

ELE 301 Proje Raporu

Osman Altınsoy, Muhammet Furkan Yıldız
Dept. of Electrical and Electronics Engineering
TOBB University of Economics and Technology
Ankara, Turkey
o.altinsoy@etu.edu.tr, mfyildiz@etu.edu.tr

Abstract—Bu proje de, tasarlanan kontrolcü ile sistemin girilen referans değerine karşı tepki vermesi ve istenilen değere oturması farklı yöntemler ile sağlanmıştır.

Index Terms—Transfer fonksiyonu, Arduino, Devre kurulumu

I. INTRODUCTION

Bu proje görev olarak bir depo ve bir bardak üzerindeki su seviyesini kontrol edebilmemiz istenmiştir. Bardak üzerinde meydana gelen değişimleri, tekrar sistemi referans düzeyine oturtacak şekilde; Açı-kapa, P, PI, PD ve PID kontrol yöntemleri kullanılarak gerçekleştirdik.

Projenin temel iki bölümü olan fiziksel tasarım ve kod yazma kısmı raporda açıklanmakta olup en son genel sonuçlar aktarılmaktadır.(Not: Çekilen video linki en sonda verilmiştir.)

II. FİZİKSEL TASARIM

Projeyi Arduino Mega kartı üzerine gömerek gerçekleştirdik. Tek yöne çalışan iki pompa kullandık. Bir tanesi bardaktaki referans seviyesi az ise tanktan doldurmak için, bir tanesi de bardaktaki referans değeri fazla ise tanka boşaltmak için. Bunun yanında pompalara gidecek voltajı ayarlamak için 2 adet npn transistör ve voltajının hızlı değişmesinin pompa içindeki salınımını azaltmak için iki adet kondansatör kullanılmıştır. Kullanıcının istediği modda sistemi kullanabilmesi için 3 adet button kullanılmıştır. Kullanıcı tarafından girilen referans değerini ölçmek için potansiyometre ve son olarak pompaları çalıştırabilmek için 1 adet pil yuvası kullanılmıştır.

1) Buttonlar: Burada Fig.1 üzerinden de görülebileceği üzere 3 adet buton kullanarak 2'lik tabandaki sayı karşılıkları ile mod değişimi gerçekleştirdik. Buttonları kontrol etmek için Arduino kartı üzerinde 22, 24 ve 26 pinlerini kullandık.

- Button3 = 0, Button2 = 0, Button1 = 0 Boş modda. (Pompaların ikisi de çalışmıyor)
- Button3 = 0, Button2 = 0, Button1 = 1 için Açı-Kapa kontrolcü.
- Button3 = 0, Button2 = 1, Button1 = 1 için P kontrolcü.
- Button3 = 0, Button2 = 1, Button1 = 1 için PI kontrolcü.
- Button3 = 1, Button2 = 0, Button1 = 0 için PD kontrolcü.
- Button3 = 1, Button2 = 0, Button1 = 1 için PID kontrolcü.

Olasak şekilde ayarlama yapılmıştır.

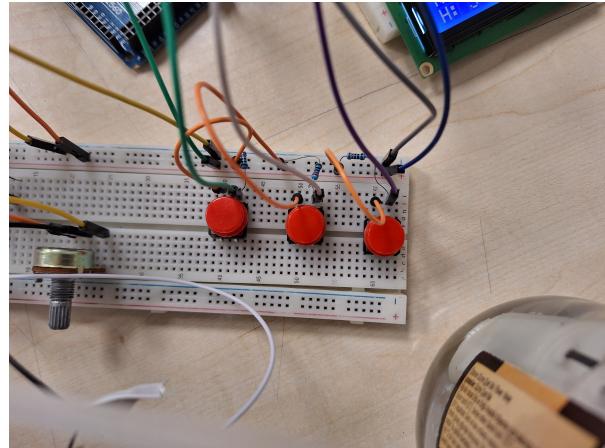


Fig. 1. Butonların devredeki kurulumu hali

2) Potansiyometre: Burada Fig.2 üzerinden görülebileceği üzere potansiyometre kullanılmıştır. Potansiyometrenin kullanım amacı kullanıcının girdiği referans değeri analog girişler üzerinden okuyarak sistem içinde gerekli ayarlamayı yapmaktadır. Burada değişken araliktaki değerler fark edilebilip kullanılabilir. Bizim amacımız 0-20 cm arasındaki değişimi kullanıcaya bildirerek istenilen referans değerinin ayarlanabilmesi sağlanmıştır. (Uyarı: Referans değeri ultrasonik sensörün pompa borusunu görmesinden dolayı 2 cm' in altında mantıklı çalışmamaktadır.)

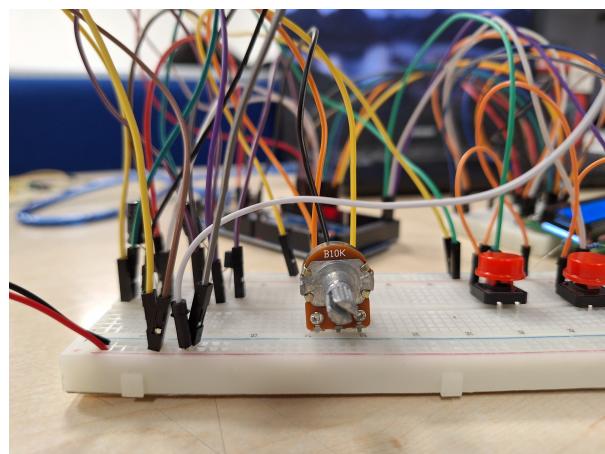


Fig. 2. Potansiyometrenin devredeki kurulumu hali

3) *Popmapalar*: Burada Fig.3 üzerinden pompalar ve o pompaların çalışmasını kontrol edebilmek adına kurulumış devreyi gözlemliyoruz. Devrede npn türünde transistör kullanılmıştır. Transistörün base bacağından voltaj değeri vererek emitter ve collector bacaklarının birbirleri arasında bağlantı olmasını, dolayısıyla 6V'luk kaynaktan orantılı mikrarda voltaj çekilmesini sağlanmıştır. Çekilen voltaj değeri pompanın çalışma hızını etkilemektedir. Bu durum P, PI, PD, PID kontrolcüler ile sistem kontrolü kısmında kullanılacaktır.

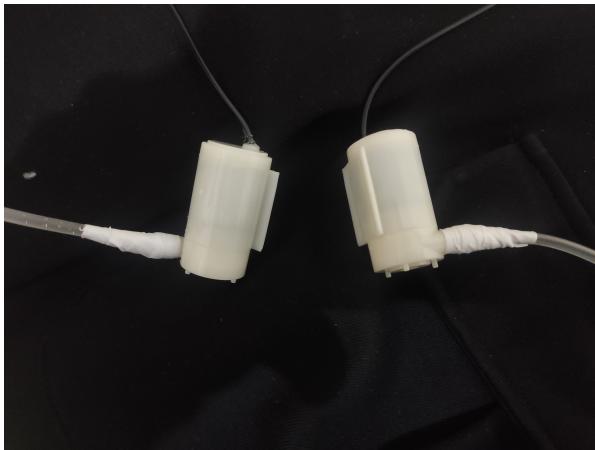


Fig. 3. Projede kullanılan pompalar

4) *Güç kaynağı*: Burada Fig.4 üzerinden de görüleceği üzere, 4 adet 1.5V'luk pil kullanılarak pompaların ihtiyaç duyduğu voltaj değeri karşılanmaktadır.

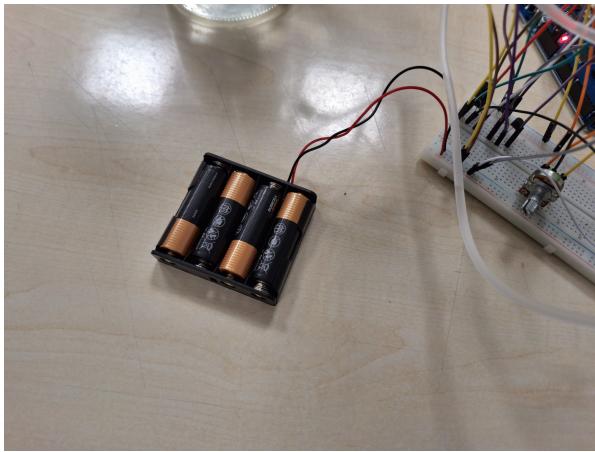


Fig. 4. Pompaları çalıştmak için kullanılan güç kaynağı. (6VV)

5) *LCD ekran modülü*: Burada Fig.5 üzerinden de görüleceği üzere LCD modül kullanılarak kullanıcının, sistemin hangi durumda ve girdi değerinin hangi değerde olduğunu görmesi sağlanmıştır. Yani kullanıcıya arayüz sağlamak için kullanılmıştır.

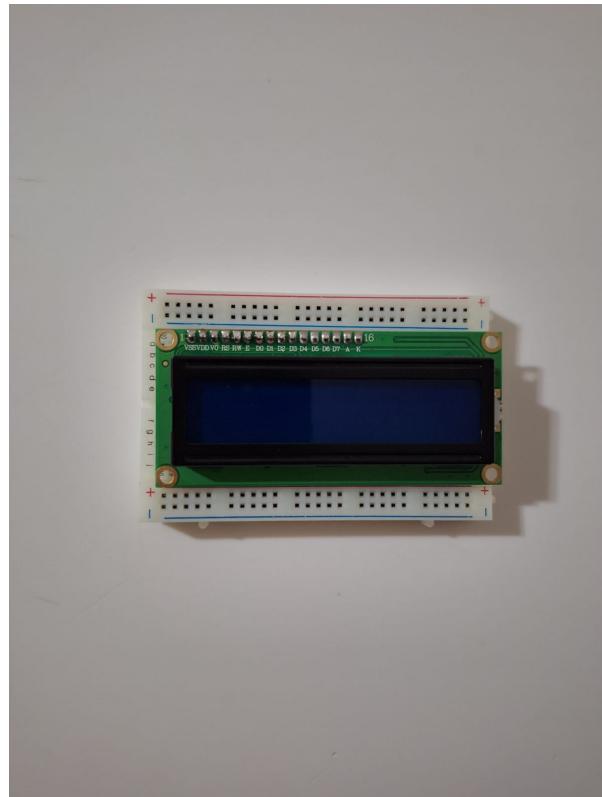


Fig. 5. Kullanıcıya arayüz sağlayan LCD ekran

6) *Ultrasonik sensör*: Burada Fig.6 üzerinden de görüleceği üzere ultrasonik sensör kullanılmıştır. Bu sayede suyun o anki yükseklik değeri hesaplanmaktadır ve gerçek değer olarak programda kullanılmaktadır.



Fig. 6. Su seviyesinin ölçülmesini sağlayan ultrasonik sensör

III. KOD YAZIMI

Bu kısımda kullandığımız karttan (Arduino Mega) dolayı kodu, C dilinde yazarak projeyi gerçekleştirdik. Teknik olarak bakarsak fiziksel tasarımda anlatılan her aşamanın kod yazma

kısımında da bir alanı var. O sebeple onlardan başlayarak kod yazımımları anlatalım.

1) Butonlar: Bu kısımda kontrolcü seçiminin sağlama amacıyla breadboard üzerine eklenilmiş olan butonlar için Arduino üzerinden kullanılacak olan pinlerin ataması, pinlerin input olarak tanımlanması, okunan pinler üzerinden işlemler yapabilmek için input değerlerinin tanımladığımız buton durumu değişkenlerine atanması, buton değeri konbinasyonlarına bağlı olarak kontrolcü seçiminin yapılmasını sağlayan kod kısmı yazılmıştır. Aşağıda bu işlemler ilgili kod kısımları ile daha detaylı şekilde gösterilemektedir.

Öncelikle aşağıdaki kod ile sırasıyla 22 24 ve 26. pinler Buton1 Buton2 ve Buton3 olarak tanımlanmıştır.

```
#define Buton1 22
#define Buton2 24
#define Buton3 26
```

Butonlar üzerinden okuma yapılacağından dolayı Buton1 (22) Buton2 (24) ve Buton3(26) pinleri void setup kısmında input olarak tanımlanmıştır.

```
pinMode(Buton1, INPUT);
pinMode(Buton2, INPUT);
pinMode(Buton3, INPUT);
```

Daha sonra ise tanımladığımız buton durumu değişkenlerine okuduğumuz Buton1 Buton2 ve Buton3 değerlerinin ataması yapılmıştır.

```
buton1_durumu = digitalRead(Buton1);
buton2_durumu = digitalRead(Buton2);
buton3_durumu = digitalRead(Buton3);
```

Butonlarla ilgili son olarak 7 numaralı figürde gösterilen kod kısmı yazılmıştır. Burada butonlara basılmadığı durumda (000 durumu) motorların çalışmasını engellenmiştir. Motorların çalışmadığını göstermek için ekrana motorların çalışmadığını ifade eden "Boş" şeklinde bir ifade bastırılmıştır. Açı kapa kontrolcüye geçildiğinde (001) kod kısmı daha sonra açıklanacak olan açı kapa kontrolcü fonksiyonuna potansiyometreden okunan referans değeri gönderilerek kontrolcünün çalışmasını sağlanmıştır. Kullanıcının hangi durumda olduğunu görmesi için Seri Port Ekranında ve LCD ekranda "Ac-kapa" yazılmıştır. P kontrolcü durumuna geçildiğinde (010) potansiyometreden okunan referans değeri P kontrolcünün fonksiyonuna parametre olarak gönderilmiştir. Kullanıcının hangi durumda olduğu görmesi için LCD ekrana ve Seri Port Ekranına "P" yazılmıştır. PI kontrolcü durumuna geçildiğinde (011) P kontrolcüde olduğu gibi potansiyometreden okunan referans değeri PI kontrolcünün fonksiyonuna parametre olarak gönderilmiştir. Kullanıcının hangi durumda olduğu görmesi için LCD ekrana ve Seri Port Ekranına "PI" yazılmıştır. PD kontrolcü durumuna geçildiğinde (100) PI kontrolcüde olduğu gibi potansiyometreden okunan referans değeri PD kontrolcünün fonksiyonuna parametre olarak gönderilmiştir. Kullanıcının hangi durumda olduğu görmesi için LCD ekrana ve Seri Port Ekranına "PD" yazılmıştır. PID kontrolcü durumuna

geçildiğinde (101) PD kontrolcüde olduğu gibi potansiyometreden okunan referans değeri PID kontrolcünün fonksiyonuna parametre olarak gönderilmiştir. Kullanıcının hangi durumda olduğu görmesi için LCD ekrana ve Seri Port Ekranına "PID" yazılmıştır.

```
if(buton1_durumu == 0 && buton2_durumu ==0 && buton3_durumu==0)
{
    analogWrite(power1, 0);
    analogWrite(power2, 0);
    lcd.print("Boş");
}
else if (buton1_durumu == 1 && buton2_durumu ==0 && buton3_durumu==0)
{
    // ac-kapa KONTROLCU
    Serial.println("AC-kapa");
    ac_kapa_kontrollor(değer);
    lcd.print("Ac-Kapa");
}
else if (buton1_durumu == 0 && buton2_durumu ==1 && buton3_durumu==0)
{
    // P KONTROLCU
    Serial.println("np");
    p_kontrollor(değer);
    lcd.print("P");
}
else if (buton1_durumu == 1 && buton2_durumu ==1 && buton3_durumu==0)
{
    // PI KONTROLCU
    Serial.println("\nPI");
    p_i_kontrollor(değer);
    lcd.print("PI");
}
else if(buton1_durumu == 0 && buton2_durumu ==0 && buton3_durumu==1)
{
    // PD KONTROLCU
    Serial.println("\nPDM");
    p_d_kontrollor(değer);
    lcd.print("PD");
}
else if(buton1_durumu == 1 && buton2_durumu ==0 && buton3_durumu==1)
{
    // PID KONTROLCU
    Serial.println("\nPID");
    p_i_d_kontrollor(değer);
    lcd.print("PID");
}
```

Fig. 7. Farklı buton kombinasyonlarına bağlı olarak kontrolcü seçiminin sağlayan Arduino kodu

2) Potansiyometre: Bu kısımda potansiyometre kullanılarak istenilen sıvı yüksekliğini ifade eden "referans" değerini Arduino'ya giriş olarak verilmiştir. Potansiyometre ayarlanabilir direnç olarak da isimlendirilmektedir. Daha önceki kısımlarda da ifade edildiği üzere potansiyometreden okunan direnç değerleri sürekli değerlerdir. Dolayısıyla potansiyometrenin analog pin olarak ifade edilmesi gerekmektedir. Aşağıdaki kodda A0 pininin potansiyometre pinini ifade eden "potpin" şeklinde tanımlanması gösterilmiştir.

```
#define potpin A0
```

Potansiyometre üzerinden okuma yapılacağından dolayı potpin (A0) pinı üzerinden okuma yapılarak kod içerisinde değişken olarak kullanabilmek için değer değişkenine atama yapılmıştır. Aşağıda kodu görülmektedir.

```
değer = analogRead(potpin);
```

analogRead fonksiyonu 0-1023 arasında değerler döndürmektedir. Su yüksekliğinin 20 cm'i geçmeyeceğini varsayıarak aşağıdaki kod ile 0-20cm aralığındaki referans yükseklik değerlerini kullanabiliyoruz. Referansın tam sayı olması için round fonksiyonu kullanılmıştır.

```
yükseklik = (20.00/1024.00)*değer;
```

```
deger=round(yukseklik);
```

A. Uzaklık Sensörü

Öncelikle aşağıdaki kod ile echo pini için 8. pin trigger pini için 7. pin tanımlanmıştır.

```
#define echoPin 8  
#define trigPin 7
```

Trigger pini ses dalgası gönderdiği için output olarak, Echo pini ses dalgasını aldığı için input olarak tanımlanmıştır.

```
pinMode(trigPin, OUTPUT);  
pinMode(echoPin, INPUT);
```

Daha sonra 8 numaralı figürde görülen mesafe isminde bir fonksiyon tanımlanarak sırasıyla echo pinine ses dalgasının gelme süresinin hesaplanması, süre üzerinden mesafe sensörü ile sensöre en yakın su seviye arasındaki mesafenin bulunması, ve bardağın uzunluğundan bu mesafe çıkarılarak bardaktaki toplam su yüksekliğinin bulunması gerçekleştirilmiştir. Fonksiyon maxrange ve minrange parametreleri alarak istenilen ölçüm aralığının dışına çıkmaması sağlanmıştır.

```
int mesafe(int maxrange, int minrange)  
{  
    long duration, distance;  
  
    digitalWrite(trigPin, LOW);  
    delayMicroseconds(2);  
    digitalWrite(trigPin, HIGH);  
    delayMicroseconds(10);  
    digitalWrite(trigPin, LOW);  
  
    duration = pulseIn(echoPin, HIGH);  
    distance = duration / 58.2;  
  
    if(distance >= maxrange || distance <= minrange)  
        return 0;  
    return 13 - distance;  
}
```

Fig. 8. Su seviyesinin ölçülmesini sağlayan fonksiyon

1) *Pompalar*: Öncelikle projede pompaların gücü üzerinden çalıştığımız için 9. pin power1 ve 6. pin power2 olarak tanımlanmıştır.

```
#define power1 9  
#define power2 6
```

Power pinleri void setup kısmında output olarak tanımlanmıştır.

```
pinMode(power1, OUTPUT);  
pinMode(power2, OUTPUT)
```

analogWrite() komutu 0,255 aralığında tam sayı değerler almaktadır. Bu gerçek dünyada 0-5V arasına denk gelmektedir. Motorun çalışma hızını ayarlayacak şekilde Kp,Ki,Kd değerleri ile yaptığımız işlemlerde sonucun bu aralıkta çıkışmasını sağladık. Örnek olarak aşağıdaki kodu kullanarak motor1'in çalışması durdurulmaktadır.

```
analogWrite(power1, 0);
```

Motor 1'in maksimum volajla çalışması için aşağıdaki kod kullanılır.

```
analogWrite(power1, 255);
```

2) *Arayüz*: Arayüzde LCD ekran modülü kullanıldı. Kullanıcı potansiyometre ile girdiği referans değeri, gerçek zamandaki su yüksekliği ve seçilen kontrolcü modu gösterilmektedir. Bunun için LiquidCrystal kütüphanesi kullanılmıştır.

```
#include <LiquidCrystal.h>
```

LCD ekrana bastırmak için lcd.print() fonksiyonu kullanılmıştır. Ekrandaki konumu ayarlamak için lcd.setCursor() fonksiyonu kullanılmıştır. Örneğin ekrana deger değişkeni bastırmak için aşağıdaki kod kullanılabilir.

```
lcd.print(String(deger));
```

Ekranda yazılan değişkenin ilk elemanını 0.satır 1. sütuna yazmak için aşağıdaki şekilde kullanılabilir.

```
lcd.setCursor(0, 1);
```

IV. MODLAR İLE İŞLEM SONUÇLARI

1) *Aç-Kapa kontrolcü*: Bu kontrolcü türünde verilen referans değer ölçülen yükseklik değerinden daha yüksekse pompa full gücü ile bardağı doldurmak için çalışacaktır. Diğer durumda ise bardağı boşaltmak için tüm gücü ile çalışacaktır. Bu mod kullanıldığı zaman ölçümlerde yüksekliğin verilen referans değerine oturduğu gözlemlenmiştir.

2) *P kontrolcü*: Bu kontrolcü türünde verilen referans değerinden ölçülen yükseklik değeri çıkarılarak hata ölçümleri yapılır. Ardından hata değeri ile önceden belirlenmiş olan K_p değeri çarpılarak elde edilen değer referans güç değeri olmaktadır. Buradan anlaşılabileceği üzeri hata değeri azaldıkça bu referans güç değeri azalmaktadır. Bu mod kullanıldığı zaman sistemin verilen referans değerine oturduğu gözlemlenmiştir.

3) *PI kontrolcü*: Bu kontrolcü türünde bir önceki kontrolcü türünden farklı olarak hesaplanan hata değerinin sadece doğrusal katı değil (K_P), sürekli olarak toplanması da referans güç değerine etki etmektedir. Bunun sonucunda beklediğimiz durum ise hatanın sürekli olarak birikesi ve ess'nin (Sürekli hal hatası'nın) azalması. Bu kontrolcü türü, P kontrolcünün istenen bir referans değerinden farklı noktaya oturmasını engellemek için kullanılır. Hataların toplanan değerinin sistem üzerinde salınıma yol açamaması için K_I değerinin düşük olması mantıklı olacaktır. Bu mod kullanıldığı zaman sistemin verilen referans değerine oturduğu gözlemlenmiştir.

4) *PD kontrolcü*: Bu kontrolcü türünde P kontrolcü türünden farklı olarak hesaplanan hata değerinin sadece doğrusal katı değil (K_P), hata üzerindeki değişimle bağlı olarak da, referans güç değerini değiştirmektedir. Bunun sonucunda hata değeri azalmaya başladıkça sisteme verilen referans güç değeri de azalmaktadır. Bu kontrolcü türü, sistemin istenen referans değeri etrafında salınım yapmasını engellemek için kullanılır. Bu sistemde bozucu etkilerin sistemi overshooting'e sokmaması için K_D değerinin düşük seçilmesi gerekmektedir. Bu mod kullanıldığı zaman sistemin verilen referans değerine oturduğu gözlemlenmiştir.

5) *PID kontrolcü*: Bu kontrolcü türününü önceki gördüğümüz kontrolcü türlerinin toplamı olarak değerlendirebiliriz (Aç-Kapa hariç). Hatanın doğrusal olarak değişmesi, toplanarak etki etmesi ve hata üzerindeki değişimin etki etmesi hep beraber referans güç değerini değiştirmektedir. Bunun sonucunda sistem en dengeli şekilde refearns değerine oturmaktadır. Önceki kısımlarda bahsedilen K_P , K_I ve K_D değeri seçimi aynı şekilde yapılmaktadır. Bu mod kullanıldığı zaman sistemin istenen referans değerine oturduğu gözlemlenmiştir.

Video linki için [buraya](#) tıklayınız.

V. MODLAR İLE İŞLEM SONUÇLARI

Uzak üzerindeki yükleme yerinde raporlarınıza kodlarınızı da ekleyin yazdığı için bütün kodu burada paylaşıyoruz:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
#define Buton1 22
#define Buton2 24
#define Buton3 26
#define echoPin 8
#define trigPin 7
#define power1 9
#define power2 6
#define potpin A0

int maximumRange = 50;
int minimumRange = 0;

int buton1_durumu = 0;
int buton2_durumu = 0;
int buton3_durumu = 0;

int olcum = 0, hata = 0, deger = 0, deger_i = 0;
double now = 0, dt = 0, last_time = 0, onceki = 0, kp, ki, kd, i_hata = 0, d_hata = 0, toplam_
float yukseklik = 0;

void setup() {
    Serial.begin(9600);
    lcd.begin(16, 2);
```

```
    pinMode(Buton1, INPUT);
    pinMode(Buton2, INPUT);
    pinMode(Buton3, INPUT);
    pinMode(power1, OUTPUT);
    pinMode(power2, OUTPUT);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
}

void loop() {
    deger = analogRead(potpin);
    yukseklik = (20.00/1024.00)*deger;
    deger_i=round(yukseklik);

    lcd.setCursor(0, 0);
    lcd.print("Ref:");
    lcd.setCursor(5, 0); // Set the cursor position
    lcd.print(String(deger_i));
    olcum = mesafe(maximumRange, minimumRange);

    lcd.setCursor(0, 1);
    lcd.print("H:");
    lcd.setCursor(3, 2);
    lcd.print(String(olcum));
    lcd.setCursor(9, 2);

    buton1_durumu = digitalRead(Buton1);
    buton2_durumu = digitalRead(Buton2);
    buton3_durumu = digitalRead(Buton3);

    if(buton1_durumu == 0 && buton2_durumu ==0 && buton3_durumu == 0)
    {
        analogWrite(power1, 0);
        analogWrite(power2, 0);
        lcd.print("Bos");
    } else if (buton1_durumu == 1 && buton2_durumu == 0 && buton3_durumu == 0)
    {
        // ac-kapa KONTROLCU
        Serial.println("AC-kapa");
        ac_kapa_kontrollor(deger_i);
        lcd.print("Ac-Kapa");
    } else if (buton1_durumu == 0 && buton2_durumu == 1 && buton3_durumu == 0)
    {
        // P KONTROLCU
        Serial.println("\np");
        p_kontrollor(deger_i);
        lcd.print("P");
    } else if (buton1_durumu == 1 && buton2_durumu == 1 && buton3_durumu == 0)
    {
        // PI KONTROLCU
        Serial.println("\nPI");
        pi_kontrollor(deger_i);
    }
}
```

```

    lcd.print("PI");
}
else if(buton1_durumu == 0 && buton2_durumu == 1 && buton3_durumu == 1)
{
    // PD KONTROLCU
    Serial.println("\nPD");
    p_d_kontrollor(deger_i);
    lcd.print("PD");
}
else if(buton1_durumu == 1 && buton2_durumu == 0 && buton3_durumu == 1)
{
    // PID KONTROLCU
    Serial.println("\nPID");
    p_i_d_kontrollor(deger_i);
    lcd.print("PID");
}

delay(100);
lcd.clear();
}

int mesafe(int maxrange, int minrange)
{
    long duration, distance;

    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    duration = pulseIn(echoPin, HIGH);
    distance = duration / 58.2;

    if(distance >= maxrange || distance <= minrange)
        return 0;
    return 13 - distance;
}

void ac_kapa_kontrollor(int ref){
    olcum = mesafe(maximumRange, minimumRange);
    hata = ref - olcum;
    if(hata > 0){
        analogWrite(power2, 255);
        analogWrite(power1, 0);
    } else if(hata < 0){
        analogWrite(power2, 0);
        analogWrite(power1, 255);
    } else{
        analogWrite(power1, 0);
        analogWrite(power2, 0);
    }
}

void p_kontrollor(int ref){
    kp = 15;
    olcum = mesafe(maximumRange, minimumRange);
    hata = ref - olcum;
    if(hata > 0 && buton3_durumu == 0) {
        if(80+hata*kp > 255){
            analogWrite(power2, 255);
        } else{
            analogWrite(power2, 80+hata*kp);
        }
        analogWrite(power1, 0);
    } else if(hata < 0 && buton3_durumu == 0) {
        if(80-1*hata*kp > 255){
            analogWrite(power1, 255);
        } else{
            analogWrite(power1, 80-1*hata*kp);
        }
        analogWrite(power2, 0);
    } else{
        analogWrite(power1, 0);
        analogWrite(power2, 0);
    }
}

void p_i_kontrollor(int ref){
    kp = 15;
    ki = 0.3;
    olcum = mesafe(maximumRange, minimumRange);
    now = millis();
    dt = (now - last_time)/1000;
    last_time = now;
    hata = ref - olcum;
    i_hata += hata*dt;
    if(hata > 0){
        toplam_hata = 80 + ((kp*hata) + (ki*i_hata));
        if(toplam_hata > 255){
            toplam_hata = 255;
        }
        analogWrite(power2, toplam_hata);
        analogWrite(power1, 0);
    } else if (hata < 0){
        toplam_hata = 80 + abs((kp*hata) + (ki*i_hata));
        if(toplam_hata > 255){
            toplam_hata = 255;
        }
        analogWrite(power1, toplam_hata);
        analogWrite(power2, 0);
    } else{
        analogWrite(power1, 0);
        analogWrite(power2, 0);
    }
}

```

```

void p_d_kontrollor(int ref){
    kp = 15;
    kd = 0.5;
    olcum = mesafe(maximumRange, minimumRange);
    now = millis();
    dt = (now - last_time)/1000;
    last_time = now;
    hata = ref - olcum;
    d_hata = (hata-onceki)/dt;
    onceki = hata;
    if(hata > 0){
        toplam_hata = 80 + ((kp*hata) + (kd*d_hata));
        if(toplam_hata > 255){
            toplam_hata = 255;
        }
        analogWrite(power2, toplam_hata);
        analogWrite(power1, 0);
    }
    else if (hata < 0){
        toplam_hata = 80 - 1*((kp*hata) + (kd*d_hata));
        if(toplam_hata > 255){
            toplam_hata = 255;
        }
        analogWrite(power1, toplam_hata);
        analogWrite(power2, 0);
    }
    else{
        analogWrite(power1, 0);
        analogWrite(power2, 0);
    }
}

```

Üst tarafta bulunamama ihtimaline karşı tekrar belirtiyorum. Video linki için [buraya](#) tıklayınız.

```

void p_i_d_kontrollor(int ref){
    kp = 15;
    ki = 0.3;
    kd = 0.5;
    olcum = mesafe(maximumRange, minimumRange);
    now = millis();
    dt = (now - last_time)/1000;
    last_time = now;
    hata = ref - olcum;
    i_hata += hata*dt;
    d_hata = (hata-onceki)/dt;
    onceki = hata;
    if(hata > 0){
        toplam_hata = 50+((kp*hata) + (kd*d_hata) +(ki*i_hata));
        if(toplam_hata > 255){
            toplam_hata = 255;
        }
        analogWrite(power2, toplam_hata);
        analogWrite(power1, 0);
    }
    else if (hata < 0){
        toplam_hata = 50-((kp*hata) + (kd*d_hata) -(ki*i_hata));
        if(toplam_hata < 0){
            toplam_hata = 0;
        }
        analogWrite(power1, toplam_hata);
        analogWrite(power2, 0);
    }
    else{
        analogWrite(power1, 0);
        analogWrite(power2, 0);
    }
}

```