İÇİNDEKİLER

1. GİRİŞ

- Projenin fikri ve amacı otopark giriş çıkış işlemlerini daha hızlı hale getirmek ve otopark giriş çıkışlarının insan kontrolü olmadan, otonom bir sistem tarafından yapılmasını sağlamaktır.
- Giriş ya da çıkışta araç plakası ve aracın ön kısmının fotoğrafı çekilip saklanabilmektedir. Bir aracın kimliği plakası üzerinden kamera yardımıyla anlık görüntüsü alınarak ve görüntü işleme teknikleri kullanılarak yazılım ortamında tespit edilerek, plaka bilgisi ile zaman bilgisi birlikte kaydedilerek elektronik ortamda tutulabilmektedir.
- Projenin hayata geçirilebilmesi için hem yazılım hemde donanım gerekmektedir. Araç otoparka girerken ve çıkarken otomatik bariyer önünde bekletilerek, kamera ile plaka bölgesinin görüntüsü çekilmektedir.
- Projede kullanıcı ve müşteri arayüzü bulunmaktadır.

2. AMAÇ

- Görüntü işleme teknikleri kullanılarak resimdeki yazıyı elde etmek
- Elektronik ve yazılımı birlikte kullanmak
- Otopark giriş ve çıkış işlemlerini hızlandırmak
- Linux tabanlı işletim sistemleri ve açık kaynaklı yazılımlar ile çalışmak
- Arayüz tasarlamak
- Bilgilerin elektronik ortamda depolanması

3. YÖNTEM

- Windows'ta PyCharm programı ile Python programlama dilinde görüntü işleme ve arayüz programları yazılmıştır. Daha sonra kodlar Raspbian OS'ta Thonny IDE veya terminalde çalışması için düzenlenmiştir.
- Kameradan görüntü almak için breadboard, direnç, jumper kullanılmıştır, programı Raspbian OS'ta Thonny IDE ile yazılmıştır.
- Arayüz QTDesigner 4 ile sürükle bırak yöntemi ile tasarlanmıştır, daha sonra .ui uzantılı dosya Python kodlarına çevirilmiştir.
- Arayüz verilerini kaydetmek için .txt dosyaları kullanılmıştır.
- İki Raspberry'nin haberleşmesi için seri port kodları yazılmıştır fakat gerçeklenememiştir.

```
camera = PiCamera()
camera.rotation = 180
camera.resolution = (640, 480)
#camera.zoom = 0.15, 0.18, 0.59, 0.63
Dlyms(100)
if pm == True:
    print("Buton bekleniyor...")
while True:
    if pm == True:
        if  RPiCamBut() == True:
            print("snapshot")
            num = num+1
            camera.capture('/home/pi/Desktop/image%s.jpg' %num)
            Dlyms(100)
    else:
        break
camera.close()
```

```
def imgprocess(imp, gSS=None, gSC=None, gDT=None):
    gray = cv2.cvtColor(imp, cv2.COLOR_BGR2GRAY)  # gri tonlama yapılması
    gray = cv2.bilateralFilter(gray, gSS, gSC, gDT)  # gürültüyü azaltmak için bulanıklık filtesi
    edged = cv2.Canny(gray, 30, 200)  # kenar tespiti
# görüntüde kontur bulma
    cnts = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:10]
    screenCnt = None
    for c in cnts:
        peri = cv2.arcLength(c, True)   #kontur yaklaştırma
        approx = cv2.approxPolyDP(c, 0.045 * peri, True)
        #bulduğumuz kontur dört noktaya sahipse çerçevedir
        if len(approx) == 4:
            screenCnt = approx
            break
    if screenCnt is None:
        detected = 0
    else:
        detected = 1
```

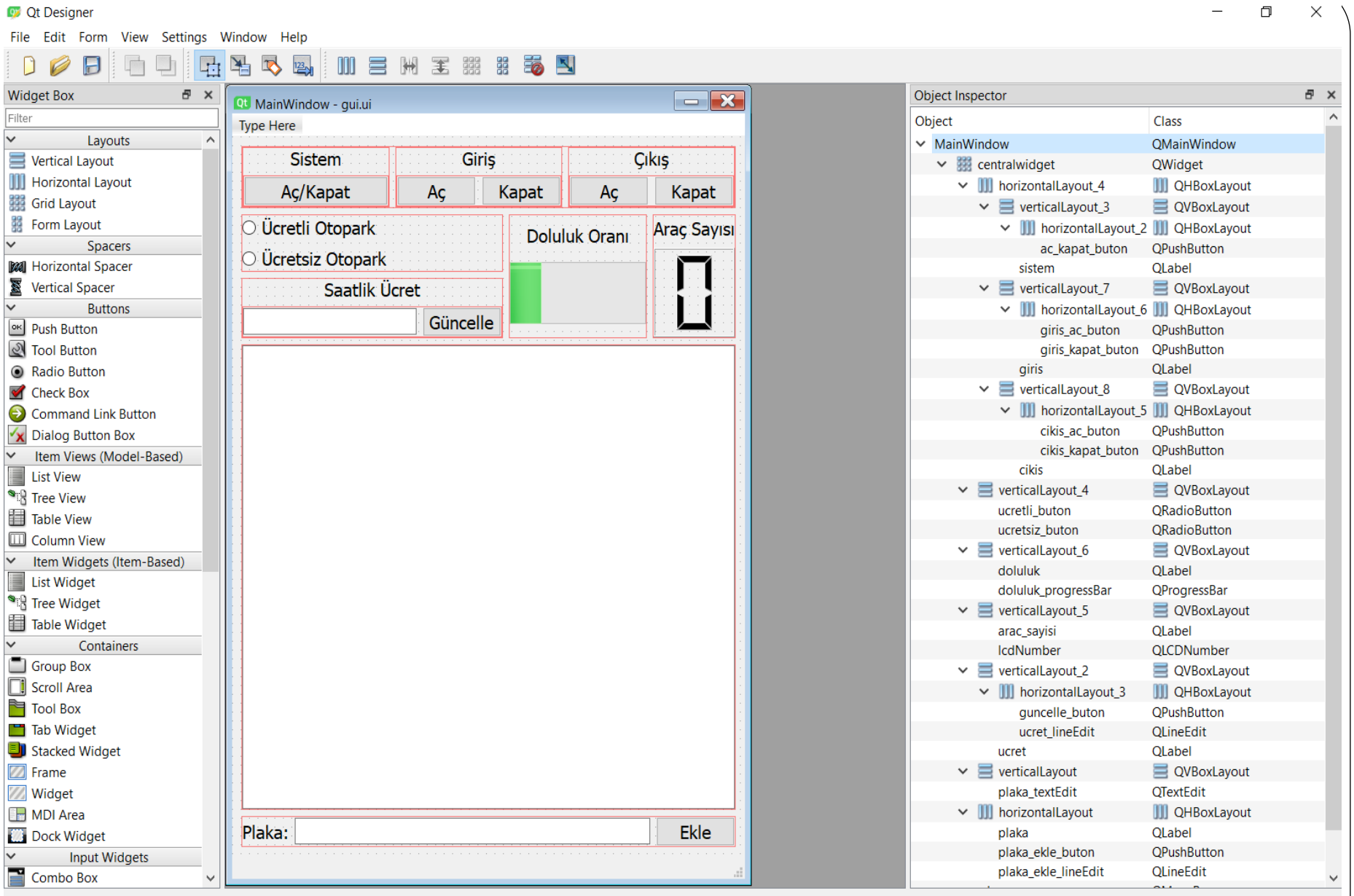
```
if detected == 1:
    cv2.drawContours(imp, [screenCnt], -1, (0, 255, 0), 3)

    # Plaka dışındaki alanın maskelenmesi
    mask = np.zeros(gray.shape, np.uint8)
    new_image = cv2.drawContours(mask, [screenCnt], 0, 255, -1)
    new_image = cv2.bitwise_and(img, img, 1, mask=mask)
    # kırpma işlemi
    (x, y) = np.where(mask == 255)
    (topx, topy) = (np.min(x), np.min(y))
    (bottomx, bottomy) = (np.max(x), np.max(y))

    Cropped = gray[topx:bottomx + 10, topy:bottomy + 10]
else:
    Cropped = None

return Cropped
```

```
def impsingleproc(srcs, lst):
    ims = srcs
    ps1 = int(lst[0])
    ps2 = int(lst[1])
    ps3 = int(lst[2])
    nims = imgprocess(ims, ps1, ps2, ps3 )
    if nims is not None:
        gri_H = 0
        gri_L = graylevel(nims)
        if gri_L > 20 and gri_L < 254:
            gri_H = gri_L + 2
            gri_L = gri_L - 20
        else:
            gri_H = 255
            gri_L = 127
        nims = cv2.threshold(nims, gri_L, gri_H, cv2.THRESH_BINARY, cv2.THRESH_OTSU)[1]
        texts = pytesseract.image_to_string(nims, config='ascii')
        assert isinstance(texts, object)
        hams = str(texts)
        if len(hams) > 3:
            # print("HNo:", hamx[:len(hamx) - 2])
            print("HNo:", hams)
            strx = plakaTR(hams)  # karakterleri filtrele
            if len(strx) > 3:
                # print(rgb, "-", p1, "-", p2, "-", p3, "HNo:", hamx)
                print("PNo:", strx)
                # print("PNo:", strx)
    return
```


Qt Designer
File Edit Form View Settings Window Help
Widget Box
Filter
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Form Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Command Link Button
Dialog Button Box
Item Views (Model-Based)
List View
Tree View
Table View
Column View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Group Box
Scroll Area
Tool Box
Tab Widget
Stacked Widget
Frame
Widget
MDI Area
Dock Widget
Input Widgets
Combo Box
MainWindow - gui.ui
Type Here
Sistem
Aç/Kapat
Giriş
Aç
Kapat
Çıkış
Aç
Kapat
Ücretli Otopark
Ücretsiz Otopark
Doluluk Oranı
Araç Sayısı
Saatlik Ücret
Güncelle
Plaka:
Ekle
Object Inspector
Object
Class
MainWindow QMainWindow
centralwidget QWidget
horizontalLayout_4 QHBoxLayout
verticalLayout_3 QVBoxLayout
horizontalLayout_2 QHBoxLayout
ac_kapat_buton QPushButton
sistem QLabel
verticalLayout_7 QVBoxLayout
horizontalLayout_6 QHBoxLayout
giris_ac_buton QPushButton
giris_kapat_buton QPushButton
giris QLabel
verticalLayout_8 QVBoxLayout
horizontalLayout_5 QHBoxLayout
cikis_ac_buton QPushButton
cikis_kapat_buton QPushButton
cikis QLabel
verticalLayout_4 QVBoxLayout
ucretli_buton QRadioButton
ucretsiz_buton QRadioButton
verticalLayout_6 QVBoxLayout
doluluk QLabel
doluluk_progressBar QProgressBar
verticalLayout_5 QVBoxLayout
arac_sayisi QLabel
lcdNumber QLCDNumber
verticalLayout_2 QVBoxLayout
horizontalLayout_3 QHBoxLayout
guncelle_buton QPushButton
ucret_lineEdit QLineEdit
ucret QLabel
verticalLayout QVBoxLayout
plaka_textEdit QTextEdit
horizontalLayout QHBoxLayout
plaka QLabel
plaka_ekle_buton QPushButton
plaka_ekle_lineEdit QLineEdit

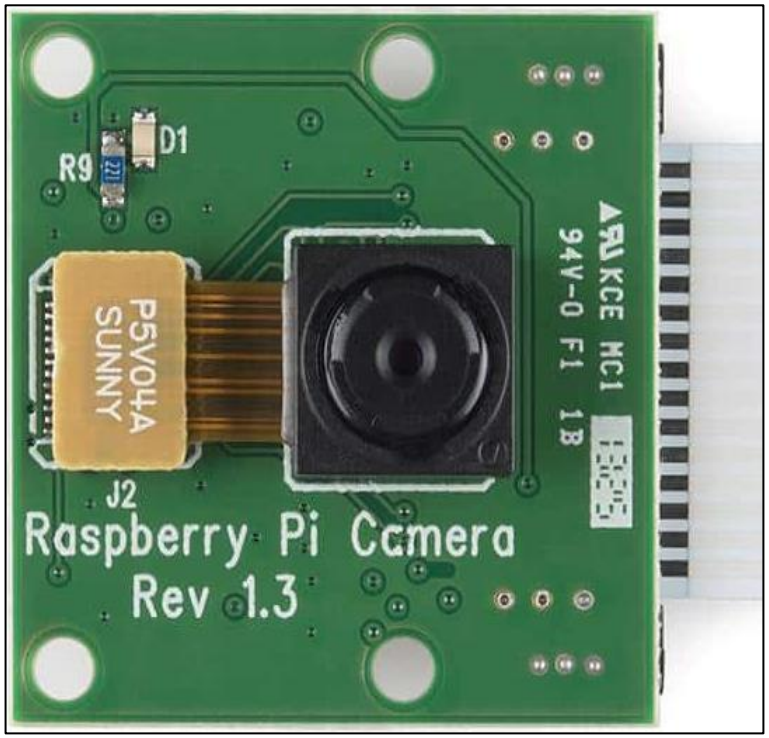
4. KULLANILAN MALZEMELER

4.1. Rasberry Pi 3 B+

Şekil 1. Raspberry PI 3 B+ (Anonim,2021b).

4.2. Raspberry Pi Camera Rev1.3

- Raspberry PI v1.3 kamera ile alınmıştır.



Şekil 2. Raspberry PI v1.3 kamera(Anonim,2021c).

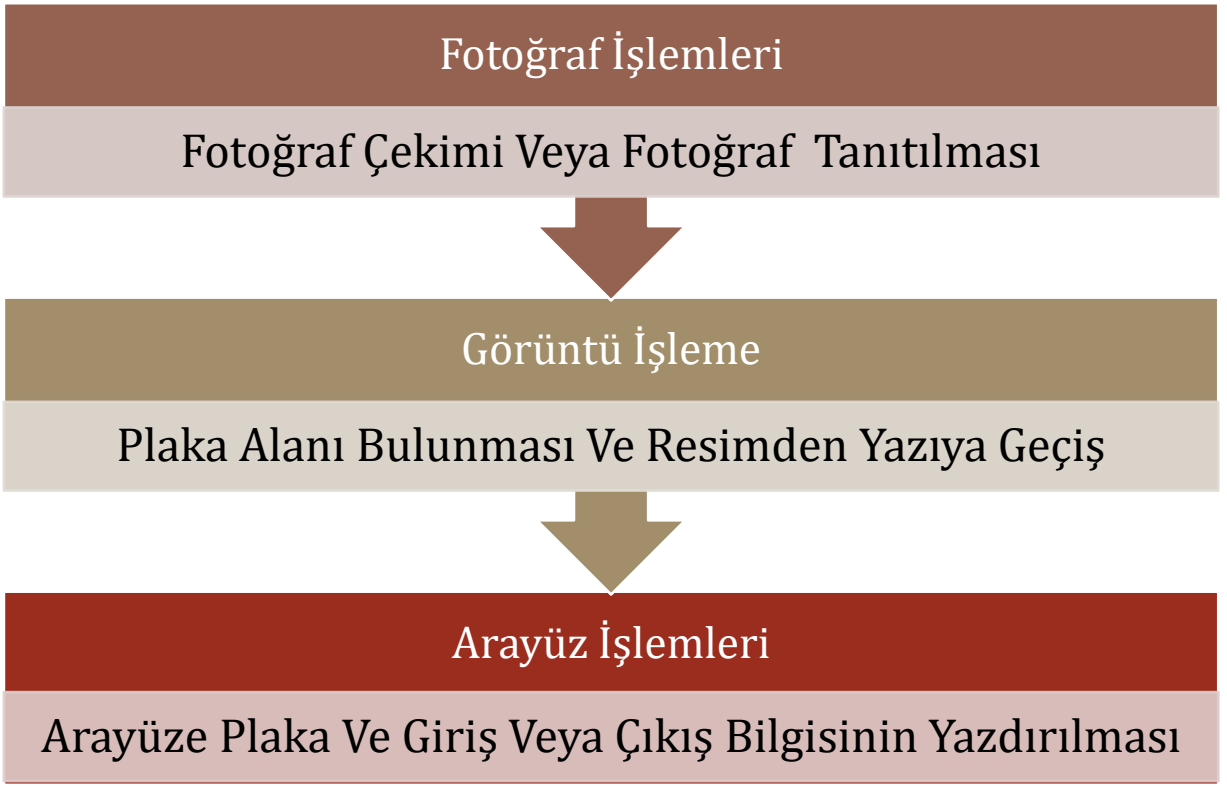
4.3. Powerbank Ve Elektronik Komponentler

- Taşınabilir olması için Raspberry Pi powerbank ile beslenmiştir. Fotoğraf çekmek için buton, direnç, jumper kablo ve breadboard kullanılmıştır. İşlemcinin ısınmaması içim fan kullanılmıştır.

Şekil 3. Çalışma Düzeneği

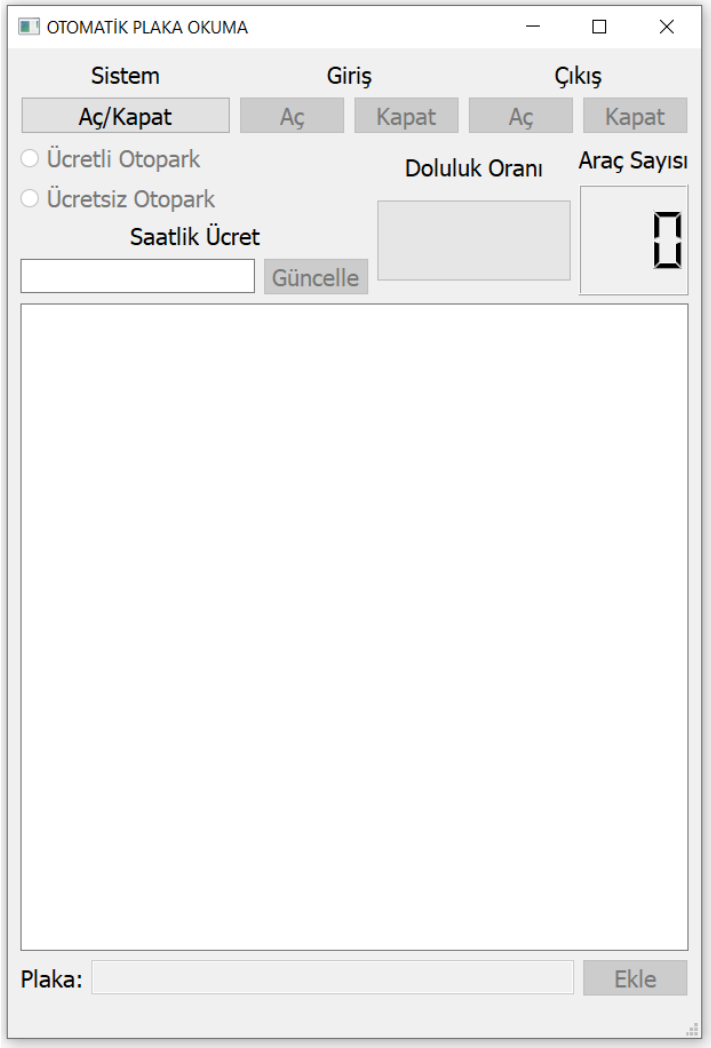
5. BLOK ŞEMASI

- Projede sistem durumunu, ayarlanan devir bilgisinin gösterimi ve motordan okunan devir bilgisinin gösterimi amacıyla kullanılmıştır.



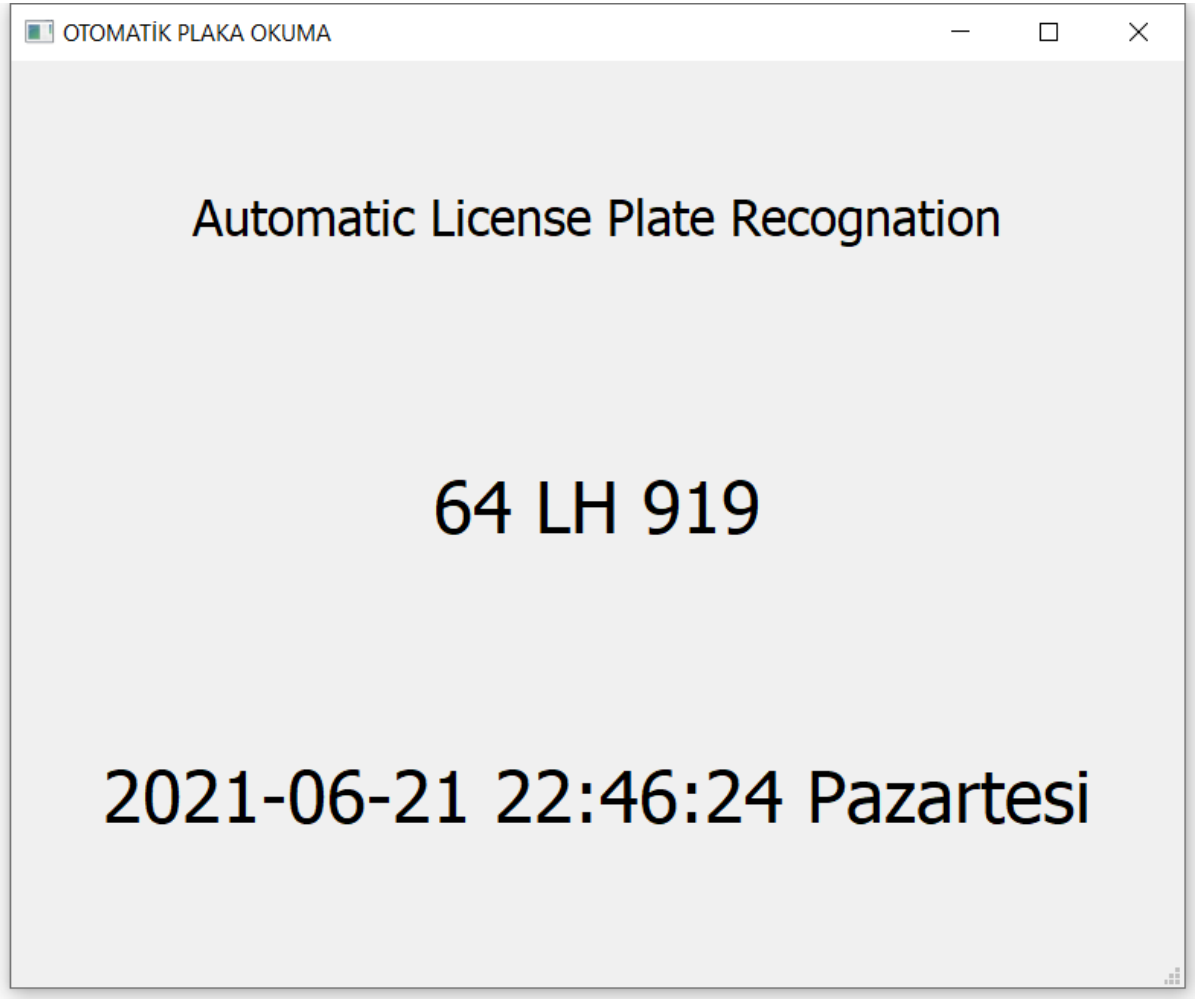
Şekil 4. Akış Diyagramı

6. ARAYÜZ



Şekil 5. Giriş Arayüzü

6. ARAYÜZ



Şekil 6. Çıkış Arayüzü

7. SONUÇ

- Otopark otomasyonu “otomatik plaka okuyan sistem” için uygulanabilirliği olan bir sistemdir. Yapılan çalışmada plaka okuma işlemi gerçekleştirilebilmiştir. Başarı oranı güvenli bir otopark otomasyonu için yeterli değildir.
- Görüntü kalitesi etkileyen unsurlardan ilki kullanılan kameranın kalitesi ve çözünürlüğüdür. İkincisi ise ortamdaki ışık yoğunluğudur. Işık yoğunluğu yetersiz kaldığında başarı çok azalmaktadır. Işığın resmin çekildiği alana dikey düşmemesi gerektiği gözlenmektedir. Işığın yüzeylere dağınık olarak gelmesi gerekmektedir
- Görüntünün elde edilmesinden kaynaklı sorunlar yazılım ortamında kısmen düzeltilebilmektedir. Bu durum işlem süresini uzatabilmektedir.
- Araç üzerindeki logo vb. şekiller kontur tespitinde hatalara neden olabilmektedir. Plakanın araca tutturulduğu vidaların doğru yerde olmamaları veya bandroller hatalı okumaya neden olabilmektedir.

DİNLEDİĞİNİZ İÇİN TEŞEKKÜRLER...