RAPOR

Proje Katılımcıları

21220030052	Cuma Aktaş	İkinci ESP Kodları ve Kontrolü
20155018	Mehmet Emin Gündüzlü	Sensör Denetçisi
20155046	Fürkan Üzüm	Birinci ESP Kodları ve Kontrolü
20155016	Ömer Faruk Özalp	Görüntü İşleme Kodları ve Kontrolü

Sonuçlar

1. Kamera ile Parmak Sayısı Tespiti
   - Bilgisayarın dahili kamerası kullanılarak gerçek zamanlı görüntü işleme gerçekleştirilir.
   - OpenCV kütüphanesi ile elde edilen görüntüde sadece eldeki parmakların olduğu alan (ROI) işlenir. ROI (Region of Interest), sadece eldeki parmakların yer aldığı kısmı ifade eder
   - Görüntü gri tonlamaya dönüştürülür, bulanıklaştırılır ve threshold yöntemiyle ikili bir görüntü haline getirilir.
   - Konturlar analiz edilerek parmakların sayısı tespit edilir.

2. ESP32'ye Parmak Sayısı Gönderimi
   - Tespit edilen parmak sayısı, ESP32 cihazının IP adresine(`http://172.20.10.2/led`) HTTP GET isteği ile gönderilir.
   - Bu sayede ESP32 cihazı, bilgisayar tarafından gönderilen veriyi alır ve işleyebilir. Parmak sayısı, ESP32 cihazına şu URL üzerinden HTTP GET isteği ile gönderilir: http://172.20.10.2/led?count=3. Buradaki 'count=3' değeri, tespit edilen parmak sayısını temsil eder.

3. ESP32'nin Parmak Sayısını İşlemesi
   - ESP32, gelen parmak sayısına göre bağlı olduğu donanımları kontrol eder.
   - ESP32, gelen sayıya göre her bir LED'in durumunu değiştirir. Örneğin, 3 parmak tespit edilirse, 3. LED'i yakar.
   - Aynı zamanda, bu sayı 7-segment display üzerinde rakamsal olarak görüntülenir. Her bir segment pini doğru şekilde kontrol edilerek sayı ekrana yansıtılır.

4. Verilerin Web Sitesine Aktarılması
   - ESP32 cihazı, tespit edilen parmak sayısını bir URL'ye (`https://vyamacli.com/iot/lab/datatofile.php`) HTTP GET isteği ile gönderir.
   - Web sitesi, alınan parmak sayısını veritabanına kaydeder ve kullanıcıların web sitesi üzerinden bu veriye erişmesini sağlar ve kayıt işlemlerini gerçekleştirir.
   - Bu işlem sayesinde, tespit edilen parmak sayısı hem donanım hem de çevrimiçi bir ortamda görüntülenir.

5. Sonuçların Görselleştirilmesi

- Parmak sayısına göre LED'lerin yanması ve 7-segment display'de rakamların görüntülenmesi, verilerin fiziksel bir çıktısı olarak kullanılır.
- Web sitesine gönderilen veriler ise, kullanıcıların uzaktan erişimle sonuçları görmesine olanak sağlar.

6. Eş Zamanlı İkinci ESP32'nin Kullanımı

- İkinci bir ESP32 cihazı aynı URL'den parmak sayısı bilgisini alır ve kendi bağlı olduğu RGB LED, buzzer ve DHT11 sensörü gibi bileşenleri kontrol eder.
- Bu cihaz, parmak sayısına bağlı olarak RGB LED ile belirli renkleri yakar, buzzer ile ses çıkışı verir ve çevresel sıcaklık/nem değerlerini ölçer.

7. Projenin Genel Çalışma Akışı

- Kamera, parmak sayısını tespit eder ve veriyi ESP32 cihazına gönderir.
- ESP32, bu bilgiyi alır, LED'leri ve 7-segment display'i kontrol eder.
- Veriler, belirlenen bir URL'ye gönderilerek hem donanımsal hem de çevrimiçi olarak kullanıcıya sunulur.

Görüntü İşleme Kodları

```
import cv2
import requests
import numpy as np
import time
from threading import Thread
ESP32_URL = "http://172.20.10.2/led"  # ESP32'nin IP adresi
class CameraStream:
    def __init__(self, src=0):
        self.cap = cv2.VideoCapture(src)
        self.cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
        self.cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)
        self.ret, self.frame = self.cap.read()
        self.running = True
        self.thread = Thread(target=self.update, args=())
        self.thread.daemon = True
        self.thread.start()
    def update(self):
        while self.running:
            self.ret, self.frame = self.cap.read()
    def read(self):
        return self.frame
    def stop(self):
        self.running = False
        self.thread.join()
        self.cap.release()
def send_led_command(finger_count):
    try:
        response = requests.get(f"{ESP32_URL}?count={finger_count}")
        print(f"ESP32 Yanıt: {response.text}")
    except Exception as e:
        print(f"ESP32'ye komut gönderirken hata: {e}")
def process_frame(frame):
    # Görüntüyü yansıt
    frame = cv2.flip(frame, 1)
    roi = frame[50:250, 50:250]
    cv2.rectangle(frame, (50, 50), (250, 250), (0, 255, 0), 2)
    gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (15, 15), 0)
    _, thresh = cv2.threshold(blur, 60, 255, cv2.THRESH_BINARY_INV +
cv2.THRESH_OTSU)
```

```
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    finger_count = 0
    if contours:
        max_contour = max(contours, key=cv2.contourArea)
        if cv2.contourArea(max_contour) > 5000:
            hull = cv2.convexHull(max_contour, returnPoints=False)
            defects = cv2.convexityDefects(max_contour, hull)
            if defects is not None:
                for i in range(defects.shape[0]):
                    s, e, f, d = defects[i, 0]
                    start = tuple(max_contour[s][0])
                    end = tuple(max_contour[e][0])
                    far = tuple(max_contour[f][0])
                    a = np.linalg.norm(np.array(start) - np.array(end))
                    b = np.linalg.norm(np.array(start) - np.array(far))
                    c = np.linalg.norm(np.array(end) - np.array(far))
                    angle = np.degrees(np.arccos((b**2 + c**2 - a**2) /
(2 * b * c)))
                    if angle < 90 and d > 10000:
                        finger_count += 1
                        cv2.circle(roi, far, 5, (0, 0, 255), -1)
            finger_count += 1

    # Parmak sayısını ekrana yaz
    cv2.putText(frame, f"Parmak Sayisi: {finger_count}", (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)

    return frame, finger_count
camera = CameraStream(0)
while True:
    frame = camera.read()
    if frame is None:
        continuee
    processed_frame, finger_count = process_frame(frame)
    cv2.imshow("Frame", processed_frame)
    send_led_command(finger_count)
    time.sleep(2)  # Veri gönderimleri arasında 2 saniye gecikme
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
camera.stop()
cv2.destroyAllWindows()
```

Birinci ESP Kodları

```
#include <WiFi.h>
#include <WebServer.h>
#include <HTTPClient.h>
const char* ssid = "Fürkan";
const char* password = "1234abcd";
WebServer server(80); // Port 80 üzerinden çalışacak
const int leds[] = {16, 17, 18, 19, 21}; // LED pinleri
const int segmentPins[] = {25, 26, 27, 32, 33, 4, 5}; // A, B, C, D, E, F, G
String fingerCount = ""; // Gelen parmak sayısı
const int digits[10][7] = {
 {1, 1, 1, 1, 1, 1, 0}, // 0
 {0, 1, 1, 0, 0, 0, 0}, // 1
 {1, 1, 0, 1, 1, 0, 1}, // 2
 {1, 1, 1, 1, 0, 0, 1}, // 3
 {0, 1, 1, 0, 0, 1, 1}, // 4
 {1, 0, 1, 1, 0, 1, 1}, // 5
 {1, 0, 1, 1, 1, 1, 1}, // 6
 {1, 1, 1, 0, 0, 0, 0}, // 7
 {1, 1, 1, 1, 1, 1, 1}, // 8
 {1, 1, 1, 1, 0, 1, 1}  // 9
};
void setup() {
 Serial.begin(115200);
 for (int i = 0; i < 5; i++) {
   pinMode(leds[i], OUTPUT);
   digitalWrite(leds[i], LOW);
 }
 for (int i = 0; i < 7; i++) {
   pinMode(segmentPins[i], OUTPUT);
   digitalWrite(segmentPins[i], LOW);
 }
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
   delay(1000);
   Serial.println("WiFi'ye bağlanıyor...");
 }
 Serial.println("WiFi'ye bağlandı!");
 Serial.print("IP Adresi: ");
 Serial.println(WiFi.localIP());
 server.on("/led", HTTP_GET, handleLedRequest);
 server.begin();
 Serial.println("HTTP Sunucu başlatıldı!");
}
void loop() {
 server.handleClient();
}
void handleLedRequest() {
 if (server.hasArg("count")) {
   fingerCount = server.arg("count");
   Serial.println("Parmak sayisi alindi: " + fingerCount);
   int count = fingerCount.toInt();
   for (int i = 0; i < 5; i++) {
     if (i < count) {
       digitalWrite(leds[i], HIGH);
     } else {
       digitalWrite(leds[i], LOW);
     }
```

```
    }
    if (count >= 0 && count <= 9) {
      for (int i = 0; i < 7; i++) {
        digitalWrite(segmentPins[i], digits[count][i]);
      }
    }
    String response = sendDataToServer(fingerCount);
    Serial.println("Site Yanıtı: " + response);
    server.send(200, "text/plain", "Parmak sayisi alindi: " + fingerCount);
  } else {
    server.send(400, "text/plain", "Parametre eksik!"); // Eksik parametre hatası
  }
}
String sendDataToServer(String count) {
  String payload = "";
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    String url = "https://vyamacli.com/iot/lab/datatofile.php?fname=grup84&sayi=" +
count;
    http.begin(url);
    int httpResponseCode = http.GET();
    if (httpResponseCode > 0) {
      payload = http.getString();
      Serial.println("Veri gönderildi: " + count);
      Serial.println("HTTP Yanıt Kodu: " + String(httpResponseCode));
    } else {
      Serial.println("Veri gönderme hatası: " + String(httpResponseCode));
    }
    http.end();
  } else {
    Serial.println("WiFi bağlantısı yok!");
  }
  return payload;
}
```

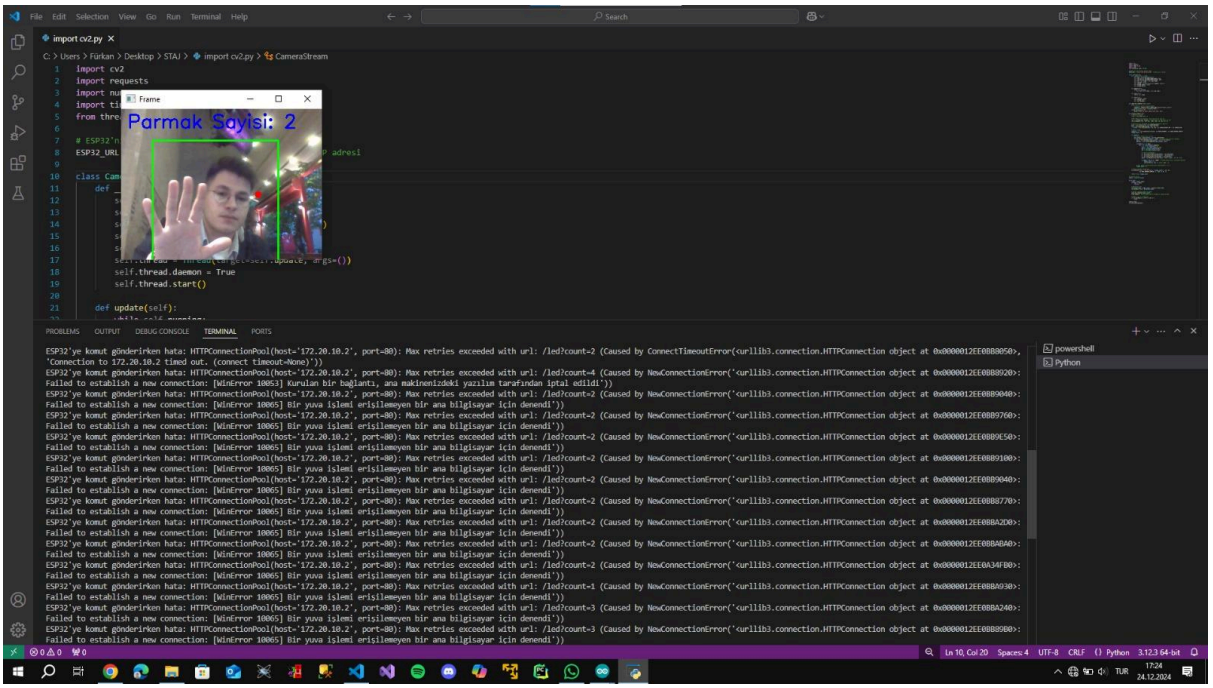
İkinci ESP Kodları

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
const char* ssid = "Ankesör";
const char* password = "covid_10";

#define DHTPIN 25
#define BUZZER_PIN 27
#define LED_RED_PIN 33
#define LED_GREEN_PIN 12
#define LED_BLUE_PIN 13
#define TRIG_PIN 26
#define ECHO_PIN 14


#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
 Serial.begin(115200);


 pinMode(BUZZER_PIN, OUTPUT);
 pinMode(LED_RED_PIN, OUTPUT);
 pinMode(LED_GREEN_PIN, OUTPUT);
 pinMode(LED_BLUE_PIN, OUTPUT);
 pinMode(TRIG_PIN, OUTPUT);
 pinMode(ECHO_PIN, INPUT);
 pinMode(DHTPIN, INPUT);


 digitalWrite(BUZZER_PIN, LOW);

 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
   delay(1000);
   Serial.println("WiFi'ye bağlanıyor...");
 }
 Serial.println("WiFi'ye bağlandı!");

  dht.begin();
 Serial.println("DHT11 ile sıcaklık ve nem ölçümü başlıyor...");
```

```
}

void loop() {
  delay(2000);

 if (WiFi.status() == WL_CONNECTED) {
   HTTPClient http;
   http.begin("https://vyamacli.com/iot/lab/datafromfile.php?fname=grup84");
   int httpResponseCode = http.GET();
   if (httpResponseCode > 0) {
     String payload = http.getString();
     Serial.println("Alınan veri: " + payload);

     int receivedValue = payload.toInt();
     handleIncomingData(receivedValue);

   } else {
     Serial.println("Veri alma hatası: " + String(httpResponseCode));
   }
   http.end();
 }
}
void handleIncomingData(int value) {
 digitalWrite(LED_RED_PIN, LOW);
 digitalWrite(LED_GREEN_PIN, LOW);
 digitalWrite(LED_BLUE_PIN, LOW);
 digitalWrite(BUZZER_PIN, LOW);

 float distance = 0;

 switch (value) {
   case 1:
     Serial.println("Kırmızı LED yanıyor.");
     digitalWrite(LED_RED_PIN, HIGH);
     break;

   case 2:
     Serial.println("Yeşil LED yanıyor ve buzzer aralıklı çalışıyor.");
     digitalWrite(LED_GREEN_PIN, HIGH);
     tone(BUZZER_PIN, 1000, 500);
     break;

   case 3:
     Serial.println("Mavi LED yanıyor. DHT11 sıcaklık ve nem değerleri:");
     digitalWrite(LED_BLUE_PIN, HIGH);
```

```
      readDHT11();
      break;

    case 4:
      Serial.println("Kırmızı ve Yeşil LED yanıyor. Mesafe sensörü çalışıyor.");
      digitalWrite(LED_RED_PIN, HIGH);
      digitalWrite(LED_GREEN_PIN, HIGH);
      distance = readUltrasonicSensor();
      Serial.print("Mesafe: ");
      Serial.print(distance);
      Serial.println(" cm");
      break;

    default:
      Serial.println("Bilinmeyen veri.");
      break;
  }
}
float readUltrasonicSensor() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH);
  float distance = duration * 0.034 / 2;
  return distance;
}

void readDHT11() {
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("DHT11 veri okuma hatası!");
    return;
  }

  Serial.print("Nem: ");
  Serial.print(humidity);
  Serial.println(" %");
  Serial.print("Sıcaklık: ");
  Serial.print(temperature);
  Serial.println(" °C");
}
```

Görüntü İşleme Ekran Görüntüsü



Birinci ESP Videosu

https://youtube.com/shorts/_X-tBcpf5Og?feature=share

İkinci ESP Videosu

https://youtu.be/iaPInkb580s