

# TURKROBOKİT ROBOTİK KODLAMA EĞİTİM SETİ

**Bu E-Kitap Robotzade ve GRobotik' in  
Robot Severlere Armağandır**

**Turkrobokit ışık izleyen robot yapımı**  
**Turkrobokit engelden kaçan robot yapımı**  
**Turkrobokit sumo robot yapımı**  
**Turkrobokit bluetooth kontrollü robot yapımı**  
**Turkrobokit çizgi izleyen robot yapımı**



## **TURKROBOKİT EĞİTİM KLAVUZU**

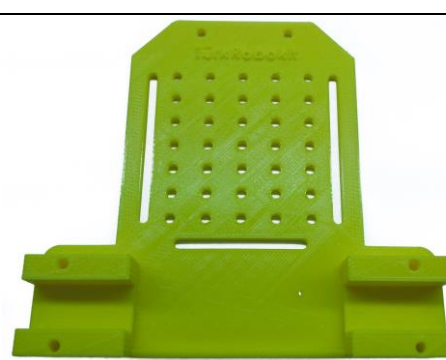


### **İçindekiler:**

- 1-) Robot kiti içeriği**
- 2-) Robot kartının incelenmesi**
- 3-) Sensörlerin tanıtımı**
- 4-) Motor Kontrol Alt Programı Kullanımı**
- 5-) Turkrobokit ışık izleyen robot yapımı**
- 6-) Turkrobokit engelden kaçan robot yapımı**
- 7-) Turkrobokit sumo robot yapımı**
- 8-) Turkrobokit bluetooth kontrollü robot yapımı**
- 9-) Turkrobokit çizgi izleyen robot yapımı**
- 10-) Turkrobokit çizgi izleyen algoritmaları**
  - a) İki çizgi arası hızlanma**
  - b) 90 derece dönüşler**
  - c) Yol ayrımında karar verme**
  - d) Kesik çizgilerden geçme**
  - e) Engelden atlama**
  - f) Zemin değiştirme**
  - g) Zamana ve referans noktalarına bağımlı olarak hızlanma**
  - h) Haydi yarışalım :D**

### 1-) Robot kiti içeriği

- a) DC 6V 600rpm redüktörlü mikro motor x 2
- b) Ana gövde x 1 , Çizgi izleyen başlık x 1, Engelden Kaçan Başlık x 1
- c) Sumo Başlık x 1, Işık izleyen Başlık x 1
- d) Türkrobokit Kontrol kartı x 1
- e) 7.4V 500mAH Li-po batarya x 1
- f) 2s-3s balanslı Li-Po şarj aleti x 1
- g) Sg-90 micro servo motor x 1
- h) HCSR04 ultrasonic mesade sensörü x 1
- i) QTR 8A çizgi sensörü x 1, QTR 1A x 2
- j) Işık izleyici Kartı x 1
- k) MZ80 endüstriyel tip kızılötesi cisim algılama sensörü x 3
- l) HC06 Bluetooth modülü x 1

### Set İçinde Kullanılan Plastik Gövde Parçaları

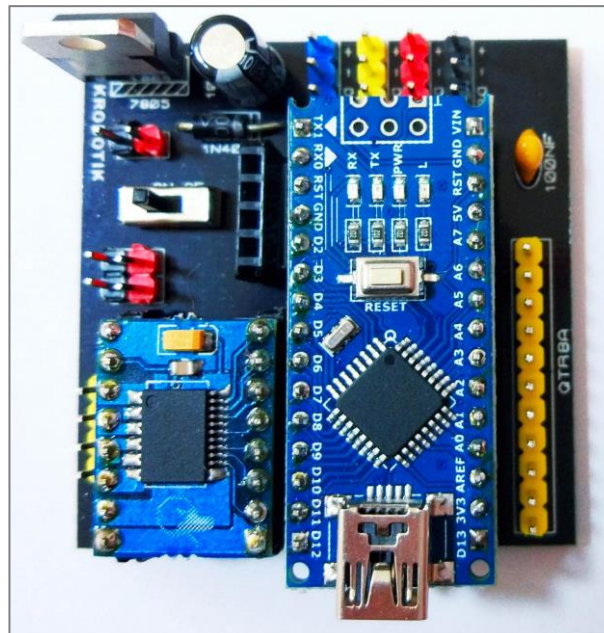
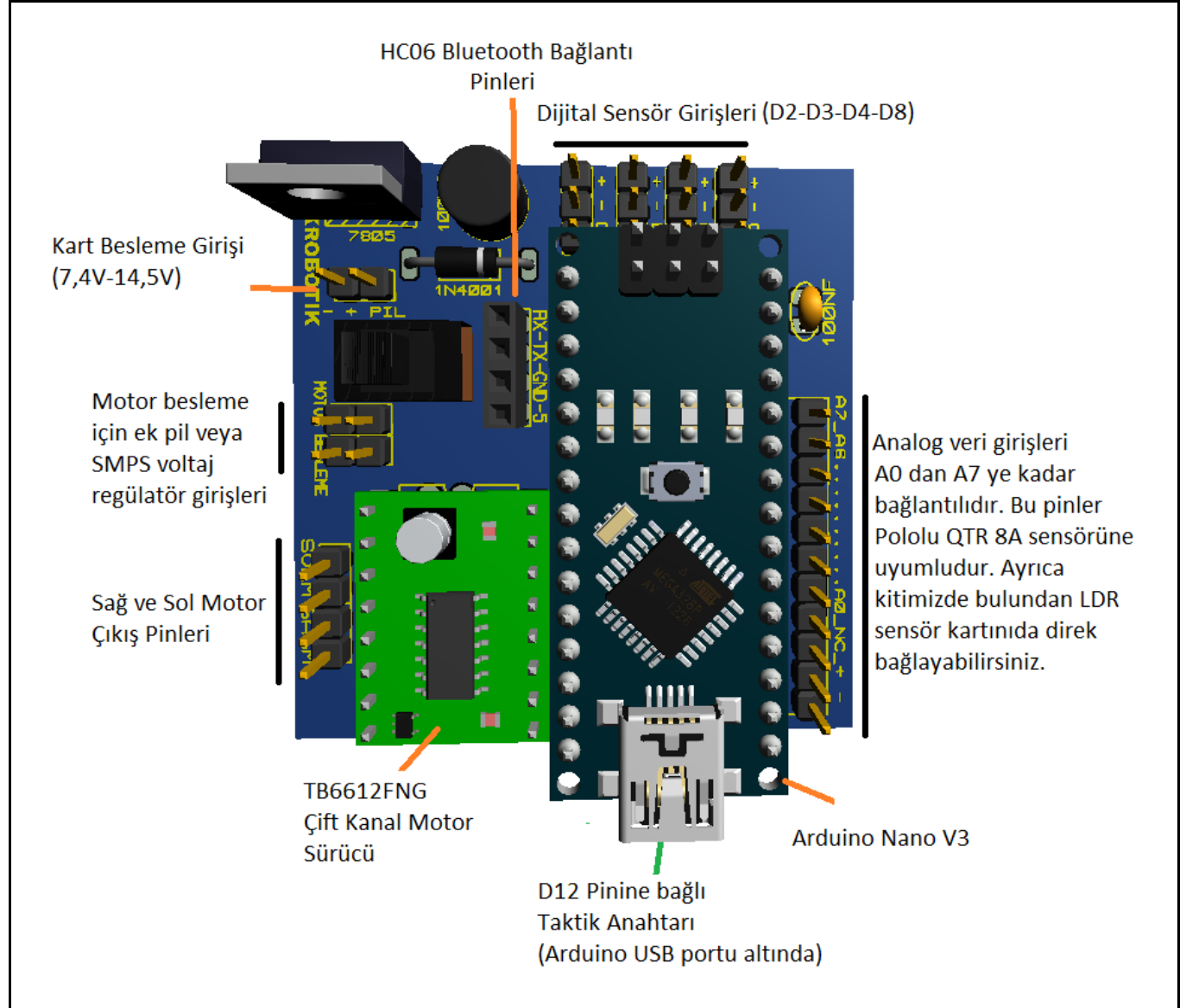
 <p>QTR-8A Sensor koruyucu</p>	 <p>Sumo Robot Başlığı</p>	 <p>Engelden Kaçan Robot Ön Başlığı</p>
 <p>LDR Modül Başlığı</p>	 <p>Ana Gövde Alt Parça</p>	 <p>Çizgi İzleyen Başlığı</p>
 <p>Bluetooth ve Işık Takip Robot Başlığı</p>	 <p>Ana Gövde Üst Parça</p>	

İçerik videolarına aşağıdaki linklerden ulaşabilirsiniz.

<https://www.youtube.com/watch?v=YD1h7LKFclo>

<https://www.youtube.com/watch?v=BxqWyT5xYcw>

## 2-) Robot kartının incelenmesi



Günümüzde kodlama ve elektronik severlerin ilgi alanları, geliştirme kartlarını kullanarak proje geliştirmeye yönelmektedir. Kaynak fazlalığından ve kodlama kolaylığından dolayı Arduino geliştirme kartları oldukça popülerdir. Bundan dolayı bizler temel robotik kartımızda Arduino Nano V3 kartını tercih ettik. Toshiba firması tarafından üretilen TB6612 motor sürücü entegresini devremizde kullanılmaktadır. TB6612FNG kartı çift kanal DC motor sürebilmektedir. Kanal başına sürekli olarak 1A çıkış verebilen kart anlık 3A e kadar desteklemektedir. Daha yüksek akım elde edebilmek için TB6612FNG kartları üst üste lehimleyerek paralel bağlanabilirler. Paralel bağlantıda akım değerleri 2 kat artabilmektedir. Motor sürücü besleme gerilimi en fazla 13,5V verilebilir. Robotik uygulamalarda motor devir hızı pil gerilimine bağlı olarak değişmektedir. Örneğin DC bir motora 12,2V gerilim altında 127PWM kontrol sinyali verdiğinizde yaklaşık olarak 1200 devirde döndüğünü farzedelim. Pil gerilimi azaldıkça devir hızı ve torkta düşecektir. Bu durum özellikle çizgi izleyen robotlarda önem arz etmektedir. Çizgi izleyen robot yarışmalarında önceden ölçüleri belli olan pistlerde başarılı olabilmek için zamana göre kodlama yapılabilir. Pil gerilimindeki değişimler zamanlarda kaymalara sebep olmaktadır. Robotunuz 16. Saniyede 23m yol alması gerekirken pildeki azalmadan dolayı 22 veya 21m yol alabilir. Bu olumsuz durumun önüne geçebilmek için gerilimin sabitlenmesi gerekmektedir. Gerilim değişimlerinin önüne geçmek için step-up veya step-down regülatör modülleri kullanılabilir. Bizim temel robotik kartımızda regülatör bağlantıları için pinleri mevcuttur. Gerekli bağlantılar kılavuzumuzun ilerleyen sayfalarında anlatılacaktır. Kartımız üzerinde kullanılan Arduino nano pinlerinin kullanım tablosu aşağıda verilmiştir.

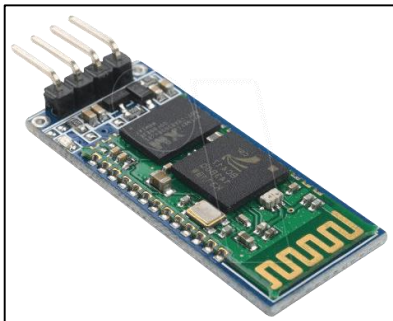
Arduino	Amacı & Kullanımı	Arduino	Amacı & Kullanımı
D0	Bluetooth bağlantısı	D11	Sağ Motor PWM (Hız Kontrol)
D1	Bluetooth bağlantısı	D12	Taktik Anahtarı
D2	Dijital sensör bağlantısı	D13	Dahili Kontrol Ledi
D3	Dijital sensör bağlantısı	A0	Analog Giriş 0
D4	Dijital sensör bağlantısı	A1	Analog Giriş 1
D5	Sol Motor PWM (Hız Kontrol)	A2	Analog Giriş 2
D6	Sol Motor Yön Kontrolü 1	A3	Analog Giriş 3
D7	Sol Motor Yön Kontrolü 2	A4	Analog Giriş 4
D8	Dijital sensör bağlantısı	A5	Analog Giriş 5
D9	Sol Motor Yön Kontrolü 2	A6	Analog Giriş 6
D10	Sol Motor Yön Kontrolü 1	A7	Analog Giriş 7

Kart ile ilgili videoya aşağıdaki linkten ulaşabilirsiniz.

<https://www.youtube.com/watch?v=FcbEt8sOGKE>

### 3-) Sensörler ve modüllerin tanıtımı

#### a) HC06 Bluetooth modülü



HC06 Bluetooth-Serial Modül Kartı, Bluetooth SSP(Serial Port Standart) kullanımı ve kablosuz seri haberleşme uygulamaları için tasarlanmıştır. Hızlı prototiplemeye imkan sağlaması, breadboard, arduino ve çeşitli devrelerde rahatça kullanılabilmesi için gerekli pinler devre kartı sayesinde dışarıya alınmıştır.

Standart pin yapısı sayesinde istenilen ortamlarda rahatça kontrol edilebilir. Bununla beraber ürünle gönderilen jumper kablolar ile bağlantılar rahatlıkla yapılabilir.

Bluetooth 2.0'ı destekleyen bu kart, 2.4GHz frekansında haberleşme yapılmasına imkan sağlayıp açık alanda yaklaşık 10

metrelik bir haberleşme mesafesine sahiptir.

Bir çok hobi, robotik ve akademik projede kullanılabilir. HC05'in aksine yalnızca slave modda kullanılabilir.



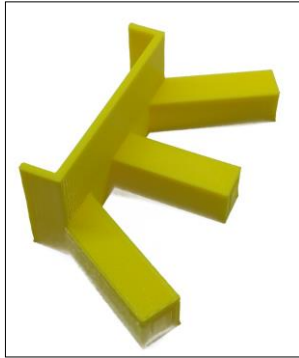
**b) MZ80 Kızılötesi cisim algılama sensörü**



80cm Menzilli Kızılötesi Sensör birçok robot projesine uygun dijital çıkışlı, yüksek kaliteli, endüstriyel kızılötesi sensördür. Arkasındaki trimpot ile menzili 3-80 cm arasında ayarlanabilir. NPN çıkışlıdır. 5V ile çalışır, tepki süresi oldukça düşüktür.( 2 ms) Montajlama aparatıyla rahatça montajlanabilir. Kablo bağlantıları aşağıdaki gibidir. (Ürün iki farklı kablo rengi kombinasyonu ile gönderilebilir);

Kombinasyon 1	Kombinasyon 2
<b>Kahverengi:</b> +5V	<b>Kırmızı:</b> +5V
<b>Mavi:</b> GND	<b>Yeşil:</b> GND
<b>Siyah:</b> Data Çıkışı	<b>Sarı:</b> Data Çıkışı

**c) LDR Modülü**



LDR Modülümüz 3 Adet ışığa duyarlı sensörden oluşmaktadır. Kartımızın QTR pinlerine direk bağlanabilmektedir. LDR ler boru içine açılı yerleştirilmiştir. Işıkların geliş açısına göre farklı değerler almaktadırlar. Modülümüzdeki sensörler analog A0,A1 ve A2 pinlerine bağlanacaktır.

**d) SG90 micro servo**



SG90 Servo Motor - Tower Pro, Futaba kumandalarla uyumlu, RC projelerde kullanılabilecek küçük boyutlu bir servo motordur. PWM sinyalleriyle kontrol edilir. Motorun dişli kutusu plastiktir. SG90 Servo Motor - Tower Pro paketi içerisine değiştirilebilir servo başlıkları dahildir.

Boyutları: 23.1mm x 12.2mm x 29mm

Ağırlık: 9g

Hız @4.8V: 0.1sn/600

Zorlanma torku @ 6V: 1.3kg-cm

Kablo Uzunluğu: 15cm

**e) HCSR04**



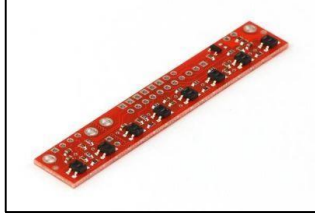
HC-SR04 Ultrasonik mesafe ölçümü yapabileceğiniz cisim sensörüdür. 2cm'den 400cm'ye kadar 3mm hassasiyetle ölçüm yapabilen bu ultrasonik sensör çeşitli uzaklık okuma, radar ve robot uygulamalarında kullanılabilir. Arduino ve tüm mikrodenetleyicilerde sorunsuz kullanabilirsiniz.

Çalışma Voltajı  
Çektiği Akım

DC 5V  
15 mA

Çalışma Frekansı	40 Hz
Maksimum Görme Menzili	4m
Minimum Görme Menzili	2cm
Görme Açısı	15°
Tetik Bacağı Giriş Sinyali	10 us TTL Darbesi
Echo Çıkış Sinyali	Giriş TTL sinyali ve Mesafe Oranı
Boyutları	45mm x 20mm x 15mm

**f) QTR8A analog çizgi izleyici sensörü**



Çizgi izleyen robotlar için tasarlanmış olan bu sensör kartı 1cm arayla yerleştirilmiş 8 IR LED/fototransistör çifti barındırmaktadır. Stabil ve sorunsuz çalıştığı için robotlarda çok fazla tercih edilmektedir.

LED çiftlerinin her biri ayrı birer MOSFET transistörle sürülmektedir ve ek hassasiyet ya da güç tasarrufu için LED'ler kapatılabilir.

Kart üzerindeki her sensör ayrı bir analog voltaj çıkışı sağlar. Her bir sensöre bir pull-up direnci bağlanmıştır. Zeminin ışığı yansıtması veya cisimle olan mesafesine göre voltaj çıkışı analog olarak değişir. Yansıma arttıkça çıkış voltajı da yükselir. QTR-8A Sensör Kartı çizgi izleyen robotlar dışında bir yakınlık veya cisim algılama sensörü olarak da kullanılabilir.

**4-) Motor Kontrol Alt Programı Kullanımı**

Yapacağımız tüm robotlarda redüktörlü DC motorlarımızı kontrol edebilmek için motor kontrol alt programımızı sıklıkla kullanacağız. Bu sebepten kodlamaya başlamadan önce mantığının çok iyi kavranması gerekmektedir. Motor kontrol alt programımızda iki temel parametre vardır bunlar sağ motor hızı ve sol motor hızı. Biz kartımızda iki kanallı, çift yönlü TB6612FNG motor sürücü kullanıyoruz. Bu sürücü de A ve B kanalları ile motorlara birbirinden bağımsız olarak yön ve hız ayarı yapılabilir. Motor sürücünde A motorunun yönünü kontrol edebilmek için 2 tane pin vardır. TB6612 hakkında detaylı bilgi için <https://www.pololu.com/file/0J86/TB6612FNG.pdf> adresini kullanabilirsiniz.

Motor kontrol alt programımızda parametre olarak -255 ile +255 arasında PWM değeri yazabiliriz. 255 PWM motorlara tam güç aktarır 0 PWM ise dinamik frenler. Ara değerler ile aktarılan güç değiştirilir ve hız da değişmiş olur. Negatif değerler motorun geri döndüreceğimizi pozitif değerler ise ileri yönde döndüreceğimizi gösterir. Parametlerden birincisi sol motor hız ve yön bilgisini ikincisi de sağ motor hız ve yön bilgisini göstermektedir. Bundan sonrasını örneklerle açıklayalım.

```
motorkontrol(127,127); delay(1000); // 1sn boyunca orta güçte ileri gider
motorkontrol(255,255); delay(250); // 250ms tam gaz ileri gider
motorkontrol(0,100); delay(500); // 500ms boyunca sol motor fren yapar sağ motor 100PWM ileri gider
motorkontrol(200,100); delay(500); // 500ms boyunca sol motor hızlı olmak üzere sağa kavis çizer
motorkontrol(-50,-50); delay(750); // 750 boyunca iki motorda 50PWM hızında geri gider
motorkontrol(0,0); delay(5000); // 5sn boyunca iki motorda frenler
motorkontrol(-200,0); delay(350); // 350ms boyunca sol motor geri döner sağ motorda frenler
```

Bu kadar örnek konunun anlaşılması için yeterlidir sanırım. Alt program kodlarımızı inceleyerek nasıl çalıştığını anlayabiliriz.

```

void motorkontrol(int solmotorpwm, int sagmotorpwm){

    if(sagmotorpwm<=0) {
        sagmotorpwm=sagmotorpwm*(-1);
        digitalWrite(sagmotor1, LOW);
        digitalWrite(sagmotor2, HIGH);
        analogWrite(sagmotorpwmpin, sagmotorpwm);
    }
    else {
        digitalWrite(sagmotor1, HIGH);
        digitalWrite(sagmotor2, LOW);
        analogWrite(sagmotorpwmpin, sagmotorpwm);
    }
    if(solmotorpwm<=0) {
        solmotorpwm=solmotorpwm*(-1);
        digitalWrite(solmotor1, LOW);
        digitalWrite(solmotor2, HIGH);
        analogWrite(solmotorpwmpin, solmotorpwm);
    }
    else {
        digitalWrite(solmotor1, HIGH);
        digitalWrite(solmotor2, LOW);
        analogWrite(solmotorpwmpin, solmotorpwm);
    }
}

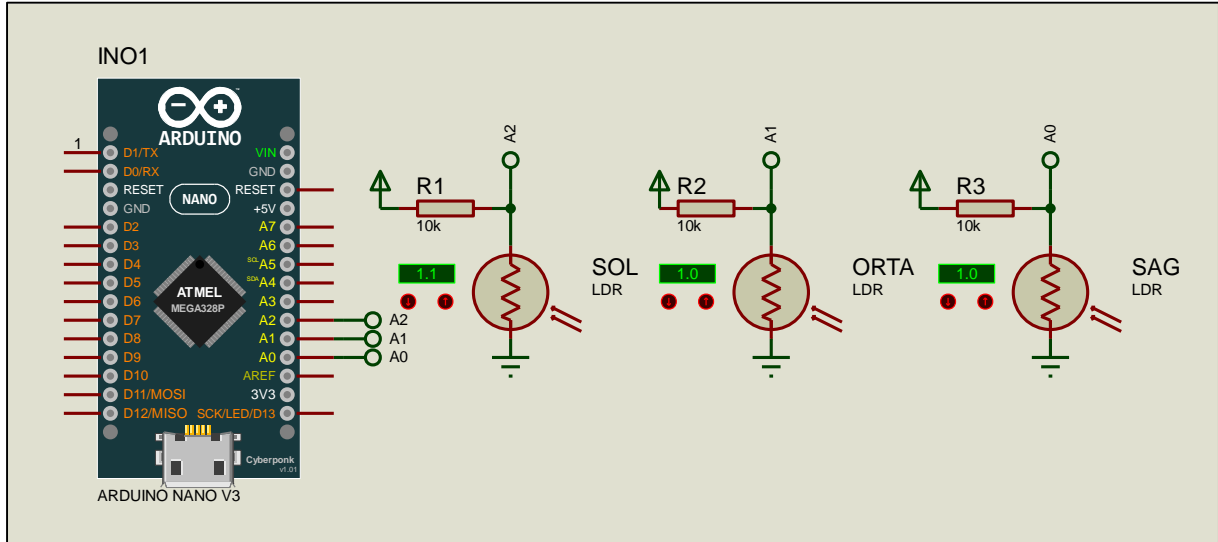
```

Artık robotlarımızı kodlamaya başlayabiliriz.

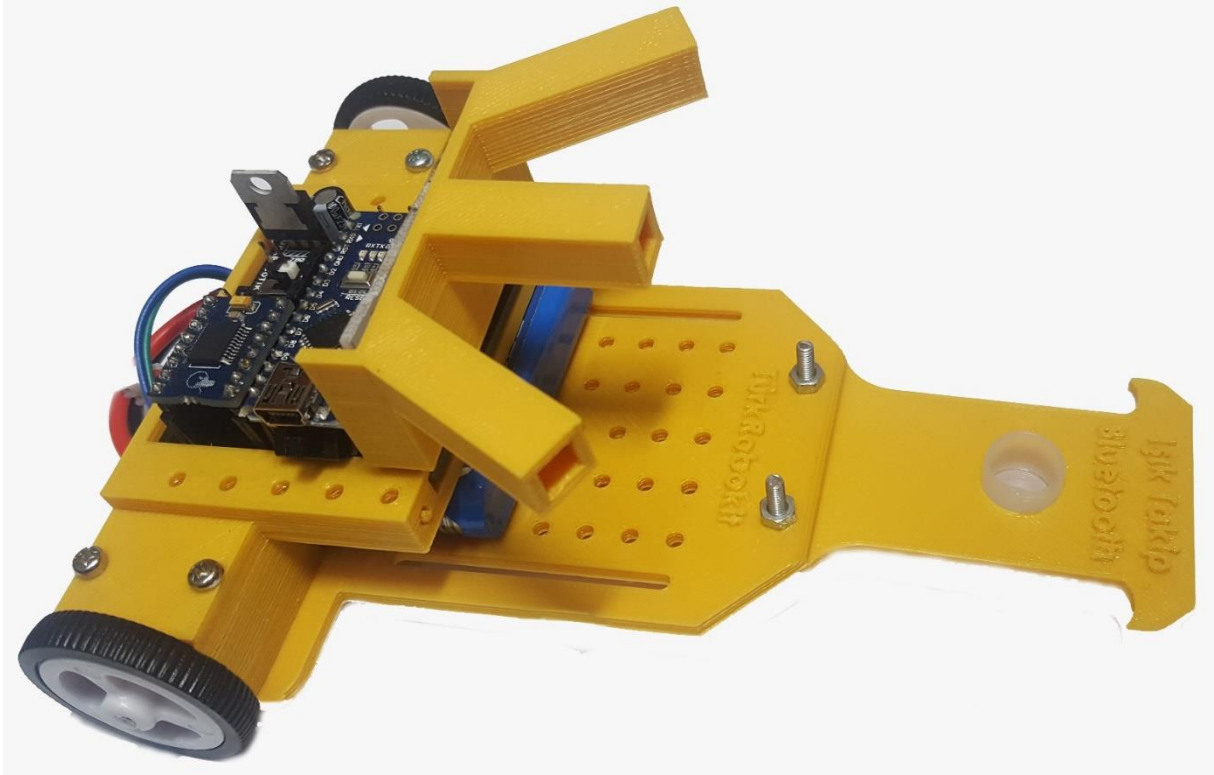
### 5-) Turkrobokit ışık izleyen robot yapımı

Türkrobokit kartımızın bulunduğu ana gövdeye 3 adet farklı yönlere bakan LDR li sensör kartımızı QTRA yazan pinlere yerleştiriyoruz. LDR lerden robota arkadan baktığımızda en sağda bulunan LDR kartımızın A0 analog girişine ortadaki LDR kartımızın A1 pini ve en soldaki LDR ise A2 pinine bağlanmış olacaktır.

LDR ler ışığa bağlı olarak direnci değişen pasif sensörlerdir. Işık sensör kartımız alttaki devre şemasına uygun olarak tasarlanmıştır. Her LDR nin bir ucunu 10K direnç üzerinden +5V a bağlanmış diğer ucu ise GND ye bağlanmıştır. LDR nin üzerine düşen ışık miktarı arttıkça Analog pindeki gerilim de düşmektedir. Yapacağımız algoritma mantığı şu şekildedir; öncelikle robotumuzu kendi etrafında birkaç tur döndürerek sensörlerden okunan en düşük değeri bulmaya çalışacağız. Programımızın ana döngüsünde sürekli sensörleri okuyacağız ve minimum değerden daha da düşük değer görürsek o yönden ışık geliyor demektir ve bu durumlara göre şartlarımızı yazacağız.







Yukarıdaki şekilde robotumuzun toplanmış resmi görülmektedir. Şimdi kod satırları üzerinden açıklamalı olarak algoritmamızı yazalım.

```
//IŞIK TAKİP ROBOTU
// pin bağlantı tanımlamalarını yapıyoruz
#define sagmotor1 10
#define sagmotor2 9
#define sagmotorpwmpin 11
#define solmotor1 6
#define solmotor2 7
#define solmotorpwmpin 5
#define sag_ldr_pin A0
#define on_ldr_pin A1
#define sol_ldr_pin A2
// programda kullanacağımız çeşitli değişkenleri tanımlıyoruz.

int sag_ldr_deger = 0;
int on_ldr_deger = 0;
int sol_ldr_deger = 0;
//sensor kalibrasyon değişkenlerini tanımlıyoruz
int sag_ldr_Min = 1023;           // minimum sensor başlangıç değeri
int on_ldr_Min = 1023;           // minimum sensor başlangıç değeri
int sol_ldr_Min = 1023;          // minimum sensor başlangıç değeri

void setup() { // pinlerin giriş çıkış tanımlamalarını yapıyoruz
pinMode(sagmotor1,OUTPUT);
pinMode(sagmotor2,OUTPUT);
pinMode(solmotor1,OUTPUT);
pinMode(solmotor2,OUTPUT);
Serial.begin(9600); //
delay(1000); //1 saniye bekliyoruz

kalibrasyon(); // ldr lerin almış olduğu minimum değeri bulacağız
}
```

```

void loop()
{
  sag_ldr_deger=analogRead(sag_ldr_pin);           // sağ ldr değerini okuyoruz
  sol_ldr_deger=analogRead(sol_ldr_pin);           // Sol ldr değerini okuyoruz
  on_ldr_deger=analogRead(on_ldr_pin);             // On ldr değerini okuyoruz

  sag_ldr_deger= sag_ldr_Min-sag_ldr_deger;        // Sensör üzerine ışık düştükçe değeride azalmakıyadı
  sag_ldr_deger=constrain(sag_ldr_deger,0,1000);   // Bu sebepten biz min. değerden okunanı çıkaracağız ve
  // Pozitif sonuçlar bulacağız. Örneğin sol ldr nin minimum değeri kalibrasyonda 245 olsun
  sol_ldr_deger=sol_ldr_Min-sol_ldr_deger; // ışık tuttuğumuzda 135 olsun formüle göre sol ldr değeri
  sol_ldr_deger=constrain(sol_ldr_deger,0,1000);   // 245-135=110 olur yani üzerine 110 ışık düşmüş diyeceğiz

  on_ldr_deger=on_ldr_Min-on_ldr_deger;
  on_ldr_deger=constrain(on_ldr_deger,0,1000);

  Serial.print("Sol değer="); Serial.print(sol_ldr_deger); // Soldan okunan değeri seri ekrana yazdırıyoruz
  Serial.print("   Orta değer="); Serial.print(on_ldr_deger); // Ondan okunan değeri seri ekrana yazdırıyoruz
  Serial.print("   Sağ değer="); Serial.println(sag_ldr_deger); // Sağdan okunan değeri seri ekrana yazdırıyoruz

  // Eğer sağdan ışık vuruyorsa robotumuzun sol motorunu hızlı sağ motorunu yavaş dondurarak sağ tarafa yöneliyoruz
  if(sag_ldr_deger>50 && sag_ldr_deger>on_ldr_deger )
  { motorkontrol(100,20); Serial.println(" Robotumuz Sağ tarafa dönüyor"); } else
  // Eğer soldan ışık vuruyorsa robotumuzun sağ motorunu hızlı sol motorunu yavaş dondurarak sol tarafa yöneliyoruz
  if(sol_ldr_deger>50 && sol_ldr_deger>on_ldr_deger)
  { motorkontrol(20,100); Serial.println(" Robotumuz Sol tarafa dönüyor"); } else
  // Eğer ortadan ışık vuruyorsa robotumuzun sağ ve sol motorunu eşit dondurarak on tarafa yöneliyoruz
  if(on_ldr_deger>50) { motorkontrol(60,60); Serial.println(" Robotumuz İleri Gidiyor"); } else
  { motorkontrol(0,0); Serial.println(" Robotumuz Duruyor"); }
  delay(10);
}

// ...

void motorkontrol(int solmotorpwm, int sagmotorpwm){
  if(sagmotorpwm<=0) {
    sagmotorpwm=abs(sagmotorpwm);
    digitalWrite(sagmotor1, LOW);
    digitalWrite(sagmotor2, HIGH);
    analogWrite(sagmotorpwmpin, sagmotorpwm);
  }
  else {
    digitalWrite(sagmotor1, HIGH);
    digitalWrite(sagmotor2, LOW);
    analogWrite(sagmotorpwmpin, sagmotorpwm);
  }
  if(solmotorpwm<=0) {
    solmotorpwm=abs(solmotorpwm);
    digitalWrite(solmotor1, LOW);
    digitalWrite(solmotor2, HIGH);
    analogWrite(solmotorpwmpin, solmotorpwm);
  }
  else {
    digitalWrite(solmotor1, HIGH);
    digitalWrite(solmotor2, LOW);
    analogWrite(solmotorpwmpin, solmotorpwm);
  }
}

```

```

void kalibrasyon(){
  // ldr lerin kalibrasyon ayarını yapıyoruz. yaklaşık 5 sn boyunca sensörlerin aldığı minimum değerler kayıt ediliyor.
  while (millis() < 6000) {
    sag_ldr_deger = analogRead(sag_ldr_pin);
    if (sag_ldr_deger < sag_ldr_Min) {
      sag_ldr_Min = sag_ldr_deger; } // sağ sensörün minimum değeri kayıt edildi.

    sol_ldr_deger = analogRead(sol_ldr_pin);
    if (sol_ldr_deger < sol_ldr_Min) {
      sol_ldr_Min = sol_ldr_deger; } // sağ sensörün minimum değeri kayıt edildi.

    on_ldr_deger = analogRead(on_ldr_pin);
    if (on_ldr_deger < on_ldr_Min) {
      on_ldr_Min = on_ldr_deger; } // ön sensörün minimum değeri kayıt edildi.

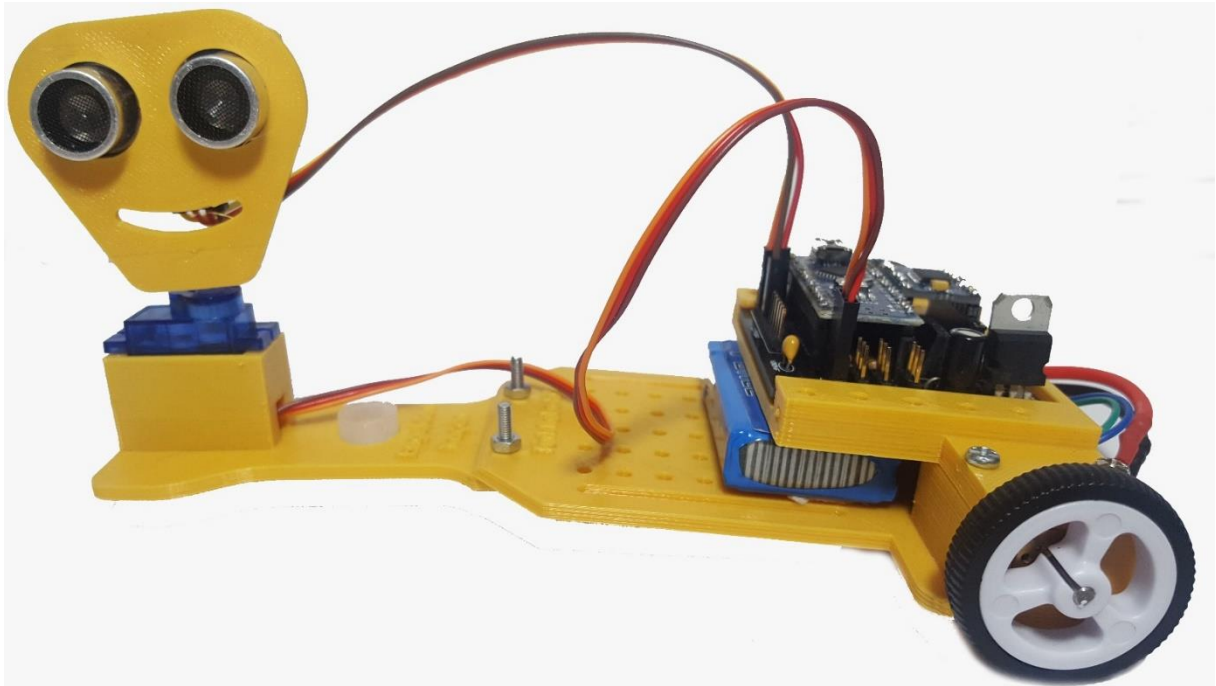
    motorkontrol(60,-60);
  }
  Serial.print("Sol min değer="); Serial.print(sol_ldr_Min); // Soldan okunan değeri seri ekrana yazdırıyoruz
  Serial.print("Orta min değer="); Serial.print(on_ldr_Min); // Ondan okunan değeri seri ekrana yazdırıyoruz
  Serial.print("Sağ min değer="); Serial.println(sag_ldr_Min); // Sağdan okunan değeri seri ekrana yazdırıyoruz
}

```

Robot videosuna aşağıdaki linkten ulaşabilirsiniz.

<https://www.youtube.com/watch?v=9Q48u7xpPjA>

#### 6-) Turkrobokit engelden kaçan robot yapımı



Engelden kaçan robotumuzu kurabilmek için ana gövdeye engelden kaçan başlığının bağlantısını yapılmalıdır. Ardından servo motorumuzun + - Data pin kablolarını Türkrobokit kartımızın Data 8 pinine bağlayacağız. Ultrasonic sensörümüzün + ve - pinlerini kart üzerindeki + ve - pinlere bağlantısını yapacağız. Sensörümüzün Trig pinini A1'a , Echo pinini de A0'e bağlantısını yapalım. Ultrasonic sensörler ses dalgası göndererek mesafe ölçümü yaparlar. Aslında sensör ölçüm yapmamaktadır trig pininden sinyal verdiğimizde ses dalgası yollamakta ve karşıya çarpan ses dalgası geri yankı yaptığında echo pininden dijital çıkış vermektedir. Yani trig pinini karşıdaki dağa doğru seslenen bir insanın ağzına, echo pinini de dağdan gelen yankıyı duyan kulaklara benzetebiliriz. Ses dalgalarının saniyedeki hızı **343.2 m** dir. Arduino da pulseIn komutu bir pindeki sinyal

değişimini mikrosaniye cinsinden bize veren komuttur. O sebepten ölçümlerimiz mikrosaniye cinsinden olacaktır. Saniyede 342.3m yol alan ses dalgası cm cinsinden (x100) 34230 yol alır. 1 mikrosaniyede(/1000000) ise 0,03432cm yol alır. Buradan  $1/0,03432=29,13$  yapar. Ses dalgamız cisme giderken ve gelirken olması gerekenden 2 kat daha fazla yol almaktadır. Bu değerler göz önüne alındığında mesafe=(olculensure/2)/29,13 formülüyle bulunur. Ayrıca yapacağımız uygulamada biz 100cm den fazla uzaklığı ölçmek istemiyoruz bu sebepten arduinomuza  $100*2*29,13= 5826$  mikrosaniye zaman aşımı sınırı koyacağız. Zaman aşımını pulseln komutunun parametresi olarak yazacağız. Robotumuzun sağlıklı çalışması için zaman aşımı koymak önemlidir aksi takdirde Arduino kulaklarının sinyal gelsin diye beklemeye alır ve başka iş yapamaz☹. Sizler daha farklı zaman aşımları koyabilirsiniz.

Robotumuz ileri yönde giderken sürekli mesafeyi ölçecek, mesafe 30cm'den az olduğunda duracak ve önce sol tarafını sonra sağ tarafını kontrol edecek. Kontrol yaparken mesafe ölçümü yapmaya devam edecek ve sağ-sol mesafeleri kıyaslayacak hangi tarafta daha fazla boşluk görürse boşluk gördüğü yöne doğru gidecek. Bizim robotumuzun çalışma senaryonu bu mantıkta kurduk sizler daha farklı algoritma geliştirebilirsiniz. Şimdi kodlarımız üzerinden açıklamalar ekleyerek devam edelim.

```
// pin bağlantı tanımlamalarını yapıyoruz
#include <Servo.h>
// servo motoru olay bir şekilde kullanabilmek için servo.h kütüphanesini ekledik.
#define sagmotor1 10
#define sagmotor2 9
#define sagmotorpwmpin 11
#define solmotor1 6
#define solmotor2 7
#define solmotorpwmpin 5
#define TRIG_PIN A1 // Ses gönderme pinini tanımladık
#define ECHO_PIN A0 // Ses alma pinini tanımladık
Servo turkrobokitservo; // servo tanımlandı
long sure, mesafe; // Mesafe hesabında kullanılan değişkenler

void setup() { // pinlerin giriş çıkış tanımlamalarını yapıyoruz
pinMode(TRIG_PIN, OUTPUT); // Ses gönderme pin türünü çıkış olarak ayarla
pinMode(ECHO_PIN, INPUT); // Yansıyan ses algılama pin türünü giriş olarak ayarla
pinMode(sagmotor1,OUTPUT); //
pinMode(sagmotor2,OUTPUT); //
pinMode(solmotor1,OUTPUT); // motor pinleri çıkış olarak ayarlandı
pinMode(solmotor2,OUTPUT); //
turkrobokitservo.attach(8); // servo motorun data pini belirlendi.
delay(1000); //1 saniye bekliyoruz
motorkontrol(0,0); // Motorlarımızı dinamik olarak frenliyoruz
turkrobokitservo.write(80); // servo motorun dik durması için ayar.
//Normalde 90 olması gerekli ama uygulamada bu tip servolar tam değerlere her zaman dönmüyorlar.
delay(1000); // ana döngüden önce 1 sn bekleme
}

void loop()
{
turkrobokitservo.write(80); // servonun karşıya bakması için
mesafeolc();
// hcscr04 için alt program. bu alt program mesafeyi ölçüp geri mesafe bilgisi yolluyor.
//Nasıl çalıştığına alt programın içine bakarak daha iyi anlayabilirsiniz.
if ( mesafe > 30 ) motorkontrol(100,100); // mesafe 30 cm den büyük ise robot düz gidiyor.
if ( mesafe <= 30 ) alanara(); // mesafe 30 cm den küçük ise robot gidebileceği boş alan arıyor.
//Nasıl çalıştığına alt programın içine bakarak daha iyi anlayabilirsiniz.
}
```

```

//mesafe 30 cm den küçük olduğu için buradayız
// bu fonksiyon geriye bir değer döndürmez. yani burada yazan komutların icra etmesini sağlar.
void alanara(){
    motorkontrol(0,0); delay(100);          // robot 100ms durur
    turkrobokitservo.write(180); delay(1000); // servo sola tam döner
    mesafeolc();                             // mesafeyi ölçer
    int mesafesag=mesafe;                    // ölçtüğü mesafeyi hafızaya alır.
    delay(100);                              // 100ms bekler
    turkrobokitservo.write(5); delay(1000);  // servo sağa tam döner
    mesafeolc();                             // mesafeyi ölçer
    int mesafesol=mesafe;                    // ölçtüğü mesafeyi hafızaya alır.
    delay(100);                              // 100ms bekler

    // burada, hafızada bulunan sağ ve sol mesafelerini karşılaştırır.
    // hangi alan boş ise robotun boş alana 90 derece dönmesini sağlar.

    if (mesafesag > mesafesol) {motorkontrol(-50,50); delay(650); motorkontrol(0,0);
        delay(50); turkrobokitservo.write(80); delay(500);}
    else {motorkontrol(50,-50); delay(650); motorkontrol(0,0);
        delay(50); turkrobokitservo.write(80); delay(500);}
    }

// bu fonksiyon geriye "mesafe" değerini döndürür.
int mesafeolc(){
    digitalWrite(TRIG_PIN, LOW);          // ses göndermeyi kapatır. 2us (mikrosaniye)
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);         // ses göndermeyi açar. 10us (mikrosaniye)
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);          // ses göndermeyi kapatır.
    sure = pulseIn(ECHO_PIN, HIGH ); // sesi dinler. mikrosaniye cinsinden bir değer alır.
    mesafe = (sure/2)/29,13;              // sesin gidip dönüşüne kadar geçen süreyi
                                          //ölçtükten sonra aradaki mesafeyi hesapladık.

    return mesafe;
}

```



```

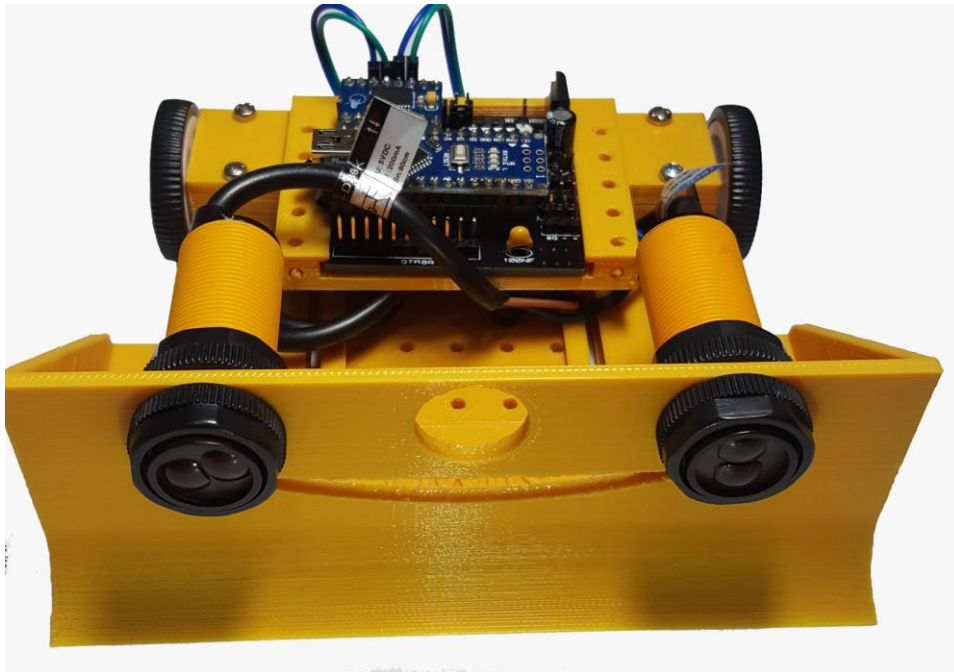
void motorkontrol(int solmotorpwm, int sagmotorpwm){
if(sagmotorpwm<=0) {
sagmotorpwm=abs(sagmotorpwm);
digitalWrite(sagmotor1, LOW);
digitalWrite(sagmotor2, HIGH);
analogWrite(sagmotorpwmpin, sagmotorpwm);
}
else {
digitalWrite(sagmotor1, HIGH);
digitalWrite(sagmotor2, LOW);
analogWrite(sagmotorpwmpin, sagmotorpwm);
}
if(solmotorpwm<=0) {
solmotorpwm=abs(solmotorpwm);
digitalWrite(solmotor1, LOW);
digitalWrite(solmotor2, HIGH);
analogWrite(solmotorpwmpin, solmotorpwm);
}
else {
digitalWrite(solmotor1, HIGH);
digitalWrite(solmotor2, LOW);
analogWrite(solmotorpwmpin, solmotorpwm);
}
}
}

```

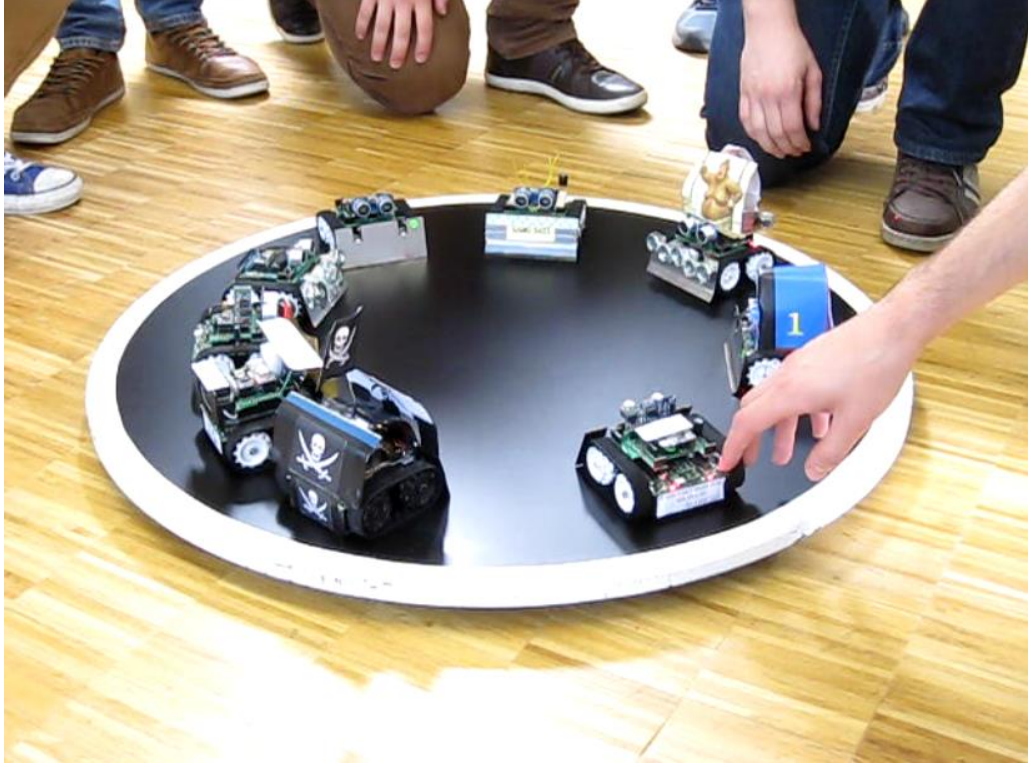
Robot videosuna aşağıdaki linkten ulaşabilirsiniz.

<https://www.youtube.com/watch?v=9FoorFeKgNw>

#### 7-) Turkrobokit sumo robot yapımı



Sumo robotlar, robotikte hobi olarak ilgilenenlerin Japon sumo güreşlerinden esinlenerek aynı güreşi robotlara yaptırmak istemeleriyle ortaya çıkmıştır. Sumo robotların güreştiği yuvarlak zemine dohyo adı verilir. Robotlar üzerlerindeki kontrast sensörleri sayesinde dohyonun çevresindeki beyaz çiziyi algılar, ring dışına çıkmamaya, ring içinde kalmaya çalışırlar. Ayrıca optik cisimden yansımali sensörleri ile rakibi algılar ve pist dışına itmeye çalışırlar.



Biz robotumuzda dohyo kenarındaki beyaz çizgiyi algılaması için QTR1A sensörleri ve rakibi algılamak için MZ80 sensörünü kullanacağız. Robotumuz her hangi bir rakip görene kadar sakın bir şekilde dohyo üzerinde dolanacaktır, sağ ve sol önünde, alt tarafta bulunan sensörlerle çizgiyi kontrol edecektir. Rakip gördüğünde atak moduna geçecek ve rakibini dohyo dışına itene kadar peşini bırakmayacaktır.

Kitimizde bağlantı oldukça kolaydır kodlara bakarak hangi sensörün hangi pine bağlı olduğunu kolaylıkla anlayabilirsiniz.

```

// SUMO ROBOT KODLUYORUZ //
// servo motoru olay bir şekilde kullanabilmek için servo.h kütüphanesini ekledik.
#define sagmotor1 10
#define sagmotor2 9
#define sagmotorpwmpin 11
#define solmotor1 6
#define solmotor2 7
#define solmotorpwmpin 5
#define solMZ 8
#define sagMZ 4
#define sagQTR A0
#define solQTR A1
#define button 2
#define taktik 12
#define led 13
int durum=6; int esik=100;

void setup() { // pinlerin giriş çıkış tanımlamalarını yapıyoruz
pinMode(sagmotor1,OUTPUT); //
pinMode(sagmotor2,OUTPUT); //
pinMode(solmotor1,OUTPUT); // motor pinleri çıkış olarak ayarlandı
pinMode(solmotor2,OUTPUT); //
pinMode(solMZ,INPUT); //
pinMode(sagMZ,INPUT); //
pinMode(button,INPUT_PULLUP); //
pinMode(taktik,INPUT_PULLUP); //
motorkontrol(0,0); // Motorlarımızı dinamik olarak frenliyoruz
Serial.begin(9600);

while(digitalRead(button)==HIGH) {motorkontrol(0,0); }//burası butona basılıncaya kadar frenler
bekle_5(); //5 Sn alt programını çalıştırır
}

void loop()
{
//
if(digitalRead(taktik)==HIGH) test(); // Arduino altındaki anahtar aktifken test alt programı çalışır
else // anahtar pasifken direk robotumuz rakip aramaya başlar
rakip_ara();
}
void rakip_ara(){
motorkontrol(150,150);
qtr_oku();
if(digitalRead(sagMZ)==LOW || digitalRead(solMZ)==LOW) {durum=0; atak(); }
}
void atak(){
while(durum<6) {
if(digitalRead(sagMZ)==LOW) durum=2;
if(digitalRead(solMZ)==LOW) durum=4;
if(digitalRead(sagMZ)==LOW && digitalRead(solMZ)==LOW) durum=3;
if(digitalRead(sagMZ)==HIGH && digitalRead(solMZ)==HIGH && durum==4) durum=5;
if(digitalRead(sagMZ)==HIGH && digitalRead(solMZ)==HIGH && durum==2) durum=1;

if(durum==3) motorkontrol(255,255); // rakip tam karşıda hızla üstüne gidilir
if(durum==2) motorkontrol(200,0); // hafif sağda, sağa doğru dönülür
if(durum==4) motorkontrol(0,200); // hafif solda, sola doğru dönülür
if(durum==5) motorkontrol(-200,200); // soldan kaybolursa tam sol döndürür
if(durum==1) motorkontrol(200,-200); // sağdan kaybolursa tam sağ döndürür
qtr_oku(); // zemin okumaya devam edilir pistin dışına çıkmamak lazım

}
}
}

```

```

void test(){
    digitalWrite(led,LOW);
    if(analogRead(sagQTR)<esik) {digitalWrite(led,HIGH); Serial.print(analogRead(sagQTR)); Serial.print("  ");}
    if(analogRead(solQTR)<esik) {digitalWrite(led,HIGH); Serial.print(analogRead(solQTR)); Serial.print("  ");}
    if(digitalRead(sagMZ)==LOW ) {digitalWrite(led,HIGH); Serial.print("Sağ MZ  ");}
    if(digitalRead(solMZ)==LOW) {digitalWrite(led,HIGH); Serial.print("Sol MZ  ");}
    Serial.println("");
    delay(10);

}

void qtr_oku(){
    if(analogRead(sagQTR)<esik && analogRead(solQTR)<esik ) gerigiti();
    if(analogRead(sagQTR)<esik) solgerigiti();
    if(analogRead(solQTR)<esik) saggerigiti();
}

void bekle_5(){
    digitalWrite(led,LOW); delay(500); digitalWrite(led,HIGH); delay(500);
    digitalWrite(led,LOW); delay(500); digitalWrite(led,HIGH); delay(500);
    digitalWrite(led,LOW); delay(500); digitalWrite(led,HIGH); delay(500);
    digitalWrite(led,LOW); delay(500); digitalWrite(led,HIGH); delay(500);
    digitalWrite(led,LOW); delay(500); digitalWrite(led,HIGH); delay(500);
    digitalWrite(led,LOW);
}

void gerigiti(){
    motorkontrol(-150,-150); delay(500);
    motorkontrol(150,-150); delay(750);
}

void saggerigiti(){
    motorkontrol(0,-150); delay(650);
    motorkontrol(150,120); delay(100);
}

void solgerigiti(){
    motorkontrol(-150,0); delay(650);
    motorkontrol(120,150); delay(100);
}

void motorkontrol(int solmotorpwm, int sagmotorpwm){
    if(sagmotorpwm<=0) {
        sagmotorpwm=abs(sagmotorpwm);
        digitalWrite(sagmotor1, LOW);
        digitalWrite(sagmotor2, HIGH);
        analogWrite(sagmotorpwmpin, sagmotorpwm);
    }
    else {
        digitalWrite(sagmotor1, HIGH);
        digitalWrite(sagmotor2, LOW);
        analogWrite(sagmotorpwmpin, sagmotorpwm);
    }
    if(solmotorpwm<=0) {
        solmotorpwm=abs(solmotorpwm);
        digitalWrite(solmotor1, LOW);
        digitalWrite(solmotor2, HIGH);
        analogWrite(solmotorpwmpin, solmotorpwm);
    }
    else {
        digitalWrite(solmotor1, HIGH);
        digitalWrite(solmotor2, LOW);
        analogWrite(solmotorpwmpin, solmotorpwm);
    }
}

```

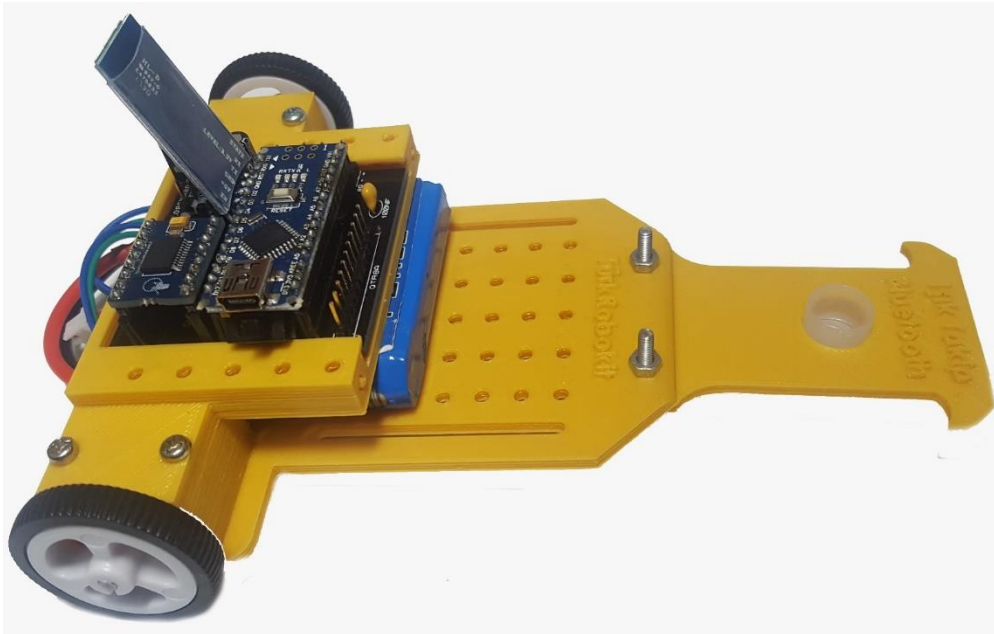
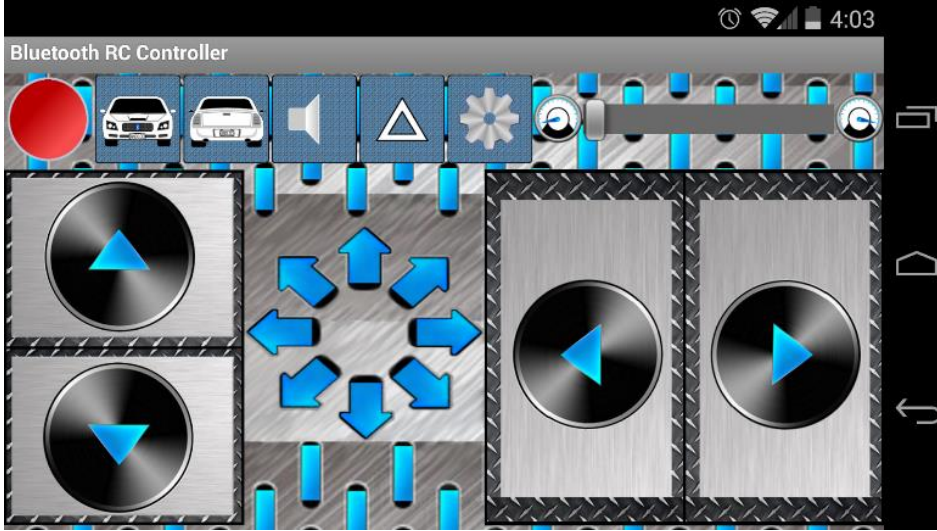


Sumo robot ile ilgili videoya ařağıdaki linkten ulaşabilirsiniz.

<https://www.youtube.com/watch?v=NTviX0mtqIY>

#### 8-) Turkrobokit bluetooth kontrollü robot yapımı

Bluetooth kontrollü araba yapabilmek için Temel robotik kartımızın bluetooth bağlantı noktasına HC06 modülü bağlanmalı ve Android tabanlı cihazınızla eşleştirilmelidir. Cihazımıza Arduino Bluetooth RC Car uygulamasını Google play üzerinden indirelim. Uygulamanın ayarlar menüsünde butonlar yazan butona tıkladığımızda hangi karakterlerin bluetooth üzerinden gönderildiğini inceleyebilirsiniz. Bu karakterler programımızın yazımında referans olacaktır. Sistemimiz gönderilen karakterlere bağlı olarak motorlara yön ve hız bilgisini vererek çalışacaktır. Konumuzun devamını kodlar üzerinde açıklamalar yaparak gerçekleştireceğiz.





```

// pin bağlantı tanımlamalarını yapıyoruz
#define sagmotor1 10
#define sagmotor2 9
#define sagmotorpwmpin 11
#define solmotor1 6
#define solmotor2 7
#define solmotorpwmpin 5
#define BUZZER 2
byte turbo=0; // robotumuza uygulamadaki dörtlü butonuna
                // bastığımızda ekstra hız vermek için turbo değişkenini tanımlıyoruz
char veri;      // bluetoothtan gelen verileri karakter olarak
                // almak için char tipi veri değişkenini tanımlıyoruz
void setup() { // pinlerin giriş çıkış tanımlamalarını yapıyoruz
pinMode(BUZZER,OUTPUT);
pinMode(sagmotor1,OUTPUT);
pinMode(sagmotor2,OUTPUT);
pinMode(solmotor1,OUTPUT);
pinMode(solmotor2,OUTPUT);
    Serial.begin(9600); // bluetooth ile 9600 baud hızında seri haberleşmeyi başlatıyoruz
    delay(1000); //1 saniye bekliyoruz

}
void loop()
{
while (Serial.available()) // Seri porttan iletişim müsait olduğunda
{
    veri=Serial.read(); // Seri porttan gelen bilgiyi okuyup veri değişkenine atıyoruz

    if(veri=='X') turbo=1; // gelen veri değeri X karakteri ise turboyu etkinleştiriyoruz
    if(veri=='x') turbo=0; // gelen veri değeri x karakteri ise turboyu pasifleştiriyoruz

    // gelen veri değeri F karakteri ise iki motoruda ileri yönde 100PWM hızında hareket ettiriyoruz.
    if(veri=='F')
    { motorkontrol(125,125);} // gelen verinin F olması ileri butonuna basıldığını gösterir

    // gelen verinin B olması geri butonuna basıldığını gösterir robot geri gider
    if(veri=='B')
    { motorkontrol(-60,-60);}

    // gelen verinin R olması sağa butonuna basıldığını gösterir robot sağa döner
    if(veri=='R')
    { motorkontrol(-60,60);}

    // gelen verinin L olması sola butonuna basıldığını gösterir robot sola döner
    if(veri=='L')
    { motorkontrol(60,-60);}

    // gelen verinin I olması sağ ileri butonuna basıldığını gösterir robot sağ ileri gider
    if(veri=='I')
    { motorkontrol(100,150);}

    // gelen verinin G olması sol ileri butonuna basıldığını gösterir robot sol ileri gider
    if(veri=='G')
    { motorkontrol(100,150);}
}
}

```

```

// gelen verinin J olması sol geri butonuna basıldığını gösterir robot sol geri gider
if(veri=='J')
{ motorkontrol(-100,-150);}

// gelen verinin H olması sağ geri butonuna basıldığını gösterir robot sağ geri gider
if(veri=='H')
{ motorkontrol(-150,-100);}

// gelen verinin S olması butonuna basılmadığını gösterir motorlar frenler
if(veri=='S')
{ motorkontrol(0,0);}

// gelen verinin V olması korna butonuna basıldığını gösterir buzzer çalıştırılır
if(veri=='V')
{ tone(BUZZER,300); }

// gelen verinin v olması kornanın pasif olduğunu gösterir buzzer susturulur
if(veri=='v')
{ noTone(BUZZER); }
}
}

void motorkontrol(int solmotorpwm, int sagmotorpwm){
if(sagmotorpwm<=0) {
sagmotorpwm=abs(sagmotorpwm);
digitalWrite(sagmotor1, LOW);
digitalWrite(sagmotor2, HIGH);
analogWrite(sagmotorpwmpin, sagmotorpwm);
}
else {
digitalWrite(sagmotor1, HIGH);
digitalWrite(sagmotor2, LOW);
analogWrite(sagmotorpwmpin, sagmotorpwm);
}
if(solmotorpwm<=0) {
solmotorpwm=abs(solmotorpwm);
digitalWrite(solmotor1, LOW);
digitalWrite(solmotor2, HIGH);
analogWrite(solmotorpwmpin, solmotorpwm);
}
else {
digitalWrite(solmotor1, HIGH);
digitalWrite(solmotor2, LOW);
analogWrite(solmotorpwmpin, solmotorpwm);
}
}
}

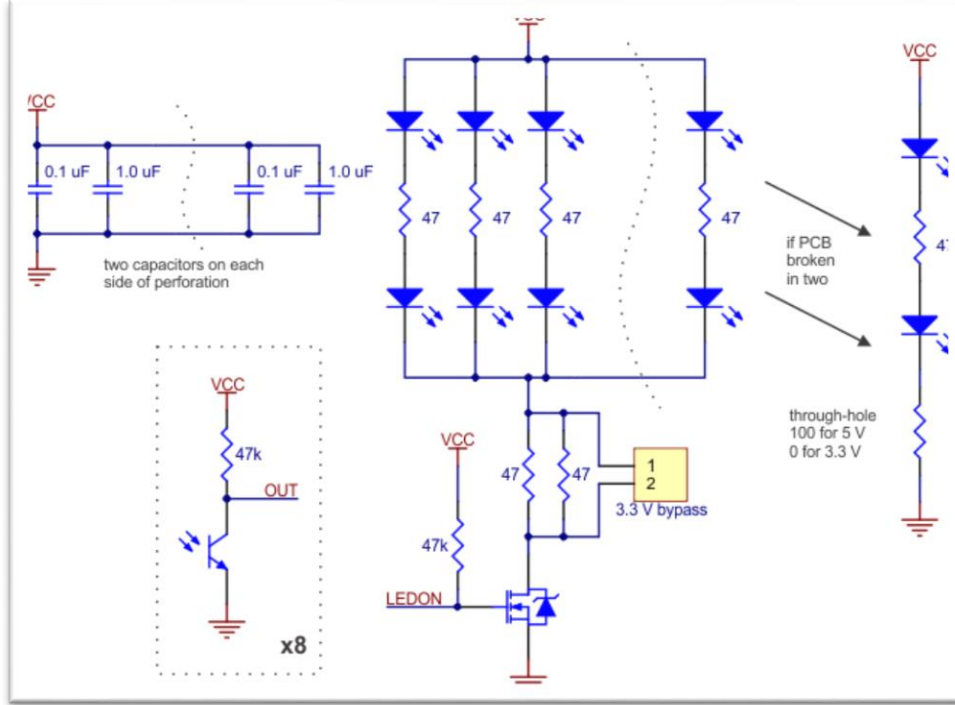
```

Robot ile ilgili videoya aşağıdaki linkten ulaşabilirsiniz.

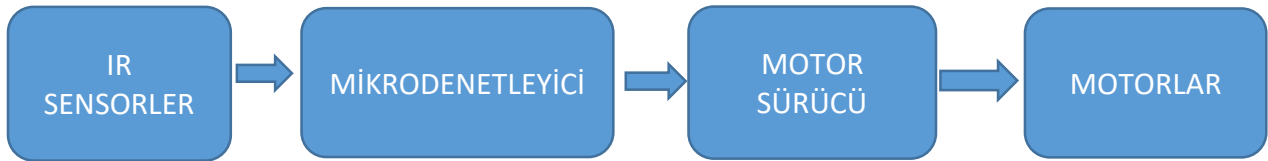
**<https://www.youtube.com/watch?v=LeH9IITPMXg>**

### 9-) Türkrobokit çizgi izleyen robot yapımı

Çizgi izleyen robotumuzu yapmaya başlamadan önce çalışma mantığını anlamamız gereklidir. Robotumuz Kızılötesi analog sensörler ile zemin üzerinde takip edeceği çizgiye göre 0-1024 arasında analog değerler üretir. Biz QTR8A sensör kullanacağız. Bu sensör üzerinde 8 adet IR alıcı ve verici barındırmaktadır. Sensör üzerinde bulunan IR modüller arası mesafe 95mm dir. Toplamda yaklaşık 8 cm alanda pozisyon ölçümü yapılabilir. Sensör üzerinde IR modüller beyaz zeminde 0 a yakın değerler verirken siyah zeminde de 1024'e yakın değerler verir. QTR8A sensörünü açık şeması alttaki şekilde verilmiştir.



Sensörlerden gelen veriler ya dijital 1 veya 0'lara çevrilir yada direk analog olarak mikrodenetleyici tarafından işlenir. Mikrodenetleyici ile robotumuzun ön sensör kısmının çizgiye göre pozisyonu hesaplanır. Hesaplanan değere göre motorlara uygulanacak hız değerleri bulunur. Mikrodenetleyicimizin pinleri motorları beslemek için yeterli olmadığından motor sürücü kullanılır. Motor sürücüyü hız ve yön bilgisi mikrodenetleyici tarafından iletilir. Motorlar hataya göre farklı hızlarda dönecek ve robotumuz çizgiyi takip edecektir. Çalışma mantığı ile ilgili işlem sırası alttaki blok diyagramda gösterilmiştir.

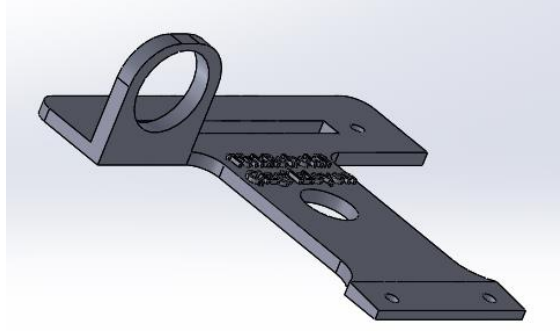


Türkrobokit setine göre robotumuzu toparlayacağız ve kodlamaları yapacağız. İsteyen arkadaşlar burada anlatacağımız bilgilerden faydalanarak kendi tasarlayacakları Arduino temelli kartlarına ve QTR8 sensörüne göre kolayca uyarlama yapabilirler.

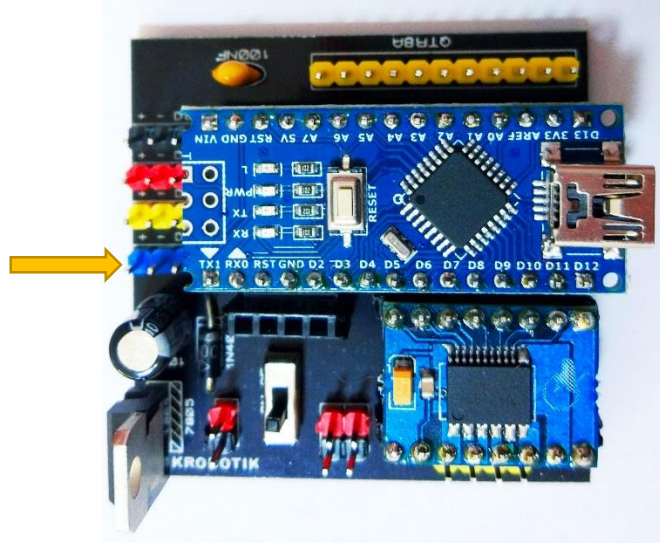
Çizgi izleyen robot yapımında kullanacak olduğumuz bileşenler aşağıdaki gibidir.

- TürkRobokit ana gövde
- TürkRobokit çizgi izleyen robot başlığı
- Türkrobokit kontrol kartı
- Mz80, qtr-8a
- 2s lipo pil
- 12mm 6v 600rpm dc motor

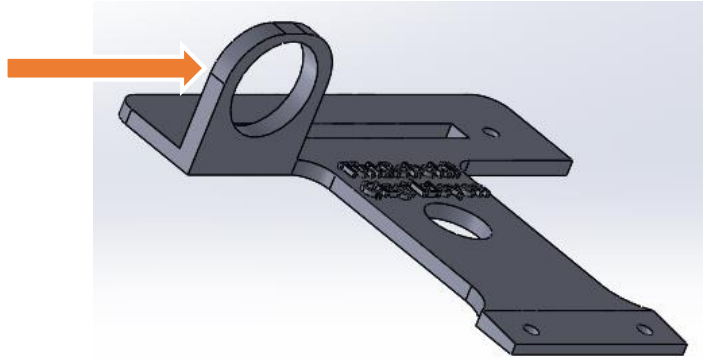
Eğitim kitimizle gelen bu donanımların nasıl birleştirildiğine bakalım. Çizgi izleyen robotumuzu kurabilmek için ana gövdeye çizgi izleyen başlığının bağlantısı yapılmalıdır.



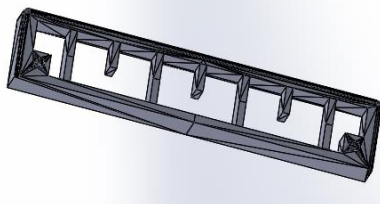
Mz80 cisim algılama sensörünü D2-D3-D4-D8 pinlerinden herhangi birine bağlayabiliyoruz. Başlangıç olarak D2 pinini tercih edelim. **Sensörler ve modüllerin tanıtımı** başlığı altında cisim algılama sensörümüzün pin yapısıyla ilgili bilgiler verilmişti. Buna göre mz80'in data ucunu (siyah veya sarı) D2 pinine, + ucunu (kahverengi veya kırmızı) kartın sensör girişi olan + pinine, - ucunu (mavi veya yeşil) kartın sensör girişi olan – pinine bağlıyoruz.



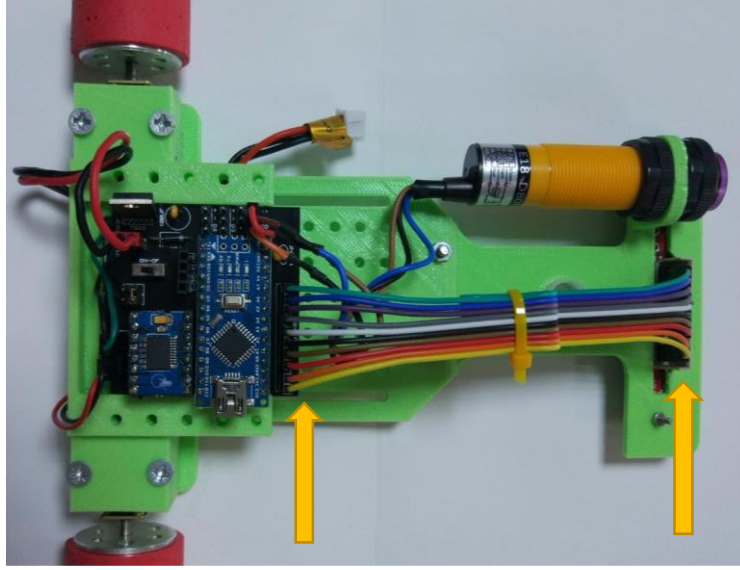
Daha sonra mz80 cisim algılama sensörünü çizgi izleyen başlığındaki yerine sıkıştırma vidalarıyla monte ediyoruz.



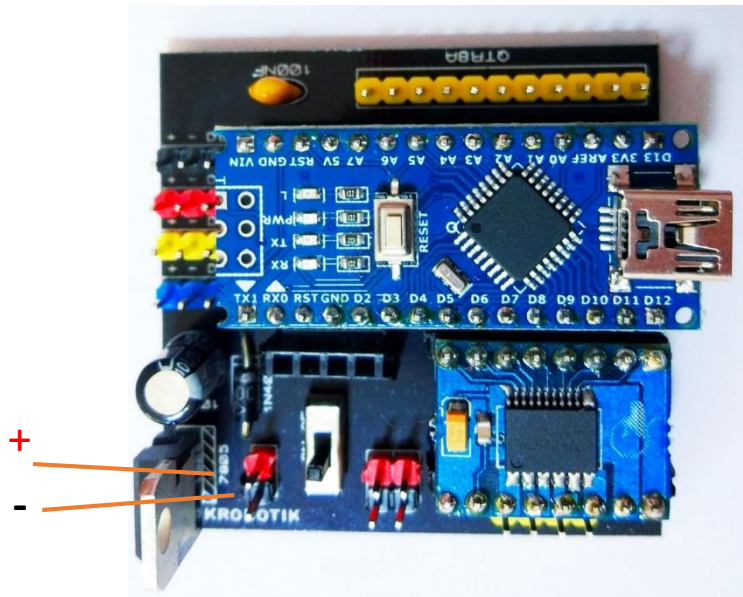
Bu işlem yapıldıktan sonra qtr-8a sensörünü çizgi izleyen başlığına monte etmemiz gerekmektedir. Çalışmalarımızda sensör zayıflığının önüne geçmek için qtr sensöre koruyucu bir kılıf tasarladık. Sensörümüzü bu kılıfa yerleştirdikten sonra m2 vidalarıyla çizgi izleyen kafaya monte ediniz.



Çizgi sensörünün bağlantısını kit ile beraber yolladığımız 20cm uzunluğundaki dişi dişi kablolar ile yapabilirsiniz. Dupont kablonun 11 adet pininden faydalanarak sensör ile TürkRobokit robot kartı arasındaki bağlantıyı alttaki şekle uygun olarak yapıyoruz. Sağ ve Sol motor kablolarımızı kartımız üzerindeki motor pinlerine bağlıyoruz.



Gerekli donanımlar bağlandıktan sonra ana gövdeye 2s lipo pilimizin TürkRobokit robot kartına bağlantısını yapıyoruz. Burada dikkat etmemiz gereken + ve - uçlarının doğru bağlanmasıdır.

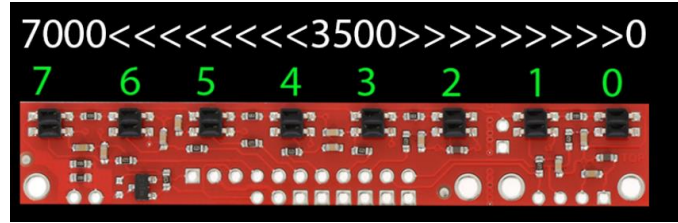




Tüm bağlantılar yapıldıktan sonra robot kartı üzerindeki arduino nano ya kod atarak robotunuzu denemeye başlayabilirsiniz.

Kodlama yaparken hemen hemen her yerde bulabileceğiniz temel yapıyı kullanacağız. Kodlamada izleyeceğimiz adımlar sırası ile

- Sabitler ve pinlere vereceğimiz isimler tanımlanacak
- QTR8A kütüphanesi ne göre parametreler yazılacak bağlı olduğu pinler tanımlanacak
- Motor sürücü ve diğer dijital pinlerimiz giriş veya çıkış olarak ayarlanacak
- Kalibrasyon kodlarımız yazılacak
- Sensör değerleri okunarak pozisyon hesabı yapılacak
- Pozisyon hesabına göre hata değerimiz bulunacak
- PID kontrol ile Motorlara uygulanacak düzeltme hızı bulunacak
- Motorlarımıza hesaplanan hız değerleri taban hızı eklenerek uygulanacaktır. Eğer hata değeri 0 ise iki motorda eşit hızda dönecektir. Hata değeri pozitif ise sağdaki motor hızlanacak, soldaki motor ise yavaşlayacaktır. Eğer hata değeri negatif çıkarsa tam tersi olacaktır. Tekrar e maddesine gidilerek sensör değerleri okunacak ve döngü tekrar başlayacaktır. E-F maddeleri programımızda loop içine yazılacak ve sürekli döngü halinde olacaktır.



Analog sensörlerimiz zeminden yaklaşık 7mm-8mm yukarıdan kızılötesi ışık göndermektedir ve zemin beyaz ise yansıma fazla siyah ise az olacaktır. Bu yansıma değerlerini direk olarak analog pinlerden okumaya kalkarsak siyahta 900 civarı beyazda ise 300 civarı bir değer görürüz ancak her sensörde bu değerler farklılık gösterir. Ayrıca pist etrafındaki ışık değerleri de olumsuz olarak parazit değerler üretir. Pistte kullanılan çizgiler ile zeminin mat veya parlak olması da değerleri değiştirir. Bu sebepten ortamdaki ışık miktarı ve zeminin durumuna göre örnekleme yapmalıyız. İşte tam burada sensörleri kalibre etmenin önemi ortaya çıkmaktadır. Örneğin 3 numaralı sensör en az 320 en fazla 860 değer versin burada 320 değerini 0 a 860 değerini de 1000 değerine normalize etmek gereklidir bu şekilde her sensör ile işlem yaparsak her sensörden beyazda 0 siyahta 1000 değeri alırız yani sensörlerin verdiği değerleri birbirine benzetiriz. Tabiki bu işlemi arka planda kütüphane gerçekleştirecek, sensör değerlerini normalize edecek ve ağırlıklı ortalama formülü ile çizginin hangi sensör üzerinde baskın olduğunu bulacaktır. Kütüphanedeki Readline komutu ile pozisyon değeri elde edilir

Çizgi izleyen robotumuzda pozisyon hesabı için QTRsensor kütüphanesini kullanacağımız belirtmiştik. Bu kütüphane çizgi üzerinde ölçüm yaparken 0-7000 arası bir değer üretir. Çizgi eğer sensörümüzün en sağ noktasında ise 0 en solda ise 7000 değeri üretir. Robotumuzun sensörü eğer çizgiyi tam ortalarsa 3500 değeri üretir. Bizim amacımız çizgi izlerken her zaman sensörü hep en ortada tutmaktır. Bu sebepten hesaplanan pozisyon değerinden 3500 değerini çıkaracağız. Yani tam ortada ise çizgi 0 hata alacağız tam sağda ise -3500 tam solda ise +3500 değeri elde edeceğiz. Yani yapacağımız ölçümlerde hata değerimiz +3500 ile -3500 arasında değişecektir. Bu hata değerini PID katsayıları ile işleme tabi tutarak motorlara vermemiz gereken hız değerlerini hesaplayacağız. PID de P oransal düzeltme uygular, I ise eski hataları toplayarak integral düzeltmesi yapar D ise türevsel düzeltme yapar. İlkel çizgi izleyenler sadece oransal düzeltme yaparak çalışıyordu bu durumda robot çizgi üzerinde salımlı olarak takip eder yani kafa sallar. Türev düzeltmesi eklendiğinde salınımlar azalır ve daha düzgün bir izleme yapar. Şayet robotumuz çizgiye yaklaşmakta zorlanıyor ve hata artıyorsa integral düzeltmesi ile hatalar toplanarak motorların tepkisi artırılabilir. Sonuçta sadece oransal hesap yeterli değildir. Bu sebepten P yerine PI, PD veya PID tercih edilir.

PID endüstride kontrol amaçlı kullanılan bir yöntemdir ve profesyonel sistemlerin çoğunda kullanılır. Örneğin bir drone düşünelim havada sürekli dengede kalabilmesi, hareket edebilmesi ve konumunu koruyabilmesi için sağlam bir algoritma gereklidir. Bu sebepten drone'larda motorların devir hızları PID ile hesaplanır. 3 Boyutlu yazıcılarda ısıtıcı tablanın sıcaklığı, nozul sıcaklığı gibi değerler sensörlerle sürekli ölçülür ve istenilen değerde kalması istenilir. Bu stabil değerler de PID ile yapılır. PID kullanımına daha bir çok örnek verilebilir.

Robotumuzda biz PID katsayılarını deneyerek bulduk. Tecrübelerimiz dayanarak bazı açıklamalarda bulunabiliriz. İyi bir çizgi izleyende ağırlık merkeziniz yere yakın olması ve iki motor arasına yakın olması istenir. Eğer motorlarınız torku yetersiz ise PID parametrelerinde katsayıları biraz artırmanın gereklidir. Aksi takdirde PID ile de kontrol zorlanır. Eğer kullanacağınız motorun torku ve tepkisi çok iyi ise katsayılar biraz düşürülmelidir. 4 tekerli çizgi izleyenlerde ufak katsayı farklılıkları pek etki göstermez ama iki tekerde oransa düzeltmeyi fazla

artırırsanız yine salınım yapar robotunuz. Biz oransal katsayı olarak  $K_p=0.05$  , türevsel katsayı olarak  $K_p=0.7$  ve integral olarak  $K_i=0.002$  değerini kullanıyoruz. Ancak kitimizde motor torku yarışmalarda kullanılan motorlara göre düşük olduğu için ve robotumuz biraz minik olduğundan kodlarda belirttiğimiz değerleri kullanacağız. Bu değerler robotun ağırlığı, genişliği, ağırlık merkezi robotun şekli, uzunluğu gibi faktörlere göre değişir. Umarım sizler kendinize en uygun değerleri kolaylıkla bulursunuz.

Bundan sonrasını kodlar ile gösterelim.

```
// QTR kütüphanesini ekliyoruz ve sabitlerimizi tanımlıyoruz
#include <QTRSensors.h>
#define tabanhiz 80
#define sagmotor1 10
#define sagmotor2 9
#define sagmotorpwmpin 11
#define solmotor1 6
#define solmotor2 7
#define solmotorpwmpin 5
#define MZ80 8
#define taktik 12
#define LED 13

// QTR8A sensörümüzü bağlı olduğu pinleri tanımlayıp
//sensörlerden gelen veriler için sensors dizisi oluşturuyoruz
QTRSensorsAnalog qtra((unsigned char[]) { A0, A1, A2, A3, A4 , A5, A6, A7} , 8);
unsigned int sensors[8];

// Değişkenlerimizi tanımlıyoruz
float Kp = 0.04;   float Kd = 0.6;   float Ki = 0.0001;   int integral = 0;
int ekhiz = 0;     int sonhata = 0;   int hata = 0;   byte K=0;
int sagmotorpwm = 0;   int solmotorpwm = 0;   unsigned char zemin = 1;
int çizgisay=0;       long baslamazamani=0;   long doksanzamani=50000;

void setup()
{
    // Dijital olarak kullanacağımız pinlerin giriş veya çıkış olarak tanımlıyoruz
    pinMode(sagmotor1, OUTPUT);
    pinMode(sagmotor2, OUTPUT);
    pinMode(sagmotorpwmpin, OUTPUT);
    pinMode(solmotor1, OUTPUT);
    pinMode(solmotor2, OUTPUT);
    pinMode(solmotorpwmpin, OUTPUT);
    pinMode(MZ80, INPUT);
    pinMode(taktik, INPUT_PULLUP);
    delay(1000); // kalibrasyon başlamadan ek süre koyuyoruz
    //Kalibrasyon kodları  arduino üzerindeki led yanık olduğu
    //sürece devam eder
    digitalWrite(LED, HIGH);
}
```

```

// Kalibrasyonun otomatik gerçekleşmesi robotumuza hafif sağ sol kafa sallama yaptırıyoruz
for (int i = 0; i < 150; i++)
{
    if ( 0 <= i && i < 5 )    hafifsagadon();
    if ( 5 <= i && i < 15 )   hafifsoladon();
    if ( 15 <= i && i < 25 )   hafifsagadon();
    if ( 25 <= i && i < 35 )   hafifsoladon();
    if ( 35 <= i && i < 40 )   hafifsagadon();
    if ( 45 <= i && i < 50 )   hafifsoladon();
    // Kalibrasyon döngüsü 50yi aştıktan sonra frenliyoruz bu esnada robotu elle de kalibre edebiliriz
    if ( i >= 50 ) {
        frenle(1);
        delay(3);
    }
    qtra.calibrate();
    delay(1);
}
// kalibrasyon döngüsünden çıkıldığını anlamak için Arduinoda bulunan yerleşik lede flaş yaptırıyoruz
// Bu esnada robotumuzun çizgi sensörünü çizgiye ortalıyoruz
flashyap();
// başlangıçta engel varsa durup bekler. MZ80 sensörü takılı olmalıdır
while (digitalRead(MZ80) == LOW) {frenle(1);}
//Serial.begin(9600);
}

...

void loop() {
    sensoroku();
    //Serial.println(hata); delay(50);

    // motorlara verilecek hız düzeltme oran hesabı PID
    integral += hata; //çizgiden uzaklaştıkça hataları toplar
    if (abs(hata) < 500) integral = 0;
    int duzeltmehizi = Kp * hata + Kd * (hata - sonhata) + Ki * integral;
    sonhata = hata;

    // Motorlara uygulanacak kesin hız ayarları
    sagmotorpwm = tabanhiz + duzeltmehizi + ekhiz ;
    solmotorpwm = tabanhiz - duzeltmehizi + ekhiz ;
    //*****

    // motorlara hız ayarlarının uygulanması
    // Burada motorlara uygulanacak PWM değerlerine sınırlandırma getirilmiştir.
    sagmotorpwm = constrain(sagmotorpwm, -50, 254);
    solmotorpwm = constrain(solmotorpwm, -50, 254);
    motorkontrol(solmotorpwm, sagmotorpwm);
}

```

```

void motorkontrol(int solmotorpwm, int sagmotorpwm){
    if(sagmotorpwm<=0) {
        sagmotorpwm=abs(sagmotorpwm);
        digitalWrite(sagmotor1, LOW);
        digitalWrite(sagmotor2, HIGH);
        analogWrite(sagmotorpwmpin, sagmotorpwm);
    }
    else {
        digitalWrite(sagmotor1, HIGH);
        digitalWrite(sagmotor2, LOW);
        analogWrite(sagmotorpwmpin, sagmotorpwm);
    }
    if(solmotorpwm<=0) {
        solmotorpwm=abs(solmotorpwm);
        digitalWrite(solmotor1, LOW);
        digitalWrite(solmotor2, HIGH);
        analogWrite(solmotorpwmpin, solmotorpwm);
    }
    else {
        digitalWrite(solmotor1, HIGH);
        digitalWrite(solmotor2, LOW);
        analogWrite(solmotorpwmpin, solmotorpwm);
    }
}

void frenle(int bekle){motorkontrol(0,0); delay(bekle);}

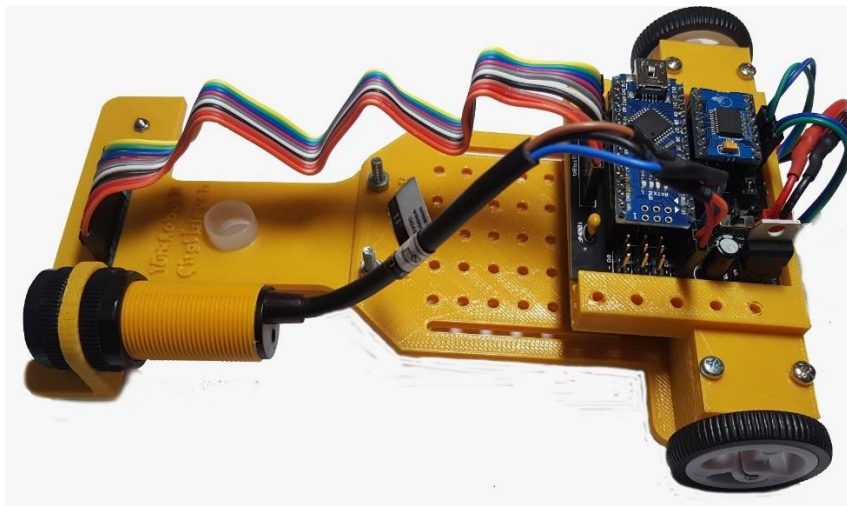
void hafifsagadon(){motorkontrol(60,-60);}

void hafifsoladon(){motorkontrol(-60,60);}

void sensoroku(){
    //Çizgi sensörü pozisyon hesap kodları
    unsigned int position = qtra.readLine(sensors,1,zemin);
    hata= position-3500;
}

void flashyap(){
    for (int x = 0; x < 10; x++) {
        digitalWrite(LED, HIGH); delay(200);
        digitalWrite(LED, LOW); delay(200);
    }
}

```

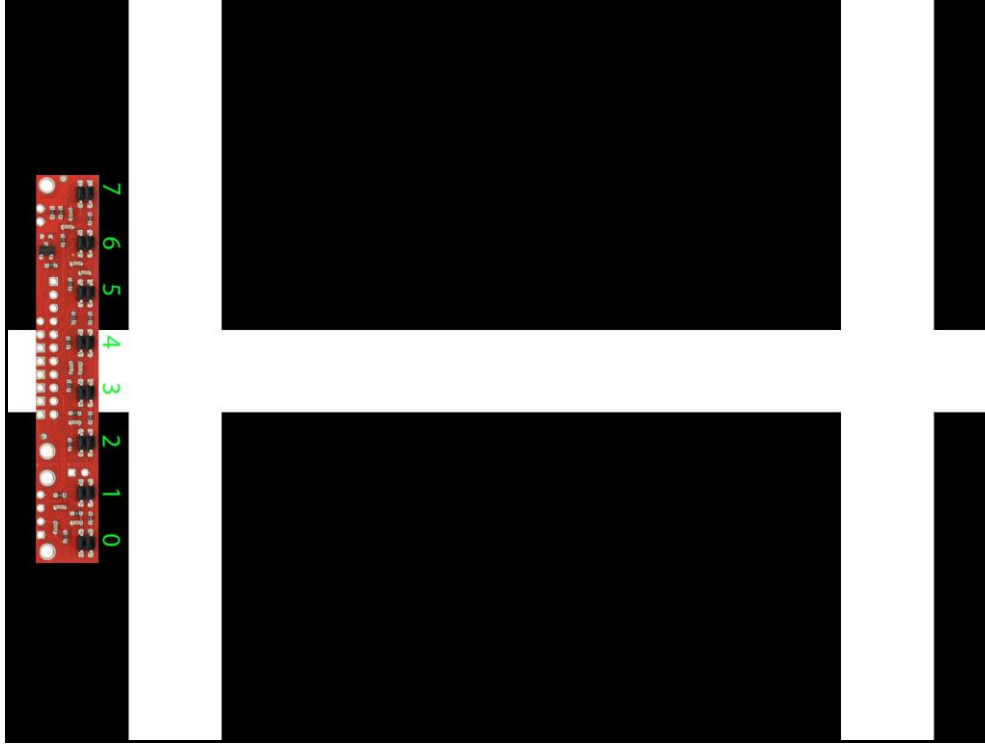


## 10-) Turkrobokit çizgi izleyen algoritmaları

Bundan sonrasında üstteki kodları yazdığınız varsayılarak kodların tamamını yazmayacağız sadece ilgili kısımlar yazılacaktır.

### a) İki çizgi arası hızlanma

Çizgi sensörümüzün beyazda 0 a yakın değerler alacağını belirtmiştik. Eğer tüm sensörler 0 a yakın değer alırsa robotumuzun QTR sensörü beyaz çizgi üzerinde demektir. İlk kodlarımızda çizgisay adından integer bir değişken oluşturmuştuk ve başlangıç değerini 0 olarak atamıştık. Robotumuz birinci çizgiyi gördüğünde çizgisay değerini bir artıracak ve 1 değerini alacaktır. Başlangıç değeri 0 olan ekhiz değişkenini çizgi sayısı 1 olduğu sürece yüksek bir değer atayabiliriz. Robotumuz ikinci bir çizgiye geldiğinde ekhiz değerini tekrar 0 yaparak robotumuzu eski yavaş hızına döndürebiliriz.



Bu işi yapabilmek için kodlarımızın en alt satırından başlayarak dikcizgioku adında bir alt program yazalım.

```
void dikcizgioku() {  
    if (sensors[0]<100 && sensors[1]<100 && sensors[2]<100 && sensors[3]<100 &&  
        sensors[4]<100 && sensors[5]<100 && sensors[6]<100 && sensors[7]<100 ) {  
        cizgisay++; motorkontrol(tabanhiz,tabanhiz); delay(50);  
    }  
}
```

Void Loop içinde ise bazı basit değişiklikler yapacağız.

```
void loop() {  
    sensoroku();  
    dikcizgioku();  
    if(cizgisay==1) ekhiz=50; else ekhiz=0;  
    if(cizgisay==2) cizgisay=0;  
    //Serial.println(hata); delay(50);  
}
```

Kodlarda görüldüğü gibi dikcizgioku() alt programımızı çağırıyoruz eğer tüm sensörler 150 den küçük yani 0 a yakın değer verirse çizgisay değerini bir artırıyoruz. Çizgisay değeri 1 olduğunda robotumuzun ortalama hızını artırmak için taban hızlara ek olarak ekhiz değişkeni kadar ekliyor ve hızımızı 50PWM artırıyoruz. Bu arada önceden bahsetmedik PWM konusunu ama 50PWM demek yaklaşık pilden gelen gerilimi %20 daha fazla motora vermek demektir. Motorlara en fazla 255PWM verilir en az da 0 PWM. 255 PWM tam güç aktarımı 0 ise dinamik frenleme demektir. Çizgi sayısı 2 olduğunda ise ekhizi tekrar 0 layarak ortalama hızımızı düşürüyoruz. Biz sürekli



kapalı bir döngü içinde robotumuzun turladığını düşünerek iki çizgi arasında sürekli hızlanmasını istediğimiz için ikinci çizgiden sonra çizgisay değerine 0 lama yapıyoruz ki tekrar çizgi gördüğünde robot gazlasın.

Bu mantıkla robotumuzu istersek 2. Çizgide frenleyebiliriz. Kartımız üzerindeki motor sürücünün dinamik frenleme özelliği var yani 0 PWM verdiğimizde motor uçlarını kısa devre ederek frenlemeyi artırır. Aksi halde robot kayarak serbest durma yapar. Yani 0 PWM = ABS fren gibi ☺

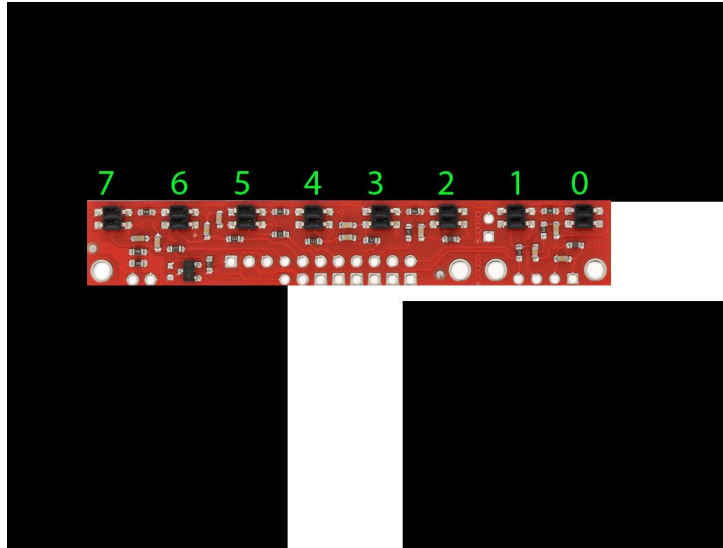
Gelin bir de çizgiye göre 10000ms yani 10sn frenleyelim.

```
void loop() {  
    sensoroku();  
    dikcizgioku();  
    if(cizgisay==1) ekhiz=50; else ekhiz=0;  
    if(cizgisay==2) {cizgisay=0; frenle(10000);}!  
    //Serial.println(hata); delay(50);  
}
```

#### b) 90 derece dönüşler

Alttaki resimden de anlaşılacağı üzere robotumuz sağ tarafa 90 derecelik bir çizginin üzerine gelmiş. 0-1-2-3 ve 4. sensörler beyazda 5-6 ve 7. sensörlerde siyah üzerinde bu durum hangi tarafa 90 derece dönüş olduğunu anlamamız için yeterlidir. Robotumuzun çizgiyi tam olarak ortalamaması ihtimali olsa bile 0-1-2-3. Sensörler 0'a yakın değerler görürken 6-7'inci sensörler siyahta olduğu için 1000'e yakın değerler döndürür. Sol tarafa 90 derece dönüş var ise bu durumda sensörlerin göreceği değerlerde simetrik olarak değişecektir.

Normalde robotumuz yavaş gidiyorsa 90 derece geçişlerde özel bir kod yazmaya gerek yoktur. Zaten standart kodlarımızda robotumuzun sensörü çizgi dışına çıktığı anda bir önceki konumu hatırlar tabiki kütüphanede arka planda gerçekleşir. Alttaki resme bakarak şunu söyleyebiliriz sağdaki sensörler beyaz üzerinde olduğu için robotun ilerlemesinden dolayı sensör boşa çıkınca çizgi algılayamaz ve pozisyon değerini 0 olarak döndürür. Hata formülümüzde 0-3500 den hatamızı -3500 buluruz. Sola dönüşte ise +3500 hata alırız.



Verilen bilgilerden faydalanarak şartımızı kolaylıkla yazabiliriz ve robotumuza sağa 90 dereceyi gördüğünde önce yavaşlamasını sağlayacağız ardından sağdaki çizgiyi yakalayana kadar kavis çizdireceğiz. Bundan sonrasını kodlarımız üzerinden açıklayalım.

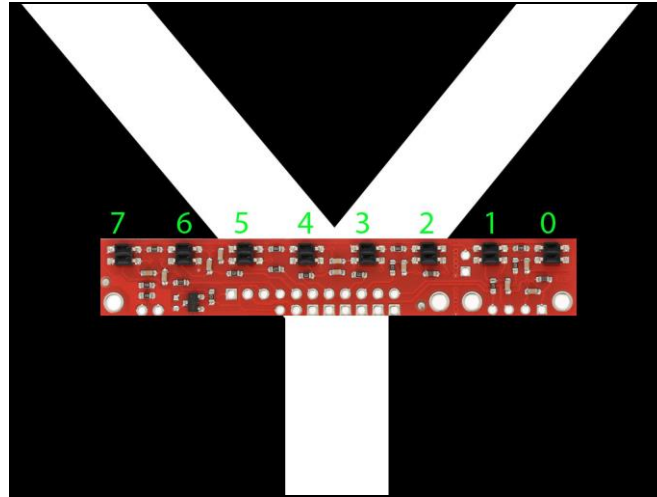
```

void sag_sol_90() {
  // Sağa 90 derece var ise şartımız alttaki şekilde olacaktır
  if (sensors[0]<100 && sensors[1]<100 && sensors[2]<100 && sensors[3]<100 && sensors[6]>500 && sensors[7]>700 ){
    // motorkontrol(-100,-250); delay(25); eğer hızlı bir motor kullanılıyorsa aktif edin
    doksanzamani=millis(); // 90 derece geçişlerde zaman bilgisini alıyoruz
    // robotumuzun ön sensör boşa çıktığında hata -3500 olur ve hata -3500 olduğunca
    // sol motoru ileri sağ motoru geri çevirip sensör okumaya devam ediyoruz
    do { sensoroku(); motorkontrol(200,-50); }
    while(hata==3500);
  }
  // Sola 90 derece var ise şartımız alttaki şekilde olacaktır
  if (sensors[7]<100 && sensors[6]<100 && sensors[5]<100 && sensors[4]<100 && sensors[1]>500 && sensors[0]>700 ){
    // motorkontrol(-250,-100); delay(25); eğer hızlı bir motor kullanılıyorsa aktif edin
    doksanzamani=millis(); // 90 derece geçişlerde zaman bilgisini alıyoruz
    // robotumuzun ön sensör boşa çıktığında hata 3500 olur ve hata 3500 olduğunca
    // sağ motoru ileri sol motoru geri çevirip sensör okumaya devam ediyoruz
    do { sensoroku(); motorkontrol(-50,200); }
    while(hata==3500);
  }
}

```

### c) Yol ayırımında karar verme

Yol ayırımında resimden de görüldüğü üzere 7 ve 0'ıncı sensörler siyahta yani 1000 e yakın değerler gösterirken 4 ve 3' üncü sensörler de tam beyazda olduğu için 0 a yakın değerler üretir. 5 ve 2'inci sensörler 3 ve 4 kadar olmasa da düşük değerler üretirler. Bu bilgiler bizim yol ayırımı için şartımız olacak.



Eğer yol ayırımı şartı gerçekleşirse kısa bir süreliğine çizgi okumadan robotumuza sağ veya sol tarafa yönlendireceğiz devamında tekrar çizgi okumaya başladığında yoluna devam edecektir. yol\_ayrimi() adında yeni bir alt program yazıp ve alt programımızı void loop() içinde çağıracağız.

Yol ayırımı alt programına K adında bir değişken ekleyeceğiz. K değeri 0 a eşitse yol ayırımında robotumuz sola dönecek 1 e eşitse sağa dönecek. Bundan sonrasını kodlarla göstereyim.

```

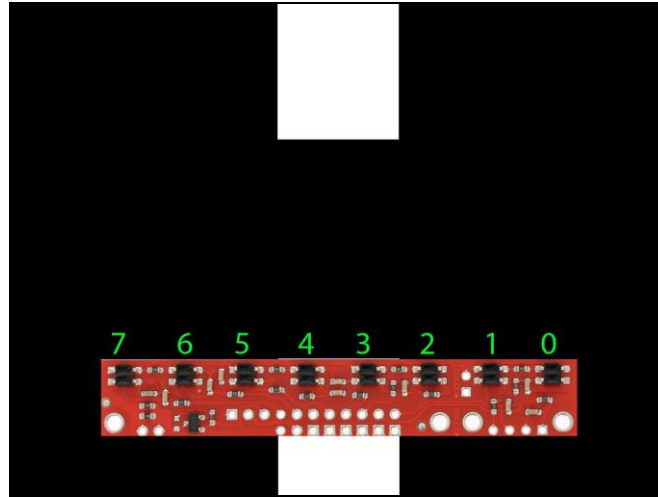
void yol_ayrimi() {
    // Eğer yol ayrımı var ise şartımız alttaki şekilde olacaktır
    if (sensors[0]>800 && sensors[2]<600 && sensors[3]<100 && sensors[4]<100 && sensors[5]<600 && sensors[7]>800 ){
        if(K==0) {
            // Sol tarafa dönmek için
            motorkontrol(80,180); delay(300);
        }

        if(K==1) {
            // Sağ tarafa dönmek için
            motorkontrol(180,80); delay(300);
        }
    }
}

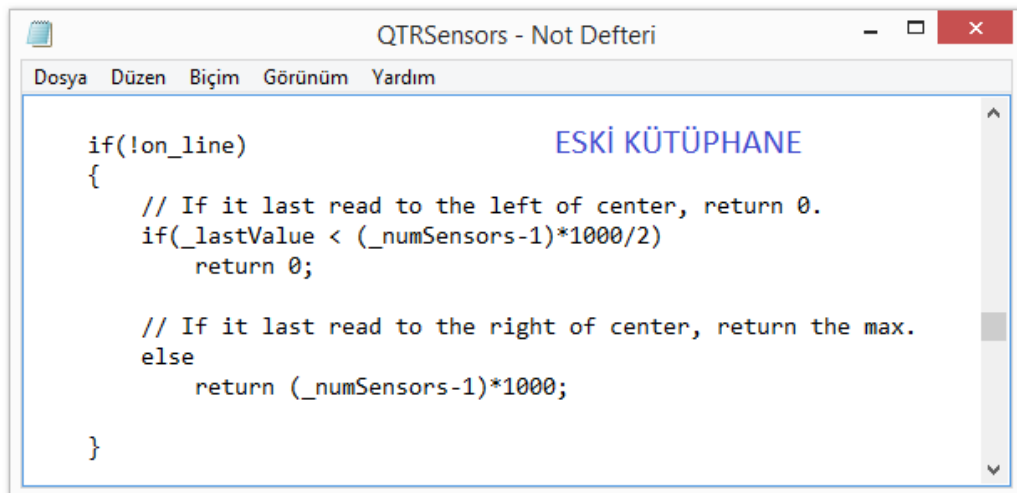
```

#### d) Kesik çizgilerden geçme

QTR kütüphanesi kullanıp kesik çizgilerden atlamak oldukça zordur en azından ben 3 günde çözmüştüm meseleyi. Kesiklerden atlamak için sensörün önceki durumu çok önemlidir. Yani çizgiyi nizami takip ettiğimizi varsayarak sensör boşa çıkmadan önce robotumuz -500 ile +500 arasında bir hata üretir. Önceki hata -500<hata<+500 arasında ise şimdi de çizgi yok ise düz devam et demeliyiz.



Biz bu durumu kütüphaneye müdahale ederek çözeceğiz. Kütüphane dosyaları içindeki QTRSensors.cpp dosyasını açacağız ve aşağıda resim olarak gösterdiğim değişikliği yaptığınızda artık kesiklerden otomatik geçer.



```
QTRSensors - Not Defteri
Dosya Düzen Biçim Görünüm Yardım

MODİFİYE KÜTÜPHANE

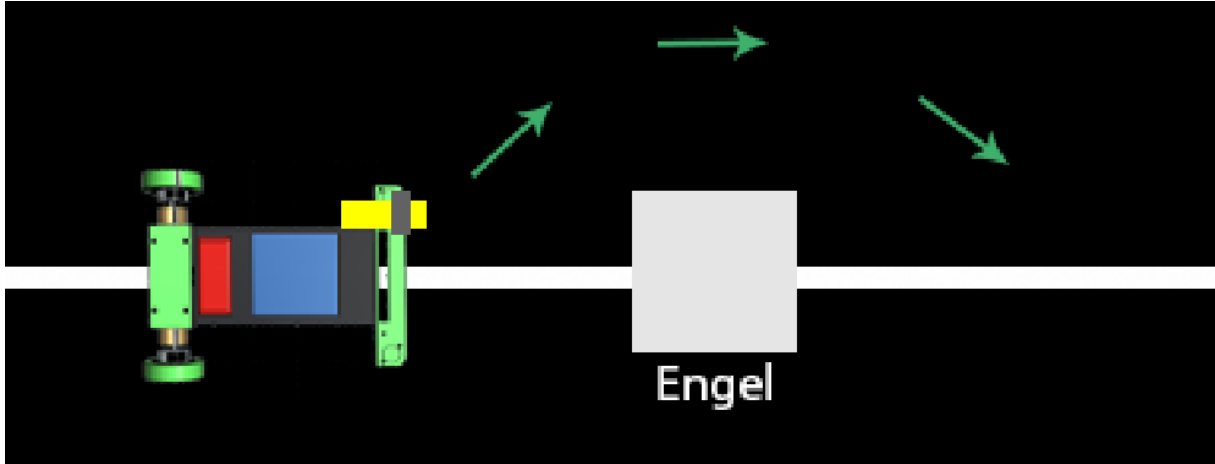
if(!on_line)
{
    // If it last read to the left of center, return 0.
    if(_lastValue < (_numSensors-2)*1000/2)
        return 0;

    if(_lastValue > (_numSensors)*1000/2)
        return (_numSensors-1)*1000;
    if(_lastValue > (_numSensors-2)*1000/2 && _lastValue <
    (_numSensors)*1000/2)
        return (_numSensors-1)*1000/2;
}
}
```

#### e) Engelden atlama

Engelden atlama veya engel görünce durmak oldukça basit ve kolay bir işlem. Engeli algılamak için genelde optik cisim algılama sensörleri kullanılır. Biz robotumuzda oldukça kolay bulunabilen MZ80 sensörünü kullanacağız. Bu sensör önüne engel geldiğinde LOW bilgisi yani 0V sinyal verir eğer önü açıksa HIGH yani 5V sinyal verir. Arkasında minik düz tornavida ile mesafe ayarı yapabileceğimiz trimpot bulunur.

Bu sensörler önündeki engel rengi beyaz olursa bayağı uzaktan algılayabilirler. Engelin rengi koyulaştıkça algılama mesafesi düşer. Mesafe ölçümü yapılması uygun değil sadece engel var veya yok bilgisi alırsınız. Neyse gelelim asıl meseleye, engelin solundan atlayacağımızı düşünelim. MZ80 engel gördüğünde robotumuzu öncelikle sol tarafa yönlendirmemiz lazım ardından biraz ileri gideceğiz sonra robotumuzu sağ tarafa yönlendirip çizgiyi bulana kadar düz götüreceğiz.



```

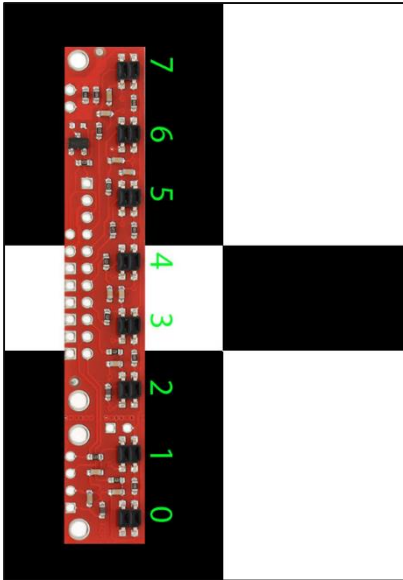
void engeldenatla(){
  if(digitalRead(MZ80) == LOW) {

    motorkontrol(0,150); delay(300); // önce sol motora fren verip sağ ileri yapıyor ve sola yönleniyoruz
    motorkontrol(150,150); delay(200); // biraz ileri gidiyoruz
    motorkontrol(150,0); delay(250); // sağ motora fren verip sol ileri yapıyor ve sola yönleniyoruz
    motorkontrol(150,150); delay(800); // biraz ileri gidiyoruz
    motorkontrol(150,0); delay(100); // biraz daha sağ yöne robotumuzu yönlendiriyoruz
    do { sensoroku(); motorkontrol(150,120); } //çizgi görene kadar kavis yaptırıyoruz
    while(hata==0);
  }
}

```

#### f) Zemin değiştirme

Zemin değiştiğinde sensörlerden alınan verilerde ters döner. Kütüphanemiz arka planda işlem yaparken whiteline adındaki bir değişkenin true yada false olması durumuna göre hesaplamaları yapar. Zemin değiştirmek için kütüphanedeki whiteline değişkenini 1 yaparsak beyaz çizgiye, 0 yaparsak siyah çizgiye göre pozisyon hesabı yapmaktadır ancak pist üzerinde zeminde değişiklik olursa kütüphaneye müdahale edemeyiz. Bu durumda kodlarımızda zemin adında bir değişken yardımıyla parametre ekledik.



Soldaki çizgi sensörünü incelediğimizde 7 ve 0'ıncı sensörler 1000 e yakın değerler görmektedir. Robotumuzdan beyaz zemin siyah çizgiye geçtiğinde ise 0 a yakın değerler almaktayız. Bu durumda zemin değişkenimizin değerini 7 ve 0'ıncı sensörlere bakarak değiştirebiliriz.

Sensör oku alt programın gerekli değişiklikleri yapalım.

```

void sensoroku(){
  //Çizgi sensörü pozisyon hesap kodları
  unsigned int position = qtra.readLine(sensors,1,zemin);
  hata= position-3500;
  // zemin değiştirme kodları
  if ( sensors[0]<100 && sensors[7]<100 ) { zemin=0; } //beyaz
  if ( sensors[0]>750 && sensors[7]>750 ) { zemin=1; } //siyah
}

```

Kodlardan da görüldüğü gibi robotumuz otomatik olarak zemin değiştirecektir. Belki dik beyaz veya siyah çizgi ile karşılaşır ne olur diye aklınıza gelebilir. Robotumuzun önceki hata değeri +500 ile -500 arasında olduğunda dikine bir çizgi ile karşılaşır hata değerini zemin

renginden bağımsız olarak 0 yapar. Bu durumda dik çizgi geçişlerinde zemin değişse de problem olmaz.

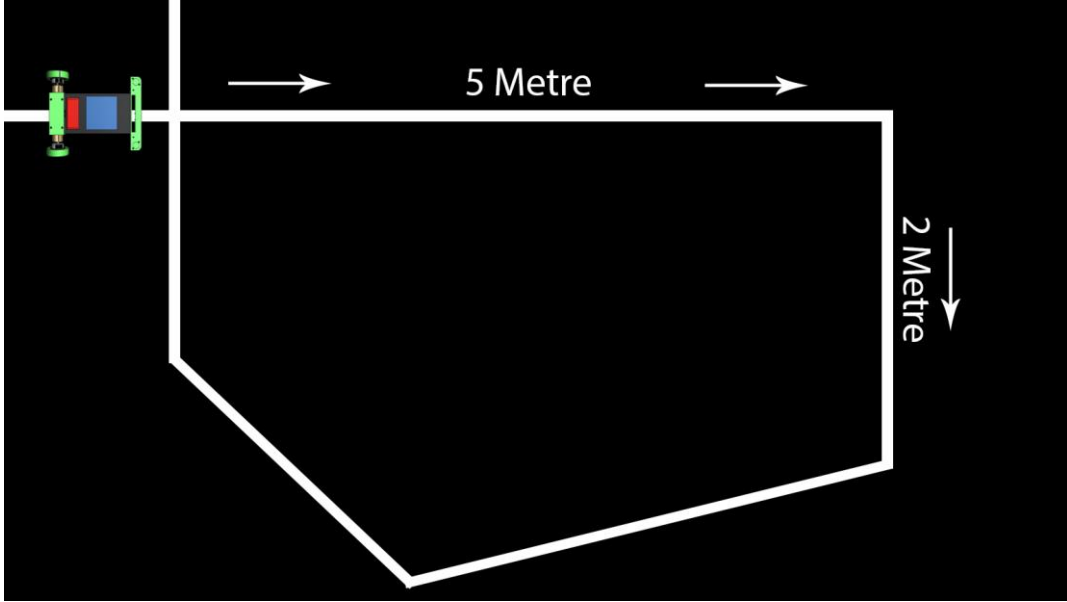
#### g) Zamana ve referans noktalarına bağımlı olarak hızlanma;

Sizlere küçük bir anımı paylaşmak istiyorum. 2017 MEB robot yarışması çizgi izleyen kategorisinde iyi bir ekibin danışmanı olan öğretmen arkadaşımınla birlikte yarışma pistinin benzeri bir pistte çalışma yapıyorduk. Bizim robotlar beyaz zemin üzerinde bulunan 90 derecelik çizgiye uğramadan direk kavis alıyordu. Robotumuzun aldığı kavis ile 200-300 ms kadar süremiz kısılıyordu. Siyah zeminden beyaza geçişte referans alıyorduk bu durumu gören kıymetli arkadaşımız PIC tabanlı robotunda aynı şekilde kodlamaya başladı ve şu cümleyi kurdu **“Timer’ım bitti”** :D .

Bildiğiniz gibi bizim kartımızda Arduino nano var. Arduino da bize göre altın değerinde bir fonksiyon var millis(). Bu fonksiyonu o kadar kullanmaya alışmıştık ki Timer’ın bitmesi diye bir kavram yoktu bizim için.

Millis() fonksiyonu Arduino çalışmaya başladıktan itibaren geçen süreyi milisaniye cinsinden bize verir. Bu durumda pistte bulunan engel yol ayrımı kasis, kesik çizgi yani farklı olan her ne varsa o andaki süreyi şartlarımız içine millis() yazarak bulabiliriz. Kronometrelerde bulunan tur zamanlama gibi düşünebilirsiniz olan her olayın süresi alıyorsunuz ama zaman akmaya devam ediyor. Zamanı almanın ne faydası olabilir diye aklınızdan geçebilir bunu bir örnekle açıklayalım. Pistimizde çizgilerin dik kesiştiği bir nokta olsun. O bölgeden geçerken robotumuz çizgisiz alt programı ile çizgiyi algılar. Kesişim noktasından sonra 5m düz bir alan ve 5m sonunda 90 derece bir geçiş ardından 2m daha düz alan olsun.





Robotumuzda motorlara 255PWM verildiğinde ortalama hızı 4m/sn gittiğini, normal zamanlarda ise 100PWM ile 1,5m/sn ortalama hızımız olduğunu varsayalım. Çizgisay alt programı içine cizgizamani=millis(); yazarsak tam kesişim noktasının süresini alabiliriz. Bu süre örneğin 12300ms olsun. Önümüzde 5m düzlük ve ardından 90 derece geçiş var. 1 sn boyunca tam hız gidip ardından hızımızı düşürür, 90 dereceyi geçtikten sonra 1,5m boyunca tam gaz yapıp hızımızı stabil bir değere çekebiliriz. Nasıl olacak dersiniz küçük bir kod satırı yazalım şıp diye anlayacaksınız.

```
if ( millis()-cizgizamani > 0 && ( millis()-cizgizamani < 1000 ) tabanhiz=255;  
if ( millis()-cizgizamani > 1000 && ( millis()-cizgizamani < 1500 ) tabanhiz=60  
if ( millis()-cizgizamani > 1500 && ( millis()-cizgizamani < 1800 ) tabanhiz=255  
if ( millis()-cizgizamani > 1800 ) tabanhiz=100
```

Bu kod satırlarını ana döngümüzün içine atarak robotumuzu zamana göre hızlanmasını ayarlayabiliriz. Kodları incelersek kesişim noktasının zaman bilgisi olan 12300ms değerini cizgizamani değişkeni içine attık. Robotumuzda zaman ilerlemeye devam ediyor. Kodlarımızdaki birinci şarta bakılırsa 12300-13300 ms arası taban 255 oldu ve robotumuz 4m hızla ilerliyor, 13300-13800ms arasında taban 60 pwm e düştü ve robotun hızı son 1m içinde azaldı, 90 dereceyi döndü, 13800-14100 ms arasında tekrar 4m hıza çıktı ve 14100ms sn den sonraki her zamanda taban 100pwm oldu robotumuz 1,5m hızla yoluna devam etti.

Bu mantıkla referans noktalarınızın zaman bilgisi alıp alt alta if blokları açarak hızlanma değerlerini sıralarsınız. Robotumuzun hızı PWM le ayarlanıyor ancak Li-Po pil gerilim değeri robot çalıştıkça düşecektir. Bu sebepten dolayı robotunuzu her piste koyduğunuzda aynı yerde hızlanıp aynı yerde frenleyemeyebilir. Bu durumu önlemek için XL6009 step-up modül kullanmanızı şiddetle tavsiye ederiz. Yaklaşık 30m lik bir pistte 4 motorlu robotumuzla 5 defaya kadar sorunsuz bir şekilde aynı noktalarda hızlanıp aynı noktalarda yavaşlayabildik.

#### h) Pistte yarışma zamanı:

##### Pist Kuralları;

- Robot başlama çizgisinden hareket yönünde gitmelidir,
- Engele çarpmadan etrafından atlayıp tekrar yolu bulmalıdır,
- Yol ayrımında 1 nolu yolu tercih edip beyaz zeminden geçmelidir
- Kesik çizgileri geçerek başlama çizgisine gitmelidir
- Tekrar hareket yönünde 2. Tura başlamalıdır
- Engelden atlayarak tekrar yolu bulmalıdır
- Yol ayrımında 2 nolu yolu seçerek bitiş çizgisinde durmalıdır.



```

void setup()
{
    // Dijital olarak kullanacağımız pinlerin giriş veya çıkış olarak tanımlıyoruz
    pinMode(sagmotor1, OUTPUT);
    pinMode(sagmotor2, OUTPUT);
    pinMode(sagmotorpwmpin, OUTPUT);
    pinMode(solmotor1, OUTPUT);
    pinMode(solmotor2, OUTPUT);
    pinMode(solmotorpwmpin, OUTPUT);
    pinMode(MZ80, INPUT);
    pinMode(taktik, INPUT_PULLUP);
    delay(1000); // kalibrasyon başlamadan ek süre koyuyoruz
    //Kalibrasyon kodları arduino üzerindeki led yanık olduğu
    //sürece devam eder
    digitalWrite(LED, HIGH);

    // Kalibrasyonun otomatik gerçekleşmesi robotumuza hafif sağ sol kafa sallama yaptırıyoruz
    for (int i = 0; i < 150; i++)
    {
        if ( 0 <= i && i < 5 ) hafifsagadon();
        if ( 5 <= i && i < 15 ) hafifsoladon();
        if ( 15 <= i && i < 25 ) hafifsagadon();
        if ( 25 <= i && i < 35 ) hafifsoladon();
        if ( 35 <= i && i < 40 ) hafifsagadon();
        if ( 45 <= i && i < 50 ) hafifsoladon();
        // Kalibrasyon döngüsü 50yi aştıktan sonra frenliyoruz bu esnada robotu
        // Arduino üzerindeki led flash yapana kadar elle de kalibre edebiliriz
        if ( i >= 50 ) {
            frenle(1);
            delay(3);
        }
        qtra.calibrate();
        delay(1);
    }
    // kalibrasyon döngüsünden çıkıldığını anlamak için Arduinoda bulunan yerleşik lede flaş yaptırıyoruz
    // Bu esnada robotumuzun çizgi sensörünü çizgiye ortalıyoruz
    flashyap();
    // başlangıçta engel varsa durup bekler. MZ80 sensörü takılı olmalıdır
    while (digitalRead(MZ80) == LOW) {frenle(1);}
    // Serial.begin(9600);

}

void loop() {
    sensoroku();
    //sensorlerioku_yaz();
    dikcizgioku();
    yol_ayrimi();
    sag_sol_90();
    engeldenatla();

    if(cizgisay==1 || cizgisay==3 ) ekhiz=100; else ekhiz=0;
    if(cizgisay==3) K=1;
    if(cizgisay==5) {frenle(10000);}

    //90 dereceden alınan zamana göre hızlanma komutlarımız
    if(200<(millis()-doksanzamani) && (millis()-doksanzamani)<1200 ) ekhiz=100;
    if(1200<(millis()-doksanzamani) && (millis()-doksanzamani)<2400 ) ekhiz=-30;
    if(2400<(millis()-doksanzamani) && (millis()-doksanzamani)<5000 ) ekhiz=100;

```

```

// motorlara verilecek hız düzeltme oran hesabı PID
integral += hata; //çizgiden uzaklaştıkça hataları toplar
if (abs(hata) < 500) integral = 0;
int duzeltmehizi = Kp * hata + Kd * (hata - sonhata) + Ki * integral;
sonhata = hata;

// Motorlara uygulanacak kesin hız ayarları
sagmotorpwm = tabanhiz + duzeltmehizi + ekhiz ;
solmotorpwm = tabanhiz - duzeltmehizi + ekhiz ;
//*****

// motorlara hız ayarlarının uygulanması
// Burada motorlara uygulanacak PWM değerlerine sınırlandırma getirilmiştir.
sagmotorpwm = constrain(sagmotorpwm, -50, 254);
solmotorpwm = constrain(solmotorpwm, -50, 254);
motorkontrol(solmotorpwm, sagmotorpwm);
}

void engeldenatla(){
  if(digitalRead(MZ80) == LOW) {

    motorkontrol(0,150); delay(300); // önce sol motora fren verip sağ ileri yapıyor ve sola yönleniyoruz
    motorkontrol(150,150); delay(200); // biraz ileri gidiyoruz
    motorkontrol(150,0); delay(250); // sağ motora fren verip sol ileri yapıyor ve sola yönleniyoruz
    motorkontrol(150,150); delay(800); // biraz ileri gidiyoruz
    motorkontrol(150,0); delay(100); // biraz daha sağ yöne robotumuzu yönlendiriyoruz
    do { sensoroku(); motorkontrol(150,120); } //çizgi görene kadar kavis yaptırıyoruz
    while(hata==0);
  }
}

void sag_sol_90(){
  // Sağa 90 derece var ise şartımız alttaki şekilde olacaktır
  if (sensors[0]<100 && sensors[1]<100 && sensors[2]<100 && sensors[3]<100 && sensors[6]>500 && sensors[7]>700 ){
    // motorkontrol(-100,-250); delay(25); eğer hızlı bir motor kullanılıyorsa aktif edin
    doksanzamani=millis(); // 90 derece geçişlerde zaman bilgisini alıyoruz
    // robotumuzun ön sensör boşa çıktığında hata -3500 olur ve hata -3500 olduğunca
    // sol motoru ileri sağ motoru geri çevirip sensör okumaya devam ediyoruz
    do { sensoroku(); motorkontrol(200,-50); }
    while(hata==3500);
  }
  // Sola 90 derece var ise şartımız alttaki şekilde olacaktır
  if (sensors[7]<100 && sensors[6]<100 && sensors[5]<100 && sensors[4]<100 && sensors[1]>500 && sensors[0]>700 ){
    // motorkontrol(-250,-100); delay(25); eğer hızlı bir motor kullanılıyorsa aktif edin
    doksanzamani=millis(); // 90 derece geçişlerde zaman bilgisini alıyoruz
    // robotumuzun ön sensör boşa çıktığında hata 3500 olur ve hata 3500 olduğunca
    // sağ motoru ileri sol motoru geri çevirip sensör okumaya devam ediyoruz
    do { sensoroku(); motorkontrol(-50,200); }
    while(hata==3500);
  }
}

void yol_ayrimi(){
  // Eğer yol ayrımı var ise şartımız alttaki şekilde olacaktır
  if (sensors[0]>800 && sensors[2]<600 && sensors[3]<100 && sensors[4]<100 && sensors[5]<600 && sensors[7]>800 ){
    if(K==0) {
      // Sol tarafa dönmek için
      motorkontrol(80,180); delay(300); }
    if(K==1) {
      // Sağ tarafa dönmek için
      motorkontrol(180,80); delay(300); }
  }
}

```

```

void sensoroku(){
    //Çizgi sensörü pozisyon hesap kodları
    unsigned int position = qtra.readLine(sensors,1,zemin);
    hata= position-3500;
    // zemin değiştirme kodları
    if ( sensors[0]<100 && sensors[7]<100 ) { zemin=0; } //beyaz
    if ( sensors[0]>750 && sensors[7]>750 ) { zemin=1; } //siyah
}

void flashyap(){
    for (int x = 0; x < 6; x++) {
        digitalWrite(LED, HIGH); delay(200);
        digitalWrite(LED, LOW); delay(200);
    }
}

void dikcizgioku(){
    if (sensors[0]<100 && sensors[1]<100 && sensors[2]<100 && sensors[3]<100 &&
        sensors[4]<100 && sensors[5]<100 && sensors[6]<100 && sensors[7]<100 ){
        cizgisay++; motorkontrol(tabanhiz,tabanhiz); delay(50);
    }
}

void sensorlerioku_yaz(){
    for (unsigned char z = 0; z < 8; z++)
    {
        Serial.print(sensors[z]);
        Serial.print('\t');
    }
    Serial.println();
    delay(250);
}

void motorkontrol(int solmotorpwm, int sagmotorpwm){
    if(sagmotorpwm<=0) {
        sagmotorpwm=abs(sagmotorpwm);
        digitalWrite(sagmotor1, LOW);
        digitalWrite(sagmotor2, HIGH);
        analogWrite(sagmotorpwmpin, sagmotorpwm);
    }
    else {
        digitalWrite(sagmotor1, HIGH);
        digitalWrite(sagmotor2, LOW);
        analogWrite(sagmotorpwmpin, sagmotorpwm);
    }
    if(solmotorpwm<=0) {
        solmotorpwm=abs(solmotorpwm);
        digitalWrite(solmotor1, LOW);
        digitalWrite(solmotor2, HIGH);
        analogWrite(solmotorpwmpin, solmotorpwm);
    }
    else {
        digitalWrite(solmotor1, HIGH);
        digitalWrite(solmotor2, LOW);
        analogWrite(solmotorpwmpin, solmotorpwm);
    }
}

void frenle(int bekle){motorkontrol(0,0); delay(bekle);}

void hafifsagadon(){motorkontrol(60,-60);}

void hafifsoladon(){motorkontrol(-60,60);}

```

Çizgi izleyen robot ile ilgili videoya aşağıdaki linkten ulaşabilirsiniz.

[https://www.youtube.com/watch?v=Xl\\_dF8ljz00](https://www.youtube.com/watch?v=Xl_dF8ljz00)



## **HAZIRLAYANLAR**

### **Giresun GRobotik Robot Takımı**

**Bulancak Mesleki ve Teknik Anadolu Lisesi  
&  
Giresun Mesleki ve Teknik Anadolu Lisesi**

**Teşekkür Ederiz...**

**SENJU**  
Mini Sumo Robot  
Profesyonel Sumo Robot Kiti

**YENİ ÜRÜN**

Yedek Parça | Kolay Kurulum | Profesyonel Tasarım

Ürünü İncele ▶



**Rapid Arduino Çizgi İzleyen Robot Kiti**

Arduino Çizgi İzleyen Robot Kitimizle Tanışın

  **Bluetooth®**  
SMART

**HEMEN İNCELE**

Proje Ödevleri ve Robot Yarışmaları için Uygundur



**ROBOT KİTLERİ**

Mobil Robot Kiti  
Mini Sumo Robot Kiti  
Makeblock Robot Kiti  
Çizgi İzleyen Robot Kiti

**ÜRÜNLERİ İNCELE**

