SELÇUK ÜNİVERSİTESİ

# ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ


**HAZIRLAYAN**

| | | |
|---|---|---|
| **Adı Soyadı** | **:** | Bilge Kaan Öztürk |
| **Öğrenci Numarası** | **:** | 213302002 |

# 1. İÇİNDEKİLER

# 2. GENEL BİLGİ

Bu proje, araçların otoparka giriş ve çıkışlarını kontrol edebilen, park alanlarının doluluk durumunu izleyebilen ve sistemle uzaktan haberleşme imkânı sunan bir Akıllı Otopark Sistemi tasarımını kapsamaktadır. Sistemde, araç algılama için ultrasonik sensörler, giriş-çıkış kontrolü için servo motorlar ve sistemin temel kontrolü için Arduino mikrodenetleyicisi kullanılmıştır.

Projenin uzaktan izlenebilirliğini ve gelişmiş veri haberleşmesini sağlamak amacıyla sisteme ESP32 mikrodenetleyicisi entegre edilmiştir. ESP32, Arduino ile haberleşerek verileri işler, gerektiğinde kullanıcıya iletir veya dış sistemlerle bağlantı kurar. Böylece otoparkın doluluk durumu uzaktan izlenebilir hale gelmiştir.

Bu sistem, düşük maliyetli bileşenlerle, şehir içi veya özel tesislerdeki otopark yönetimini daha verimli ve kullanıcı dostu hale getirmeyi amaçlamaktadır.

# 3. PROJE HAKKINDA BİLGİLER

## 3.1. Kullanılan Malzeme ve Ekipman

| Malzeme Adı | Adet | Görevi / Açıklama |
|---|---|---|
| ARDUİNO UNO | 1 | Sistemin ana kontrolcüsü; sensör verilerini işleyip servo motorları kontrol eder. |
| ESP32 | 1 | Arduino ile haberleşerek verileri işler ve uzaktan iletişim sağlar. |
| HC-SR04 ULTRASONİK SENSÖR | 2 | Giriş ve çıkış noktalarında araç algılamak için kullanılır. |
| TCRT5000 KIZILÖTESİ SENSÖR | 6 | Park alanlarının doluluk durumunu tespit etmek için kullanılır. |
| SG90 SERVO MOTOR | 2 | Otopark bariyerlerini kontrol eder (açma-kapama). |
| LED (KIRMIZI) | 6 | Parkın dolu/boş durumunu göstermek için kullanılır. |
| DİRENÇ (220Ω - 330Ω) | 7 | LED'lerin güvenli çalışması için.1 ader 10K direnç ESP32'nin Arduino ile güvenli iletişimi için Arduino TX,ESP RX pinine bağlanmıştır. |
| JUMPER KABLO | - | Tüm bileşenleri bağlamak için kullanılır. |
| USB KABLOSU | 2 | Arduino ve ESP32'nin programlanması ve güç verilmesi için. |

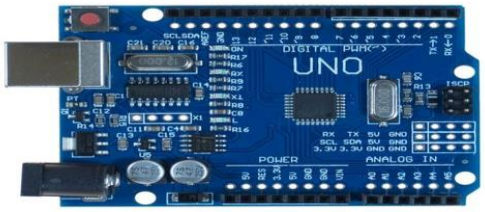
## 3.2. Projenin Mekanik Bileşenleri

| Bileşen Adı | Açıklama |
|---|---|
| **Bariyer Kolu (Kaldırma kolu)** | Servo motorla entegre çalışan ve araç geçişini fiziksel olarak engelleyen kol sistemidir. Genellikle hafif plastik ya da pleksi malzemeden yapılır. |
| **Servo Montaj Aparatları** | Servo motorun bariyere ve gövdeye sabitlenmesini sağlayan L tipi ya da özel üretilmiş plastik/metal bağlantı elemanlarıdır. |
| **TCRT5000 Sensör Sabitleme Kasası** | Park alanlarındaki kızılötesi sensörlerin doğru konumlandırılması için kullanılan küçük taşıyıcı yapılar. 3D baskı veya plastik parçalar olabilir. |
| **ESP32 ve Arduino Koruma Kutusu** | Elektronik kartların dış etkenlerden korunması için kullanılan kutu veya muhafaza kabı. |
| **Montaj Zemini (Mukavva)** | Tüm sistemi taşıyan ana yüzey. Projede hafiflik amacıyla mukavva malzeme tercih edilmiştir. |
| **Kablo Kanalı** | Kabloların düzenli şekilde döşenmesini ve bağlantı güvenliğini sağlar. |

## 3.3. Projenin Elektronik Donanımı

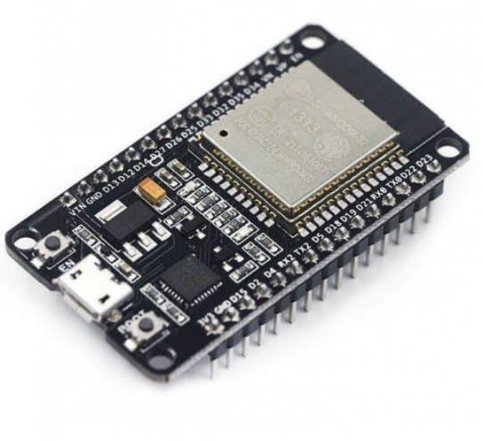
- **Arduino Uno:**

Projenin ana kontrol ünitesi olarak görev yapar. Sensörlerden gelen analog ve dijital verileri okur, işleyerek servo motor ve LED kontrollerini sağlar.



- **ESP32:**

Arduino ile haberleşerek verilerin seri haberleşme yoluyla iletilmesini sağlar. Ayrıca verilerin uzaktan izlenmesi ve kontrolü için IOT aracı olarak kullanulmıştır.



- **HC-SR04 Ultrasonik Sensörler:**

Otopark giriş ve çıkış noktalarında kullanılır. Araçların varlığını mesafe ölçümü yaparak algılar.Trigger pinine gelen sinyal ile gönderilen ses dalgasının dönüş süresi echo pininin verdiği çıktı ile hesaplanır ve buna göre bir aracın varlığı tespit edilir.


- **TCRT5000 Kızılötesi Sensörler:**

Her bir park alanının dolu veya boş olduğunu algılamak için kullanılır. Sensörler, zemine yerleştirilen ışık yansımasını ölçerek aracın varlığını tespit eder.

- **SG90 Servo Motorlar:**

Otopark giriş ve çıkış bariyerlerinin açılıp kapanmasını sağlar. Arduino tarafından kontrol edilir.


- **LED'ler (Kırmızı):**

Park alanlarının tamamen dolu olduğunu görsel olarak belirtmek için kullanılır.

- **Dirençler:**

LED'lerin ve sensörlerin güvenli çalışması için gerekli dirençler devrede yer alır.

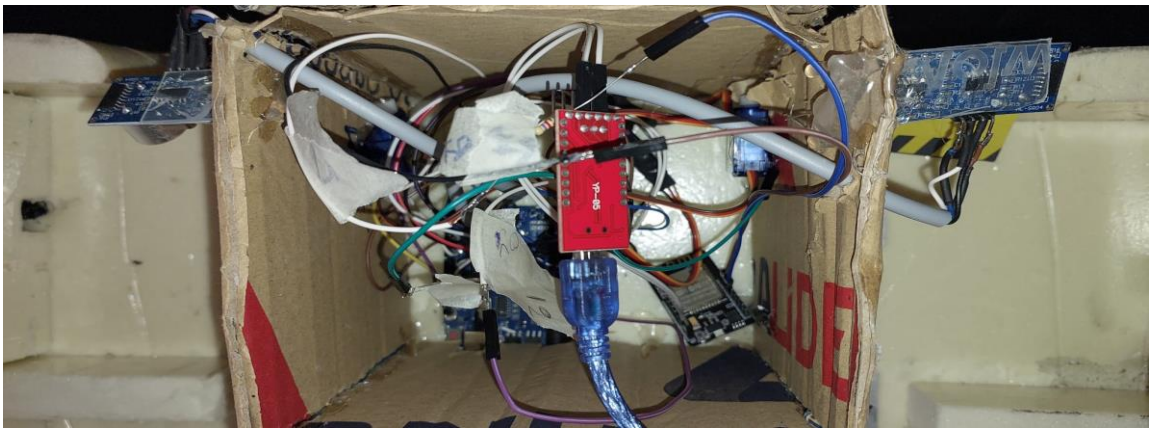
# 4.PROJE MİMARİSİ

## 4.1.Projenin Elektronik Mimarisi

Projenin elektronik mimarisi, Arduino Uno'ya bağlı altı adet TCRT5000 sensörü ile park yerlerinde araç tespiti yapılmasını sağlar. Ayrıca, iki adet HC-SR04 ultrasonik sensör sayesinde otoparkın giriş ve çıkış noktalarındaki barikatların kontrolü gerçekleştirilir. Bu barikatlar, ultrasonik sensörlerden gelen sinyallere bağlı olarak SG90 servo motorlar aracılığıyla kaldırılır ve araç geçişi tamamlanana kadar kapılar açık konumda bekler.


Otopark içerisindeki doluluk durumu ise, her park alanına yerleştirilen TCRT5000 kızılötesi sensörler aracılığıyla tespit edilir. Bu sensörlerden elde edilen veriler doğrultusunda, dolu olan park alanları kırmızı LED ile gösterilir. LED'lerin Arduino üzerinden sağlıklı şekilde çalışabilmesi için devreye 220 ohm'luk dirençler eklenmiştir.

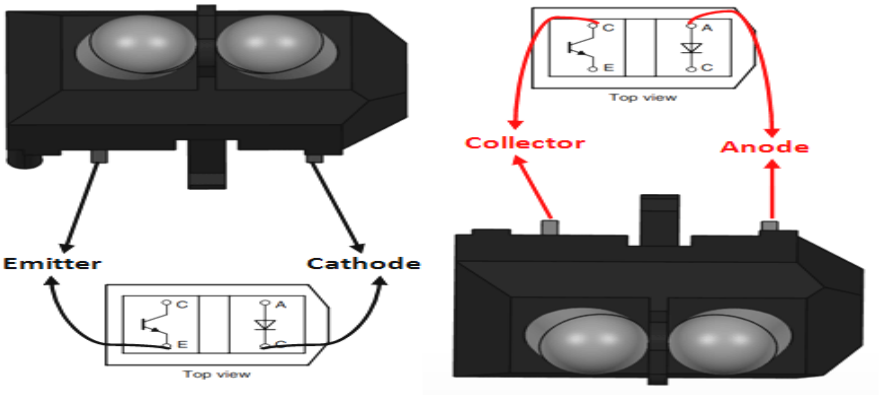

Arduino ile ESP32 arasında haberleşme sağlamak amacıyla, Arduino'nun "RX" pini ESP32'nin "TX" pinine, Arduino'nun "TX" pini ise ESP32'nin "RX" pinine bağlanmıştır. Ancak iletişimin yalnızca tek yönlü olması gerektiğinden, yalnızca Arduino "TX" – ESP32 "RX" bağlantısı kullanılması yeterlidir. ESP32'nin UART haberleşmesinde 3.3V ile çalışmasına karşın, Arduino'nun 5V ile çalışması nedeniyle, ESP32'nin zarar görmesini önlemek adına bağlantı hattına 10k ohm'luk bir direnç yerleştirilmiştir.



### 4.2.Projenin Yazılım Mimarisi

Bu projede kullanılan iki mikrodenetleyici de Arduino IDE ile programlanabildiği için, geliştirme sürecinde tek bir IDE kullanılması yeterli olmuştur. Arduino Uno'nun kodlanmasında, sensörlerin gerçek zamanlı, doğru ve senkronize şekilde çalışması önemli bir gereklilik olmuştur.

TCRT5000 kızılötesi sensörün yapısı gereği; VCC ucu anoda, GND ucu katoda, analog çıkış ucu ise sensör içindeki fototransistörün kollektörüne bağlanmıştır. Bu yapı sayesinde, sensör sürekli olarak kızılötesi ışın gönderir. Gönderilen ışının yüzeyden yansıma miktarına bağlı olarak, fototransistörün beyzine ulaşan ışık miktarı değişir ve buna göre elektrik akışına izin verilip verilmediği belirlenir.




Bu prensip doğrultusunda, kod yapımız ve donanım bağlantılarımız bu davranışa uygun şekilde tasarlanmıştır. Her bir TCRT5000 sensörüne karşılık gelecek şekilde bir adet LED atanmış ve bu LED'ler aracılığıyla her park yerinin durumu görsel olarak gösterilmiştir.

```
// TCRT5000 sensör ve LED pinleri
const int tcrtPins[6] = {A0, A1, A2, A3, A4, A5}; // TCRT5000 sensör pinleri
const int ledPins[6]  = {3, 4, 2, 6, 5, 7};       // Her sensöre karşılık gelen LED
```

Ardından, giriş ve çıkış barikatlarında kullanılan HC-SR04 ultrasonik sensörler için trig ve echo pinlerinin bağlantıları yapılmıştır. Trig pini, Arduino'dan gelen darbe sinyali ile bir ses dalgası gönderir. Bu dalga, önündeki nesneye çarpıp geri döndüğünde, echo pini üzerinden alınır.


**Vcc-** Connects to 5V of positive voltage for power
**Trig-** A pulse is sent here for the sensor to go into ranging mode for object detection
**Echo-** The echo sends a signal back if an object has been detected or not. If a signal is returned, an object has been detected. If not, no object has been detected.
**GND-** Completes electrical pathway of the power.

Sensörden gelen bu sinyale göre mesafe, “Mesafe = Ses hızı × Geçen zaman / 2” formülüyle hesaplanır. Buradaki “2” ifadesi, ses dalgasının aynı mesafeyi hem gidiş hem de dönüş yönünde katetmesinden dolayı kullanılmaktadır.

```
// Ultrasonik sensör pinleri
const int trigGiris = 8;
const int echoGiris = 9;
const int trigCikis = 10;
const int echoCikis = 11;
```

Servo motorların giriş ve çıkış pinleri tanımlandıktan sonra, altı adet park yeri için gerekli değişkenler belirlenir. Otopark sisteminin ilk kez enerjilenmesi sırasında tüm park yerlerinin dolu olarak görünmesini engellemek amacıyla, dolu park yerlerini takip eden “doluYer” adlı değişken başlangıçta 0 değerine eşitlenir.

```
// Servo motorlar
Servo servoGiris;
Servo servoCikis;

// Servo pinleri
const int servoGirisPin = 12;
const int servoCikisPin = 13;

int kapasite = 6;
int doluYer = 0;

```

UART iletişiminin 9600 baud hızında gerçekleşebilmesi için Arduino Uno'nun baud değeri bu hıza göre ayarlanır. Ardından, TCRT5000 sensörlerinin bağlı olduğu pinler giriş (input), bu sensörlere karşılık gelen LED pinleri ise çıkış (output) olarak tanımlanır. Aynı şekilde, HC-SR04 ultrasonik sensörlerin trig pinleri çıkış, echo pinleri ise giriş olarak ayarlanır. Giriş ve çıkış noktalarında

kullanılan servo motorlar da ilgili servo pinleri üzerinden tanımlanarak sisteme dahil edilir.

```
void setup() {
  Serial.begin(9600); // Hem bilgisayara hem ESP'ye veri gönderir

  // TCRT ve LED pin ayarları
  for (int i = 0; i < 6; i++) {
    pinMode(tcrtPins[i], INPUT);
    pinMode(ledPins[i], OUTPUT);
  }

  // Ultrasonik sensör pinleri
  pinMode(trigGiris, OUTPUT);
  pinMode(echoGiris, INPUT);
  pinMode(trigCikis, OUTPUT);
  pinMode(echoCikis, INPUT);

  // Servo motor bağlantıları
  servoGiris.attach(servoGirisPin);
  servoCikis.attach(servoCikisPin);
```

Ardından, döngü (loop) içerisinde TCRT5000 sensörlerinden gelen veriler okunur ve bu verilere göre her sensöre karşılık gelen LED'lerin durumu belirlenir. TCRT5000 sensörünün önüne bir cisim geldiğinde sensör LOW (0) değeri üretir. Bu durumda, ilgili LED'in bağlı olduğu çıkış pini HIGH (1) yapılır ve LED yanar. Döngü içerisinde kullanılan "i" değişkeni ile hangi sensör algılama yaptıysa, ona karşılık gelen LED'in aktif hale gelmesi sağlanır.

```
void loop() {
  doluYer = 0;
  String ledDurum = "";

  // TCRT5000 ile park yeri kontrolü
  for (int i = 0; i < 6; i++) {
    int tcrtValue = digitalRead(tcrtPins[i]);

    if (tcrtValue == LOW) {
      digitalWrite(ledPins[i], HIGH);
      doluYer++;
      ledDurum += "1";
    } else {
      digitalWrite(ledPins[i], LOW);
      ledDurum += "0";
    }
  }
```

Ardından, dolu park yerlerine ait veriler ikili (binary) formatta hem ESP32'ye hem de bağlıysa bilgisayara, UART seri iletişim yöntemiyle gönderilir.

```
// Durumu hem bilgisayara hem ESP'ye gönder
Serial.print("Dolu Yer: ");
Serial.println(doluYer);
Serial.println(ledDurum); // Örnek: 101100
```

**aracVarMi** fonksiyonu, giriş ve çıkış noktalarında araç olup olmadığını kontrol etmek için kullanılır. Bu fonksiyon, öncelikle trig pinini kısa süreli LOW yapar, ardından HIGH seviyesine çekip tekrar LOW seviyesine getirir. Trig pini HIGH olduğunda sensör ultrasonik bir darbe gönderir.

Gönderilen bu ses dalgasının bir cisme çarpıp geri dönüş süresi echo pini üzerinden ölçülür. Bu ölçüm kullanılarak mesafe hesaplanır ve mesafe belli bir eşik değerin altındaysa, o noktada araç olduğu sonucuna varılır.

```
// Ultrasonik sensör ile araç algılama
bool aracVarMi(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long sure = pulseIn(echoPin, HIGH, 20000);
  int mesafe = sure * 0.034 / 2;

  return (mesafe > 0 && mesafe < 15);
}
```

Projede kullanılan iki servo motordan biri ters bağlandığı için, kapı açma işlemi için iki farklı fonksiyon yazılmıştır. Birinci fonksiyon, servo motoru 15 ms aralıklarla enerjilendirerek açma hareketini gerçekleştirir; servo önce 10 dereceden 65 dereceye yavaşça hareket ettirilir, ardından 3 saniye bu konumda bekletilir ve sonrasında tekrar yavaş yavaş 10 dereceye geri döner.

```
// DÜZ çalışan servo motor için yavaş aç-kapat
void kapiyiYavasAcKapat(Servo &servoMotor) {
  for (int pos = 10; pos <= 65; pos++) {
    servoMotor.write(pos);
    delay(15);
  }

  delay(3000);

  for (int pos = 65; pos >= 10; pos--) {
    servoMotor.write(pos);
    delay(15);
  }
}
```

İkinci fonksiyon ise tam tersine çalışır; servo önce 65 dereceden 10 dereceye hareket ettirilir, 3 saniye beklenir ve ardından tekrar 65 dereceye geri döndürülür. Böylece, ters bağlanmış olan servo motor için uygun kapı açma-kapama hareketleri sağlanmış olur.

```
// TERS çalışan servo motor için yavaş aç-kapat
void kapiyiYavasAcKapatTers(Servo &servoMotor) {
  for (int pos = 65; pos >= 10; pos--) {
    servoMotor.write(pos);
    delay(15);
  }

  delay(3000);

  for (int pos = 10; pos <= 65; pos++) {
    servoMotor.write(pos);
    delay(15);
  }
}
```

Ardından, bu fonksiyonlar döngü (loop) içinde kullanılmıştır. Öncelikle, birinci fonksiyon araç algılandığında içerideki araç sayısını kontrol ederek, park yeri dolu değilse kapıyı açacak şekilde çağrılır. Buna karşın, çıkışta kullanılan ikinci fonksiyon ise içerideki araç sayısını kontrol etmeden, araç algılandığı anda kapıyı açar ve geçişe izin verir.

```
  // GİRİŞ KONTROLÜ
  if (aracVarMi(trigGiris, echoGiris)) {
    if (doluYer < kapasite) {
      Serial.println("Giriş: Araç algılandı ve boş yer var.");
      kapiyiYavasAcKapatTers(servoGiris);
    } else {
      Serial.println("Giriş: Kapasite dolu, kapı açılmıyor.");
    }
  }

  // ÇIKIŞ KONTROLÜ
  if (aracVarMi(trigCikis, echoCikis)) {
    Serial.println("Çıkış: Araç algılandı.");
    kapiyiYavasAcKapat(servoCikis);
  }

  delay(500);
}
```

ESP kodları içinse öncelikle kullanacağımız wifi ismini ve şifresini tanımlıyoruz.

```
1 #include "esp_camera.h"
2 #include <WiFi.h>
3 #include <WebServer.h>
4
5 const char* ssid = "BUM";
6 const char* password = "bum12321";
```

Ardından web server başlatılır ve park durumu “000000” ayarlanır bu sayede güç verildiği anda park yerleri son kalan bilgiler gösterilmez.

```
WebServer server(80);

// Gelen veri: P1, P2, P3, P4, P5, P6
// Gösterim sırası: P5, P4, P6, P3, P1, P2 --> A1, A2, A3, B1, B2, B3
String parkDurumu = "000000";  // Başlangıçta tüm park yerleri boş
```

HTML ve CSS kodlamaları ile web server'ımızın önce 300ms'de bir yenilenemesini bu sayede verilerin sürekli güncel kalmasını sağlıyoruz.Sonraki satırlar sırası ile sekmenin ismi ve başlığı tanımlanır.

```
void handleRoot() {
  String html = "<!DOCTYPE html><html><head><meta charset='utf-8'>";
  html += "<meta http-equiv='refresh' content='0.3'>";  // Sayfayı 300ms'de bir yenile
  html += "<title>Park Durumu</title>";
  html += "<style>body{font-family:Arial;text-align:center;} .box{display:inline-block;width:100px;height:100px;margin:10px;font-size:20px;line-height:100px;border-radiu
  html += "<h1>Bilge Kaan Öztürk</h1>";

```

Park yerlerinin görsel sıralamasına göre bir liste oluşturulur ve bu liste, anlık park durumu verilerini içeren parkDurumu değişkenine göre güncellenir. parkDurumu değişkenindeki veriler, ikili (binary) formatta durum değişkenine aktarılır. Daha sonra, durum değişkeninin her bir binary basamağı ayrılarak, her park yerine karşılık gelen kutucuklara (görsel göstergelere) yüklenir. Bu sayede, her bir park yerinin dolu veya boş olduğu bilgisi görsel arayüzde doğru şekilde temsil edilir.

```
  // A1-A2-A3-B1-B2-B3 görsel sırasına göre: P5, P4, P6, P3, P1, P2
  int goruntuSirasi[6] = {4, 3, 5, 2, 0, 1};
  String label[6] = {"A1", "A2", "A3", "B1", "B2", "B3"};

  for (int i = 0; i < 6; i++) {
    char durum = parkDurumu[goruntuSirasi[i]];
    String sinif = (durum == '1') ? "dolu" : "bos";
    html += "<div class='box " + sinif + "'>" + label[i] + "</div>";
  }

  html += "</body></html>";
  server.send(200, "text/html", html);
}
```

ESP32'nin Access Point (AP) modunda çalışması sağlanarak, cihazın kendi Wi-Fi ağı üzerinden bağlantı kurabilmesi amaçlanmıştır. Serial.begin(9600); komutu ile seri haberleşme başlatılır ve veri aktarım hızı 9600 baud olarak ayarlanır. WiFi.softAP(ssid, password); satırı, ESP32'nin belirtilen SSID (ağ adı) ve şifre ile bir kablosuz erişim noktası oluşturmasını sağlar. Ardından, WiFi.softAPIP(); fonksiyonu ile ESP32'nin erişim noktası IP adresi alınır ve seri monitöre yazdırılır. server.on("/", handleRoot); satırı, sunucuya gelen ana sayfa isteğini ("/") handleRoot adlı işleyici fonksiyona yönlendirir. Son olarak, server.begin(); ile web sunucusu başlatılarak istemcilerin bağlantı kurmasına olanak tanınır.

```
void setup() {
  Serial.begin(9600);

  // ESP32'yi Access Point modunda başlat
  WiFi.softAP(ssid, password);
  IPAddress IP = WiFi.softAPIP();
  Serial.print("ESP32 IP Adresi: ");
  Serial.println(IP);

  server.on("/", handleRoot);
  server.begin();
}

```

ESP32'nin sürekli olarak çalıştığı loop() fonksiyonu ilk olarak, server.handleClient(); komutu sayesinde gelen istemci (web tarayıcısı) istekleri dinlenir ve yanıtlanır. Ardından, Arduino'dan gelen UART verisi kontrol edilir. Serial.available() fonksiyonu ile seri portta veri olup olmadığı kontrol edilir ve varsa Serial.readStringUntil('\n') komutu ile gelen veri satır sonuna kadar okunur. trim() fonksiyonu ile gereksiz karakterler (örneğin \n veya \r) temizlenir. Sonrasında, alınan verinin 6 karakter uzunluğunda olup olmadığı ve içerisinde en az bir '0' ya da '1' karakteri bulunup bulunmadığı kontrol edilir. Bu şartları sağlayan veri parkDurumu adlı değişkene atanır ve seri monitöre "Yeni Park Verisi: " ifadesiyle birlikte yazdırılır. Bu yapı sayesinde ESP32, Arduino'dan gelen park yeri bilgilerini düzenli olarak alır ve günceller.

```
48 void loop() {
49   server.handleClient();
50
51   // Arduino'dan gelen UART verisini oku
52   if (Serial.available()) {
53     String gelenVeri = Serial.readStringUntil('\n');
54     gelenVeri.trim();  // \n, \r gibi karakterleri temizle
55
56     // Gelen veri 6 karakter uzunluğunda ve sadece 0/1 içermelidir
57     if (gelenVeri.length() == 6 && gelenVeri.indexOf('1') != -1 || gelenVeri.indexOf('0') != -1) {
58       parkDurumu = gelenVeri;
59       Serial.println("Yeni Park Verisi: " + parkDurumu);
60     }
61   }
62 }
```

## 5. PROJE SONUÇLARI VE YORUMLANMASI

Bu proje kapsamında geliştirilen Arduino ve ESP32 tabanlı Akıllı Otopark Sistemi, düşük maliyetli elektronik bileşenlerle park yönetimini otomatikleştirmeyi başarmıştır. Sistemde kullanılan TCRT5000 sensörleri, her bir park alanının doluluk durumunu başarıyla tespit etmiş; HC-SR04 ultrasonik sensörleri ile araç giriş-çıkış kontrolü etkin biçimde gerçekleştirilmiştir.

Servo motorlarla entegre edilen bariyer sistemi, araç algılandığında otomatik olarak açılıp kapanarak giriş ve çıkışı düzenlemiştir. Sistem, ESP32 üzerinden Arduino ile haberleşerek park durumu verilerini başarıyla iletmiş ve uzaktan izlenebilir bir yapı oluşturulmuştur.

Sonuç olarak, sistem;

- Araç varlığını doğru bir şekilde algılayabilmiş,
- Doluluk durumuna göre LED uyarısı verebilmiş,
- Giriş ve çıkış noktalarını otomatik şekilde kontrol edebilmiş,

- Donanımlar arası haberleşmeyi sorunsuz gerçekleştirmiştir.

Bu proje, küçük ölçekli otoparklarda kolayca uygulanabilecek, esnek ve ekonomik bir çözüm sunmaktadır. Geliştirilmeye açık yapısıyla, mobil uygulama entegrasyonu, web arayüzü ya da plakadan tanıma gibi özelliklerle genişletilmesi mümkündür.

## 6.KAYNAKÇA


- https://www.youtube.com
- https://www.chatgpt.com
- https://www.selcuk.edu.tr/Birim/Bolum/teknoloji-elektrik_elektronik_muhendisligi/15640/muhendislik-tasarimi-dersi-belgeleri/56424
- https://www.robotistan.com