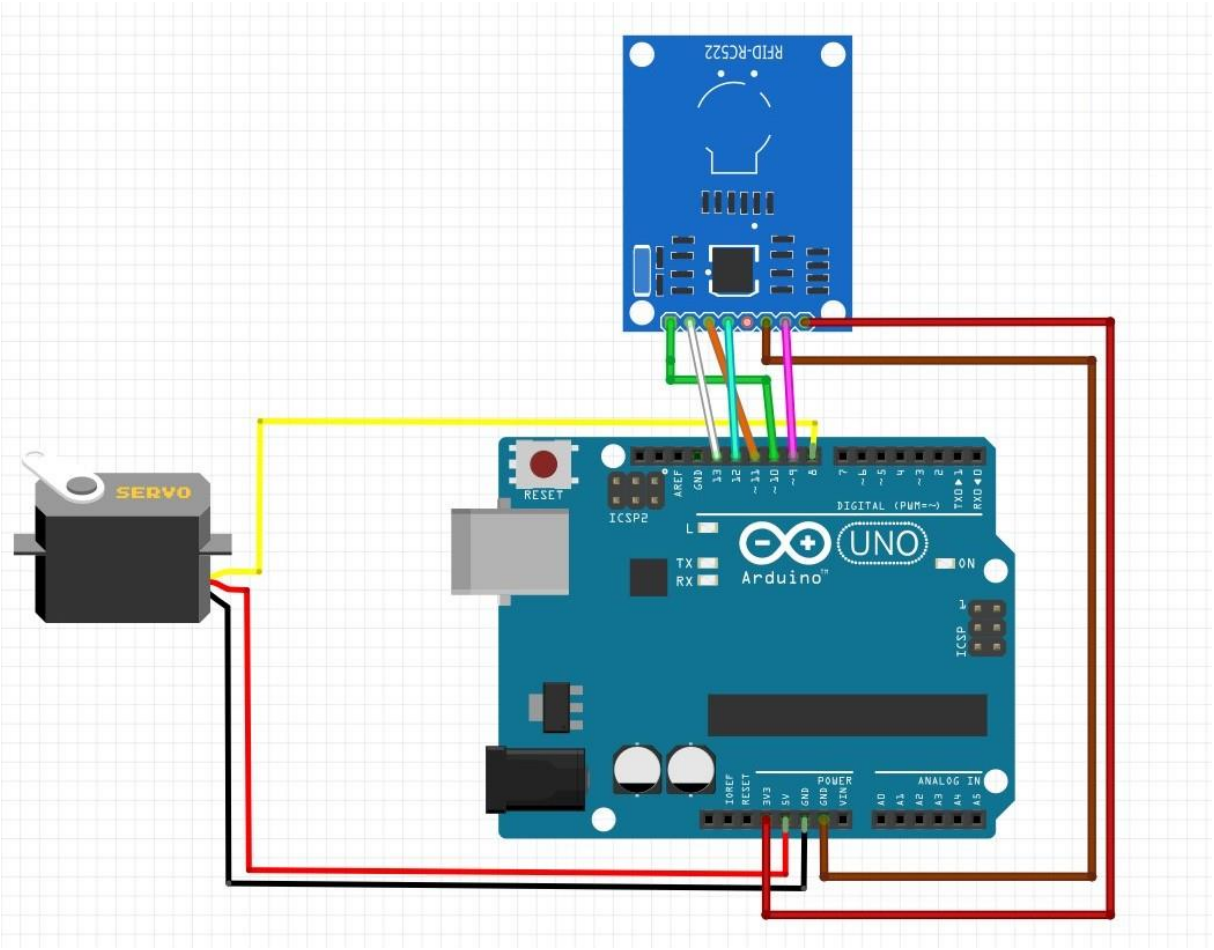
- Yandaki kod parçası Arduino modülünün çalıştırdığı kod parçasıdır.
- Bu kısımda void loop() kısmında görülen kod bölümü arayüzden Bariyer aç ve Bariyer Kapat butonlarının tıklanması ile gönderilen ‘0’ ve ‘1’ değerlerine göre motorun konumunun belirlendiği bölümdür.
- 0 değeri motorun 0° konumuna gelmesine sağlar.
- 1 değeri ise motorun 90° konumuna gelmesine sağlar.
- Kodun üst kısmında görüldüğü gibi servoPin 8 olarak belirlenmiştir. Bu servo motorun sinyal pininin Arduino üzerindeki PWM 8 pini olduğu anlamına gelir.
- RFID modülünün pinleri ise 10 ve 9 olarak belirlenmiştir. 10 Numaralı pin RFID nin SDA pini 9 ise Reset pinidir.
- “gelenVeri” olarak tanımlanan char tipindeki değişken bilgisayardan gönderilen 0 ve 1 değerlerini tutmaktadır.
- Void setup kısmında 9600 bant hızında seri haberleşme ve SPI haberleşme başlatılmıştır.
- Motor açılışta 0° konumuna yani bariyerin kapalı olduğu konuma getirilmiştir.
- Devamındaki bölümde ise RFID kartın okutulduğunda alınan ID verisinin kontrol edilmesi ve eğer doğru ise motorun 90° konumuna getirilmesi sağlanmıştır.

```
#include <SPI.h>
#include <RFID.h>
#include <Servo.h>
char gelenVeri;
int servoPin = 8;
Servo motor;
RFID lrt720(10, 9);

void setup() {
  Serial.begin(9600);
  SPI.begin();
  lrt720.init();
  motor.attach(servoPin);
  motor.write(0);


}

void loop() {
  //motor.write(0);
  if(Serial.available()>0)
  {
    gelenVeri = Serial.read();
    if(gelenVeri == '1'){
      motor.write(90);
      }
     else if(gelenVeri =='0'){
      motor.write(0);
      }
     }
```

```
if (lrt720.isCard()) {
  if (lrt720.readCardSerial()) {
    Serial.println("Kart Bulundu ID: ");
    char cardID[5];
    sprintf(cardID, "%02X%02X%02X%02X%02X", lrt720.serNum[0],
    lrt720.serNum[1], lrt720.serNum[2], lrt720.serNum[3],
    lrt720.serNum[4]);
    String cardIDString = String(cardID);
    Serial.println(cardIDString);

    if (strcmp(cardID, "C3964C958C") == 0) {
      motor.write(90);
      Serial.println("ID Doğru");
    } else {
      motor.write(0);
      Serial.println("ID Yanlış");
    }
  }
}
lrt720.halt();
```

4.ELEKTRONİK DEVRE

4.1.1 Devre Şeması;



- Görselde görünen sistemin kurulmuş devre şemasıdır.
- Şemada görünen Pinlerin bağlantı noktaları aşağıdaki gibidir.

SDA →PWM 10

SCK→ PWM 13

MOSI→PWM 11

MISO→PWM 12

IRQ→ BOŞ BIRAKILMIŞTIR

GND→ ARDUİNO GND

RST→ PWM 9

VCC→ ARDUİNO 3.3V

4.1.2 Devrede Kullanılan Elemanlar;

- **Arduino Uno:** Devrenin kontrolü için kullanılmıştır. Kullanılma nedeni kolay programlanması ve kaynak koduna erişim kolaylığıdır.
- **Servo 9 gr:** Test düzeneği olarak kullanılmıştır. Ucuzluğundan dolayı tercih edilmiştir.
- **RFID-RC522:** Devrede kullanılan RFID modülüdür. Modülün kullanılma nedeni olası aksaklıklarda müşterinin kendisine verilen RFID kartı ile bariyeri açması içindir.

(Arduino Uno)

(Micro Servo Motor)

(RFID Modülü)

(RFID Kart)

5.SONUÇ VE ÖNERİLER

Otomasyon sistemlerinin hayatımızdaki geniş alanı ile birlikte bu proje ile birlikte 24 saat başında birisinin bulunmasına ihtiyaç duymayan bir sistem geliştirdik. Bu sayede kaynakların daha doğru kullanımı ve zamandan tasarruf amaçlanmıştır. Sonuç olarak sistem bir otoparkın veyahut bir evin girişindeki bariyerinin RFID veya Görüntü işleme ile kontrolünü amaçlamış ve başarmıştır. Sistemin kullanıldığında ortaya çıkacak bazı sorunlar daha kaliteli malzemelerin kullanımı ile çözülecektir. Kod ve algoritmalar kısmında karşılaşılacak olan sorunlar ise düzeltilip mükemmele yakın bir proje elde edilebilir.

Proje video linki:
https://drive.google.com/file/d/1YuIt_TiXoFIny0noCO1awoVUBm_umYdU/view?usp=sharing

9.KAYNAKÇA

- https://www.youtube.com/watch?v=4BfvPCMvcPA
- https://www.youtube.com/watch?v=-LeuA7jIx8E&list=PLpmnLzMgTTpAucGwBMaLhcUOy60FN5-T8
- https://medium.com/@adem.akdogan/opencv-k%C3%BCt%C3%BCphanesi-ile-g%C3%B6r%C3%BCnt%C3%BC-i%CC%87%C5%9Fleme-uygulamal%C4%B1-af50033f7d8
- https://bayramblog.medium.com/python-opencv-ile-plaka-tan%C4%B1ma-plate-recognition-basit-ef97cf7a7c7c
- https://www.youtube.com/watch?v=4fJL8E36qSk
- https://www.youtube.com/watch?v=0u7GKpIIwQI
- https://www.youtube.com/watch?v=a31R73pkhcY
- https://www.youtube.com/watch?v=bQ4waLaxsTE