Raspberry Pi ile Akıllı Araç Park Sensörü Sistemi Geliştirilmesi

Ekip Üyeleri

- Emre KAYA 2022123151
- Ahmet Oğuzhan CEYHAN 2022123049
- Talha ANAY 2022123061
- Tunahan KARA 2022123038
- Serkan DEMİR 2022123021
- Görkem DAĞ 2022123033
- Mahmut KAYHAN 2022123039
- Furkan YOĞURTÇU 2022123023

Proje Konusu : Akıllı Araç Park Sensörü Sistemi

Projenin Tanıtımı: Günümüzde artan araç sayısı ve park yeri sorunları, park etme sürecini zorlaştırmaktadır. Bu proje, Raspberry Pi mikrodenetleyicisi kullanarak geliştirilen akıllı bir park sensörü sistemi ile bu soruna çözüm sunmayı amaçlamaktadır. Sistem, araçların park manevralarını kolaylaştırmak, park esnasında oluşabilecek kazaları önlemek ve sürücülere daha güvenli bir park deneyimi sunmak için tasarlanmıştır.

Amaç: Bu projenin temel amacı, ultrasonik sensörler yardımıyla aracın çevresindeki engelleri algılayan ve sürücüyü sesli ve görsel uyarılarla bilgilendiren düşük maliyetli ve etkili bir park sensörü sistemi geliştirmektir. Proje aynı zamanda Raspberry Pi'nin mikrodenetleyici olarak araç elektroniği uygulamalarında kullanım potansiyelini göstermeyi hedeflemektedir.

Kullanılan Teknolojiler Ve Araçlar

Raspberry Pi: Raspberry Pi, düşük maliyetli, kredi kartı boyutunda bir bilgisayardır. Bu projede, Raspberry Pi'nin işlem gücü, sensörlerden gelen verileri işlemek ve uyarıları üretmek için kullanılacaktır. Raspberry Pi'nin GPIO (General Purpose Input/Output) pinleri, sensörler ve diğer elektronik bileşenlerle arayüz oluşturmak için kullanılacaktır.

Ultrasonik Sensörler (HC-SR04): Bu sensörler, ses dalgalarını kullanarak mesafeyi ölçerler. Verici (transmitter) bir ses dalgası gönderir ve alıcı (receiver) bu dalganın bir engelden yansıdıktan sonra geri dönmesini bekler. Geri dönüş süresine göre mesafe hesaplanır. HC-SR04 sensörü, düşük maliyeti ve kolay kullanımı nedeniyle bu proje için uygun bir seçimdir.

Buzzer (Sesli Uyarı): Buzzer, engellerin yaklaştığını belirtmek için sesli uyarı üretir. Mesafe azaldıkça uyarı frekansı artar.

LED Göstergeler : Farklı renklerde LED'ler (yeşil, sarı, kırmızı) kullanarak mesafeyi görsel olarak göstermek mümkündür.

Breadboard : Breadboard, elektronik devrelerin geçici olarak kurulması ve test edilmesi için kullanılan, lehim gerektirmeyen bir platformdur. Üzerindeki delikler ve dahili bağlantılar sayesinde elektronik bileşenler ve kablolar kolayca yerleştirilir ve bağlanır.GPIO pinlerinden gelen sinyaller, breadboard aracılığıyla sensörler ve diğer bileşenlere yönlendirilmiştir.Örneğin, HC-SR04 **sensörünün** Trigger ve Echo pinleri breadboard üzerinden bağlanmıştır.

GPIO(General Purpose Input/Output): Ultrasonik sensör (HC-SR04) ve aktif buzzer gibi bileşenlerin Raspberry Pi ile haberleşmesi için kullanılmıştır.

Yazılım Dili (Python): Python, Raspberry Pi üzerinde yaygın olarak kullanılan, okunabilir ve öğrenmesi kolay bir programlama dilidir. Sensörlerden veri okuma, veri işleme ve uyarı üretme işlemleri Python ile gerçekleştirilmiştir.
Ayrıca, Raspberry Pi ile birlikte kullanılan
RPi.GPIO kütüphanesini içermektedir.

Mikroişlemci ve Yazılım Modelinin İşleyişi

- **Sensör Veri Okuma:** Python kodu, Raspberry Pi'nin GPIO pinlerini kullanarak ultrasonik sensörlerden mesafe verilerini okur.
- **Veri İşleme:** Okunan veriler, gürültü ve hataları filtrelemek için işlenir. Mesafe değerleri belirli aralıklarla karşılaştırılarak engelin ne kadar yakın olduğu belirlenir.
- **Uyarı Üretme:** Mesafe belirli bir eşiğin altına düştüğünde, buzzer aktif hale getirilir ve LED'ler yakılır. Mesafe azaldıkça buzzer'ın ses çıkarma sıklığı artar.

Sistemin Çalışma Prensibi

Geliştirilen sistem, ultrasonik sensörler aracılığıyla çevredeki nesnelerle olan mesafeyi algılar ve bu verileri Raspberry Pi'ye iletir. Sistemin işleyişi şu adımları takip edecektir:

1. **Mesafe Ölçümü:**

   Ultrasonik sensör, belirli aralıklarla ses dalgaları gönderir ve dalgaların bir yüzeye çarparak geri dönme süresini hesaplayarak mesafeyi belirler.

2. **Veri İşleme:**

   Raspberry Pi, sensörden gelen verileri işler ve bu verileri yazılım algoritmasına aktarır. Yazılım, ölçülen mesafeye göre uygun tepkiyi belirler.

3. **Görsel ve Sesli Uyarılar:**

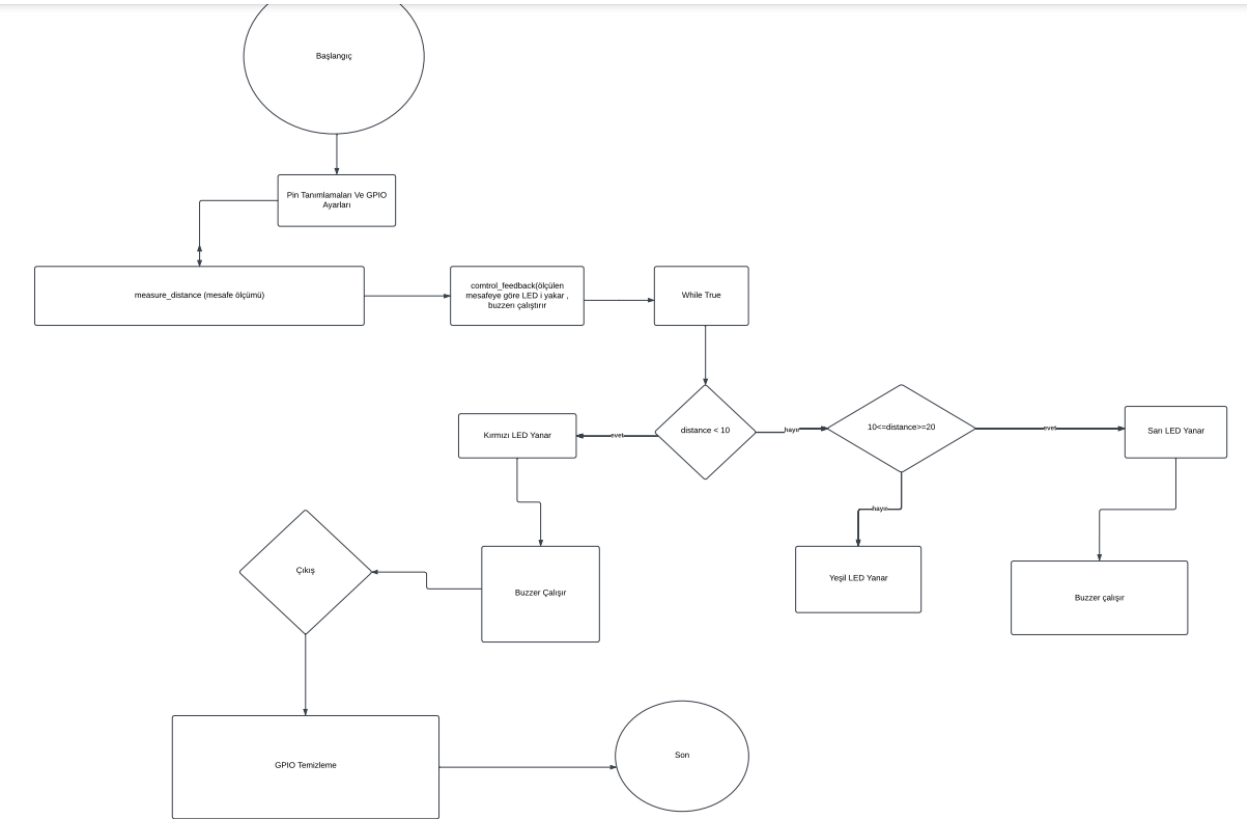
   - Nesne mesafesi, belirlenen eşik değerlerin altında ise aktif buzzer devreye girerek sesli bir uyarı verir. Aynı zamanda, mesafeye göre LED'ler farklı renklerle yanarak görsel bir uyarı sağlar.

4. **Gerçek Zamanlı Çalışma:**

   Sistem, mesafe ölçümünü sürekli olarak tekrarlayarak sürücünün park etme sırasında engellere olan uzaklığını gerçek zamanlı olarak bildirir.

Yazılım ve Algoritma

Proje Akış Şeması



Park Sensörü İçin Kullanılan Akış Şeması

Kodlar Ve Açıklamaları

```
import RPi.GPIO as GPIO
import time
```

- `RPi.GPIO`: Raspberry Pi'nin GPIO pinlerini kontrol etmek için kullanılan bir kütüphane.
- `time`: Zaman gecikmeleri ve zaman ölçümleri için kullanılan standart Python modülü.

```
TRIG = 23
ECHO = 24
```

```
BUZZER = 18
LEDS = {"RED": 22, "YELLOW": 27, "GREEN": 17}
```

- **TRIG ve ECHO**: Ultrasonik sensörün tetikleme (TRIG) ve yankı (ECHO) pinleri. TRIG, ses dalgalarını başlatırken, ECHO pininden yansıyan sinyaller okunur.
- **BUZZER**: Sesli uyarılar için kullanılan buzzer'ın bağlandığı pin.
- **LEDS**: Farklı durumlara göre yanan kırmızı, sarı ve yeşil LED'lerin pinleri.

GPIO Ayarları

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)
GPIO.setup(BUZZER, GPIO.OUT)
for color in LEDS.values():
GPIO.setup(color, GPIO.OUT)
```

- `GPIO.setmode(GPIO.BCM)`: Pin numaralandırmasını **BCM (Broadcom)** modunda yapar.
- `GPIO.setup()`: Pinleri giriş (**IN**) veya çıkış (**OUT**) olarak tanımlar.
- **LED Pinleri**: LED'lerin her biri çıkış olarak ayarlanır.

Mesafe Ölçüm Fonksiyonu

```
def measure_distance():
    GPIO.output(TRIG, False)
    time.sleep(0.2)

    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)

    while GPIO.input(ECHO) == 0:
```

```
        pulse_start = time.time()

    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150
    return round(distance, 2)
```

- `TRIG` **ile Ses Dalgası Gönderme**:
  - TRIG pini 10 mikro saniye boyunca **HIGH** yapılır, bu da ultrasonik sensörün ses dalgaları yaymasını sağlar.
- **ECHO ile Sesin Yankısını Ölçme**:
  - ECHO pini **0**'dan **1**'e geçtiğinde ses dalgası gönderilmeye başlanır, **1**'den tekrar **0**'a geçtiğinde yankı alınmış olur.
  - Yankının süresi ölçülerek mesafe hesaplanır (**mesafe = süre × 17150**).
  - Hesaplanan değer santimetreye çevrilir ve döndürülür.

LED ve Buzzer Kontrol Fonksiyonu

```
def control_feedback(distance):
    if distance < 10:
        GPIO.output(LEDS["RED"], True)
        GPIO.output(BUZZER, True)
    elif 10 <= distance < 20:
        GPIO.output(LEDS["YELLOW"], True)
        GPIO.output(BUZZER, True)
        time.sleep(0.1)
        GPIO.output(BUZZER, False)
    elif 20 <= distance <= 50:
        GPIO.output(LEDS["GREEN"], True)
        GPIO.output(BUZZER, True)
        time.sleep(0.5)
        GPIO.output(BUZZER, False)
```

```
    else:
        GPIO.output(BUZZER, False)
```

- `distance`: Mesafeyi temsil eden bir sayı (örneğin, sensör tarafından ölçülen bir mesafe). Mesafeye bağlı olarak farklı renkli LED'leri ve buzzer'ı kontrol eder.
- **Mesafe 10 cm'den küçükse**:
  - Kırmızı LED (`LEDS["RED"]`) yanar.
  - Buzzer sürekli öter.
- **Mesafe 10 cm ile 20 cm arasındaysa**:
  - Sarı LED (`LEDS["YELLOW"]`) yanar.
  - **Buzzer kısa aralıklarla öter**:
    - 0.1 saniye boyunca aktif hale gelir.
    - Ardından kapatılır.
- **Mesafe 20 cm ile 50 cm arasındaysa**:
  - Yeşil LED (`LEDS["GREEN"]`) yanar.
  - **Buzzer daha uzun aralıklarla öter**:
    - 0.5 saniye boyunca aktif hale gelir.
    - Ardından kapatılır.
- **Mesafe 50 cm'den büyükse**:
  - Buzzer tamamen kapatılır.
  - LED kontrolü yapılmamış; mevcut LED'ler kapatılmamışsa yanmaya devam eder.

Ana Döngü

```
try:
    while True:
        distance = measure_distance()
        print(f"Mesafe: {distance} cm")
```

```
        control_feedback(distance)
except KeyboardInterrupt:
    print("Çıkış yapılıyor...")
    GPIO.cleanup()
```

- **Sonsuz Döngü**: Sürekli olarak mesafe ölçümü yapılır ve duruma göre LED'ler ile buzzer kontrol edilir.
- **Klavye Kesintisi (`KeyboardInterrupt`)**: Kullanıcı programı durdurduğunda, `GPIO.cleanup()` fonksiyonu çalıştırılarak pinler temizlenir ve güvenli bir şekilde çıkılır.

Genel Amaç:

Bu kod, bir ultrasonik sensör yardımıyla mesafeyi ölçer ve belirli mesafe aralıklarında görsel ve işitsel geri bildirim sağlar. Örneğin:

- **Kırmızı LED ve sürekli buzzer**: Tehlikeli derecede yakın.
- **Sarı LED ve kısa aralıklarla buzzer**: Yakın ama tehlikeli değil.
- **Yeşil LED ve uzun aralıklarla buzzer**: Güvenli mesafe.

Beklenen Sonuçlar

Bu proje kapsamında, Raspberry Pi ve ultrasonik sensörler kullanılarak geliştirilen araç park sensörünün, belirli bir mesafeden itibaren engelleri algılayarak görsel ve sesli uyarılar vermesi beklenmektedir. Sensörlerin mesafe ölçümleri doğru ve kararlı bir şekilde çalışmalı, aktif buzzer aracılığıyla araç sürücüsüne etkili bir şekilde sesli uyarılar iletilebilmelidir.

- **Doğru Mesafe Tespiti:** Sensörlerin, belirlenen 3 aşamada doğru mesafe ölçümü yapması.
- **Anlık Tepki:** Algılanan mesafelere bağlı olarak hızlı ve doğru bir şekilde buzzer ve LED'lerin tetiklenmesi.
- **Kullanıcı Dostu Çalışma:** Sistemin düşük maliyetli, kurulumu kolay ve pratik bir şekilde çalışması beklenmektedir.

Alternatif Teknolojiler ve Seçilen Teknolojilerin Tercih Sebepleri

Projemizin geliştirilmesinde kullanılan teknolojiler, bütçe, erişilebilirlik, öğrenim potansiyeli ve uygulama kolaylığı gibi çeşitli kriterlere dayanılarak belirlenmiştir. Ancak, bu teknolojilere alternatif olarak kullanılabilecek diğer seçenekler de bulunmaktadır. Aşağıda, alternatif teknolojiler ve seçilen teknolojilerin neden tercih edildiği açıklanmıştır:

1. Ultrasonik Sensörler yerine Lidar Sensörleri

- **Alternatif:** Lidar (Light Detection and Ranging) sensörleri, ultrasonik sensörlere kıyasla daha hassas ve geniş bir algılama kapasitesine sahiptir. Lidar, ışık dalgaları kullanarak çevredeki nesnelerin yüksek çözünürlüklü bir haritasını oluşturabilir.
- **Neden Ultrasonik Sensörler Seçildi?**
  - **Maliyet:** Ultrasonik sensörler, Lidar'a kıyasla çok daha düşük maliyetlidir.
  - **Prototip Basitliği:** Lidar, karmaşık veri işleme gerektirirken ultrasonik sensörler, basit ve doğrudan veri aktarımı sağlar.
  - **Uygulama Uygunluğu:** Proje kapsamında araç park sensörü gibi kısa mesafeli uygulamalar için ultrasonik sensörlerin doğruluğu yeterlidir.

2. Aktif Buzzer yerine Hoparlör

- **Alternatif:** Hoparlörler, kullanıcıya daha çeşitli ve özelleştirilebilir sesli uyarılar sağlayabilir.
- **Neden Aktif Buzzer Seçildi?**
  - **Basitlik:** Buzzer, düşük donanım gereksinimleriyle çalışabilir ve tek bir dijital sinyalle kolayca kontrol edilebilir.
  - **Maliyet:** Hoparlöre kıyasla daha ucuzdur ve karmaşık bir ses sürücüsü gerektirmez.
  - **Proje İhtiyacına Uygunluk:** Araç park sensörü gibi uygulamalarda basit sesli uyarılar yeterlidir.

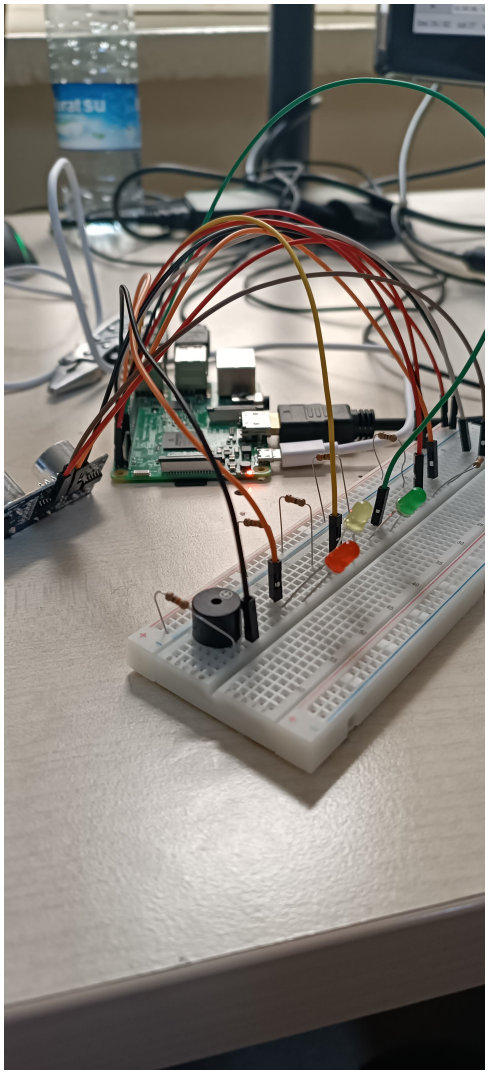
3. LED’ler yerine LCD veya Ekran Kullanımı

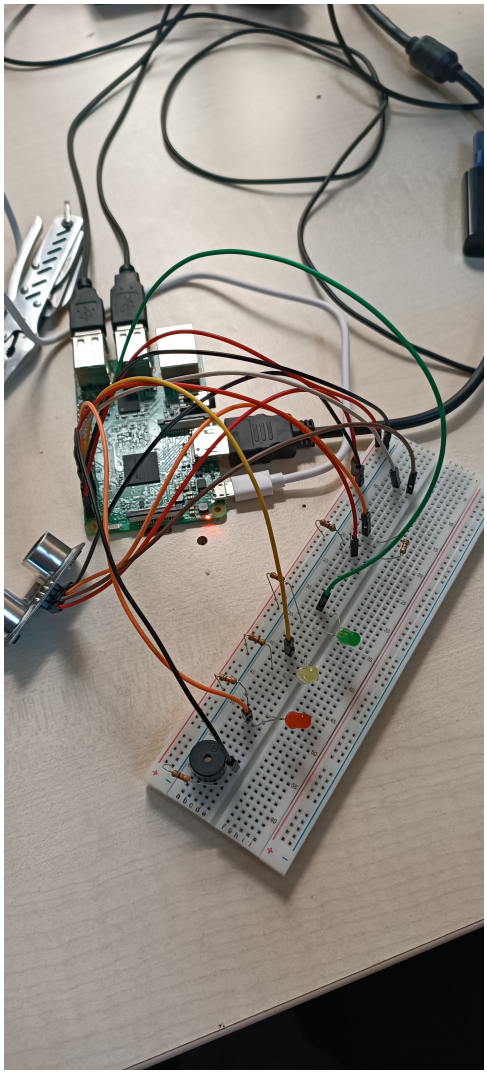
- **Alternatif:** LCD veya diğer ekran türleri, mesafe bilgisini dijital olarak göstererek daha detaylı bilgi sağlayabilir.
- **Neden LED'ler Seçildi?**
  - **Basit ve Hızlı Çıkış:** LED'ler, mesafe bilgisini sürücülere hızlı bir şekilde iletmek için görsel bir araç sağlar.
  - **Maliyet ve Güç Tüketimi:** LED'ler, düşük maliyetli ve enerji verimli bir çözümdür.
  - **Prototip Basitliği:** Daha karmaşık bir ekran yerine LED'lerle görsel uyarılar vermek, sistemin sadeliğini korur.

4. Raspberry Pi yerine Arduino

- **Alternatif:** Arduino, basit mikrodenetleyici projelerinde yaygın olarak kullanılan bir platformdur. Özellikle sadece sensör verisi işlemek ve çıkış sağlamak gibi işlemler için uygun olabilir.
- **Neden Raspberry Pi Seçildi?**
  - **Çoklu İşlem Yeteneği:** Raspberry Pi, bir işletim sistemi (Raspberry Pi OS) çalıştırabilir ve birden fazla görevi eş zamanlı olarak gerçekleştirebilir. Bu, yazılım geliştirme sürecinde esneklik sağlar.
  - **Geniş Kütüphane Desteği:** Python gibi yüksek seviyeli programlama dillerini destekler ve yazılım geliştirme için geniş bir topluluk desteği sunar.
  - **Görsel Çıkış İmkanı:** Raspberry Pi, HDMI çıkışı ile sistemin çalışmasını görselleştirmek veya daha karmaşık kullanıcı arayüzleri oluşturmak için kullanılabilir.

Proje Entegrasyonundan Görüntüler





Kullanılan Kaynaklar

Raspberry Pi Official : https://www.raspberrypi.com

Youtube/Core Electronics : https://www.youtube.com/@Core-Electronics

ChatGPT : https://chatgpt.com/

Youtube/Robotistan : https://www.youtube.com/@robotistan

Patika/Phyton : https://academy.patika.dev/courses/python-temel