Nevşehir Hacı Bektaş Veli Üniversitesi

Bilgisayar Mühendisliği Bölümü

Bilgisayar Tasarım ve Uygulamaları Derisi

GENEL AMAÇLI TAŞIYICI ROBOT

İlayda Buse ÖZTÜRK

20260810027

Uygulama

Genel Amaçlı Taşıyıcı Robot

Aracın Tasarımı

Robot, aşağıdaki temel bileşenlerden oluşur:

Gövde: Robotun ana gövdesi, plastikten yapılmıştır ve yükleri taşıyabilecek kadar sağlamdır.

Tekerlekler: Robot, hareket etmek için dört tekerleğe sahiptir. Tekerlekler, robotun farklı yüzeylerde hareket etmesini sağlar. Tank tipi tekerlek sistemine sahiptir.

Elektronikler: Robot, hareketini kontrol etmek için aşağıdaki elektronik bileşenleri kullanır:

- **Ultrasonik mesafe sensörleri:** Bu sensörler, robotun çevresini algılamak için kullanılır. Robot, sensörlerden gelen verileri kullanarak önündeki engelleri tespit eder ve onlardan kaçınabilir.
- **IR alıcı verici sensör modülü:** Bu sensör modülü, robotun arka tekerleklerinin dönüp dönmediğini algılayabilmek için kullanılır. Bu sayede mototrlar çalıştığı halde robotun takılması durumu tespit edilir.
- **NRF24L01 haberleşme modülü:** Bu modül, robotun kumanda ile iletişim kurmasını sağlar. Robot, bu modül sayesinde uzaktan kontrol edilebilir ve veri alışverişi yapabilir.
- **Redüktörlü Motorlar:** Redüktörlü motorlar, bir motorun torkunu artırarak daha yüksek bir yük taşıma kapasitesi sağlayan motorlardır. Bu, motorun dönüş hızını düşürerek ve dişli mekanizması kullanarak yapılır. Robotunuzda kullandığınız redüktörlü motorlar, 6V ve 250 RPM'dir. Bu, motorların 6V'luk bir güç kaynağı ile çalıştığını ve dakikada 250 devir yaptığını gösterir. Motorların dönüş hızı, robotun hareket hızını etkiler. L298N Çift Motor Sürücü Kartı, redüktörlü motorları kontrol etmek için kullanılan bir elektronik karttır. Bu kart, motorların dönüş hızını ve yönünü kontrol etmek için kullanılabilir.
- **Mikroişlemci (Arduino Uno):** Bu işlemci, robotun tüm elektronik bileşenlerini kontrol eder.
- **Güç kaynağı:** Robot, hareket etmek ve elektronikleri çalıştırmak için bir güç kaynağına ihtiyaç duyar. Robotumda, iki adet Aspilsan INR18650A28 2900 mAh 3.7V 25A Li-

ion Şarjlı Pil kullandım. Bu piller, 7.4V'luk bir voltajda toplam 5.8 Ah'lik bir kapasiteye sahiptir. Bu piller, lityum iyon teknolojisine sahiptir. Lityum iyon piller, yüksek enerji yoğunluğuna sahip oldukları için robotlar için ideal bir güç kaynağıdır.

Aracın Karşıladığı Gereksinimler

Yük taşıma kapasitesi: Robot, en fazla 1 kilogramlık yükleri taşıyabilir.

Hareket kabiliyeti: Robot, düz zeminlerde ve engebeli arazilerde hareket edebilir.

Kontrol: Robot, hareketini otomatik olarak kontrol edebilir aynı zamanda manuel kontrol de mümkündür.

Kumandanın Tasarımı

Gövde: Uzaktan kumanda gövdesi, kartondan yapılmıştır hafiftir.

Elektronikler: Uzaktan kumanda, aşağıdaki elektronik bileşenleri kullanır:

- **NRF24L01 haberleşme modülü:** Bu modül, uzaktan kumandanın robotla iletişim kurmasını sağlar.
- **Arduino Uno mikroişlemci:** Bu işlemci, uzaktan kumandanın tüm elektronik bileşenlerini kontrol eder.
- **16x2 Karakter LCD Ekran:** Bu ekran, uzaktan kumandadaki bilgileri gösterir.
- **LCD I2C Seri Arayüz Modülü:** Bu modül, LCD ekranı Arduino Uno'ya bağlamak için kullanılır.
- **Joystick:** Bu joystick, robotun hareketini kontrol etmek için kullanılır.

Kumandanın Karşıladığı Gereksinimler

Hareket kontrolü testi: Uzaktan kumanda kullanılarak robotun hareketi kontrol edilmiştir. Robot, uzaktan kumandadaki joystick kullanılarak sorunsuz bir şekilde hareket ettirilebilmiştir.

Mesafe kontrolü testi: Açık alanda 800 metreye kadar haberleşme sağlayabilmektedir

Bilgi görüntüleme testi: Uzaktan kumandadaki LCD ekran kullanılarak robot hakkında bilgiler görüntülenmiştir. Aracın hangi sürüş modunda olduğunu ve araçla kumanda arasındaki veri aktarımında kopukluk olup olmadığını görmek mümkündür.

Alternatif Kullanım Alanları

Robotlar, insan gücünün yerine geçebilecek, daha verimli ve daha güvenli bir şekilde çalışabilen araçlardır. Taşıyıcı robotlar ise, özellikle ağır ve hacimli yükleri taşımak için ideal araçlardır. Taşıyıcı robot yapmayı seçtim çünkü bu robotların, çeşitli endüstrilerde ve günlük yaşamda önemli bir rol oynayabileceğini düşünüyorum. Örneğin, taşımacılık, depolama ve hizmet sektörlerinde taşıyıcı robotlar kullanılarak işgücü verimliliği artırılabilir ve maliyetler düşürülebilir. Ayrıca, engelli veya yaşlı kişilerin günlük yaşamda daha bağımsız hareket etmesine yardımcı olmak için taşıyıcı robotlar kullanılabilir.

Taşıyıcı robotlar, aşağıdaki gibi çeşitli alanlarda kullanılabilecek potansiyele sahiptir:

Taşımacılık: Taşıyıcı robotlar, depolama alanları arasında yük taşımak, kargo taşımacılığı yapmak veya havaalanlarında bagaj taşımak için kullanılabilir. Örneğin, bir fabrikada üretilen ürünler, taşıyıcı robotlar kullanılarak depoya taşınabilir. Bu, işçilerin manuel olarak yük taşımasını gerektirmeyen, daha verimli ve daha güvenli bir süreçtir.

Depolama: Taşıyıcı robotlar, depolarda malzemeleri taşımak ve istiflemek için kullanılabilir. Örneğin, bir depodaki malzemeler, taşıyıcı robotlar kullanılarak raflara kaldırılabilir. Bu, işçilerin manuel olarak yük taşımasını gerektirmeyen, daha verimli ve daha güvenli bir süreçtir.

Hizmet sektörü: Taşıyıcı robotlar, restoranlarda yemek taşımak, hastanelerde hastaları taşımak veya mağazalarda ürünleri taşımak için kullanılabilir. Örneğin, bir restoranda yemekler, taşıyıcı robotlar kullanılarak müşterilere servis edilebilir. Bu, garsonların manuel olarak yemek taşımasını gerektirmeyen, daha verimli ve daha güvenli bir süreçtir.

Engelli ve yaşlı bakım: Taşıyıcı robotlar, engelli veya yaşlı kişilerin günlük yaşamda daha bağımsız hareket etmesine yardımcı olmak için kullanılabilir. Bu, engelli kişilerin günlük yaşamda daha aktif ve bağımsız olmasına yardımcı olur.

Taşıyıcı robotların potansiyelini daha da artırmak için aşağıdakiler yapılabilir:

Yük taşıma kapasitesi artırılabilir. Bu, robotların daha ağır ve hacimli yükleri taşımasını sağlayacaktır. Örneğin, 100 kilogramlık bir yük taşıma kapasitesine sahip bir taşıyıcı robot, bir fabrikada üretilen malzemeleri depoya taşımak için kullanılabilir.

Hareket kabiliyeti geliştirilebilir. Bu, robotların daha zorlu arazilerde ve engelli ortamlarda hareket etmesini sağlayacaktır. Örneğin, arazi koşullarına uyum sağlayabilen bir taşıyıcı robot, bir inşaat sahasında malzemeler taşımak için kullanılabilir.

Otomatik kontrol sistemi geliştirilebilir. Bu, robotların daha güvenli ve verimli bir şekilde çalışmasını sağlayacaktır. Örneğin, çevresini algılayabilen ve engelleri erken tespit edebilen bir taşıyıcı robot, daha güvenli bir şekilde hareket edecektir.

Bu öneriler, taşıyıcı robotların daha geniş bir yelpazede kullanılmasını ve daha verimli ve güvenli bir şekilde çalışmasını sağlayacaktır. Taşıyıcı robotların tasarımı ve yapımında kullanılan malzemeler ve teknolojiler, robotun kullanım alanına göre değişebilir. Örneğin, bir fabrikada kullanılacak bir taşıyıcı robotun, sağlam ve dayanıklı malzemelerden yapılması gerekir. Ayrıca, robotun hareket kabiliyetinin yüksek olması gerekir. Taşıyıcı robotların kontrol sistemi, robotun otomatik veya manuel olarak kontrol edilmesine olanak tanımalıdır. Otomatik kontrol sistemi, robotun çevresini algılamasını ve engelleri tespit etmesini sağlar. Manuel kontrol sistemi ise, kullanıcının robotu uzaktan kontrol etmesini sağlar. Taşıyıcı robotlar, insan gücünün yerine geçebilecek, daha verimli ve daha güvenli bir şekilde çalışabilen araçlardır. Bu robotların, çeşitli endüstrilerde ve günlük yaşamda önemli bir rol oynayabileceği düşünülmektedir.

Öneriler

Robotun performansını daha da iyileştirmek için aşağıdaki öneriler yapılabilir:

Daha hassas ultrasonik mesafe sensörleri kullanılabilir. Bu, aracın engelleri daha doğru bir şekilde tespit etmesini sağlayacaktır. Mevcut ultrasonik mesafe sensörleri, yaklaşık 5 cm'lik bir hata payına sahiptir. Daha hassas ultrasonik mesafe sensörleri kullanılarak, bu hata payı azaltılabilir. Bu, robotun engelleri daha doğru bir şekilde tespit etmesini ve daha güvenli bir şekilde hareket etmesini sağlayacaktır.

Daha yüksek çözünürlüklü bir IR alıcı verici sensör modülü kullanılabilir. Bu, aracın nesnelere olan mesafeyi daha doğru bir şekilde ölçmesini sağlayacaktır. Mevcut IR alıcı verici sensör modülü, yaklaşık 1 cm'lik bir hata payına sahiptir. Daha yüksek çözünürlüklü IR alıcı verici sensör modülleri kullanılarak, bu hata payı azaltılabilir. Bu, robotun nesnelere olan mesafeyi daha doğru bir şekilde ölçmesini ve daha güvenli bir şekilde hareket etmesini sağlayacaktır.

Daha güçlü redüktörlü motorlar kullanılabilir. Bu, aracın daha ağır yükleri taşıyabilmesini sağlayacaktır. Mevcut redüktörlü motorlar, 1 kilogramlık bir yük taşıyabilmektedir. Daha güçlü redüktörlü motorlar kullanılarak, bu yük taşıma kapasitesi artırılabilir. Bu, robotun daha ağır yükleri taşıyabilmesini sağlayacaktır.

Daha güçlü bir mikroişlemci kullanılabilir. Bu, robotun daha karmaşık görevleri yerine getirmesini sağlayacaktır. Mevcut mikroişlemci, temel görevleri yerine getirebilmektedir. Daha güçlü bir mikroişlemci kullanılarak, robot daha karmaşık görevleri yerine getirebilir. Bu, robotun daha geniş bir yelpazede kullanılmasını sağlayacaktır.

Aracın gövdesi, daha sağlam ve dayanıklı malzemelerden yapılabilir. Bu, robotun daha uzun ömürlü olmasına ve daha ağır yükler taşımasına olanak sağlayacaktır.

Aracın sensörleri, daha geniş bir açıda algılayabilecek şekilde geliştirilebilir. Bu, robotun daha geniş bir çevreyi algılamasını ve daha güvenli bir şekilde hareket etmesini sağlayacaktır.

Joystick, daha hassas ve ergonomik bir şekilde tasarlanabilir. Mevcut joystick, yaklaşık 5 mm'lik bir hassasiyete sahiptir. Daha hassas bir joystick kullanılarak, bu hassasiyet artırılabilir. Bu, robotun daha hassas bir şekilde kontrol edilmesini sağlayacaktır. Ayrıca, joystickin ergonomik bir şekilde tasarlanması, kullanıcının daha rahat bir şekilde robotu kontrol etmesini sağlayacaktır.

LCD ekran, daha yüksek çözünürlüklü bir ekran kullanılabilir. Mevcut LCD ekran, 16x2 karakter çözünürlüğüne sahiptir. Daha yüksek çözünürlüklü bir LCD ekran kullanılarak, robot hakkında daha fazla bilgi görüntülenebilir. Örneğin, robotun hızı, yönü, pil seviyesi ve sensörlerden gelen veriler gibi bilgiler LCD ekranda görüntülenebilir.

Robotun pil ömrü, artırılabilir. Bu, robotun daha uzun süre kontrol edilmesini sağlayacaktır. Ayrıca, pillerin verimli bir şekilde kullanılmasını sağlayan bir pil tasarruf modu kullanılabilir.

Haberleşme modülleri güçlendirilebilir. Araç ve kumanda arası haberleşmeyi güçlendirmek için, daha hızlı bir haberleşme modülü kullanılabilir. Mevcut haberleşme modülü, 2.4 GHz frekansında çalışır ve yaklaşık 2 Mbps veri hızına sahiptir. Daha hızlı bir haberleşme modülü kullanılarak, bu veri hızı artırılabilir. Bu, robotun kontrolünü daha hızlı ve daha hassas bir şekilde gerçekleştirmeyi sağlayacaktır. Örneğin, 5.8 GHz frekansında çalışan ve yaklaşık 10 Mbps veri hızına sahip bir haberleşme modülü kullanılabilir. Bu, araç ve kumanda arasındaki haberleşmenin daha hızlı ve daha kararlı olmasını sağlayacaktır.

Aracın Kodları

```
// Gerekli kütüphaneleri dahil et
#include "SPI.h"
#include "RF24.h"
#include "nRF24L01.h"

// Radyo modülü için pinleri ve ayarları tanımla
#define CE_PIN 9
#define CSN_PIN 10
#define INTERVAL_MS_SIGNAL_LOST 1000
#define INTERVAL_MS_SIGNAL_RETRY 250
// NRF24L01 radyo modülünü başlat
RF24 radio(CE_PIN, CSN_PIN);
const byte address[6] = "00001"; // Radyo iletişimi için adres

// Radyo iletişimi için veri yapısı
struct payload {
  int data1;
  int data2;
  int data3;
  byte data4;
  byte data5;
};
payload payload; // Veri aktarımı için yapı
unsigned long lastSignalMillis = 0; // Son sinyal alma zamanını tutan değişken

// Pin ve değişken tanımlamaları
int verici_arac_hata_mesaji_engel = 38; // Engel hatası için mesaj kodu
int verici_arac_hata_mesaji_takilma = 49; // Takılma hatası için mesaj kodu

int joys_yatay; // Joystick'ten gelen yatay veri
int joys_dikey; // Joystick'ten gelen dikey veri
int joys_buton; // Joystick'ten gelen buton verisi
int alici_otonom_surus_sinyali = 87; // Otonom sürüş başlatma sinyali
int alici_otonom_surusten_cikis_sinyali = 88; // Otonom sürüşten çıkış sinyali
```

```
// Sol motor için pinler
#define sol_motor_1 A3
#define sol_motor_2 A2
// Sağ motor için pinler
#define sag_motor_1 A4
#define sag_motor_2 A5

// Sensör pinleri
int sag_sensor_echo = 4;// sağ sensör
int sag_sensor_trigger = 5;// sağ sensör

int sol_sensor_echo = 6;// sol sensör
int sol_sensor_trigger = 7;// sol sensör

int on_sensor_echo = 2; // Ön sensör
int on_sensor_trigger = 3; // Ön sensör

int takilma_sayac=0;
int takilma_sayaci = 0;
int takilma_sayaci_son_durum = 0;

int engel_tanima_sayac = 0;
int engel_tanima_sayaci = 0;
int engel_tanima_sayaci_son_durum = 0;

int takilma_algilayici_sensor = A0; // Takılma algılayıcı sensör pini

void setup(void)
{
  Serial.begin(115200); //Seri iletişimi 115200 hızında başlat
  radio.begin(); //Radyo modülünü başlat
  radio.setAutoAck(false); //Otomatik onaylama özelliğini kapalı duruma getir
  radio.setDataRate(RF24_250KBPS); //Veri aktarım hızını 250 kbps olarak
ayarla
  radio.setPALevel(RF24_PA_MIN); //Güç seviyesini en düşük seviyeye ayarla
  radio.setPayloadSize(sizeof(payload)); //Veri paketinin boyutunu "payload"
yapısının boyutuna göre ayarla
  radio.openReadingPipe(0, address); //Alıcı modülü için adres tanımla
  radio.startListening(); //Radyo modülünü dinlemeye başla
  // Pinleri giriş/çıkış olarak tanımla:
  pinMode(on_sensor_trigger, OUTPUT);//Ön sensör tetikleyici pini çıkış olarak
ayarla
  pinMode(on_sensor_echo, INPUT);//Ön sensör eko pini giriş olarak ayarla

  pinMode(sol_sensor_trigger, OUTPUT);//Sol sensör tetikleyici pini çıkış
olarak ayarla
  pinMode(sol_sensor_echo, INPUT);//Sol sensör eko pini giriş olarak ayarla
```

```
  pinMode(sag_sensor_trigger, OUTPUT);//Sağ sensör tetikleyici pini çıkış
olarak ayarla
  pinMode(sag_sensor_echo, INPUT);//Sağ sensör eko pini giriş olarak ayarla

  //Sol motor pinlerini çıkış olarak ayarla
  pinMode(sol_motor_1, OUTPUT);
  pinMode(sol_motor_2, OUTPUT);
  //Sağ motor pinlerini çıkış olarak ayarla
  pinMode(sag_motor_1, OUTPUT);
  pinMode(sag_motor_2, OUTPUT);

  pinMode(takilma_algilayici_sensor, INPUT);// Takılma algılayıcı sensör
pinini giriş olarak ayarla
}

void loop(void) {
  unsigned long currentMillis = millis();//Geçen zamanı milisaniye cinsinden
al
  lastSignalMillis = currentMillis;//Son sinyal alma zamanını güncelle

  // Payload verilerini hazırla
  payload.data1 = joys_buton;
  payload.data2 = joys_yatay;
  payload.data3 = joys_dikey;
  payload.data4 = alici_otonom_surus_sinyali;
  payload.data5 = alici_otonom_surusten_cikis_sinyali;

  radio.read(&payload, sizeof(payload));// Radyodan veri okumaya çalış

  // Motor pinlerinin durumunu oku
  int motor_pin_durumu=analogRead(sol_motor_1) || analogRead(sol_motor_2) ||
analogRead(sag_motor_1) || analogRead(sag_motor_2);
  // Sensör ölçümleri yap
  long on_sensor_zaman, sol_sensor_zaman, sag_sensor_zaman, sag_mesafe,
sol_mesafe, on_mesafe;

  // Ön sensörü tetikle ve mesafeyi ölç
  digitalWrite(on_sensor_trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(on_sensor_trigger, HIGH);
  delayMicroseconds(5);
  digitalWrite(on_sensor_trigger, LOW);
  on_sensor_zaman = pulseIn(on_sensor_echo, HIGH);
  on_mesafe = on_sensor_zaman/29/2;

  // Sol sensörü tetikle ve mesafeyi ölç
  digitalWrite(sol_sensor_trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(sol_sensor_trigger, HIGH);
```

```
  delayMicroseconds(5);
  digitalWrite(sol_sensor_trigger, LOW);
  sol_sensor_zaman = pulseIn(sol_sensor_echo, HIGH);
  sol_mesafe = sol_sensor_zaman/29/2;

  // Sağ sensörü tetikle ve mesafeyi ölç
  digitalWrite(sag_sensor_trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(sag_sensor_trigger, HIGH);
  delayMicroseconds(5);
  digitalWrite(sag_sensor_trigger, LOW);
  sag_sensor_zaman = pulseIn(sag_sensor_echo, HIGH);
  sag_mesafe = sag_sensor_zaman/29/2;

  // Engel tanıma ve takılma durumlarını kontrol et
  if (millis() %20000>=18500 && millis() %20000-1500<=20000){//Her 30 saniyede
bir
    if(motor_pin_durumu<=255){//Motorlar hareket etmiyorsa
      if(engel_tanima_sayaci>=6){//Engel tanıma sayısı 6'dan büyükse
        while(1){//Sonsuz döngüye gir
          // Motorları durdur
          analogWrite(sol_motor_1, 0);
          analogWrite(sol_motor_2, 0);
          analogWrite(sag_motor_1, 0);
          analogWrite(sag_motor_2, 0);

          engel_tanima_sayac==0;// Engel tanıma sayısını sıfırla
          engel_sayac();// Engel sayacı fonksiyonunu çağır

          lastSignalMillis = currentMillis;// Son sinyal alma zamanını
güncelle

          // Payload verilerini hazırla
          payload.data1 = joys_buton;
          payload.data2 = joys_yatay;
          payload.data3 = joys_dikey;
          payload.data4 = alici_otonom_surus_sinyali;
          payload.data5 = alici_otonom_surusten_cikis_sinyali;

          radio.read(&payload, sizeof(payload));// Radyodan veri oku
          if(payload.data1==HIGH){break;}// Joystick düğmesi basılıysa
döngüden çık
        }
      }else{engel_tanima_sayac==0;}
    }
  }
  engel_sayac();
```

```
  if (millis() %10000>=8500 && millis() %10000-1500<=10000){//Her 5 dakikada
bir
    if(motor_pin_durumu<=255){//Motorlar hareket etmiyorsa
      if(takilma_sayaci<=4){//Takılma sayısı 4'ten küçükse
        while(1){ //Sonsuz döngüye gir
          // Motorları durdur
          analogWrite(sol_motor_1, 0);
          analogWrite(sol_motor_2, 0);
          analogWrite(sag_motor_1, 0);
          analogWrite(sag_motor_2, 0);

          takilma_sayac==0; // Takılma sayısını sıfırla
          takil_sayac();// Takılma sayacı fonksiyonunu çağır
          lastSignalMillis = currentMillis;// Son sinyal alma zamanını
güncelle

          // Payload verilerini hazırla
          payload.data1 = joys_buton;
          payload.data2 = joys_yatay;
          payload.data3 = joys_dikey;
          payload.data4 = alici_otonom_surus_sinyali;
          payload.data5 = alici_otonom_surusten_cikis_sinyali;

          radio.read(&payload, sizeof(payload));// Radyodan veri oku

          if(payload.data1==HIGH){break;}// Joystick düğmesi basılıysa
döngüden çık
        }
      }else{takilma_sayac==0;}
      }
  }
  takil_sayac();// Takılma sayacı fonksiyonunu çağır
  takilma_sayac=digitalRead(takilma_algilayici_sensor);// Takılma sayısını
güncelle

  if(on_mesafe >30){// Ön mesafe 30 cm'den büyükse
    engel_tanima_sayac++;//Engel tanıma sayısını bir artır
    // Motorları ileri yönde çalıştır
    analogWrite(sol_motor_1, 255);
    analogWrite(sol_motor_2, 0);
    analogWrite(sag_motor_1, 255);
    analogWrite(sag_motor_2, 0);
  }else{// Ön mesafe 30 cm'den küçükse
    // Motorları geriye doğru çalıştır
    analogWrite(sol_motor_1, 0);
    analogWrite(sol_motor_2, 255);
    analogWrite(sag_motor_1, 0);
    analogWrite(sag_motor_2, 255);
    delay(500);// 500 milisaniye bekle
```

```
    // Motorları ileri yönde çalıştır
    analogWrite(sol_motor_1, 255);
    analogWrite(sol_motor_2, 0);
    analogWrite(sag_motor_1, 0);
    analogWrite(sag_motor_2, 255);
    delay(1000);// 1000 milisaniye bekle
  }

  // ENGEL KONTROLÜ VE MANEVRA
  // Sol mesafe 30 cm'den küçük, sağ mesafe 30 cm'den büyük ve ön mesafe 30
cm'den küçük ise:
  if(sol_mesafe <=30 && sag_mesafe>30 && on_mesafe <=30){
    engel_tanima_sayac++;// Engel tanıma sayısını artır
    sag_oto();// Sağa dönüş fonksiyonunu çağır
  }
  // Sağ mesafe 30 cm'den küçük, sol mesafe 30 cm'den büyük ve ön mesafe 30
cm'den küçük ise:
  else if (sag_mesafe <= 30 && sol_mesafe > 30 && on_mesafe <= 30) {
    engel_tanima_sayac++;  // Engel tanıma sayısını artır
    sol_oto();             // Sola dönüş fonksiyonunu çağır
  }
  // Sağ, sol ve ön mesafeler 30 cm'den küçük ise:
  else if (sag_mesafe <= 30 && sol_mesafe <= 30 && on_mesafe <= 30) {
    engel_tanima_sayac++;  // Engel tanıma sayısını artır
    tam_donus_oto();       // Tam dönüş fonksiyonunu çağır
  }

  // RADYO KONTROLÜ:
  while (payload.data5 == 88) {  // Otonom sürüşten çıkış sinyali 88 olmadığı
sürece:
    lastSignalMillis = currentMillis;  // Son sinyal alma zamanını güncelle

    // Payload verilerini hazırla
    payload.data1 = joys_buton;
    payload.data2 = joys_yatay;
    payload.data3 = joys_dikey;
    payload.data4 = alici_otonom_surus_sinyali;
    payload.data5 = alici_otonom_surusten_cikis_sinyali;

    radio.read(&payload, sizeof(payload));  // Radyodan veri oku

    // Motorları durdur
    analogWrite(sol_motor_1, 0);
    analogWrite(sol_motor_2, 0);
    analogWrite(sag_motor_1, 0);
    analogWrite(sag_motor_2, 0);

    // Joystick verilerine göre motorları kontrol et:
    if (payload.data3 >= 900) {  // Joystick ileri
```

```
      analogWrite(sol_motor_1, 255);
      analogWrite(sag_motor_1, 255);
    } else if (payload.data3 <= 200) {  // Joystick geri
      analogWrite(sol_motor_2, 255);
      analogWrite(sag_motor_2, 255);
    } else if (payload.data2 >= 900) {  // Joystick sağa
      analogWrite(sol_motor_1, 255);
      analogWrite(sag_motor_2, 255);
    } else if (payload.data2 <= 200) {  // Joystick sola
      analogWrite(sol_motor_2, 255);
      analogWrite(sag_motor_1, 255);
    }

    if (payload.data4 == 87) {break;}// Otonom sürüş sinyali alınırsa döngüden
çık
  }
}

void sol_oto() {// Sola dönüş fonksiyonu
  // Sol motorları ileri yönde çalıştır
  analogWrite(sol_motor_1, 0);
  analogWrite(sol_motor_2, 255);
  // Sağ motorları geri yönde çalıştır
  analogWrite(sag_motor_1, 0);
  analogWrite(sag_motor_2, 255);
  delay(500);// 500 milisaniye bekle
  // Sol motorları geri yönde çalıştır
  analogWrite(sol_motor_1, 0);
  analogWrite(sol_motor_2, 255);
  // Sağ motorları ileri yönde çalıştır
  analogWrite(sag_motor_1, 255);
  analogWrite(sag_motor_2, 0);
  delay(500);// 500 milisaniye bekle
}

void sag_oto() {// Sağa dönüş fonksiyonu
  // Sol motorları geri yönde çalıştır
  analogWrite(sol_motor_1, 255);
  analogWrite(sol_motor_2, 0);
  // Sağ motorları ileri yönde çalıştır
  analogWrite(sag_motor_1, 0);
  analogWrite(sag_motor_2, 255);
  delay(500);// 500 milisaniye bekle
  // Sol motorları ileri yönde çalıştır
  analogWrite(sol_motor_1, 0);
  analogWrite(sol_motor_2, 255);
  // Sağ motorları geri yönde çalıştır
  analogWrite(sag_motor_1, 255);
  analogWrite(sag_motor_2, 0);
```

```
  delay(500);  // 500 milisaniye bekle
}

void tam_donus_oto(){// Tam dönüş fonksiyonu
  // Sol motorları ileri yönde çalıştır
  analogWrite(sol_motor_1, 255);
  analogWrite(sol_motor_2, 0);
  // Sağ motorları geri yönde çalıştır
  analogWrite(sag_motor_1, 0);
  analogWrite(sag_motor_2, 255);
  delay(1200);// 1200 milisaniye bekle
}

void engel_sayac(){// Robotun önündeki engellerin sayısını izleme fonksiyonu.
  if (engel_tanima_sayac != engel_tanima_sayaci_son_durum){ // Her 5
milisaniyede bir çalış
    if (engel_tanima_sayac == HIGH){// Engele çarpıldığında sayacı bir
artırır.
      engel_tanima_sayaci++;
    }
  delay(5);
  }
  engel_tanima_sayaci_son_durum = engel_tanima_sayac;
}

void takil_sayac(){// Takılma sayacı fonksiyonu
  takilma_sayac=digitalRead(takilma_algilayici_sensor);// Sensörden veri alır.
  if (takilma_sayac != takilma_sayaci_son_durum){// Geçmiş durumla
karşılaştırır.
    if (takilma_sayac == HIGH){// Engele takılmışsa, sayacı bir artırır.
      takilma_sayaci++;
    }
  delay(5);// 5 milisaniye bekler.
  }
  takilma_sayaci_son_durum = takilma_sayac;// Değişiklik varsa, yeni durumu
kaydeder.
}
```

Kumandanın Kodları

```
//kumanda
// Gerekli kütüphaneleri dahil ediyoruz
#include <LiquidCrystal_I2C.h>  // I2C LCD ekranı kontrol etmek için kütüphane
#include "SPI.h"                // SPI iletişim için kütüphane
#include "RF24.h"               // NRF24L01 radyo modülünü kullanmak için
kütüphane
#include "nRF24L01.h"           // NRF24L01 için ek tanımlamalar
```

```
#define CE_PIN 9                // NRF24L01 modülünün CE pini
#define CSN_PIN 10               // NRF24L01 modülünün CSN pini

#define INTERVAL_MS_TRANSMISSION 250  // Veri iletimi arasındaki aralık, 250
ms

RF24 radio(CE_PIN, CSN_PIN);// NRF24L01 radyo modülünü başlatıyoruz

const byte address[6] = "00001";// Haberleşme için adres belirleyelim

// Veriyolu üzerinden gönderilecek veri yapısı
struct payload {
  int data1;
  int data2;
  int data3;
  byte data4;
  byte data5;
};

payload payload;// Veriyolu üzerinden gönderilecek veri değişkeni

// Hata mesajları ve sinyalleri için değişkenler
int alici_arac_hata_mesaji_engel = 38;  // Araç engel algıladığında gönderilen
hata mesajı kodu
int alici_arac_hata_mesaji_takilma = 49; // Araç takıldığında gönderilen hata
mesajı kodu
int verici_otonom_surus_sinyali = 87;   // Otonom sürüş başlatma sinyali
int verici_otonom_surusten_cikis_sinyali = 88; // Otonom sürüşten çıkış
sinyali

// Joystick pinleri
int joys_dik = A2;  // Joystick dikey ekseni
int joys_yat = A1;  // Joystick yatay ekseni
int joys_but = A0;  // Joystick butonu

// Joystick değerlerini tutmak için değişkenler
int joys_buton;
int joys_dikey;
int joys_yatay;

// LCD ekranda yazı kaydırma için değişkenler
int scrollPosition = 0;
int scrollPosition2 = 0;
int scrollPosition3 = 0;

void setup() {
  Serial.begin(115200);  // Seri iletişimi başlat
```

```
  // Radyo modülünü ayarla
  radio.begin();
  radio.setAutoAck(false);  // Otomatik onaylama kapalı
  radio.setDataRate(RF24_250KBPS);  // Veri aktarım hızı 250 kbps
  radio.setPALevel(RF24_PA_MAX);  // Güç seviyesi maksimum
  radio.setPayloadSize(sizeof(payload));  // Veri paketi boyutu
  radio.openWritingPipe(address);  // Veri göndermek için adresi ayarla

  // Joystick pinlerini giriş olarak ayarla
  pinMode(joys_dik, INPUT);
  pinMode(joys_yat, INPUT);
  pinMode(joys_but, INPUT);

  lcd.begin();  // LCD ekranı başlat
  lcd.backlight();  // Arka ışığı aç

  // LCD ekranda açılış mesajları göster
  lcd.home();
  lcd.setCursor(4, 0);
  lcd.print("Veriler");
  lcd.setCursor(3, 1);
  lcd.print("Aliniyor...");
  delay(500);
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("Baglaniyor...");
  delay(500);
  lcd.clear();
  lcd.setCursor(0, 1);
  lcd.print("=>=>Baglandi<=<=");
  delay(500);
  lcd.clear();
}

void loop() {
  // Joystick değerlerini oku
  joys_buton = digitalRead(joys_but);
  joys_dikey = analogRead(joys_dik);
  joys_yatay = analogRead(joys_yat);

  // LCD ekrana "Araç Otonomda" mesajını yazdır
  lcd.setCursor(0, 0);
  lcd.print("Arac Otonomda");
  delay(50);
  lcd.clear();

  // Veri paketini oluştur ve gönder
  payload.data1 = digitalRead(joys_but);
  payload.data2 = analogRead(joys_yat);
```

```
  payload.data3 = analogRead(joys_dik);
  payload.data5 = verici_otonom_surusten_cikis_sinyali;  // Otonom sürüşten
çıkış sinyali gönder

  radio.write(&payload, sizeof(payload));  // Veri paketini gönder
  delay(INTERVAL_MS_TRANSMISSION);  // Gönderimler arasında bekle

  // Joystick butonu basılı ve joystick sol alt köşede ise manuel sürüşe geç
  while (joys_buton == HIGH && joys_dikey <= 200 && joys_yatay <= 200) {
    basadon:  // Döngü etiketi
    lcd.setCursor(0, 0);
    lcd.print("manuel surustuseniz");  // LCD ekrana manuel sürüş mesajını
yazdır

    joys_yon_atama();    // Joystick yön atama fonksiyonunu çalıştır

    // Veri paketini oluştur ve gönder
    payload.data1 = digitalRead(joys_but);
    payload.data2 = analogRead(joys_yat);
    payload.data3 = analogRead(joys_dik);
    payload.data4 = verici_otonom_surus_sinyali;  // Otonom sürüş başlatma
sinyali gönder

    radio.write(&payload, sizeof(payload));  // Veri paketini gönder

    // Seri iletişime veri yazdır
    Serial.print("Data1:");
    Serial.println(payload.data1);
    Serial.print("Data2:");
    Serial.println(payload.data2);
    Serial.println("Sent");
    delay(INTERVAL_MS_TRANSMISSION);  // Gönderimler arasında bekle

    // Joystick butonu basılı ve joystick sol üst köşede ise otonom sürüşe dön
    if (joys_buton == HIGH && joys_dikey >= 900 && joys_yatay <= 200) {
      break;  // Döngüden çık
    }

    // Döngüyü baştan başlat
    goto basadon;
  }
}

void joys_yon_atama() {
  // Joystick değerlerini oku
  joys_buton = digitalRead(joys_but);
  joys_dikey = analogRead(joys_dik);
  joys_yatay = analogRead(joys_yat);
}
```