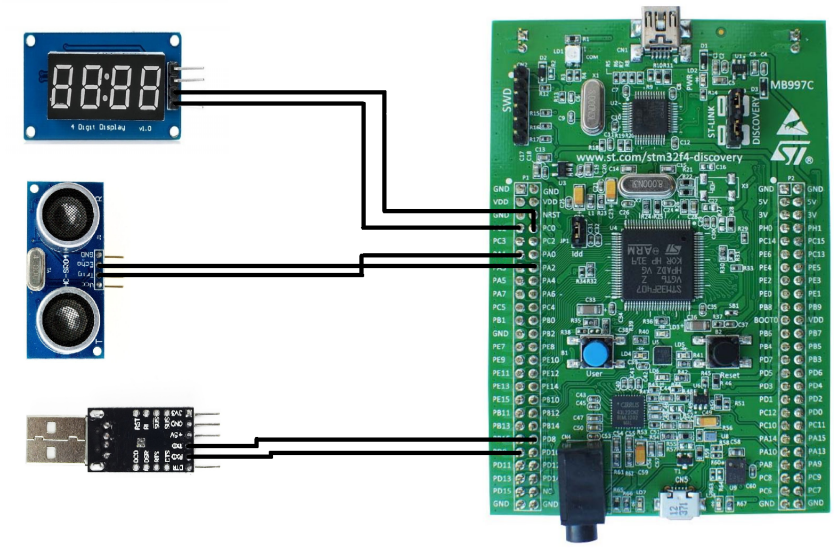
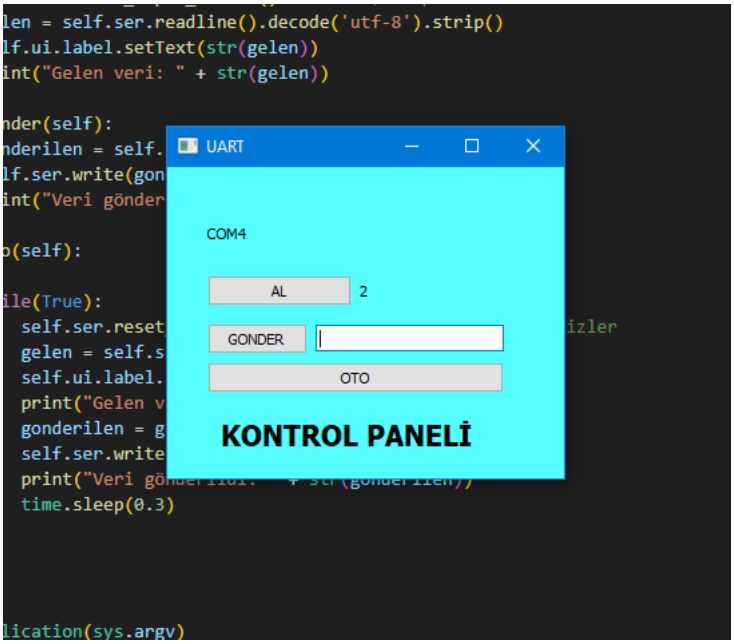
KONTROL PANELİ

Stm32f407vgt6 discovery board ile UART haberleşmesi yapılmaktadır. HC-SR04 ile mesafe ölçümü yapılıp ölçülen değeri bilgisayara göndererek bilgisayarın mikrodenetleyiciye komut göndermesi planlanmıştır. Ölçülen mesafe değerine göre 3 adet led belirli değer aralikları için yanmaktadır ve 4 digit display üzerinden mesafe kullanıcıya gösterilmektedir.



Python PyQt5 ile bir kullanıcı arayüzü tasarlanıp bu arayüzden veri akışı kontrol edilmektedir. Yine python ile port okuması ve port üzerinden veri göndermesi yapılmaktadır.



Kullanıcı arayüzü

Main kodu

SeriHaberlesme.py

```python
import sys, serial, time
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import QApplication, QMainWindow
from pencere import Ui_MainWindow

class pencere(QMainWindow):                                 # QMainWindow'dan türetilen GUI için ana sınıf

    def __init__(self):
        super().__init__()
        self.ui = Ui_MainWindow()                           # Kullanıcı arayüzünü başlatma
        self.ui.setupUi(self)                               # Arayüz setup
        self.setToolTip("DevamEt!!")                        # fare sabit kaldığında altında çıkan yazı

        self.ui.pushButton.clicked.connect(self.verial)     # butona basıldığında gidilen fonksiyon
        self.ui.pushButton_2.clicked.connect(self.gonder)   # butona basıldığında gidilen fonksiyon
        self.ui.pushButton_3.clicked.connect(self.oto)      # butona basıldığında gidilen fonksiyon

        self.ser = serial.Serial(       # seri haberleşme
        port='COM4',                    # Linux'ta /dev/ttyUSB4 veya /dev/ttyS4 olabilir
        baudrate=115200,                # veri akış hızı
        timeout=2                       # bekleme
        )

        if self.ser.is_open:                                # port açıkmı
            port = str(self.ser.name)                       # port adını alır
            self.ui.label_2.setText(port)                   # port adını label üzerine yazar

    def verial(self):                                       # tek seferde veri almak için
        self.ser.reset_input_buffer()                       # Giriş tamponunu temizler
        gelen = self.ser.readline().decode('utf-8').strip() # porttan gelen veriyi okur
        self.ui.label.setText(str(gelen))                   # veriyi label üzerine yazar
        print("Gelen veri: " + str(gelen))                  # terminale yazar
```

```python
    def gonder(self):                               # tek seferde veri göndermek için
        gonderilen = self.ui.lineEdit.text() + "."  # verinin sonuna . koyar, stm32'in verinin bittiğini anlaması için
        self.ser.write(gonderilen.encode('utf-8'))  # port üzerinden veriyi gönderir
        print("Veri gönderildi: " + str(gonderilen))# terminale yazar

    def oto(self):                                  # veri alış verişini otomatik yapar

        while(True):
            self.ser.reset_input_buffer()           # Giriş tamponunu temizler
            gelen = self.ser.readline().decode('utf-8').strip()  # porttan gelen veriyi okur
            self.ui.label.setText(str(gelen))       # veriyi label üzerine yazar
            print("Gelen veri: " + str(gelen))      # terminale yazar
            gonderilen = gelen +  "."               # gelen veriyi noktalar
            self.ser.write(gonderilen.encode('utf-8'))  # port üzerinden veriyi gönderir
            print("Veri gönderildi: " + str(gonderilen))# terminale yazar
            time.sleep(0.3)                         # 0.3 saniye bekler, gözlemleme için

app = QApplication(sys.argv)    # PyQt5 uygulamasını başlatma
win = pencere()                 # Ana pencerenin bir örneğini oluşturma
win.show()                      # main window gösterme
app.exec_()                     # Uygulamanın ana döngüsünü yürütme
```

Arayüz kodu

pencere.py

```python
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(297, 233)
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(85, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(85, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
        brush = QtGui.QBrush(QtGui.QColor(85, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(85, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
        MainWindow.setPalette(palette)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label_4 = QtWidgets.QLabel(self.centralwidget)
        self.label_4.setGeometry(QtCore.QRect(40, 190, 221, 21))
        font = QtGui.QFont()
        font.setPointSize(16)
        font.setBold(True)
        font.setWeight(75)
        self.label_4.setFont(font)
        self.label_4.setObjectName("label_4")
        self.widget = QtWidgets.QWidget(self.centralwidget)
        self.widget.setGeometry(QtCore.QRect(30, 30, 222, 139))
        self.widget.setObjectName("widget")
        self.verticalLayout_3 = QtWidgets.QVBoxLayout(self.widget)
        self.verticalLayout_3.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout_3.setObjectName("verticalLayout_3")
        self.verticalLayout_2 = QtWidgets.QVBoxLayout()
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        self.verticalLayout = QtWidgets.QVBoxLayout()
        self.verticalLayout.setObjectName("verticalLayout")
        self.label_2 = QtWidgets.QLabel(self.widget)
        self.label_2.setObjectName("label_2")
```

```python
        self.verticalLayout.addWidget(self.label_2)
        self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_2.setObjectName("horizontalLayout_2")
        self.pushButton = QtWidgets.QPushButton(self.widget)
        self.pushButton.setObjectName("pushButton")
        self.horizontalLayout_2.addWidget(self.pushButton)
        self.label = QtWidgets.QLabel(self.widget)
        self.label.setObjectName("label")
        self.horizontalLayout_2.addWidget(self.label)
        self.verticalLayout.addLayout(self.horizontalLayout_2)
        self.horizontalLayout = QtWidgets.QHBoxLayout()
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.pushButton_2 = QtWidgets.QPushButton(self.widget)
        self.pushButton_2.setObjectName("pushButton_2")
        self.horizontalLayout.addWidget(self.pushButton_2)
        self.lineEdit = QtWidgets.QLineEdit(self.widget)
        self.lineEdit.setObjectName("lineEdit")
        self.horizontalLayout.addWidget(self.lineEdit)
        self.verticalLayout.addLayout(self.horizontalLayout)
        self.verticalLayout_2.addLayout(self.verticalLayout)
        self.pushButton_3 = QtWidgets.QPushButton(self.widget)
        self.pushButton_3.setObjectName("pushButton_3")
        self.verticalLayout_2.addWidget(self.pushButton_3)
        self.verticalLayout_3.addLayout(self.verticalLayout_2)
        MainWindow.setCentralWidget(self.centralwidget)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "UART"))
        self.label_4.setText(_translate("MainWindow", "KONTROL PANELİ"))
        self.label_2.setText(_translate("MainWindow", "TextLabel"))
        self.pushButton.setText(_translate("MainWindow", "AL"))
        self.label.setText(_translate("MainWindow", "0000"))
        self.pushButton_2.setText(_translate("MainWindow", "GONDER"))
        self.pushButton_3.setText(_translate("MainWindow", "OTO"))
```

QtDesigner ile hazırlandı

Stm32 kodu

Değişkenler

```c
char Uart_Data[10] = "";    // string çevirme için

uint8_t rx_buffer[20];      // 10 baytlık gelen veri
uint8_t rx_index = 0;       // Sıradaki byte konumu
uint8_t rx_byte;            // Tek baytlık okuma
uint8_t data_ready = 0;     // veri durumu

uint32_t time;              // HCSR04 ile ölçülen süre
uint16_t distance;          // ölçülen sürenin 58' bölünmesi ile elde edilen mesafe (cm)

int gelen_sayi = 1111;      // port üzerinden gelen veri için değisken
```

HC-SR04 okuması

```c
uint32_t read_hcsr04(){                              // HC-SR04 okuması

    uint32_t local_time=0;                           // Yankının dönüs süresini (mikro saniye cinsinden) tutacak sayaç değiskeni

    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, 1);         // TRIG pini (örneğin PA1)
    DWT_Delay_us(10);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, 0);

    while(!HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_2));     // ECHO pini 1 olana kadar bekle (yankı baslasın)

    while(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_2)){      // ECHO pini 1 olduğu sürece
        local_time++;                                // Her 1 mikro saniyede sayaç artırılır
        DWT_Delay_us(1);                             // 1 mikro saniye bekle
    }

    return local_time;                               // Ölçülen süreyi geri döndür

}
```

Init kodları

```c
  MX_GPIO_Init();
  MX_USART3_UART_Init();
  MX_TIM1_Init();
  /* USER CODE BEGIN 2 */
  //------------------------------------------------------------------------------------------------
  __HAL_UART_ENABLE_IT(&huart3, UART_IT_RXNE);   // UART RX kesmesi aktif hale geldi
  HAL_UART_Receive_IT(&huart3, &rx_byte, 1);     // UART3'ten 1 byte'lık veri almak için kesmeli alım baslatılıyor

  DWT_Delay_Init();                              // DWT zamanlayıcıyı baslat (mikro saniyelik hassasiyetli delay için)

  tm1637Init();                                  // TM1637 4-digit 7 segment display modülünü baslat

  tm1637SetBrightness(8);                        // Ekran parlaklığı maksimum seviyeye ayarlandı
  tm1637DisplayDecimal(1111, 1);                 // 1111 sayısı ekrana yazıldı, noktalar aktif
  //------------------------------------------------------------------------------------------------
  /* USER CODE END 2 */
```

While döngüsü

```c
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    //------------------------------------------------------------------------------------------------

    time = read_hcsr04();                                   // Yankının dönüs süresi mikro saniye cinsinden alınır
    distance = time / 58;                                   // Mesafe cm cinsine çevrilir (yaklasık 343 m/s sese göre dönüs süresi üzerinden)
    HAL_Delay(100);

    sprintf(Uart_Data, "%u\n", distance);                   // String'e çevir

    HAL_UART_Transmit(&huart3, Uart_Data, strlen(Uart_Data) , 10);    // UART3 üzerinden veriyi gönder

    //------------------------------------------------------------------------------------------------

    if (data_ready)                                         // Eğer UART üzerinden bir veri geldi ve tam veri tamamlandıysa
    {
        data_ready = 0;                                     // Veri islendi, bayrağı sıfırla
        rx_buffer[rx_index] = '\0';                         // Buffer'ı C string formatına çevir (null-terminate)

        gelen_sayi = atoi((char*)rx_buffer);                // Gelen string veriyi integer'a çevir


        if (gelen_sayi >= 1 && gelen_sayi <= 10)
        {
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);
        }

        else if (gelen_sayi >= 11 && gelen_sayi <= 20)
        {
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);
        }

        else if (gelen_sayi >= 21 && gelen_sayi <= 30)
        {
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);
        }

        else
        {
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET);
        }

        memset(rx_buffer, 0, sizeof(rx_buffer));            // Tüm buffer temizlenir
        rx_index = 0;
    }

    tm1637SetBrightness(8);
    tm1637DisplayDecimal(gelen_sayi, 1);                    // UART'tan gelen sayıyı ekranda göster

    //------------------------------------------------------------------------------------------------
```

UART Callback fonksiyonu

```c
//----------------------------------------------------------------------------------------------------
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){     // UART kesmesiyle her bir karakter alındığında çağrılan callback fonksiyonu

    if (huart->Instance == USART3)                           // Bu callback yalnızca USART3 için çalısacak sekilde sınırlandırılmıs
    {
        if (rx_byte == '.')                                  // Nokta karakteri geldiğinde verinin tamamlandığı varsayılır
        {
            data_ready = 1;                                  // Ana döngüde islenecek veri hazır bayrağı
        }
        else
        {
            if (rx_index < sizeof(rx_buffer) - 1)            // Buffer tasması önleniyor, boyut sınırı kontrol ediliyor
            {
                rx_buffer[rx_index++] = rx_byte;             // Alınan karakteri buffer'a ekle
            }
        }

        HAL_UART_Receive_IT(&huart3, &rx_byte, 1);           // Bir sonraki karakter alımı baslatılır
    }
}
//----------------------------------------------------------------------------------------------------
```

Baudrate = 115200
Word Length = 8 Bits
Stop Bits = 1
PD8 (TX) = RX
PD9 (RX) = TX
PC0 (Output) = CLK (4 digit display)
PC1 (Output) = DIO (4 digit display)
PA1 (Output) = Trig (HC-SR04)
PA2 (Input) = Echo (HC-SR04)
Clock = HSE - 168MHz