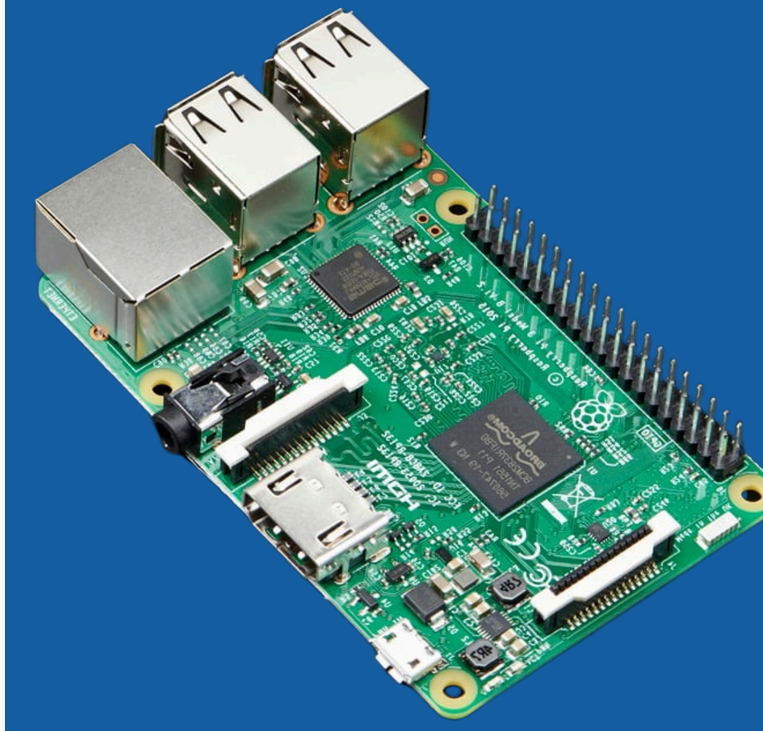
Raspberry Pi

ARAÇ PARK SENSÖRÜ

MIKROIŞLEMCILER

İçindekiler



Projenin Tanıtımı:

Bu proje, HC-SR04 ultrasonik sensör kullanılarak park sensörü sisteminin geliştirilmesini amaçlamaktadır. Proje, Raspberry Pi mikrodenetleyicisi ile sensör verilerini işleyerek araçların park ederken çevredeki engelleri algılamasını ve sürücüyü uyararak güvenli bir park süreci sağlamayı hedeflemektedir.

Amacı:

Projenin amacı, park alanlarında sürücülere rehberlik ederek:

- Araç hasarlarını önlemek,
- Dar alanlarda park işlemini kolaylaştırmak,
- Park esnasında çevresel güvenliği artırmak gibi faydalar sağlamaktır.

Projenin Kapsamı ve Özeti

PROJE KAPSAMI:

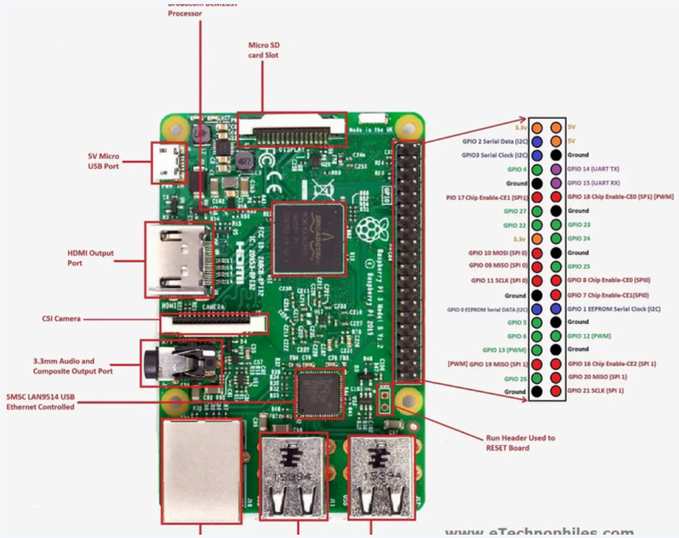
Projenin kapsamı, araçlar için akıllı park sensörü sisteminin geliştirilmesini içerir. Bu sistem, araç çevresindeki engelleri algılamak, sürücüyü görsel ve işitsel olarak uyarmak ve park etme sürecini kolaylaştırmak amacıyla tasarlanmıştır. Proje kapsamında Raspberry Pi 3 mikroişlemcisi, ultrasonik sensörler, LED ışıklar, ve buzzer. Sistem, Python programlama diliyle geliştirilmiş bir yazılım modeli üzerinden çalışacak, mesafe ölçümlerini yaparak kullanıcıya anlık olarak bilgi sağlayacaktır. Ayrıca, sistemin tasarımı, farklı park senaryolarına uygun şekilde çalışabilecek hassasiyette olacaktır.

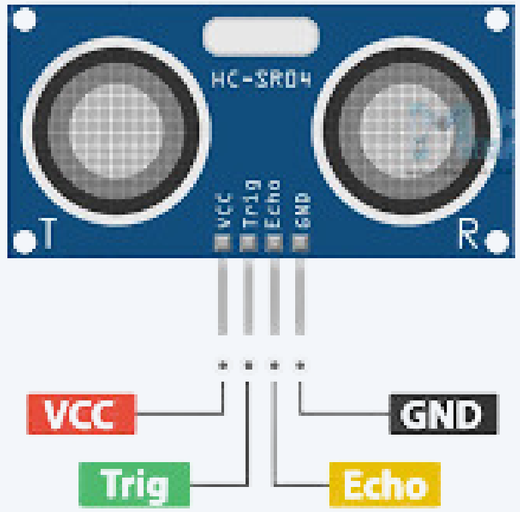
Özet:

“Bu proje, araç park etme sırasında mesafe algılama ihtiyacını karşılamak için geliştirilmiştir. HC-SR04 sensöründen alınan veriler, Raspberry Pi ile işlenerek mesafe bilgisi görsel veya sesli bir uyarı şeklinde sürücüye iletilmektedir. Bu sistem, hem bireysel araç sahipleri hem de ticari araç filoları için etkili bir çözüm sunmayı amaçlamaktadır.”

DONANIM Bileşenleri

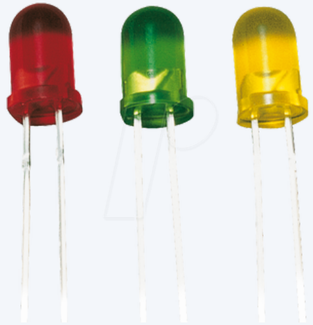
- **Raspberry Pi:** Ana mikroişlemci olarak kullanılacak ve sistemin beyin işlevini yerine getirecektir.
- **Ultrasonik Mesafe Sensörleri (HC-SR04):** Engelleri algılamak için kullanılacak sensörlerdir. Bu sensörler, mesafe ölçümü yaparak çevredeki nesneleri tespit eder.



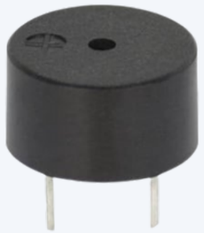


- **Buzzer (Sesli Uyarı):** Engeller yakın olduğunda sürücüyü sesli olarak uyaracak donanım bileşeni.
- **LED Işıklar (Görsel Uyarı):** Park sensörünün durumuna göre yeşil, sarı ve kırmızı LED'lerle görsel uyarılar yapılacaktır.

DONANIM Bileşenleri

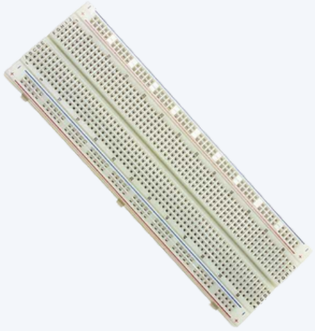


LED’ler



Buzzer

- **Güç Kaynağı:** Raspberry Pi için 5V adaptör.
- **Devre Tahtası:** Sensör ve LED/Buzzer gibi bileşenlerin kolayca bağlanmasını sağlar.
- **SD kartı:** Raspberry Pi işletim sistemi ve proje dosyalarını depolamak için.



Devre Tahtası

SD kartı

Adaptör VE micro kablosu

DONANIM Bileşenleri

- **Jumper kablosu:** Donanım bileşenlerini Raspberry Pi’nin GPIO pinlerine bağlamak için.
- **Wifi:** İletişim sağlamak için.
- **Klavye ve Fare:** İletişim sağlamak için.
- **Dirençler (330Ω ve 10kΩ):**

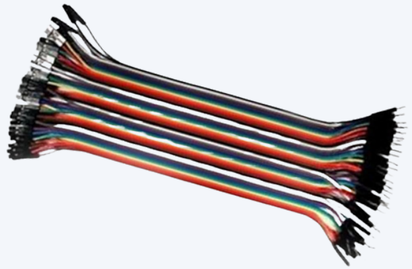
LED'ler için uygun akım sınırlamasını sağlar. **(330Ω)**

Ultrasonik sensör sinyallerinin düzenlenmesinde de kullanılır. **(10kΩ)**

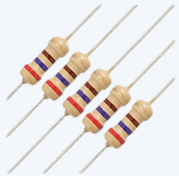
Klavye ve Fare: Raspberry Pi’yi kurulum sırasında yapılandırmak için.

Klavye ve Fare

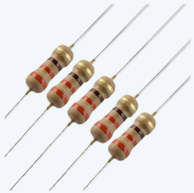
Wifi



Jumper kablosu



(10kΩ)



(330Ω)

Yazılımın Genel Yapısı

1. **Python Programlama Dili:** Raspberry Pi'de GPIO pinlerini kontrol etmek için kullanılır.
   - **Python:** Raspberry Pi üzerinde çalışan yazılım dilidir. Sensörlerden veri okumak, işlemek ve çıktıları sürücüye iletmek için Python kullanılacaktır.
   - **Raspberry Pi OS (Raspberry Pi işletim sistemi):** Projenin çalışacağı ana işletim sistemi.

2. **Kütüphaneler:**

RPi.GPIO: GPIO pinlerini kontrol etmek için.

time: Gecikme süreleri ve zaman ölçümleri için.

3. **Çalışma Mantığı:**

Ultrasonik sensör **(HC-SR04)** mesafe ölçümü yapar.

Ölçülen mesafeye göre **LED veya buzzer** ile sürücüye geri bildirim sağlar.

Eğer mesafe kritik bir değerin altına düşerse uyarı sesleri sıklaşır.

Kod Detayları ve Açıklamaları

1. Pin Tanımları

TRIG ve ECHO Pinleri: HC-SR04 ultrasonik sensörün giriş ve çıkış pinleri.

LED'ler (Yeşil, Sarı, Kırmızı): Mesafeye göre yanacak LED'ler.

BUZZER: Kritik mesafelerde sesli uyarı için kullanılan pin.

Kod Detayları ve Açıklamaları

2. PWM Kullanımı

PWM (Pulse Width Modulation) ile LED parlaklığını ve buzzer ses seviyesini kontrol ediyorsunuz. PWM sayesinde: LED'ler tam parlaklık veya düşük parlaklıkta çalışabilir. Buzzer sesi farklı yoğunluklarda çalınabilir.

3. Mesafe Ölçümü

TRIG pinine kısa bir tetikleme sinyali gönderilir.

ECHO pininden geri dönen sinyalin süresine göre mesafe hesaplanır.

Hesaplama: distance = (signal_time * 34300) / 2

4. LED ve Buzzer Kontrolü

Mesafe aralıklarına göre farklı eylemler gerçekleştiriliyor:

< 10 cm: Kırmızı LED yanar, buzzer çalar.

10–25 cm: Sarı LED yanar.

25–100 cm: Yeşil LED yanar.

> 100 cm: Tüm LED'ler ve buzzer kapatılır.

5. Programı Sonlandırma

Kullanıcı, klavyeden kesme **(Ctrl+C)** ile programı durdurabilir. Ayrıca, mesafe 100 cm'yi aştığında program kendiliğinden duruyor.

Yazılım Kodları

Python ile yazılan temel mesafe ölçüm kodu şu şekildedir:

```
import RPi.GPIO as GPIO
import time

# Uyarıları devre dışı bırak ve GPIO modunu BCM olarak ayarla
# BCM: GPIO pinlerini pin numaralarına göre kullanmayı sağlar.
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

# Pinlerin tanımlanması (TRIG, ECHO sensör için; LED'ler ve buzzer kontrol için)
TRIG = 4
ECHO = 18
GREEN = 17
YELLOW = 27
RED = 22
BUZZER = 23 # Buzzer için pin tanımlaması
```

Yazılım Kodları

```
# Giriş/Çıkış pinleri ayarlanıyor
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)
GPIO.setup(GREEN, GPIO.OUT)
GPIO.setup(YELLOW, GPIO.OUT)
GPIO.setup(RED, GPIO.OUT)
GPIO.setup(BUZZER, GPIO.OUT) # Buzzer pini çıkış olarak ayarlandı

# LED'ler ve buzzer için PWM (Pulse Width Modulation) başlatılıyor
# PWM, LED parlaklığı ve buzzer ses frekansını kontrol etmek için kullanılıyor.
pwm_green = GPIO.PWM(GREEN, 1000) # Yeşil LED için PWM, 1kHz
pwm_yellow = GPIO.PWM(YELLOW, 1000) # Sarı LED için PWM
pwm_red = GPIO.PWM(RED, 1000) # Kırmızı LED için PWM
pwm_buzzer = GPIO.PWM(BUZZER, 1000) # Buzzer için PWM

# PWM başlangıcı, %0 parlaklık/ses (kapalı)
pwm_green.start(0)
```

Yazılım Kodları

```
# PWM başlangıcı, %0 parlaklık/ses (kapalı)
pwm_green.start(0)
pwm_yellow.start(0)
pwm_red.start(0)
pwm_buzzer.start(0)
# Yeşil LED'i açan fonksiyon
def green_light():
 pwm_green.ChangeDutyCycle(100) # Yeşil LED tam parlaklıkta
 pwm_yellow.ChangeDutyCycle(0) # Sarı LED kapalı
 pwm_red.ChangeDutyCycle(0) # Kırmızı LED kapalı
 pwm_buzzer.ChangeDutyCycle(0) # Buzzer kapalı
# Sarı LED'i açan fonksiyon
def yellow_light():
 pwm_green.ChangeDutyCycle(0) # Yeşil LED kapalı
 pwm_yellow.ChangeDutyCycle(100) # Sarı LED tam parlaklıkta
 pwm_red.ChangeDutyCycle(0) # Kırmızı LED kapalı
 pwm_buzzer.ChangeDutyCycle(0) # Buzzer kapalı
```

Yazılım Kodları

```
# Kırmızı LED'i açan fonksiyon
def red_light():
 pwm_green.ChangeDutyCycle(0) # Yeşil LED kapalı
 pwm_yellow.ChangeDutyCycle(0) # Sarı LED kapalı
 pwm_red.ChangeDutyCycle(100) # Kırmızı LED tam parlaklıkta
 pwm_buzzer.ChangeDutyCycle(50) # Buzzer %50 yoğunlukta çalar (orta seviye ses)

# Tüm LED'leri ve buzzer'ı kapatan fonksiyon
def turn_off_all():
 pwm_green.ChangeDutyCycle(0) # Yeşil LED kapalı
 pwm_yellow.ChangeDutyCycle(0) # Sarı LED kapalı
 pwm_red.ChangeDutyCycle(0) # Kırmızı LED kapalı
 pwm_buzzer.ChangeDutyCycle(0) # Buzzer kapalı
# Ultrasonik sensörle mesafe ölçümü yapan fonksiyon
def get_distance():
 # TRIG pinine kısa bir tetik sinyali gönder
 GPIO.output(TRIG, GPIO.HIGH) time.sleep(0.00001) # 10 mikro saniye tetik sinyali
 GPIO.output(TRIG, GPIO.LOW)
```

Yazılım Kodları

```
# ECHO pinindeki sinyalin gidiş-dönüş süresini ölç
 while GPIO.input(ECHO) == GPIO.LOW:
 start = time.time() # Sinyal başlangıç zamanı
 while GPIO.input(ECHO) == GPIO.HIGH:
 end = time.time() # Sinyal bitiş zamanı

 # Sinyal süresine göre mesafeyi hesapla (cm)
 signal_time = end - start
 distance = (signal_time * 34300) / 2 # Mesafe = zaman * ses hızı / 2
 return distance

# Ana döngü (program burada sürekli çalışır)
try:
 while True:
 distance = get_distance() # Mesafeyi ölç
 print("Distance: %.1f cm" % distance) # Ölçülen mesafeyi ekrana yazdır
```

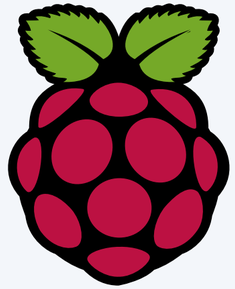
Yazılım Kodları

```
 # Mesafeye göre LED'leri ve buzzer'ı kontrol et
 if distance < 10:
 red_light() # 10 cm'den az mesafe: Kırmızı LED ve buzzer aktif
 elif 10 <= distance < 25:
 yellow_light() # 10-25 cm arası: Sarı LED aktif
 elif 25 <= distance <= 100:
 green_light() # 25-100 cm arası: Yeşil LED aktif
 else:
 turn_off_all() # 100 cm'den büyük mesafede: Tüm uyarılar kapalı
 # Eğer mesafe 100 cm'yi geçerse programdan çık
 if distance > 100:
 print("Distance exceeds limit. Stopping the program.")
 break # Döngüden çık
 time.sleep(0.05) # Ölçümler arasında 50ms bekle
# Kullanıcı programı manuel durdurursa (Ctrl+C)
except KeyboardInterrupt:
 print("Program stopped by user.")
```

Yazılım Kodları

```
# Program sonlandığında kaynakları serbest bırak
finally:
 pwm_green.stop() # Yeşil LED PWM durdur
 pwm_yellow.stop() # Sarı LED PWM durdur
 pwm_red.stop() # Kırmızı LED PWM durdur
 pwm_buzzer.stop() # Buzzer PWM durdur
 GPIO.cleanup() # GPIO pinlerini temizle
```

Programlama dili
(Python)



Raspberry Pi OS

Mikroişlemci Yazılım Modeli ve Donanım ile Etkileşim

1. Yazılım Modeli

- **Raspberry Pi Üzerindeki Yazılım:**

Python programlama dili kullanılarak HC-SR04 ultrasonik sensörün verileri işlenmiştir.

Yazılım, sensörden alınan tetikleme (trigger) ve yansıma (echo) sinyallerini ölçerek mesafeyi hesaplar.

- **Sensör Verisi Okuma:** HC-SR04 sensöründen mesafe verisi alınır.

- **Mesafe Hesaplama:**

HC-SR04 sensörü, bir ultrasonik dalga gönderir ve bu dalganın engelden geri dönme süresini ölçer.

Mesafe şu formül ile hesaplanır:

distance = (signal_time * 34300) / 2 # Mesafe = zaman * ses hızı / 2

- **signal_time (sinyal zamanı):** Sinyalin bir noktadan yayılıp geri dönmesi için geçen süreyi temsil eder. Yani, sesin yayılması ve geri dönmesi arasındaki toplam süredir.
- **34300:** Bu sayı, sesin havada 1 saniyede kat ettiği mesafeyi temsil eder. 343 m/s, sesin hava ortamındaki hızıdır (yaklaşık olarak 20°C'deki havada).
- **/ 2:** Ses sinyali bir noktaya gider ve sonra geri döner. Yani toplam yol iki katı olduğundan, mesafeyi doğru hesaplamak için bu sonucu 2'ye böleriz.

- **Uyarı Sistemi:**

Mesafe belirli bir eşik değerin altına düştüğünde (örneğin 20 cm), sürücüye sesli veya görsel bir uyarı gönderilir.

Mikroişlemci Yazılım Modeli ve Donanım ile Etkileşim

2. Donanım ile Etkileşim

- **HC-SR04 Sensörünün Çalışması:**

Raspberry Pi’nin GPIO pinleri üzerinden sensör ile iletişim kurulur.

Trigger pini bir ultrasonik dalga gönderir.

Echo pini dalganın geri dönme süresini Raspberry Pi’ye bildirir.

- **Donanım ve Yazılım Bağlantısı:**

GPIO Pinleri:

Sensörün tetikleme ve okuma işlemleri için GPIO pinleri atanmıştır.

- Örneğin:

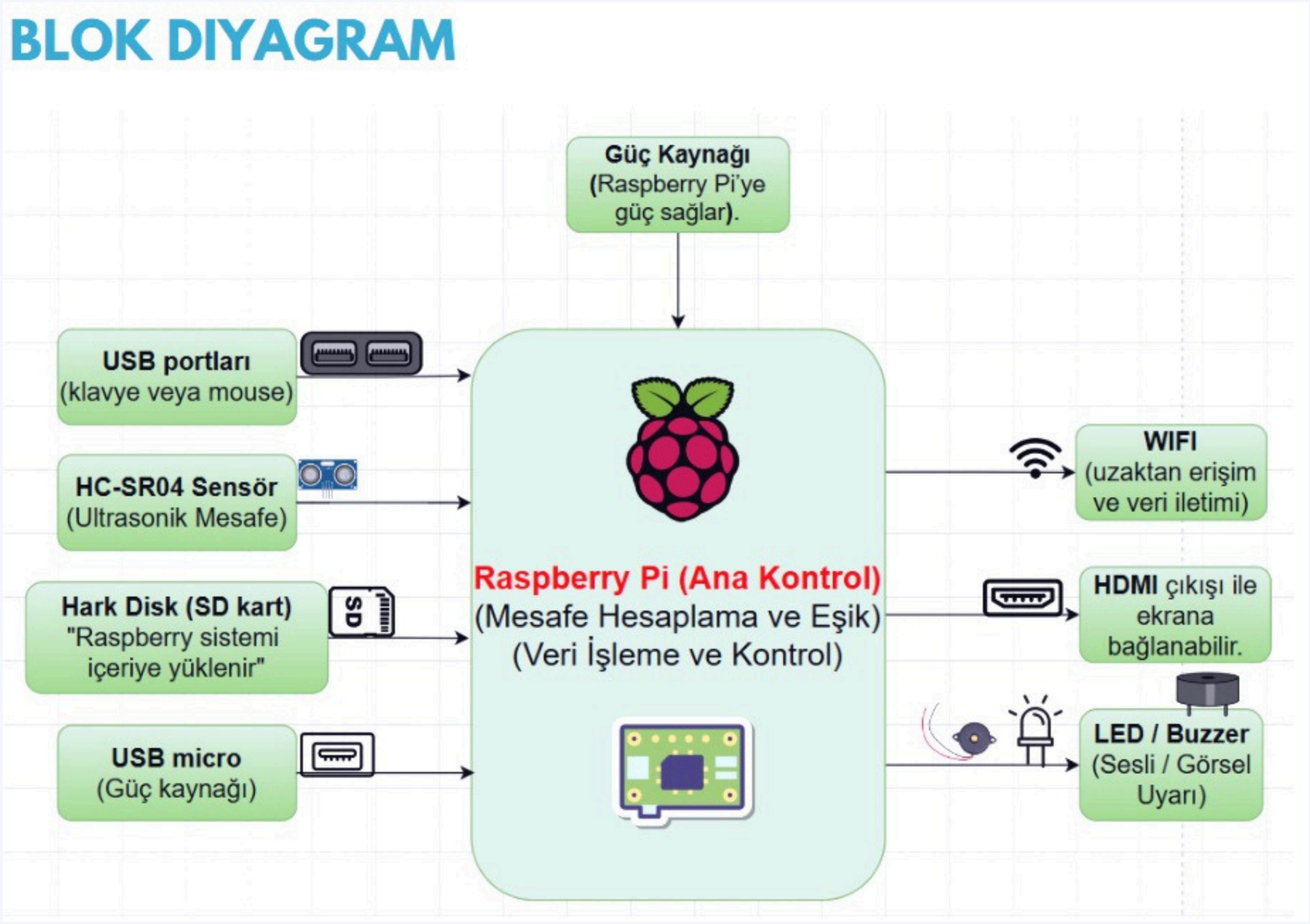
Trigger pini: GPIO 23

Echo pini: GPIO 24

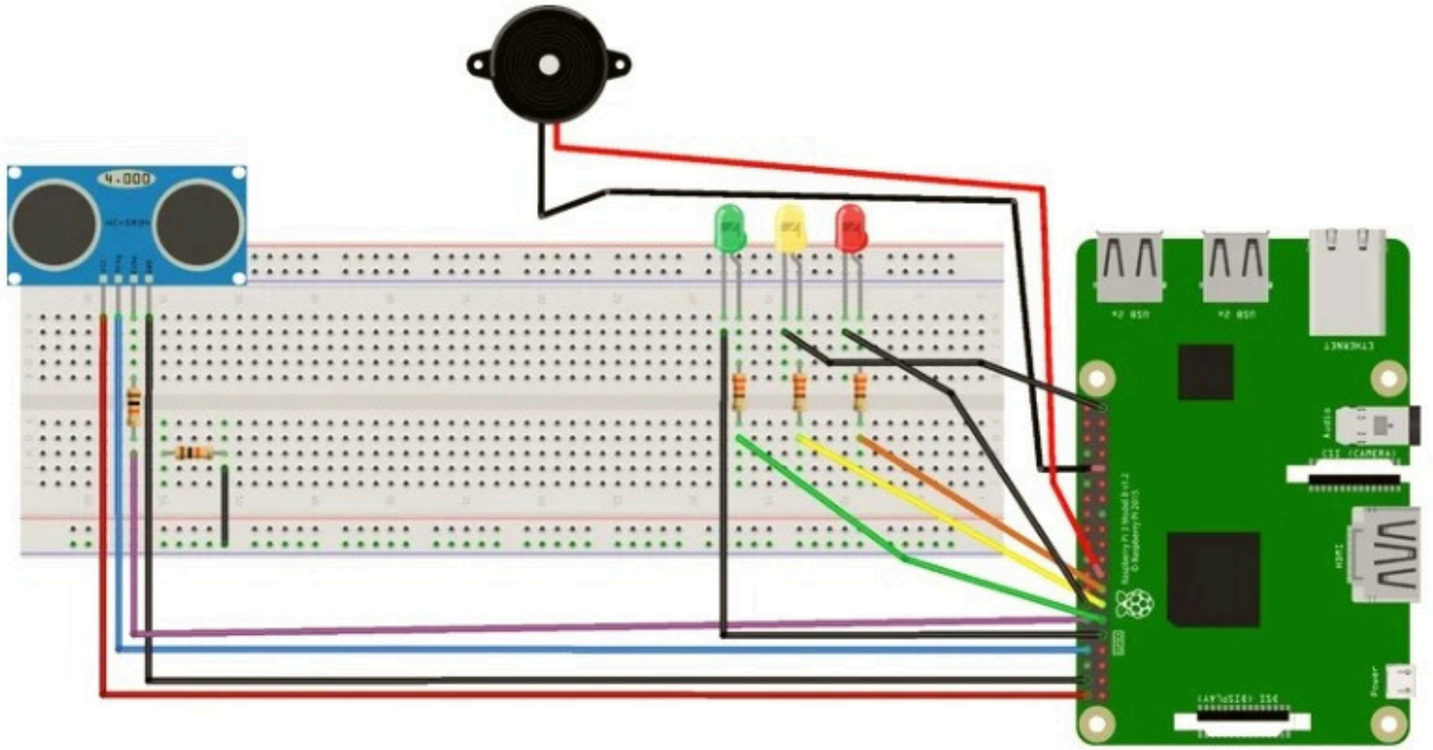
- **Gerçek Zamanlı İşlem:**

Raspberry Pi, bu pinlerden gelen veriyi sürekli okur ve mesafeyi gerçek zamanlı olarak hesaplar.

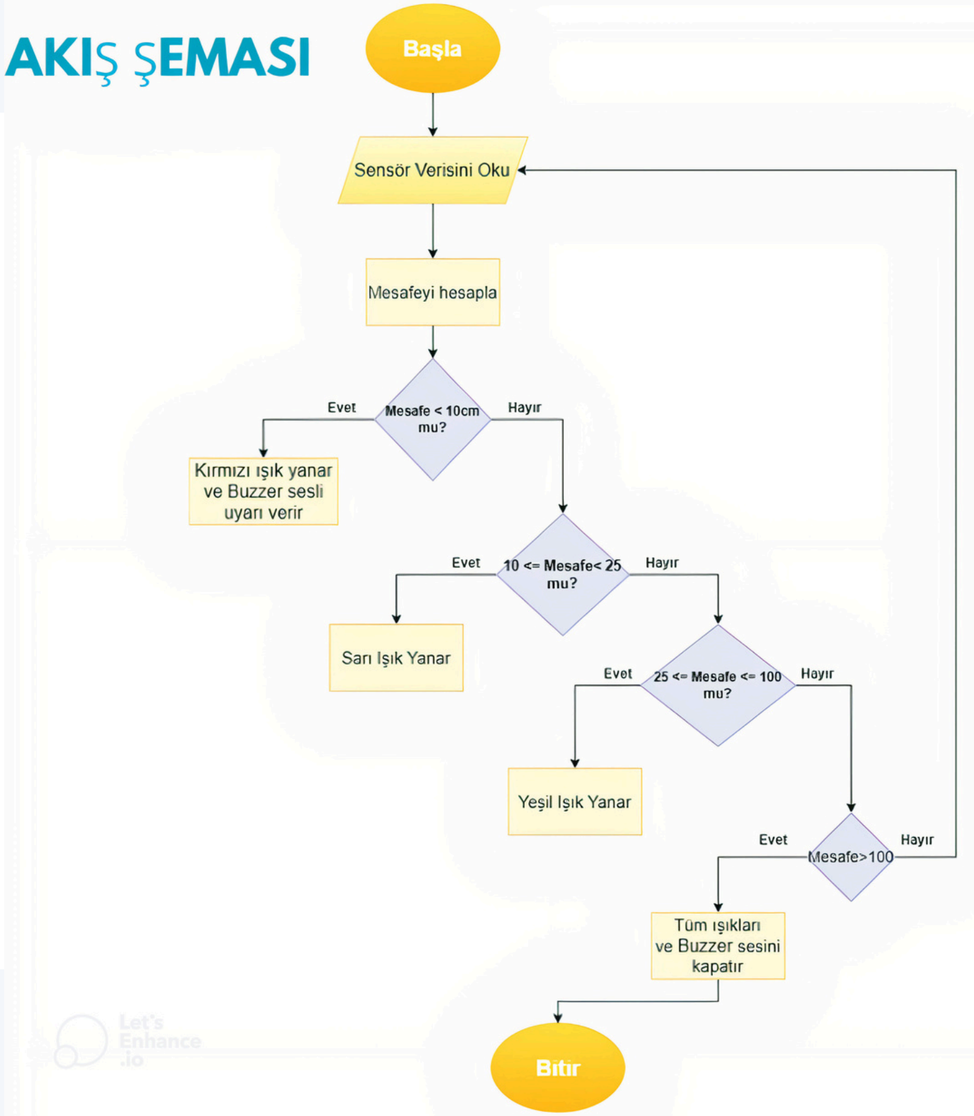
BLOK DIYAGRAM



DEVRE ŞEMASI



AKIŞ ŞEMASI



Karşılaşılan Zorluklar ve Çözümler

Zorluk: Ultrasonik sensörlerin kararlı sonuç vermemesi.

Çözüm: Sinyal işleme algoritmalarını optimize etmek.

Zorluk: Raspberry Pi'nin GPIO pinlerinde bağlantı sorunları.

Çözüm: Pin yerleşimini yeniden düzenleyip temas kontrolü yapmak.

Zorluk: Yazılımda gecikmelerin önlenmesi.

Çözüm: Kod optimizasyonu ve algoritma geliştirme.

Zorluk: Donanım bağlantı sorunları.

Çözüm: Jumper kablolarının sağlam şekilde bağlanması.

Zorluk: Enerji tüketimi.

Çözüm: Yüksek kaliteli güç adaptörü kullanılması.

Zorluk: Kablolama karmaşası.

Çözüm: Kabloların etiketlenmesi ve düzenli yerleştirilmesi.

Zorluk: Mesafe Ölçüm Doğruluğu: Ultrasonik sensörlerin doğru mesafe ölçmemesi bazen sorun olabilir.

Çözüm: Sensörlerin hizalanmasına dikkat edilmesi ve sensörlerin düzgün çalışabilmesi için ortamın uygunluğunun kontrol edilmesi gerekir.

Zorluk: Fiziksel Kurulum Problemleri: Sensörlerin montajı, doğru yerleştirilmesi ve düzgün çalışması zorlu olabilir.

Çözüm: Sensörlerin doğru şekilde yerleştirilmesi ve test edilmesi gerekmektedir.

Projenin Sonuçları

Bu araç park sensörü projesi, Raspberry Pi ve ultrasonik sensörler kullanarak araçların park işlemini daha güvenli ve kullanıcı dostu hale getirmeyi hedeflemiştir. Proje başarıyla tamamlandığında elde edilen sonuçlar şunlardır:

1. Hedeflenen Fonksiyonun Gerçekleştirilmesi

Engel Tespiti: Ultrasonik sensörler sayesinde araç, arkasındaki engelleri doğru bir şekilde algılar. Bu sensörler, park halindeki bir araç için yeterli mesafe sağlanıp sağlanmadığını belirler. Yakın mesafe engelleri (örneğin, duvar veya başka bir araç) algılandığında sistem doğru şekilde uyarı verir.

Uyarı Sistemi: Park yaparken sürücüyü uyaran sesli ve görsel bir sistem kurulmuştur. Engel yaklaştıkça buzzer sesi şiddetini artırır ve LED ışıkları (ya da ekran) aracın durumu hakkında bilgi verir. Örneğin, çok yakın bir mesafe algılandığında, buzzer sürekli çalar ve kırmızı LED yanar. Bu, sürücüyü büyük bir engelin varlığı hakkında net bir şekilde bilgilendirir.

Hızlı Tepki Süresi: Sistemin tepki süresi oldukça hızlıdır, yani engellerin tespiti ve uyarı verilmesi arasındaki süre çok kısadır. Bu, kullanıcıların park işlemi sırasında anlık kararlar almasına olanak tanır.

2. Kullanıcı Dostu Bir Sistem

Kolay Kullanım: Proje, karmaşık bir kullanıcı arayüzü gerektirmez. Buzzer ve LED'lerin uyumlu çalışması, sürücünün kolayca ne zaman engelin yakın olduğunu anlamasına yardımcı olur. Sistemin sade tasarımı, kullanıcıya herhangi bir ek çaba harcatmadan araç park etme sürecini daha güvenli hale getirir.Düşük Maliyet: Raspberry Pi ve ultrasonik sensörler kullanarak, araç park sensörlerinin maliyeti

Projenin Sonuçları

Düşük Maliyet: Raspberry Pi ve ultrasonik sensörler kullanarak, araç park sensörlerinin maliyeti oldukça düşük tutulmuştur. Bu, özellikle küçük işletmeler veya bireysel kullanıcılar için daha erişilebilir bir çözüm sunar.

3. Donanım ve Yazılımın Verimli Etkileşimi

Donanımın Doğru Çalışması: Raspberry Pi ile ultrasonik sensörler arasında başarılı bir entegrasyon sağlanmıştır. Raspberry Pi, sensörlerden aldığı veriyi doğru şekilde işler ve buna göre uyarı sistemini devreye sokar. Yazılımın donanım ile olan etkileşimi, verinin gerçek zamanlı olarak alınmasını ve işlenmesini sağlar.

Yazılımın İstikrarlı Performansı: Python programı, sürekli olarak sensör verilerini okumakta ve sistemin gerektiği şekilde çalışmasını sağlamaktadır. Yazılımda yapılan optimizasyonlar sayesinde işlemci yükü minimum seviyeye indirilmiştir.

Sonuçların Değerlendirilmesi

Bu proje, araç park sensörü sisteminin etkinliğini ve verimliliğini kanıtlamaktadır. Raspberry Pi kullanılarak geliştirilen bu sistem, düşük maliyetle güvenli park etme sağlama amacı güdüyor ve bu hedefe ulaşmıştır. Geliştirme alanlarına odaklanarak daha kapsamlı ve yüksek performanslı bir sistem oluşturulabilir. Bu sonuçlar, araç park etme konusunda daha akıllı ve kullanıcı dostu bir çözümün mümkün olduğunu göstermektedir.

Olası Geliştirme Alanları

1. Sensör Hassasiyetinin Artırılması:

Daha yüksek hassasiyetli ultrasonik sensörler veya **LiDAR teknolojisi** entegre edilebilir.

2. Görsel ve Sesli Uyarıların İyileştirilmesi:

- Ekranlı bir gösterge paneli eklenerek mesafe bilgisi görsel olarak daha ayrıntılı sunulabilir.
- Uyarı ses tonları farklı mesafe aralıkları için özelleştirilebilir.

3. Kablosuz Bağlantı Özellikleri:

- Sisteme Wi-Fi veya Bluetooth eklenerek mobil uygulama entegrasyonu sağlanabilir.
- Kullanıcılar, park sensöründen gelen uyarıları telefonlarından takip edebilir.

4. Çoklu Sensör Kullanımı:

- Birden fazla HC-SR04 sensörü kullanılarak 360 derece algılama yapılabilir.
- Bu, araçların sadece arka tarafını değil, yan ve ön taraflarını da kapsayan bir güvenlik sistemi sunar.

5. Dış Ortam Koşullarına Dayanıklılık:

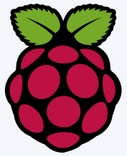
- Sensörler ve sistem, su geçirmezlik ve sıcaklık dayanıklılığı gibi özelliklerle açık hava koşullarına uygun hale getirilebilir.

6. Yapay Zeka Entegrasyonu:

Sistemin, engellerin türünü tanıyabilmesi için görüntü işleme veya yapay zeka teknikleri kullanılabilir.

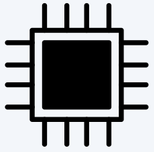
7. Gelişmiş Görsel Ekranlar:

Daha fazla bilgi sağlayan dokunmatik ekranlar ya da grafiksel arayüzler eklenebilir.



Raspberry Pi

TEŞEKKÜRLER



MIKROIŞLEMCILER