



**Mühendislik Fakültesi**  
**Bilgisayar Mühendisliği Bölümü**  
**Gömülü Sistemler Dersi**  
**Proje Raporu**

Proje Konusu
Akıllı Otopark Sistemi

Öğrenci Bilgileri	
Öğr. No	21100011027
Ad Soyad	Senanur İncekara

<b>Doç. Dr.</b> <b>Muhammed KARAALTUN</b>
--

**Haziran 2024**  
**Konya**

## Projede Aranılan Özellikler

Gömülü Sistemler dersi Proje Ödevi olarak her öğrenci Arduino tabanlı basit bir sistem geliştirecektir. Sistemde mutlaka görsel yazılım kullanılacaktır.

### ➤ Genel Özellikler:

- Arduino Tabanlı Sistem bağlantı şeması için fritzing paket programı kullanılacaktır.
- ADC, DAC, Timer, (Zamanlayıcı), EEPROM, Kütüphane. Projenizde bu özelliklerden en az 3 adet kullanılmalı
- Arduino yazılımı geliştirmek için Arduino veya (visual studio, visual C++, vb.) programlama dili kullanılabilir. Kodlar düzgün bir şekilde raporun Ekler bölümünün altında yer alacaktır.

### ➤ Önerilen Projeler:

- Şehir aydınlatma sistemi veya Ev otomasyon sistemi, bu sistemde (ADC, DAC, Timer, (Zamanlayıcı), EEPROM) kullanılacaktır, arduinoya bağlı olan aygıtların bilgileri veri tabanına kaydedilecektir ve aygıtların bilgileri görsel olarak görsel yazılım kullanarak tasarlanacaktır.

## Proje Raporu Hazırlama Kuralları

Her öğrenci proje ödevini aşağıda belirlenen kurallara göre hazırlayacaktır.

### 1. Rapor Bölümleri

- Kapak
- İçindekiler
- Özet
- Projede kullanılan donanım ve yazılımla ilgili bilgileri (vermiş olduğum notlardaki gibi) Donanımda kullanılan malzemelerin açıklaması, özellikleri ve kullanım şekli gibi bilgiler. Yazılımda kullanılan komutların ve fonksiyonların açıklaması.
- Kaynaklar
- Ekler (Kodlar)

### 2. Konu Anlatımı

- Etik kurallara uygun olarak, öğrenci konuyu kendi cümleleri ile sade bir şekilde anlatmalıdır. Yararlanılan kaynaklar belirtilmelidir.

### 3. Sayfa Düzeni

- Kenar Boşlukları: 2.5 cm
- Sayfa Numarası: Sağ Alt Köşede ("Sayfa No/Toplam Sayfa" şeklinde. Örn: 1/10, 2/10 şeklinde.)

### 4. Metin Özellikleri

- Paragraf Girintisi Yok
- Metin iki yana yaslı
- Tek satır aralıklı
- 12 Punto ve Normal
- Paragraflar arasında 1 boşluk
- Ana Başlıklar: 14 Punto ve Kalın, Alt Başlıklar:12 Punto ve Kalın
- Tüm Metin Fontu: Times New Roman

### 5. Şekiller ve Tablolar

- Şekiller sayfa içerisinde ortalı olmalıdır.
- Şekil açıklama metni şekil altında tek satıra sığmıyorsa ortalı, sığmıyorsa iki yana yaslı sol kenara yaslı olmalıdır.
- Tablolar sayfa içerisinde sola yaslı olmalıdır
- Tablo açıklama metni tablo üstünde ve sola yaslı olmalıdır.

### 6. Kaynaklar

- Araştırma ödevinin hazırlanmasında yararlanılan kaynaklar, metin içerisinde kullanım sırasına göre rapor sonunda ve "Kaynaklar" başlığı altında sıra numaraları verilerek listelenecektir.

**Gömülü sistemleri projesi raporu Word veya PDF dosyası formatında sisteme yüklenecektir. Dosya adı öğrenci numarası olacaktır; son gönderme tarihi 26 Mayıs saat 17:00.**

Başarılar Dilerim...

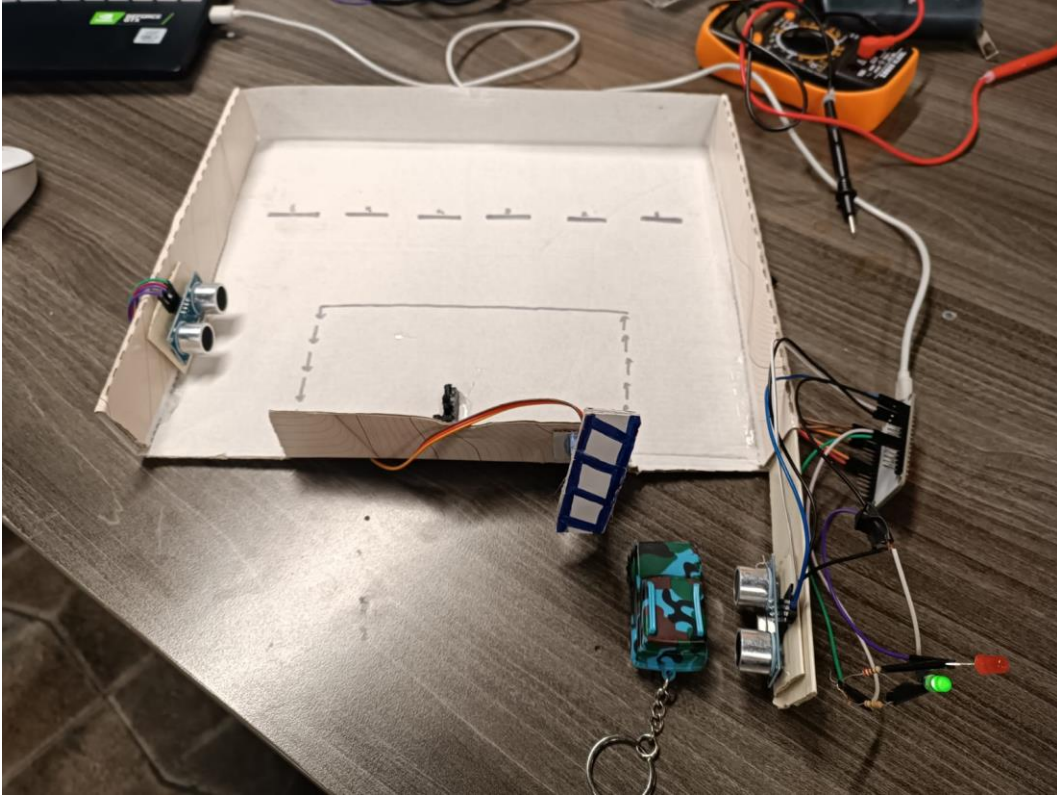
## **İçindekiler**

1. Projenin Amacı ve Hedefi.....	4
2. Projede kullanılan donanımlar ve yazılımlar ile ilgili bilgileri.....	5
2.1. Donanımlar .....	5
2.2. Yazılımlar .....	9
3. Projenin yapım aşamaları.....	17
Kaynaklar .....	24

## 1. Projenin Amacı ve Hedefi

Bu projenin amacı, akıllı bir otopark yönetim sistemi geliştirerek otoparkın verimli kullanımını sağlamak ve sürücülere kolaylık sunmaktır. Proje ortamında gerçekleştirilen işlemler şu şekildedir:

- Otoparkın doluluk durumunu anlık olarak izlemek ve LED göstergelerle sürücülere bilgi vermek.
- Otopark kapısını otomatik olarak kontrol etmek.
- Otopark doluluk durumuna göre yeni bir arabanın girişine izin vermek
- Araçların giriş ve çıkış zamanlarını kaydederek veri tabanında gözlemlemek
- Araçların çıkış zamanında otoparka ödenmesi gereken tutarı hesaplamak
- Masaüstü uygulaması ile otopark durumunu izlemek ve kontrol etmek.



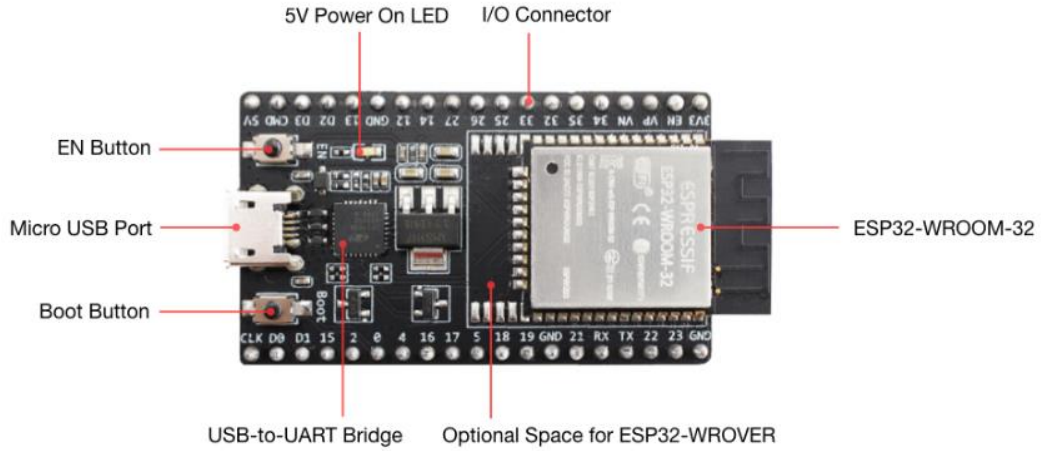
Şekil 1. Projenin görseli

Proje, ESP 32[1] kartı ile Ardunio IDE ortamında geliştirilmiş ve Visual Studio kullanılarak c# programlama dilinde bir masaüstü uygulama haline getirilmiştir.

## 2. Projede kullanılan donanımlar ve yazılımlar ile ilgili bilgileri

### 2.1. Donanımlar

#### 2.1.1. ESP32 WROOM



ESP32 WROOM, Espressif Systems tarafından geliştirilen güçlü bir mikrodenetleyici modüldür. Özellikle IoT (Internet of Things - Nesnelerin İnterneti) projeleri için tasarlanmıştır ve yüksek performans, düşük güç tüketimi ve çeşitli bağlantı özellikleri sunar.

- Dijital Çıkış Pini Sayısı: 34 adet genel amaçlı giriş/çıkış (GPIO) pini bulunur.
- PWM Çıkışı Sayısı: 16 adet PWM (Pulse Width Modulation) kanalı bulunur. Bu kanallar, motor kontrolü, LED parlaklık ayarı gibi görevler için kullanılabilir.
- Analog Giriş Sayısı: 18 adet 12-bit ADC (Analog to Digital Converter) kanalı bulunur. Bu kanallar analog sinyallerin dijital verilere dönüştürülmesi için kullanılır.
- Çalışma Gerilimi: 2.2V- 3.6V aralığında çalışır. genellikle 3.3V kullanılır.
- Mikrodenetleyici: ESP32 WROOM, Xtensa® dual-core 32-bit LX6 mikrodenetleyici çekirdeğine sahiptir.
- EEPROM: ESP32 WROOM'da EEPROM bulunmaz, ancak flash bellek kullanılabilir. Kullanıcı, flash bellek üzerinde veri saklamak için yazılım bazlı çözümler kullanabilir.
- SRAM: 520 KB SRAM bulunmaktadır.
- Saat Hızı: 240 MHz'e kadar saat hızı (işlemci hızı) destekler.
- I/O için Akım: Her GPIO pini için maksimum akım 40 mA'dir.

### 2.1.2. 2 Adet HC-SR04 Ultrasonik Mesafe Sensörü



HC-SR04, ultrasonik mesafe ölçümü yapmak için kullanılan bir sensördür. Çeşitli mesafeleri hassas bir şekilde ölçmek için tasarlanmıştır. Proje içerisinde 2 adet mesafe sensörü kullanılmıştır. 1. Mesafe sensörü arabanın otoparka girişinde bulunmakta. Burada tespit edilen bir araba olduğunda servo motorun çalışması sağlanıyor ve kapı açılıyor. 2. Mesafe sensörü otopark çıkışında bulunmaktadır. Burada tespit edilen bir araba olduğunda veri tabanından ilgili arabanın bilgileri silinmektedir.

- Çalışma Prensibi: HC-SR04, ultrasonik dalgalar göndererek ve bu dalgaların geri dönme süresini ölçerek mesafeyi hesaplar. Ses dalgaları bir nesneye çarptığında geri yansır. Sensör, bu yansıyan dalgaları alarak mesafeyi belirler.
- Ölçüm Mesafesi: 2 cm ile 400 cm (4 metre) arasında mesafe ölçümü yapabilir.
- Çözünürlük: 3 mm'ye kadar yüksek hassasiyet sunar.
- Çalışma Voltajı: 5V DC.
- Açısı: 15 dereceye kadar geniş bir algılama açısı sunar.
- Pin Yapısı:
  - ✓ VCC: 5V güç kaynağı.
  - ✓ Trig: Ölçüm sinyalini başlatmak için kullanılan tetikleme pini.
  - ✓ Echo: Geri dönen yankı sinyalini almak için kullanılan pin.
  - ✓ GND: Toprak bağlantısı.

### 2.1.3. Tower Pro SG90 Mini (9gr) Servo Motor



Tower Pro SG90 Servo Motor, küçük boyutu ve düşük ağırlığı ile birçok elektronik proje ve uygulamada yaygın olarak kullanılan bir servo motordur. Proje içerisinde 1. mesafe sensörünün algıladığı arabanın otoparka girişinde kullanılmaktadır. Veri tabanı üzerinden çekilen bilgilere göre otopark dolu değil ise servo motor çalışıyor ve otopark kapısını açıyor.

- Ağırlık: 9 gram
- Boyut: 22.2 x 11.8 x 31 mm
- Çalışma Voltajı: 4.8V - 6V
- Hareket Aralığı: 0° ile 180° arasında hareket edebilir.
- Hız: 60° dönüşü yaklaşık 0.1 saniyede tamamlar
- Bağlantı Pinleri:
  - ✓ Turuncu: PWM sinyali (kontrol sinyali) Servo motorun açılma pozisyonunu kontrol etmek için kullanılan Pulse Width Modulation (PWM) sinyali bu pinden alınır.
  - ✓ Kırmızı: VCC (besleme voltajı) Servo motorun çalışması için gerekli olan güç kaynağını sağlar
  - ✓ Kahverengi: GND (toprak) Güç kaynağının negatif terminaline bağlanarak devrenin tamamlanmasını sağlar.

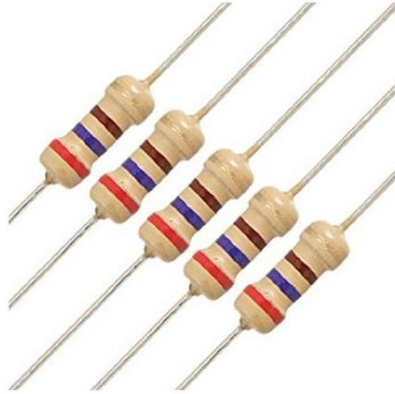
#### 2.1.4. 2 Adet 5mm led



Ledler elektrik enerjisini ışığa dönüştüren yarı iletken bir devre elemanıdır. Belirli miktarda voltaj verildiğinde etrafa ışık saçmaktadırlar. Projede yeşil ve kırmızı renkte ledler kullanılmıştır. Bu ledler projede otoparkın dolu olup olmadığı bilgisini vermektedir. Otopark dolu ise kırmızı değil ise yeşil led yanmaktadır.

- Boyut : 5 mm
- Uzunluk: Çoğunlukla 8-9 mm
- Akım Tüketimi: 20mA
- Çalışma voltajı : 1.5 - 3 V
- Açısı: Genellikle 20° ile 30° arasında
- Bağlantı Pinleri:
  - ✓ Anot (Uzun Bacak): Pozitif terminal
  - ✓ Katot (Kısa Bacak): Negatif terminal

#### 2.1.5. 2 Adet 220 Ohm direnç



220 ohm direnç, elektronik devrelerde akım sınırlama, voltaj bölme ve diğer çeşitli uygulamalar için yaygın olarak kullanılan bir bileşendir. Proje içerisinde ledlerin bağlantısında kullanılmıştır.

- Direnç Değeri: 220 ohm
- Güç Derecesi: Genellikle 1/4 watt (0.25W) veya 1/2 watt (0.5W) gücünde gelir.
- Tolerans:  $\pm 5\%$



### 2.1.6. Jumper kablo



Jumper kablo, elektronik devrelerde bağlantı yapmak için kullanılan esnek ve yalıtılmış tel parçalarıdır. Bu kablolar devre elemanları arasında geçici bağlantılar kurmak için kullanılır. Hızlı ve kolay bağlantı imkânı sağlar ve farklı renklerde gelir, bu da bağlantıların takip edilmesini kolaylaştırır.

## 2.2. Yazılımlar

### 2.2.1 Gömülü yazılım kısmında:

- Arduino IDE: C ve C ++ dilleri ile yazılmış bir platformlar arası uygulamadır ve Arduino'ya özgü fonksiyonlar ve kütüphaneler içerir. Arduino uyumlu kartlara program yazmak ve yüklemek için kullanılır.[1]

### 2.2.2 Arayüz yazılım kısmında:

- Visual Studio 2022: Visual Studio, Microsoft tarafından geliştirilen geliştirme döngüsünün tamamını tek bir yerde tamamlamak için kullanabileceğiniz güçlü bir IDE aracıdır. Kod yazmak, düzenlemek, hata ayıklamak ve derlemek ve ardından uygulamanızı dağıtmak için kullanabileceğiniz kapsamlı bir tümleşik geliştirme ortamıdır. Farklı platformlarda (Windows, macOS, Linux) çeşitli programlama dilleriyle (C#, C++, Python, JavaScript vb.) uygulama geliştirme imkanı sağlar [2].
- C#: Microsoft tarafından geliştirilmiş, modern ve nesne yönelimli bir programlama dilidir. 2000 yılında .NET Framework ile tanıtılmıştır ve günümüzde geniş bir kullanım alanına sahiptir. C#, güçlü tip güvenliği, otomatik bellek yönetimi, olay tabanlı programlama ve LINQ gibi özellikler sunar. Web uygulamaları, masaüstü uygulamaları, mobil uygulamalar, oyun geliştirme ve bulut tabanlı hizmetler gibi birçok alanda kullanılır. [3]

### 2.2.3 Veri Tabanı

- MSSQL: Microsoft tarafından geliştirilen ve yönetilen bir ilişkisel veri tabanı yönetim sistemidir. SQL Server, büyük ve karmaşık veri tabanlarını depolamak, yönetmek, sorgulamak ve işlemek için kullanılan bir yazılım ürünüdür. Veri depolama, veri güvenliği, yedekleme, geri yükleme, veri entegrasyonu, analiz ve raporlama gibi çeşitli veri tabanı yönetimi işlevlerini destekler [4].

Proje içerisinde 2 adet veri tabanı tablosu kullanılmıştır.

### 1. Araba Tablosu:

Kullanılan Veri tabanı:

- Veri Tabanı Adı: sensem
- Tablo Adı: Araba
- Kolonlar:
  - ArabaID: Benzersiz kimlik numarası (Primary Key).
  - GirisSaati: Arabanın otoparka giriş yaptığı saat.
  - GirisTarihi: Arabanın otoparka giriş yaptığı tarih.

→ Bu tablo, otoparka giren araçların bilgilerini saklamak için kullanılır. Her araç giriş yaptığında, bu tabloya bir yeni kayıt eklenir.

### 2. ArabaCikis2 Tablosu:

Kullanılan Veri tabanı:

- Veri Tabanı Adı: sensem
- Tablo Adı: ArabaCikis2
- Kolonlar:
  - ArabaID: Benzersiz kimlik numarası (Primary Key).
  - GirisSaati: Arabanın otoparka giriş yaptığı saat.
  - GirisTarihi: Arabanın otoparka giriş yaptığı tarih.
  - CikisTarihi: Arabanın otoparktan çıkış yaptığı tarih.
  - CikisSaati: Arabanın otoparktan çıkış yaptığı saati.
  - OdemeTutari: Arabanın park süresine bağlı olarak ödemesi gereken tutar.

→ Bu tablo, otoparktan çıkan araçların bilgilerini saklamak için kullanılır. Her araç çıkış yaptığında, bu tabloya bir yeni kayıt eklenir.

```
otopark.sql - DESK...SS.sensem (sa (51))* X
create database sensem;
USE sensem;

CREATE TABLE Araba (
  ArabaID INT IDENTITY(1,1) PRIMARY KEY,
  GirisSaati Varchar(20) NOT NULL,
  GirisTarihi Varchar(20) NOT NULL
);

CREATE TABLE ArabaCikis2 (
  ArabaID INT IDENTITY(1,1) PRIMARY KEY,
  GirisTarihi VARCHAR(20) NOT NULL,
  GirisSaati VARCHAR(20) NOT NULL,
  CikisTarihi VARCHAR(20) NOT NULL,
  CikisSaati VARCHAR(20) NOT NULL,
  OdemeTutari DECIMAL(10, 2) NOT NULL
);

INSERT INTO Araba (GirisSaati, GirisTarihi) VALUES ('10:30:00', '22.05.2024');

INSERT INTO ArabaCikis2 (GirisTarihi, GirisSaati, CikisTarihi, CikisSaati, OdemeTutari)
VALUES ('22.05.2024', '10:30:00', '22.05.2024', '11:30:00', 100.00);

SELECT * FROM ArabaCikis2;
```

75 %

Results Messages

	ArabaID	GirisTarihi	GirisSa...	CikisTarihi	CikisSaati	OdemeTut...
1	1	22.05.2024	10:30:00	22.05.2024	11:30:00	100.00

	ArabaID	GirisSa...	GirisTarihi
1	1	10:30:00	22.05.2024
2	2	11:30:00	22.05.2024
3	3	11:45:00	22.05.2024
4	4	14:40:00	22.05.2024

## 2.2.4 Yazılım Komutları

### Timer (Zamanlayıcı) Kullanımı

```
unsigned long previousMillisServo = 0; // Servo kontrolü için önceki milisaniye
unsigned long servoDelay = 3000; // Servo için bekleme süresi
unsigned long previousMillisSensor = 0; // Sensör kontrolü için önceki milisaniye
unsigned long sensorInterval = 3000; // Sensör kontrolü için zaman aralığı

unsigned long previousMillisDistance2 = 0; // İkinci sensör kontrolü için önceki milisaniye
unsigned long distance2Interval = 1000; // İkinci sensör kontrolü için zaman aralığı

bool servoAt180 = false; // Servo motorun 180 derecede olup olmadığını belirten bayrak
unsigned long servoAt180Time = 0; // Servo motorun 180 dereceye ulaştığı zaman
unsigned long servoDuration = 3000; // Servo motorun 180 derecede kalma süresi
```

```
unsigned long currentMillis = millis(); // Mevcut milisaniyeyi al

// Sensörleri belirli aralıklarla kontrol et
if (currentMillis - previousMillisSensor >= sensorInterval) {
    previousMillisSensor = currentMillis; // Önceki milisaniyeyi güncelle
```

```
// Mesafeye göre servo kontrolü
if (distance < 7) { // Eğer mesafe 7 cm'den küçükse
    if (notEmptyPlace < 6) { // Ve dolu park yeri sayısı 6'dan azsa
        myServoControl.servoTurn(180); // Servo motoru 180 dereceye çevir
        previousMillisServo = currentMillis; // Önceki milisaniyeyi güncelle
        servoAt180 = true; // Servo'nun 180 derecede olduğunu belirten bayrağı set et
        servoAt180Time = currentMillis; // Servo'nun 180 dereceye ulaştığı zamanı kaydet
    }
} else { // Eğer mesafe 7 cm'den büyükse
    myServoControl.servoTurn(90); // Servo motoru 90 dereceye çevir
    servoAt180 = false; // Servo'nun 180 derecede olmadığını belirten bayrağı resetle
}

// Gecikme süresinden sonra servo'yu varsayılan konuma döndür
if (currentMillis - previousMillisServo >= servoDelay) {
    myServoControl.servoTurn(90); // Servo motoru 90 dereceye çevir
}

// Servo'nun 180 derecede belirli bir süre kalıp kalmadığını kontrol et
if (servoAt180 && currentMillis - servoAt180Time >= servoDuration) {
    myServoControl.servoTurn(90); // Servo motoru varsayılan konuma döndür
    servoAt180 = false; // Servo'nun 180 derecede olmadığını belirten bayrağı resetle
}
```

## Kütüphane (Library) Kullanımı

```
#include "servoMotorControls.h" // Servo motor kontrolü için oluşturulmuş özel kütüphane

1 // servoMotorControls.h
2
3 //servo motor kontrolü için sınıfın tanımını içerir. Sınıfın fonksiyonları ve değişkenleri burada tanımlanmıştır
4 #ifndef SERVO_MOTOR_CONTROLS_H
5 #define SERVO_MOTOR_CONTROLS_H
6
7 #include <ESP32Servo.h> // ESP32 için servo motor kütüphanesi
8
9 // servoMotorControls sınıfının tanımı
10 class servoMotorControls {
11 public:
12     // Kurucu fonksiyon (constructor), servo pinini parametre olarak alır
13     servoMotorControls(byte servoPin);
14
15     // Servo motoru belirli bir dereceye döndürmek için kullanılan fonksiyon
16     void servoTurn(int degree);
17
18 private:
19     Servo myServo; // Servo motor nesnesi
20     byte _servoPin; // Servo motorun bağlı olduğu pin
21 };
22
23 #endif // SERVO_MOTOR_CONTROLS_H
24
```

```
1 // servoMotorControls.cpp
2
3 #include "servoMotorControls.h" // Servo motor kontrol sınıfının başlık dosyası
4
5 // Sınıfın kurucusu (constructor)
6 servoMotorControls::servoMotorControls(byte servoPin) {
7     _servoPin = servoPin; // Servo pinini belirler
8     myServo.attach(servoPin); // Servo motoru belirlenen pine bağlar
9 }
10
11 // Servo motoru belirli bir dereceye döndürmek için kullanılan fonksiyon
12 void servoMotorControls::servoTurn(int degree) {
13     myServo.write(degree); // Servo motoru belirtilen dereceye döndürür
14 }
15
```

```
1 başvuru
private int GetEmptyParkCount()
{
    int totalPark = 6; // Toplam park yeri sayısı
    int arabaCounts = GetArabaCount(); // Mevcut araba sayısını al
    return totalPark - arabaCounts; // Boş park yeri sayısını hesapla
}
```

Bu fonksiyon, toplam park yeri sayısından mevcut araba sayısını çıkararak boş park yeri sayısını hesaplar ve bu değeri geri döndürür. Park yeri sayısı sabit olarak 6 olarak belirlenmiştir.

```

private int GetArabaCount()
{
    int count = 0;
    string connectionString = "Data Source=DESKTOP-2JMETKU;Initial Catalog=sensem;Integrated Security=True;Encrypt=False";

    using (SqlConnection con = new SqlConnection(connectionString))
    {
        try
        {
            con.Open(); // Veritabanı bağlantısını açtı
            string query = "SELECT COUNT(*) FROM Araba"; // Araba sayısını sorgulayan SQL komutu
            using (SqlCommand cmd = new SqlCommand(query, con))
            {
                count = (int)cmd.ExecuteScalar(); // Sorgu sonucu
            }
        }
        catch (Exception ex)
        {
            // Hata mesajını görmek için
            MessageBox.Show("Araba count retrieval error: " + ex.Message);
        }
    }

    //araba sayısını döndür
    return count;
}

```

Bu fonksiyon, SQL bağlantısı ile veri tabanında kayıtlı olan araba sayısını sorgular ve bu değeri döndürür

```

private void Form1_Load(object sender, EventArgs e)
{
    string[] portNames = SerialPort.GetPortNames();
    comboBoxPortName.Items.AddRange(portNames);

    string connectionString = "Data Source=DESKTOP-2JMETKU;Initial Catalog=sensem;Integrated Security=True;Encrypt=False";
    using (SqlConnection con = new SqlConnection(connectionString))
    {
        try
        {
            con.Open(); //veri tabanı bağlantısını aç
            //arabaları giriş saatlerine göre sırala
            string query = "SELECT * FROM Araba ORDER BY GirişSaati DESC";
            using (SqlCommand cmd = new SqlCommand(query, con))
            {
                SqlDataAdapter adapter = new SqlDataAdapter(cmd); // Veri adaptörü oluştur
                DataTable table = new DataTable(); // Veri tablosu oluştur
                adapter.Fill(table); // Verileri tabloya doldur
                dataGridView1.DataSource = table; // Verileri dataGridView'e bağla
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Veri çekerken hata oluştu: " + ex.Message);
        }
    }

    int arabaCount = GetArabaCount(); //araba sayısını al
    int emptyParks = GetEmptyParkCount(); // boş park sayısını al
    labelArabaCount.Text = arabaCount.ToString(); // araba sayısını güncelle
    labelEmptyParks.Text = emptyParks.ToString(); // boş park sayısını güncelle
}

```

Bu fonksiyon, form yüklendiğinde çalışır. Seri port isimlerini alır ve combobox'a ekler. Ayrıca veri tabanından arabaların bilgilerini alarak DataGridView'e doldurur. Mevcut araba sayısı ve boş park yeri sayısını günceller



```

private void button1_Click(object sender, EventArgs e)
{
    try
    {
        if (sr.IsOpen)
        {
            sr.Close(); // Seri port açıksa kapat
        }

        if (comboBoxPortName.SelectedItem != null)
        {
            sr.PortName = comboBoxPortName.SelectedItem.ToString(); // Seçilen port ismini al
            sr.BaudRate = 9600; // Baud rate ayarla
            sr.Open(); // Seri portu aç
            button1.Enabled = false; // button 1 in durumunu false yap
            button2.Enabled = true; //button 2 nin durumunu true (etkin) yap
            label3.Text = "AÇIK"; // Durum etiketini güncelle
            sr.WriteLine("sayi:" + labelArabaCount.Text); // Araba sayısını seri porttan gönder
        }
        else
        {
            // Port seçilmediyse uyarı gönder
            MessageBox.Show("Lütfen bir port seçin.");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Seri port açılırken hata oluştu: " + ex.Message);
    }
}

```

Bu fonksiyon, "Bağlan" butonuna tıklandığında çalışır. Seçilen seri portu açar ve arabaların sayısını seri port üzerinden gönderir. Butonların durumlarını günceller

```

private List<int> GetArabaIDs()
{
    //araba ID leri için liste oluşturdum
    List<int> arabaIDs = new List<int>();
    string connectionString = "Data Source=DESKTOP-2JMETKU;Initial Catalog=sensem;Integrated Security=True;Encrypt=False";

    using (SqlConnection con = new SqlConnection(connectionString))
    {
        try
        {
            //veri tabanı bağlantısını aç
            con.Open();
            //ArabaID lerini sorgulayan SQL komutu
            string query = "SELECT ArabaID FROM Araba";
            using (SqlCommand cmd = new SqlCommand(query, con))
            {
                SqlDataReader reader = cmd.ExecuteReader(); //sorgu sonucunu oku
                while (reader.Read())
                {
                    arabaIDs.Add(reader.GetInt32(0)); //sorgu sonuçlarını listeye ekle
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Araba ID Hatası: " + ex.Message);
        }
    }

    return arabaIDs;
}

```

Bu fonksiyon, SQL bağlantısı ile veri tabanındaki araba ID'lerini sorgular ve bir liste olarak döndürür.

```

private void SerialDataReceivedHandler(object sender, SerialDataReceivedEventArgs e)
{
    try
    {
        SerialPort sp = (SerialPort)sender;

        string inData = sp.ReadExisting().Trim(); // Gelen veriyi al
        Console.WriteLine("Received data: " + inData);

        if (inData == "1") //sensör 1 den nesne geldiyse - araba giriş yapmak istiyorsa
        {
            if (GetArabaCount() < 6) // otopark dolu değil ise - otoparktaki araba sayısı 6 dan küçük ise
            {
                HandleNewCarEntry(); // yeni bir araba girişi sağla
            }
            else
            {
                MessageBox.Show("Otoparkta Boş Alan Yok");
            }
        }
        else if (inData == "2") //sensör 2 de nesne algılandıysa - araba çıkış yapmak istiyorsa
        {
            HandleCarExit(); // araba çıkışını sağla
        }
        else
        {
            //beklenmeyen veri geldiğinde
            Console.WriteLine("Unexpected data received: " + inData);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Data received handler error: " + ex.Message);
    }
}

```

Bu fonksiyon, seri porttan veri alındığında çalışır. Gelen veriyi okur ve temizler. Veriye göre yeni bir araba girişi veya çıkışı sağlar.

```

private void HandleNewCarEntry()
{
    DateTime currentTime = DateTime.Now; //güncel tarih
    string girisSaati = currentTime.ToString("HH:mm:ss"); //saat formatına dönüştür
    string girisTarihi = currentTime.ToString("dd.MM.yyyy"); //tarih formatına dönüştür

    string connectionString = "Data Source=DESKTOP-2JMETKU;Initial Catalog=sensem;Integrated Security=True;Encrypt=False";
    using (SqlConnection con = new SqlConnection(connectionString))
    {
        try
        {
            con.Open(); // Veritabanı bağlantısını aç
            string query = "INSERT INTO Araba (GirisSaati, GirisTarihi) VALUES (@GirisSaati, @GirisTarihi)"; // Yeni araç girişi sorgusu
            using (SqlCommand cmd = new SqlCommand(query, con))
            {
                cmd.Parameters.AddWithValue("@GirisSaati", girisSaati); // Giriş saatini ekle
                cmd.Parameters.AddWithValue("@GirisTarihi", girisTarihi); // Giriş tarihini ekle
                cmd.ExecuteNonQuery(); // Sorguyu çalıştır
            }

            this.Invoke(new Action() =>
            {
                Form1_Load(null, null); //Formu yeniden yükle
                int arabaCount = GetArabaCount();
                labelArabaCount.Text = arabaCount.ToString(); // araba sayısını güncelle
                sr.WriteLine("sayi:" + labelArabaCount.Text); // Yeni araba sayısını seri porttan gönder
            });

            // Yeni araç girişi mesajı göster
            MessageBox.Show("Yeni Araç Giriş Yapıldı\nGiriş Tarihi: " + currentTime);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Araba giriş hatası: " + ex.Message);
        }
    }
}

```

Bu fonksiyon, yeni bir araba girişi olduğunda çalışır. Güncel tarih ve saat bilgilerini alır ve SQL sorgusu ile veri tabanına kaydeder. Sonrasında formu yeniden yükler ve kullanıcıya güncel park yeri durum bilgisini verir.

```
private void HandleCarExit()
{
    List<int> arabaIDs = GetArabaIDs(); // Araç ID'lerini al
    if (arabaIDs.Count > 0)
    {
        Random rnd = new Random();
        int indexToDelete = arabaIDs[rnd.Next(arabaIDs.Count)]; // Rastgele bir araç ID seç

        string connectionString = "Data Source=DESATOP-2\METU;Initial Catalog=sense;Integrated Security=True;Encrypt=False"; // Veritabanı bağlantısı
        using (SqlConnection con = new SqlConnection(connectionString))
        {
            try
            {
                con.Open(); // Veritabanı bağlantısını aç

                // Araç giriş zamanını al
                DateTime girisZamani;
                string selectQuery = "SELECT GirisSaati, GirisTarihi FROM Araba WHERE ArabaID = @ID"; // Giriş bilgilerini almak için SQL sorgusu
                using (SqlCommand selectCmd = new SqlCommand(selectQuery, con))
                {
                    selectCmd.Parameters.AddWithValue("@ID", indexToDelete); // Araç ID'si parametresini ekle
                    using (SqlDataReader reader = selectCmd.ExecuteReader())
                    {
                        if (reader.Read())
                        {
                            girisZamani = DateTime.ParseExact(reader["GirisTarihi"].ToString() + " " + reader["GirisSaati"].ToString(), "dd.MM.yyyy HH:mm:ss",
                                CultureInfo.InvariantCulture);
                        }
                        else
                        {
                            MessageBox.Show("Araba bilgisi bulunamadı."); // Araç bilgisi bulunamadıysa mesaj göster
                            return;
                        }
                    }
                }

                // Şu anki zaman
                DateTime cikisZamani = DateTime.Now;

                // Kalan süre
                TimeSpan kalanSure = cikisZamani - girisZamani;
                int totalMinutes = (int)kalanSure.TotalMinutes; // Toplam dakikayı al
                double odeme = totalMinutes * 5; // Dakika başına 5 TL ücret

                // Ücreti göster ve ödeme onayı iste
                DialogResult result = MessageBox.Show($"Araba ID: {indexToDelete} kaldığı Süre: {kalanSure.Days} gün {kalanSure.Hours} saat {kalanSure.Minutes} dakika. Ödeme tutarı: {odeme} TL. Ödemeyi onaylıyor musunuz?");
                if (result == DialogResult.Yes)
                {
                    // ArabaCikis2 tablosu için IDENTITY_INSERT'i etkinleştir
                    string enableIdentityInsertQuery = "SET IDENTITY_INSERT ArabaCikis2 ON";
                    using (SqlCommand enableIdentityInsertCmd = new SqlCommand(enableIdentityInsertQuery, con))
                    {
                        enableIdentityInsertCmd.ExecuteNonQuery();
                    }

                    // Kayıtları ArabaCikis2 tablosuna ekle
                    string insertQuery = "INSERT INTO ArabaCikis2 (ArabaID, GirisTarihi, GirisSaati, CikisTarihi, CikisSaati, OdemeTutari) VALUES (@ID, @GirisTarihi, @GirisSaati, @CikisTarihi, @CikisSaati, @OdemeTutari)";
                    using (SqlCommand insertCmd = new SqlCommand(insertQuery, con))
                    {
                        insertCmd.Parameters.AddWithValue("@ID", indexToDelete); // Araç ID'si parametresini ekle
                        insertCmd.Parameters.AddWithValue("@GirisTarihi", girisZamani.ToString("dd.MM.yyyy")); // Giriş tarihi parametresini ekle
                        insertCmd.Parameters.AddWithValue("@GirisSaati", girisZamani.ToString("HH:mm:ss")); // Giriş saati parametresini ekle
                        insertCmd.Parameters.AddWithValue("@CikisTarihi", cikisZamani.ToString("dd.MM.yyyy")); // Çıkış tarihi parametresini ekle
                        insertCmd.Parameters.AddWithValue("@CikisSaati", cikisZamani.ToString("HH:mm:ss")); // Çıkış saati parametresini ekle
                        insertCmd.Parameters.AddWithValue("@OdemeTutari", odeme); // Ödeme tutarı parametresini ekle
                        insertCmd.ExecuteNonQuery(); // Sorguyu çalıştır
                    }

                    // ArabaCikis2 tablosu için IDENTITY_INSERT'i devre dışı bırak
                    string disableIdentityInsertQuery = "SET IDENTITY_INSERT ArabaCikis2 OFF";
                    using (SqlCommand disableIdentityInsertCmd = new SqlCommand(disableIdentityInsertQuery, con))
                    {
                        disableIdentityInsertCmd.ExecuteNonQuery();
                    }

                    // Araba tablosundan kaydı sil
                    string deleteQuery = "DELETE FROM Araba WHERE ArabaID = @ID";
                    using (SqlCommand deleteCmd = new SqlCommand(deleteQuery, con))
                    {
                        deleteCmd.Parameters.AddWithValue("@ID", indexToDelete); // Araç ID'si parametresini ekle
                        deleteCmd.ExecuteNonQuery(); // Sorguyu çalıştır
                    }

                    this.Invoke(new Action() =>
                    {
                        Form1_Load(null, null); // Formu yeniden yükle
                        int arabaCount = GetArabaCount(); // Araç sayısını al
                        LabelArabaCount.Text = arabaCount.ToString(); // Araç sayısını etikete yaz
                        LabelArabaCount.Text = arabaCount.ToString(); // Seri porttan araç sayısını gönder
                    });

                    MessageBox.Show("Araba çıkışı yapıldı ve ödeme alındı."); // Çıkış ve ödeme mesajı göster
                }
                else
                {
                    MessageBox.Show("Araba çıkışı yapılamaz. Lütfen önce ödemeyi yapın."); // Ödeme yapılmadıysa mesaj göster
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Araba çıkış hatası: " + ex.Message); // Hata durumunda mesaj göster
            }
        }
    }
}
```

Bu fonksiyon, park yerinden çıkış yapan bir arabanın bilgilerini günceller ve kullanıcıyı bilgilendirir. Öncelikle, mevcut araba ID'leri alınır ve rastgele bir araba ID'si seçilir. Veri tabanı bağlantısı açılarak, seçilen arabanın giriş tarihi ve saati alınır. Çıkış zamanı ile parkta kalan süre hesaplanır ve her dakika için 5 TL üzerinden ödeme tutarı belirlenir. Kullanıcıdan ödeme onayı alınırsa, çıkış bilgileri ArabaCikis2 tablosuna kaydedilir ve Araba tablosundan ilgili kayıt silinir. Form yeniden yüklenir ve güncel araba sayısı güncellenir. Hata durumlarında kullanıcıya mesaj gösterilir. Bu işlem, park yeri yönetimini otomatikleştirir ve kullanıcıya güncel bilgi sağlar.



```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (sr.IsOpen)
    {
        sr.Close(); //seri port açık ise kapat
    }
}
```

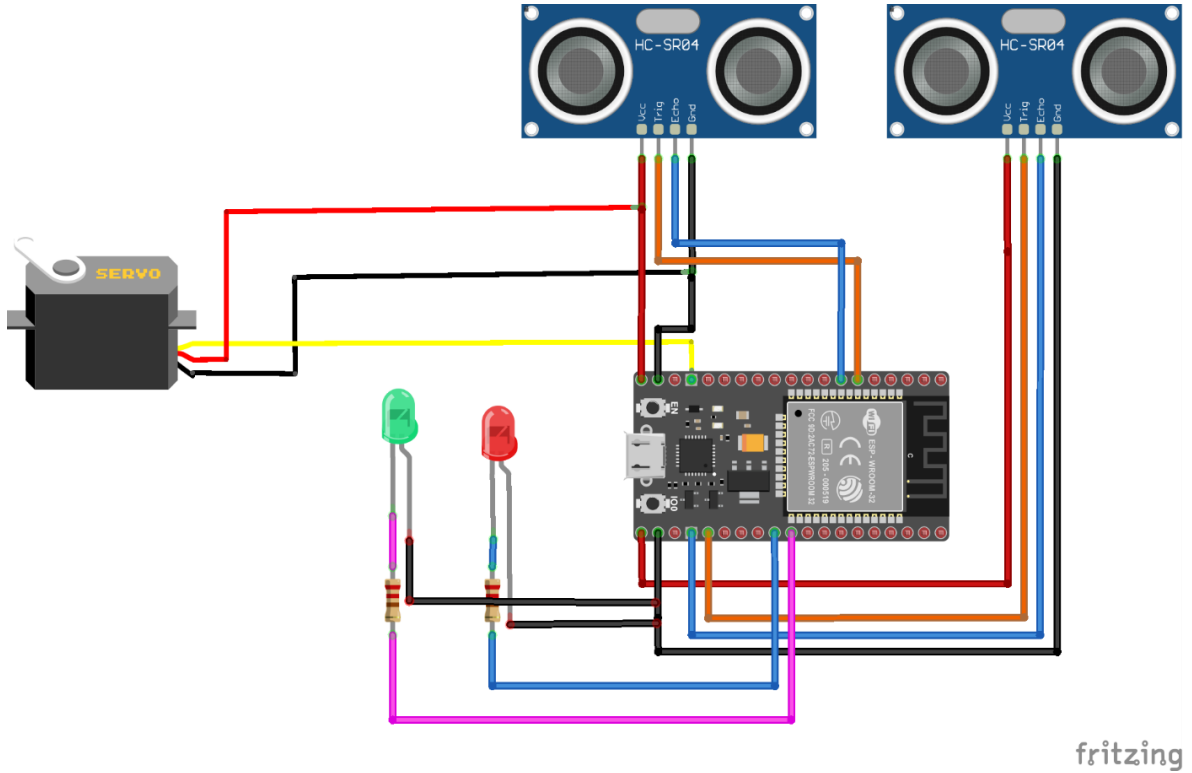
Bu fonksiyon, form kapanırken çalışır. Seri port açıksa kapatır.

```
private void button2_Click(object sender, EventArgs e)
{
    if (sr.IsOpen)
    {
        sr.Close();
        button1.Enabled = true;
        button2.Enabled = false;
        label3.Text = "KAPALI"; //"PORT DURUMU: " KAPALI olarak güncellendi
    }
}
```

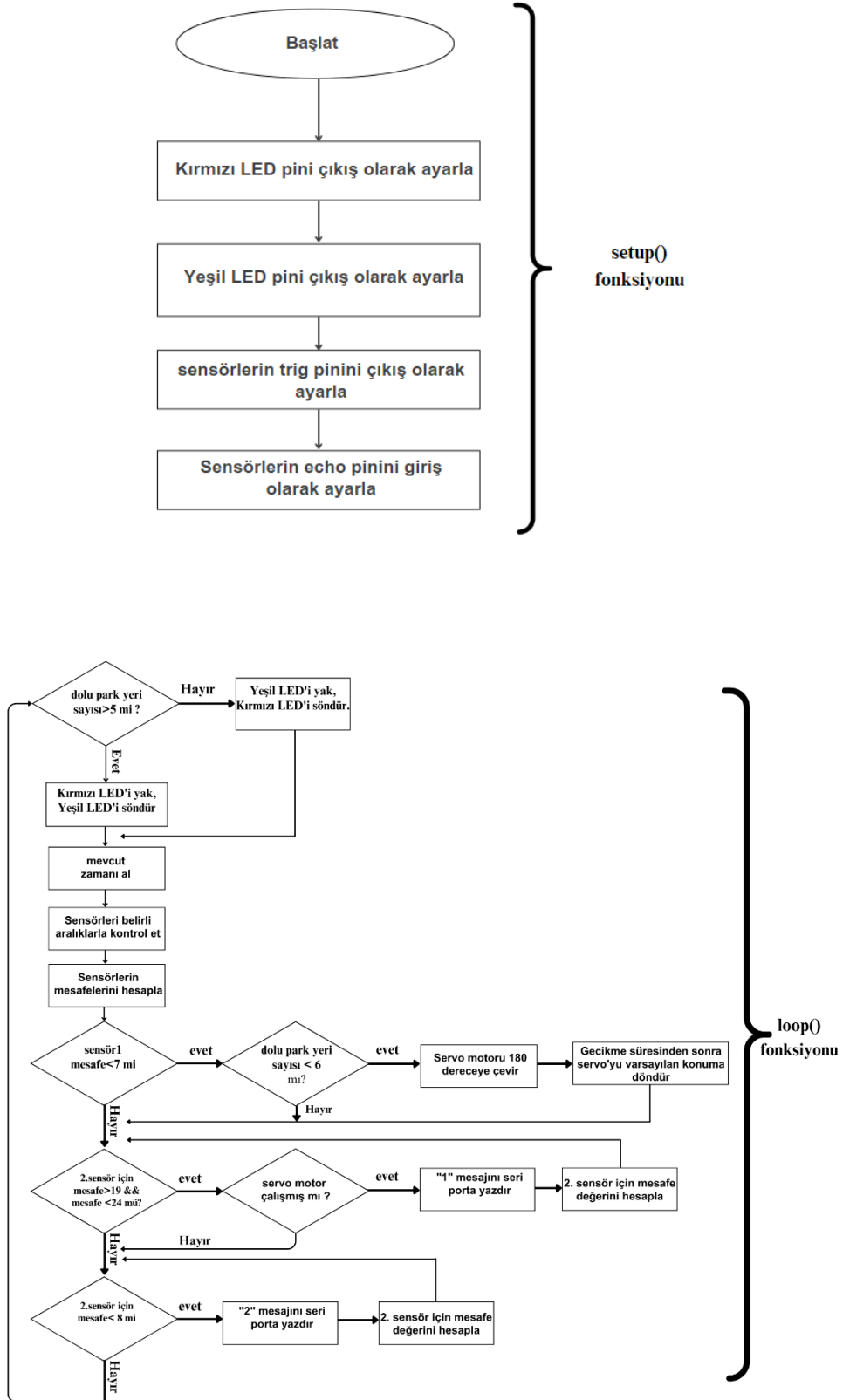
Bu fonksiyon, "Bağlantıyı Kes" butonuna tıklandığında çalışır. Seri portu kapatır ve butonların durumlarını günceller.

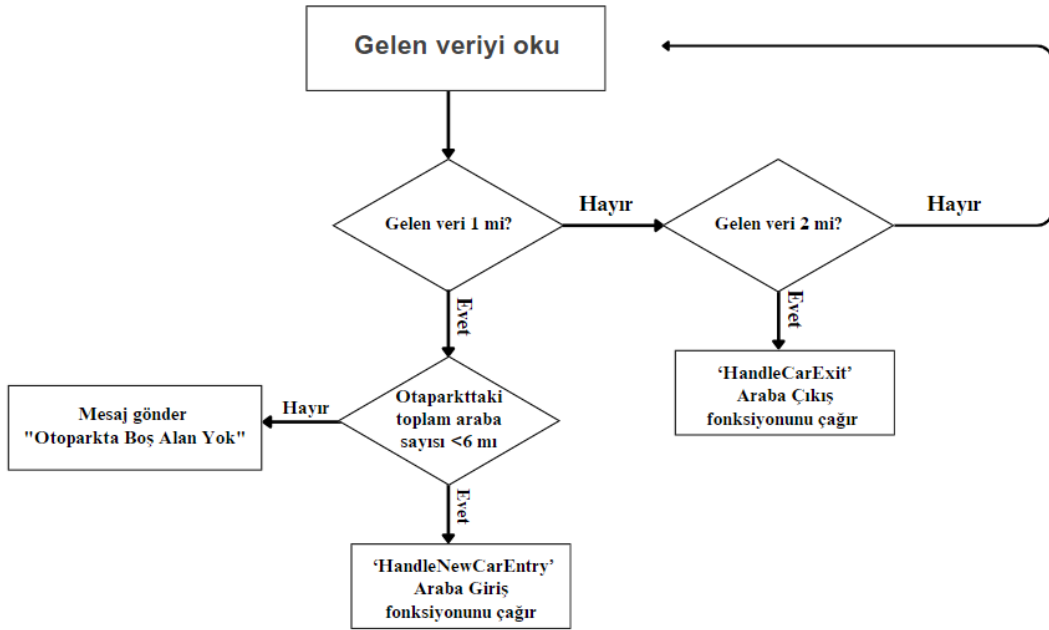
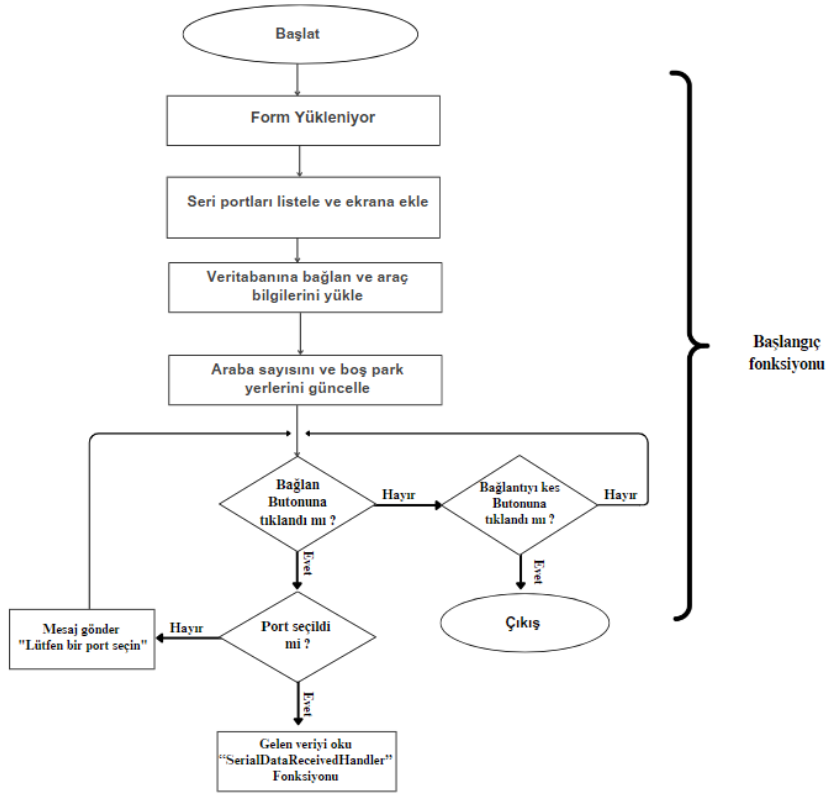
### 3. Projenin yapım aşamaları

#### 3.1. Fritzing çizim şeması

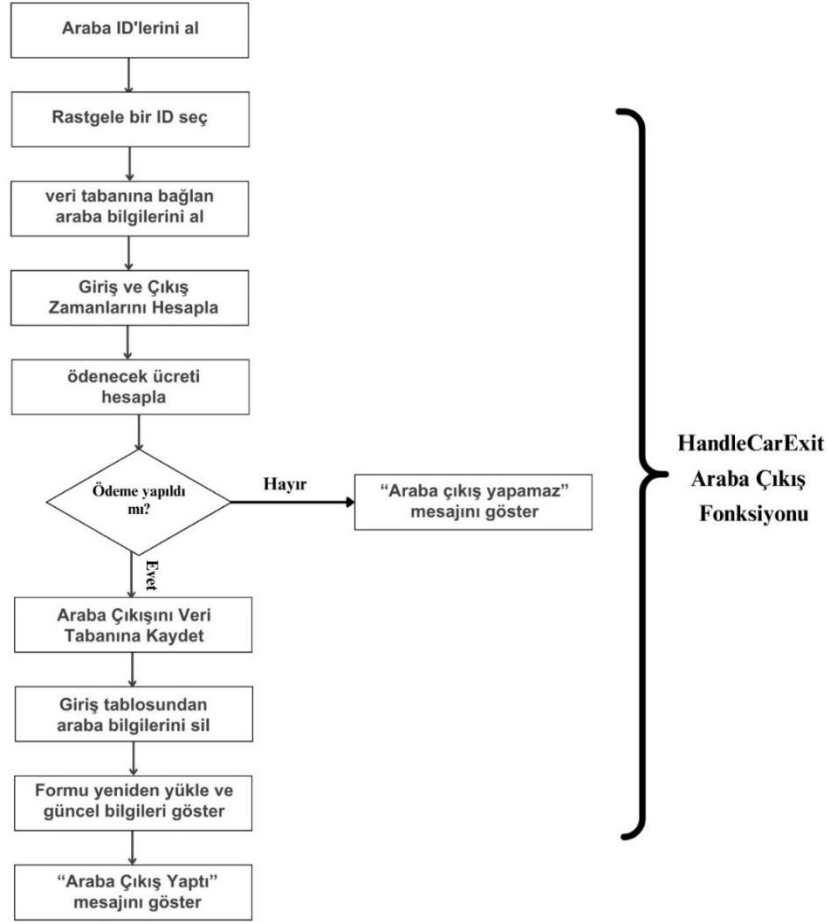


### 3.2.Akış Şemaları





**SerialDataReceivedHandler  
Fonksiyonu**



### 3.3. Çalışma adımları

SenSem Otoparki

## SenSem OTOPARK OTOMASYONU

ArabaID	GirisSaati	GirisTarihi
237	11:29:57	05.06.2024
236	11:29:32	05.06.2024
235	11:29:00	05.06.2024
234	11:28:55	05.06.2024
233	11:28:47	05.06.2024

Port Seçin:

**Bağlan** **Bağlantıyı Kes**

Araba Sayısı: 6

Port Durum: **KAPALI**

Lütfen bir port seçin.

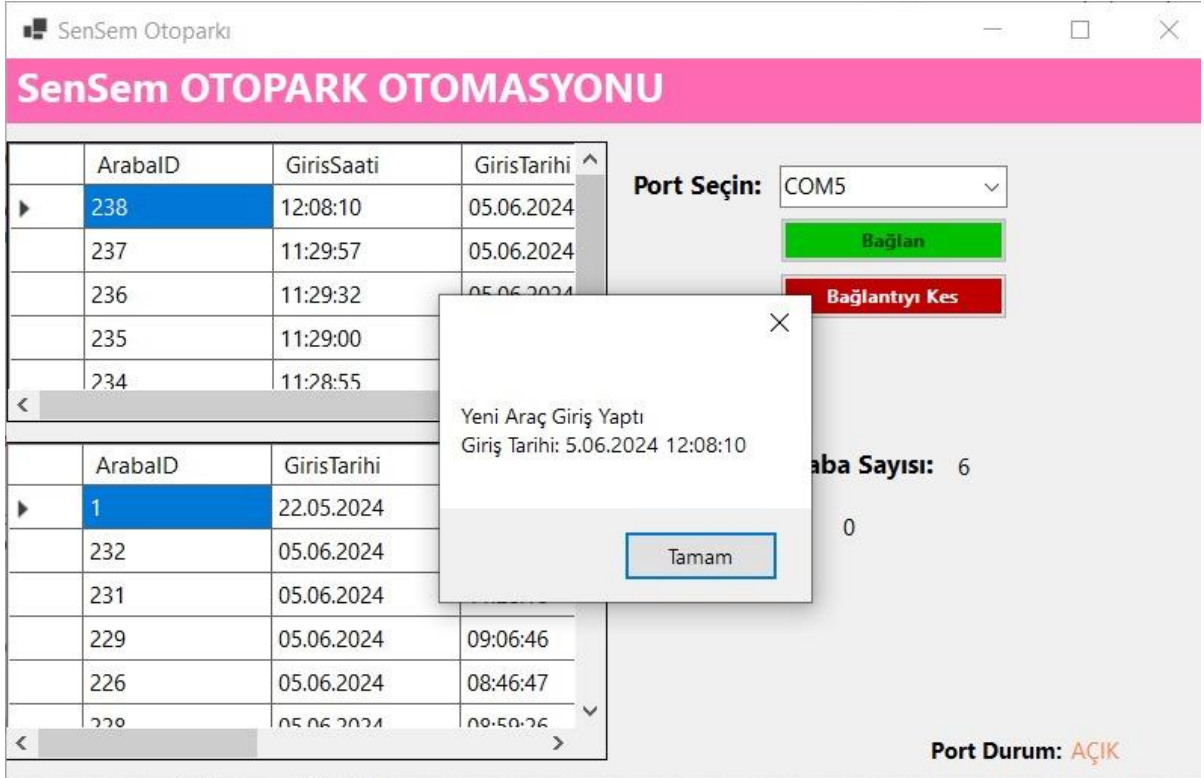
**Tamam**

### 1. Seri Port Bağlantısının Açılması

Uygulamayı çalıştırdığımızda ilk olarak ana form yüklenir ve otopark içerisinde mevcut araç bilgilerinin olduğu ve otoparktan çıkış yapan araç bilgilerinin bulunduğu tablolar getirilerek kullanıcıya gösterilir. Uygulama içerisinde işlemlerin yapılabilmesi için ilk olarak kullanıcının mevcut seri portlardan birini seçmesi ve bağlantıyı açması gerekmektedir. Bu işlem, “comboBoxPortName” bileşeninden port seçimi yapılarak ve bağlan butonuna basılarak gerçekleştirilir. Kullanıcı, seri port bağlantısını açmak için bağlan butonuna bastığında, seçili port adı ve baud hızı ayarlanarak seri port bağlantısı açılır. Bağlantı başarılı olduğunda, bağlan butonu devre dışı bırakılır ve kapat butonu etkinleştirilir. Ayrıca, bağlantının durumu “AÇIK” olarak etiketlenir.

### 2. Mevcut Araç ve Boş Park Yeri Sayısının Gösterilmesi

Seri port bağlantısı açıldığında veya herhangi bir işlem gerçekleştirildiğinde, veri tabanına bağlanılarak mevcut araç sayısı ve boş park yeri sayısı hesaplanır. Bu bilgiler, “labelArabaCount” ve “labelEmptyParks” etiketlerinde kullanıcıya gösterilir.



ArabaID	GirisSaati	GirisTarihi
238	12:08:10	05.06.2024
237	11:29:57	05.06.2024
236	11:29:32	05.06.2024
235	11:29:00	05.06.2024
234	11:28:55	05.06.2024

ArabaID	GirisTarihi
1	22.05.2024
232	05.06.2024
231	05.06.2024
229	05.06.2024
226	05.06.2024
228	05.06.2024

Port Seçin: COM5

Bağlan

Bağlantıyı Kes

Araba Sayısı: 6

Boş Park Yeri Sayısı: 0

Port Durum: AÇIK

Yeni Araç Giriş Yaptı  
Giriş Tarihi: 5.06.2024 12:08:10

Tamam

### 3. Yeni Araç Girişi

Yeni bir araç girişi olduğunda, sistem bu durumu seri port üzerinden gelen '1' sinyali ile algılar. Bu durumda, veri tabanına mevcut araç için otomatik bir ID, giriş tarihi, giriş saati bilgileri ile yeni bir araç kaydı eklenir ve uygulama üzerindeki mevcut araç sayısı ve boş park alan sayıları güncellenir.

SenSem Otoparkı

## SenSem OTOPARK OTOMASYONU

ArabaID	GirisSaati	GirisTarihi
237	11:29:57	05.06.2024
236	11:29:32	05.06.2024
235	11:29:00	
234	11:28:55	
233	11:28:47	

Port Seçin: COM5

Bağlan

Bağlantıyı Kes

Araba Sayısı: 6

Port Durum: AÇIK

Ödeme Onayı

Araba ID: 233

Kaldığı Süre: 0 gün 0 saat 37 dakika

Ödenecek Tutar: 185,00 TL

Araba ödemedi mi?

Evet Hayır

SenSem Otoparkı

## SenSem OTOPARK OTOMASYONU

ArabaID	GirisSaati	GirisTarihi
237	11:29:57	05.06.2024
236	11:29:32	05.06.2024
235	11:29:00	05.06.2024
234	11:28:55	
233	11:28:47	

Port Seçin: COM5

Bağlan

Bağlantıyı Kes

Araba Sayısı: 5

Port Durum: AÇIK

Araba çıkış yaptı ve ödeme alındı.

Tamam

SenSem Otoparkı

## SenSem OTOPIARK OTOMASYONU

ArabaID	GirisSaati	GirisTarihi
237	11:29:57	05.06.2024
236	11:29:32	05.06.2024
235	11:29:00	05.06.2024
234	11:28:5	
233	11:28:4	
1	22.05.2	
231	05.06.2	
229	05.06.2024	09:06:46
226	05.06.2024	08:46:47
228	05.06.2024	08:59:26
225	05.06.2024	08:46:44

Port Seçin: COM5

Bağlan

Bağlantıyı Kes

Araba çıkışı yapılamaz. Lütfen önce ödemeyi yapın.

Tamam

Port Durum: AÇIK

#### 4. Araç Çıkışı

Bir araç çıktığında, sistem bu durumu seri port üzerinden gelen "2" sinyali ile algılar. Bu durumda, veri tabanından rastgele seçilen bir araç ID' si ile giriş ve çıkış zamanlarını hesaplar ve dakikada 5 TL olacak şekilde bir tutar belirler. Sonrasında, kullanıcıya yöneltilen "ödeme yapıldı mı?" sorusuna verilen yanıt doğrultusunda, aracın çıkış işlemi onaylanır. Ödeme onayı alındığında, aracın mevcut bilgileri "ArabaCikis2" tablosuna aktarılır ve "Araba" tablosundan silinir. Ardından, otoparktaki mevcut araç sayısı ve boş park alanı bilgileri güncellenir.

SenSem Otoparkı

## SenSem OTO PARK OTOMASYONU

ArabaID	GirisSaati	GirisTarihi
238	12:08:10	05.06.2024
237	11:29:57	05.06.2024
236	11:29:32	05.06.2024
235	11:29:00	05.06.2024
234	11:28:55	05.06.2024

Port Seçin: COM5

Bağlan

Bağlantıyı Kes

Otoparktaki Araba Sayısı: 6

Boş Alan Sayısı: 0

ArabaID	GirisTarihi	GirisSaati
1	22.05.2024	10:30:00
232	05.06.2024	11:28:41
231	05.06.2024	11:28:18
229	05.06.2024	09:06:46
226	05.06.2024	08:46:47
228	05.06.2024	08:50:26

Port Durum: KAPALI

### 5. Seri Port Bağlantısının Kapatılması

Kullanıcı, kapat butonuna basarak seri port bağlantısını kapatabilir. Bu işlem, seri portun kapatılmasını sağlar ve bağlantı port durumu “KAPALI” olarak etiketlenir. Kapat butonu devre dışı bırakılır ve bağlan butonu tekrar etkinleştirilir.

### 6. Uygulamanın Kapatılması

Uygulama kapatıldığında, açık olan seri port bağlantısı varsa kapatılır ve kaynaklar serbest bırakılır. Bu, sistemin güvenli bir şekilde kapanmasını ve sonraki açılıшта sorunsuz çalışmasını sağlar.

## Kaynaklar

- [1] [https://tr.wikipedia.org/wiki/Arduino\\_IDE](https://tr.wikipedia.org/wiki/Arduino_IDE)
- [2] <https://learn.microsoft.com/tr-tr/visualstudio/get-started/visual-studio-ide?view=vs-2022>
- [3] <https://tr.wikipedia.org/wiki/C%E2%99%AF>
- [4] [https://tr.wikipedia.org/wiki/Microsoft\\_SQL\\_Server](https://tr.wikipedia.org/wiki/Microsoft_SQL_Server)

## Kaynak kodları – Ekler

### Sketch\_apr26a.ino dosyası

```
#include <Wire.h> // I2C haberleşme protokolünü kullanmak için Wire kütüphanesini dahil eder.
```



```
#include "servoMotorControls.h" // Servo motor kontrol kütüphanesini projeye dahil eder.

#define led_pin_red 18 // Kırmızı LED'in pin numarasını 18 olarak tanımlar.
#define led_pin_green 19 // Yeşil LED'in pin numarasını 19 olarak tanımlar.

#define echoPin 2 // İlk mesafe sensörünün echo pini için pin numarasını 2 olarak tanımlar.
#define trigPin 4 // İlk mesafe sensörünün trig pini için pin numarasını 4 olarak tanımlar.

#define echoPin2 33 // İkinci mesafe sensörünün echo pini için pin numarasını 33 olarak tanımlar.
#define trigPin2 32 // İkinci mesafe sensörünün trig pini için pin numarasını 32 olarak tanımlar.

const byte servoPin = 12; // Servo motorun bağlı olduğu pin numarasını 12 olarak tanımlar.

servoMotorControls myServoControl(servoPin); // Servo motor kontrol nesnesini oluşturur ve servo pinini parametre olarak geçirir.

long duration, distance, duration2, distance2; // Mesafe sensörlerinden alınan süre ve mesafe değerlerini saklamak için değişkenler tanımlar.
int notEmptyPlace = 3; // Otoparkta dolu yer sayısını saklamak için bir değişken tanımlar ve başlangıç değerini 3 olarak atar.

unsigned long servoActivatedTime = 0; // Servo motorun son aktive edildiği zamanı saklamak için bir değişken tanımlar ve başlangıç değerini sıfır olarak atar.
bool isServoActive = false; // Servo motorun aktif olup olmadığını saklamak için bir bayrak tanımlar ve başlangıç değerini false olarak atar.
bool ArabaGirdiMi = false; // Arabanın içeri girdiği kontrolünü saklamak için bir bayrak tanımlar ve başlangıç değerini false olarak atar.

unsigned long previousMillis = 0; // Önceki millis süresini saklamak için bir değişken tanımlar ve başlangıç değerini sıfır olarak atar.
const long interval = 1000; // Her bir okuma arasındaki zaman aralığını saklamak için bir değişken tanımlar ve başlangıç değerini 1000 (1 saniye) olarak atar.

void setup() {
    Serial.begin(9600); // Seri haberleşmeyi başlatır ve baud hızını 9600 olarak ayarlar.
    pinMode(led_pin_red, OUTPUT); // Kırmızı LED pinini çıkış olarak ayarlar.
    pinMode(led_pin_green, OUTPUT); // Yeşil LED pinini çıkış olarak ayarlar.
    pinMode(trigPin, OUTPUT); // İlk mesafe sensörünün trig pinini çıkış olarak ayarlar.
    pinMode(echoPin, INPUT); // İlk mesafe sensörünün echo pinini giriş olarak ayarlar.
    pinMode(trigPin2, OUTPUT); // İkinci mesafe sensörünün trig pinini çıkış olarak ayarlar.
    pinMode(echoPin2, INPUT); // İkinci mesafe sensörünün echo pinini giriş olarak ayarlar.
}

void loop() {
    unsigned long currentMillis = millis(); // Şu anki millis zamanını alır.

    if (notEmptyPlace > 5) {
```

```

    digitalWrite(led_pin_red, HIGH); // Kırmızı LED'i yakar.
    digitalWrite(led_pin_green, LOW); // Yeşil LED'i söndürür.
} else {
    digitalWrite(led_pin_green, HIGH); // Yeşil LED'i yakar.
    digitalWrite(led_pin_red, LOW); // Kırmızı LED'i söndürür.
}

if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis; // Önceki millis zamanını günceller.

    // Mesafe sensörü 1
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance = duration / 58.2;

    // Mesafe sensörü 2
    digitalWrite(trigPin2, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin2, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin2, LOW);
    duration2 = pulseIn(echoPin2, HIGH);
    distance2 = duration2 / 58.2;
}

// Seri porttan veri okunur
if (Serial.available() > 0) {
    String data = Serial.readStringUntil('\n'); // Seri porttan gelen veriyi okur.
    Serial.println(data); // Okunan veriyi seri porta yazdırır.
    int delimiterIndex = data.indexOf(':'); // Veri içerisindeki ":" karakterinin indeksini
    bulur.
    if (delimiterIndex != -1) {
        String valueStr = data.substring(delimiterIndex + 1); // ":" karakterinden sonraki kısmı
        alır.
        notEmptyPlace = valueStr.toInt(); // Alınan kısmı integer'a dönüştürür ve dolu park
        yeri sayısını günceller.
    } else {
        Serial.println("Delimiter not found!"); // Eğer ":" karakteri bulunamazsa hata mesajı
        yazdırır.
    }
}

// Servo kontrolü
if (distance < 7 && notEmptyPlace < 6 && !isServoActive) {
    myServoControl.servoTurn(180); // Servo motoru 180 derece döndürür.
    servoActivatedTime = currentMillis; // Servo motorun aktive edildiği zamanı kaydeder.
}

```

```

    isServoActive = true; // Servo motorun aktif olduğunu işaretler.
    ArabaGirdiMi = false; // Arabanın içeri girdiği kontrolünü sıfırlar.
}

// Servo motoru 5 saniye sonra kapar
if (isServoActive && currentMillis - servoActivatedTime >= 5000) {
    myServoControl.servoTurn(90); // Servo motoru 90 derece döndürür.
    isServoActive = false; // Servo motorunun aktif olduğunu işaretler.
}

if (distance2 > 19 && distance2 < 24 && isServoActive && !ArabaGirdiMi) {
    ArabaGirdiMi = true; // Arabanın içeri girdiğini işaretler.
    Serial.println("1"); // Seri porta "1" yazdırır.
    while (distance2 > 19 && distance2 < 24) {
        digitalWrite(trigPin2, LOW);
        delayMicroseconds(2);
        digitalWrite(trigPin2, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin2, LOW);
        duration2 = pulseIn(echoPin2, HIGH);
        distance2 = duration2 / 58.2;
    }
} else if (distance2 < 8) {
    Serial.println("2"); // Seri porta "2" yazdırır.
    while (distance2 < 8) {
        digitalWrite(trigPin2, LOW);
        delayMicroseconds(2);
        digitalWrite(trigPin2, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin2, LOW);
        duration2 = pulseIn(echoPin2, HIGH);
        distance2 = duration2 / 58.2;
    }
}
}
}

```

### **servoMotorControls.cpp dosyası**

```

// servoMotorControls.cpp

#include "servoMotorControls.h" // Servo motor kontrol sınıfının başlık dosyası

// Sınıfın kurucusu (constructor)
servoMotorControls::servoMotorControls(byte servoPin) {
    _servoPin = servoPin; // Servo pinini belirler
    myServo.attach(servoPin); // Servo motoru belirlenen pine bağlar
}

// Servo motoru belirli bir dereceye döndürmek için kullanılan fonksiyon
void servoMotorControls::servoTurn(int degree) {

```

```
myServo.write(degree); // Servo motoru belirtilen dereceye döndürür  
}
```

### **servoMotorControls.h dosyası**

```
// servoMotorControls.h  
  
//servo motor kontrolü için sınıfın tanımını içerir. Sınıfın fonksiyonları ve değişkenleri  
burada tanımlanmıştır  
#ifndef SERVO_MOTOR_CONTROLS_H  
#define SERVO_MOTOR_CONTROLS_H  
  
#include <ESP32Servo.h> // ESP32 için servo motor kütüphanesi  
  
// servoMotorControls sınıfının tanımı  
class servoMotorControls {  
public:  
    // Kurucu fonksiyon (constructor), servo pinini parametre olarak alır  
    servoMotorControls(byte servoPin);  
  
    // Servo motoru belirli bir dereceye döndürmek için kullanılan fonksiyon  
    void servoTurn(int degree);  
  
private:  
    Servo myServo; // Servo motor nesnesi  
    byte _servoPin; // Servo motorun bağlı olduğu pin  
};  
  
#endif // SERVO_MOTOR_CONTROLS_H
```

### **Masaüstü uygulaması kaynak kodlar Form1.cs dosyası**

```
using System; // Temel C# kütüphanelerini içerir, temel veri türleri ve yöntemler sağlar.  
using System.Windows.Forms; // Windows Forms uygulamaları için gerekli sınıfları içerir.  
using System.IO.Ports; // Seri port iletişimi için gerekli sınıfları içerir.  
using System.Data.SqlClient; // SQL Server veritabanı bağlantısı ve işlemleri için gerekli  
sınıfları içerir.  
using System.Data; // Veri setleri ve veri tabloları gibi veri yapıları ile çalışmak için gerekli  
sınıfları içerir.  
using System.Collections.Generic; // Generic koleksiyon sınıflarını (List, Dictionary, vb.)  
içerir.
```

```
namespace Otopark_Otomasyonu  
{  
    public partial class Form1 : Form  
    {  
        SerialPort sr = new SerialPort(); // Seri port nesnesi oluşturuluyor  
  
        public Form1()  
        {  
            InitializeComponent();  
        }  
    }  
}
```

```

        sr.DataReceived += new
SerialDataReceivedEventHandler(SerialDataReceivedHandler); // Seri porttan veri
alındığında çalışacak olan olay işleyici atanıyor
    }

    // Boş park sayısını hesaplayan fonksiyon
    private int GetEmptyParkCount()
    {
        int totalPark = 6; // Toplam park sayısı
        int arabaCounts = GetArabaCount(); // Parkta bulunan araç sayısını al
        return totalPark - arabaCounts; // Boş park sayısını hesapla
    }

    // Parkta bulunan araç sayısını veritabanından alan fonksiyon
    private int GetArabaCount()
    {
        int count = 0;
        string connectionString = "Data Source=DESKTOP-2JMETKU;Initial
Catalog=sensem;Integrated Security=True;Encrypt=False"; // Veritabanı bağlantı dizesi

        using (SqlConnection con = new SqlConnection(connectionString))
        {
            try
            {
                con.Open(); // Veritabanı bağlantısını aç
                string query = "SELECT COUNT(*) FROM Araba"; // Araç sayısını almak
için SQL sorgusu
                using (SqlCommand cmd = new SqlCommand(query, con))
                {
                    count = (int)cmd.ExecuteScalar(); // Sorguyu çalıştır ve sonucu al
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Araba count retrieval error: " + ex.Message); // Hata
durumunda mesaj göster
            }
        }
        return count; // Araç sayısını döndür
    }

    // Form yüklendiğinde çalışacak olan olay işleyici
    private void Form1_Load(object sender, EventArgs e)
    {
        string[] portNames = SerialPort.GetPortNames(); // Mevcut seri port isimlerini al
        comboBoxPortName.Items.AddRange(portNames); // Seri port isimlerini
combobox'a ekle

        string connectionString = "Data Source=DESKTOP-2JMETKU;Initial
Catalog=sensem;Integrated Security=True;Encrypt=False"; // Veritabanı bağlantı dizesi

```

```

using (SqlConnection con = new SqlConnection(connectionString))
{
    try
    {
        con.Open(); // Veritabanı bağlantısını aç
        string query = "SELECT * FROM Araba ORDER BY GirişSaati DESC"; //
        Araç verilerini almak için SQL sorgusu
        using (SqlCommand cmd = new SqlCommand(query, con))
        {
            SqlDataAdapter adapter = new SqlDataAdapter(cmd); // Sorgu sonucunu
            almak için adapter oluştur
            DataTable table = new DataTable(); // DataTable oluştur
            adapter.Fill(table); // DataTable'ı doldur
            dataGridView1.DataSource = table; // DataGridView'a veri kaynağı olarak
            ata
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Veri çekerken hata oluştu: " + ex.Message); // Hata
        durumunda mesaj göster
    }
}

```

```

using (SqlConnection con = new SqlConnection(connectionString))
{
    try
    {
        con.Open(); // Veritabanı bağlantısını aç
        string query = "SELECT * FROM ArabaCikis2 ORDER BY CikisTarihi
        DESC, CikisSaati DESC"; // Çıkış yapmış araç verilerini almak için SQL sorgusu
        using (SqlCommand cmd = new SqlCommand(query, con))
        {
            SqlDataAdapter adapter = new SqlDataAdapter(cmd); // Sorgu sonucunu
            almak için adapter oluştur
            DataTable table = new DataTable(); // DataTable oluştur
            adapter.Fill(table); // DataTable'ı doldur
            dataGridView2.DataSource = table; // DataGridView'a veri kaynağı olarak
            ata
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("ArabaCikis2 verilerini çekerken hata oluştu: " +
        ex.Message); // Hata durumunda mesaj göster
    }
}

```

```

int arabaCount = GetArabaCount(); // Parktaki araç sayısını al
int emptyParks = GetEmptyParkCount(); // Boş park sayısını al

```

```

        labelArabaCount.Text = arabaCount.ToString(); // Araç sayısını etikete yaz
        labelEmptyParks.Text = emptyParks.ToString(); // Boş park sayısını etikete yaz
    }

    // Seri port açma butonuna tıklandığında çalışacak olay işleyici
    private void button1_Click(object sender, EventArgs e)
    {
        try
        {
            if (sr.IsOpen)
            {
                sr.Close(); // Eğer seri port açıksa kapat
            }

            if (comboBoxPortName.SelectedItem != null)
            {
                sr.PortName = comboBoxPortName.SelectedItem.ToString(); // Seçilen seri
port adını al
                sr.BaudRate = 9600; // Baud rate ayarla
                sr.Open(); // Seri portu aç
                button1.Enabled = false; // Açma butonunu devre dışı bırak
                button2.Enabled = true; // Kapama butonunu etkinleştir
                label3.Text = "AÇIK"; // Etiket "AÇIK" yaz
                sr.WriteLine("sayı:" + labelArabaCount.Text); // Seri porttan araç sayısını
gönder
            }
            else
            {
                MessageBox.Show("Lütfen bir port seçin."); // Port seçilmemişse mesaj göster
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Seri port açılırken hata oluştu: " + ex.Message); // Hata
durumunda mesaj göster
        }
    }

    // Araç ID'lerini veritabanından alan fonksiyon
    private List<int> GetArabaIDs()
    {
        List<int> arabaIDs = new List<int>();
        string connectionString = "Data Source=DESKTOP-2JMETKU;Initial
Catalog=sensem;Integrated Security=True;Encrypt=False"; // Veritabanı bağlantı dizesi

        using (SqlConnection con = new SqlConnection(connectionString))
        {
            try
            {
                con.Open(); // Veritabanı bağlantısını aç

```

```

        string query = "SELECT ArabaID FROM Araba"; // Araç ID'lerini almak için
SQL sorgusu
        using (SqlCommand cmd = new SqlCommand(query, con))
        {
            SqlDataReader reader = cmd.ExecuteReader(); // Sorgu sonucunu okuyucu
ile al
            while (reader.Read())
            {
                arabaIDs.Add(reader.GetInt32(0)); // Her bir ID'yi listeye ekle
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Araba ID Hatası: " + ex.Message); // Hata durumunda
mesaj göster
        }
    }

    return arabaIDs; // Araç ID'lerini döndür
}

```

```

// Seri porttan veri alındığında çalışacak olay işleyici
private void SerialDataReceivedHandler(object sender, SerialDataReceivedEventArgs
e)
{
    try
    {
        SerialPort sp = (SerialPort)sender;
        string inData = sp.ReadExisting().Trim(); // Gelen veriyi al ve boşlukları temizle

        // Gelen veriyi konsola yaz
        Console.WriteLine("Received data: " + inData);

        if (inData == "1")
        {
            if (GetArabaCount() < 6)
            {
                HandleNewCarEntry(); // Yeni araç girişi işlemi
            }
            else
            {
                MessageBox.Show("Otoparkta Boş Alan Yok"); // Otoparkta boş alan yoksa
mesaj göster
            }
        }
        else if (inData == "2")
        {
            HandleCarExit(); // Araç çıkışı işlemi
        }
    }
}

```



```

        else
        {
            Console.WriteLine("Unexpected data received: " + inData); // Beklenmeyen
            veri alındığında konsola yaz
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Data received handler error: " + ex.Message); // Hata
        durumunda mesaj göster
    }
}

// Yeni araç girişi işlemi
private void HandleNewCarEntry()
{
    DateTime currentTime = DateTime.Now; // Şu anki zamanı al
    string girisSaati = currentTime.ToString("HH:mm:ss"); // Giriş saatini al
    string girisTarihi = currentTime.ToString("dd.MM.yyyy"); // Giriş tarihini al

    string connectionString = "Data Source=DESKTOP-2JMETKU;Initial
    Catalog=sensem;Integrated Security=True;Encrypt=False"; // Veritabanı bağlantı dizesi
    using (SqlConnection con = new SqlConnection(connectionString))
    {
        try
        {
            con.Open(); // Veritabanı bağlantısını aç
            string query = "INSERT INTO Araba (GirisSaati, GirisTarihi) VALUES
            (@GirisSaati, @GirisTarihi)"; // Araç giriş bilgilerini eklemek için SQL sorgusu
            using (SqlCommand cmd = new SqlCommand(query, con))
            {
                cmd.Parameters.AddWithValue("@GirisSaati", girisSaati); // Giriş saati
                parametresini ekle
                cmd.Parameters.AddWithValue("@GirisTarihi", girisTarihi); // Giriş tarihi
                parametresini ekle
                cmd.ExecuteNonQuery(); // Sorguyu çalıştır
            }

            this.Invoke(new Action(() =>
            {
                Form1_Load(null, null); // Formu yeniden yükle
                int arabaCount = GetArabaCount(); // Araç sayısını al
                labelArabaCount.Text = arabaCount.ToString(); // Araç sayısını etikete yaz
                sr.WriteLine("sayi:" + labelArabaCount.Text); // Seri porttan araç sayısını
                gönder
            }
            ));

            MessageBox.Show("Yeni Araç Giriş Yaptı\nGiriş Tarihi: " + currentTime); //
            Yeni araç girişi mesajı göster
        }
    }
}

```

```

        catch (Exception ex)
        {
            MessageBox.Show("Araba giriş hatası: " + ex.Message); // Hata durumunda
mesaj göster
        }
    }
}

// Araç çıkışı işlemi
private void HandleCarExit()
{
    List<int> arabaIDs = GetArabaIDs(); // Araç ID'lerini al
    if (arabaIDs.Count > 0)
    {
        Random rnd = new Random();
        int indexToDelete = arabaIDs[rnd.Next(arabaIDs.Count)]; // Rastgele bir araç ID
seç

        string connectionString = "Data Source=DESKTOP-2JMETKU;Initial
Catalog=sensem;Integrated Security=True;Encrypt=False"; // Veritabanı bağlantı dizesi
        using (SqlConnection con = new SqlConnection(connectionString))
        {
            try
            {
                con.Open(); // Veritabanı bağlantısını aç

                // Araç giriş zamanını al
                DateTime girisZamani;
                string selectQuery = "SELECT GirisSaati, GirisTarihi FROM Araba
WHERE ArabaID = @ID"; // Giriş bilgilerini almak için SQL sorgusu
                using (SqlCommand selectCmd = new SqlCommand(selectQuery, con))
                {
                    selectCmd.Parameters.AddWithValue("@ID", indexToDelete); // Araç
ID'si parametresini ekle
                    using (SqlDataReader reader = selectCmd.ExecuteReader())
                    {
                        if (reader.Read())
                        {
                            girisZamani = DateTime.ParseExact(reader["GirisTarihi"].ToString()
+ " " + reader["GirisSaati"].ToString(), "dd.MM.yyyy HH:mm:ss", null); // Giriş zamanını
al
                        }
                        else
                        {
                            MessageBox.Show("Araba bilgisi bulunamadı."); // Araç bilgisi
bulunamadıysa mesaj göster
                            return;
                        }
                    }
                }
            }
        }
    }
}

```

```

// Şu anki zaman
DateTime cikisZamani = DateTime.Now;

// Kaldığı süreyi hesapla
TimeSpan kalinanSure = cikisZamani - girisZamani;
int totalMinutes = (int)kalinanSure.TotalMinutes; // Toplam dakikayı al
double odeme = totalMinutes * 5; // Dakika başına 5 TL ücret

// Ücreti göster ve ödeme onayı iste
DialogResult result = MessageBox.Show($"Araba ID:
{indexToDelete}\nKaldığı Süre: {kalinanSure.Days} gün {kalinanSure.Hours} saat
{kalinanSure.Minutes} dakika\nÖdenecek Tutar: {odeme:F2} TL\nAraba ödemeyi yaptı
mı?", "Ödeme Onayı", MessageBoxButtons.YesNo);
if (result == DialogResult.Yes)
{
    // ArabaCikis2 tablosu için IDENTITY_INSERT'i etkinleştir
    string enableIdentityInsertQuery = "SET IDENTITY_INSERT
ArabaCikis2 ON";
    using (SqlCommand enableIdentityInsertCmd = new
SqlCommand(enableIdentityInsertQuery, con))
    {
        enableIdentityInsertCmd.ExecuteNonQuery();
    }

    // Kayıtları ArabaCikis2 tablosuna ekle
    string insertQuery = "INSERT INTO ArabaCikis2 (ArabaID, GirisTarihi,
GirisSaati, CikisTarihi, CikisSaati, OdemeTutari) " +
        "VALUES (@ID, @GirisTarihi, @GirisSaati, @CikisTarihi,
@CikisSaati, @OdemeTutari)";
    using (SqlCommand insertCmd = new SqlCommand(insertQuery, con))
    {
        insertCmd.Parameters.AddWithValue("@ID", indexToDelete); // Araç
ID'si parametresini ekle
        insertCmd.Parameters.AddWithValue("@GirisTarihi",
girisZamani.ToString("dd.MM.yyyy")); // Giriş tarihi parametresini ekle
        insertCmd.Parameters.AddWithValue("@GirisSaati",
girisZamani.ToString("HH:mm:ss")); // Giriş saati parametresini ekle
        insertCmd.Parameters.AddWithValue("@CikisTarihi",
cikisZamani.ToString("dd.MM.yyyy")); // Çıkış tarihi parametresini ekle
        insertCmd.Parameters.AddWithValue("@CikisSaati",
cikisZamani.ToString("HH:mm:ss")); // Çıkış saati parametresini ekle
        insertCmd.Parameters.AddWithValue("@OdemeTutari", odeme); //
Ödeme tutarı parametresini ekle
        insertCmd.ExecuteNonQuery(); // Sorguyu çalıştır
    }

    // ArabaCikis2 tablosu için IDENTITY_INSERT'i devre dışı bırak
    string disableIdentityInsertQuery = "SET IDENTITY_INSERT
ArabaCikis2 OFF";

```

```

        using (SqlCommand disableIdentityInsertCmd = new
SqlCommand(disableIdentityInsertQuery, con))
        {
            disableIdentityInsertCmd.ExecuteNonQuery();
        }

        // Araba tablosundan kaydı sil
        string deleteQuery = "DELETE FROM Araba WHERE ArabaID = @ID";
        using (SqlCommand deleteCmd = new SqlCommand(deleteQuery, con))
        {
            deleteCmd.Parameters.AddWithValue("@ID", indexToDelete); // Araç
ID'si parametresini ekle
            deleteCmd.ExecuteNonQuery(); // Sorguyu çalıştır
        }

        this.Invoke(new Action(() =>
        {
            Form1_Load(null, null); // Formu yeniden yükle
            int arabaCount = GetArabaCount(); // Araç sayısını al
            labelArabaCount.Text = arabaCount.ToString(); // Araç sayısını etikete
yaz
            sr.WriteLine("sayı:" + labelArabaCount.Text); // Seri porttan araç
sayısını gönder
        }));

        MessageBox.Show("Araba çıkış yaptı ve ödeme alındı."); // Çıkış ve
ödeme mesajı göster
        }
        else
        {
            MessageBox.Show("Araba çıkışı yapılamaz. Lütfen önce ödemeyi
yapın."); // Ödeme yapılmadıysa mesaj göster
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Araba çıkış hatası: " + ex.Message); // Hata durumunda
mesaj göster
    }
}
}

// Form kapandığında çalışacak olay işleyici
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (sr.IsOpen)
    {
        sr.Close(); // Eğer seri port açıksa kapat
    }
}

```

```
}

// Seri portu kapatma butonuna tıklandığında çalışacak olay işleyici
private void button2_Click(object sender, EventArgs e)
{
    if (sr.IsOpen)
    {
        sr.Close(); // Seri portu kapat
        button1.Enabled = true; // Açma butonunu etkinleştir
        button2.Enabled = false; // Kapama butonunu devre dışı bırak
        label3.Text = "KAPALI"; // Etikete "KAPALI" yaz
    }
}
}
```