SELÇUK
ÜNİVERSİTESİ

T.C

SELÇUK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
ELEKTRİK ELEKTRONİK BÖLÜMÜ
TASARIM PROJESİ RAPORU

BLUETOOTH KONTROLLÜ DENGE ROBOTU
(BLUETOOTH CONTROLLED BALANCE ROBOT)

Hazırlayan
161202112
Mustafa ÜNLÜ

Proje Danışmanı
Dr. Akif DURDU

2018 - KONYA

1. GİRİŞ

Tek tekerlekli ya da top robotlar gibi iki tekerlekli robotlar da diğer tip robotlara oranla denge problemi yaşarlar. Bunun sebebi dengede kalabilmek için harekete devam etme gerekliliklerdir. İki tekerli robotlarda robotun ağırlık merkezinin dingilin altında olması gerekmektedir. Bunu başarmak için genellikle batarya gibi ağır parçalar, gövdenin alt kısmına yerleştirilir. İki tekerlekli robotlar, tekerlekleri birbirlerine paralel ya da arkalı önlü dizilmiş olabilir. İki tekerleklli robotlar dengede durmak için hareketlerine devam etmek zorundadırlar. Aracın düşme yönüne doğru dönüş hareketi yapması dengesini yeniden kazanmasını sağlar. Sağ ve sol tekerleri bulunan bir robotun dengede durmak için en az iki sensöre ihtiyacı vardır. Bunlardan biri tilt açısını ölçmek için bir sensör ve diğeri robotun pozisyonunu öğrenmek için motor enkoderleridir.

2. MALZEMELERİN TANITIMI

2.1 6V 250 Rpm Motor ve Tekerlek Seti

Plastik redüktörlü motor ve tekerlek seti basit uygulamalarda kullanabileceğiniz bir üründür. Motorda iki ayrı noktadan mil çıkışı olduğu için sağ ve sol kullanımlarda rahatlıkla kullanılabilir.

ÖZELLIKLER

-Çalışma Voltajı: 3-6V

-Hız: 250 Rpm(@6V)

-Ağırlık: 29gr

TEKER ÖLÇÜLERI

- Çapı: 70mm

-Kalınlık: 30mm

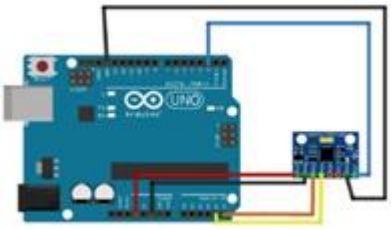
-Ağırlık: 45gr

2.2 MPU6050 6 EKSEN İVME VE GYRO SENSÖRÜ GY-521

MPU-6050 çeşitli hobi, multicopter ve robotik projelerinde sıklıklı kullanılan üzerinde 3 eksenli bir gyro ve 3 eksenli bir açısal ivme ölçer bulunduran 6 eksenli bir IMU sensör kartıdır. Kart üzerinde voltaj regulatörü bulunduğundan 3 ile 5 V arası bir besleme voltajı ile çalıştırılabilir. İvme ölçer ve gyro çıkışlarının her ikisi de ayrı kanallardan I²C çıkışı vermektedir. Her eksende 16 bitlik bir çözünürlükle çıkış verebilmektedir. Pinler arası boşluk standart olarak ayarlandığı için breadboard veya farklı devre kartlarında rahatlıkla kullanılabilir.

ÖZELLIKLER

-Çalışma gerilimi: 3-5V
-Gyro ölçüm aralığı: + 250 500 1000 2000 ° / s
-Açısal ivme ölçer ölçüm aralığı: ± 2 ± 4 ± 8 ± 16 g İletişim: Standart I²C



Şekil b. MPU6050 6 Eksen İvme ve Gyro Sensörü gy-521 ve Çalışma Prensibi

2.3 L298N MOTOR SÜRÜCÜ KARTI

Kart üzerinde 1 adet L298N motor sürücü entegresi mevcuttur. Kanal başına 2A'e kadar akım verebilmektedir. Motor voltajı 6-15V arası kullanılabilmektedir. Besleme gerilimi 5V ise kart üzerindeki Vcc-5V jumpere kısa devre yapılarak kullanılmaktadır. Kartın kullanılmayan tüm pinleri kart üzerindeki 4'lü konnektörlere çevrilerek genel kullanım için bırakılmıştır.

ÖZELLIKLER

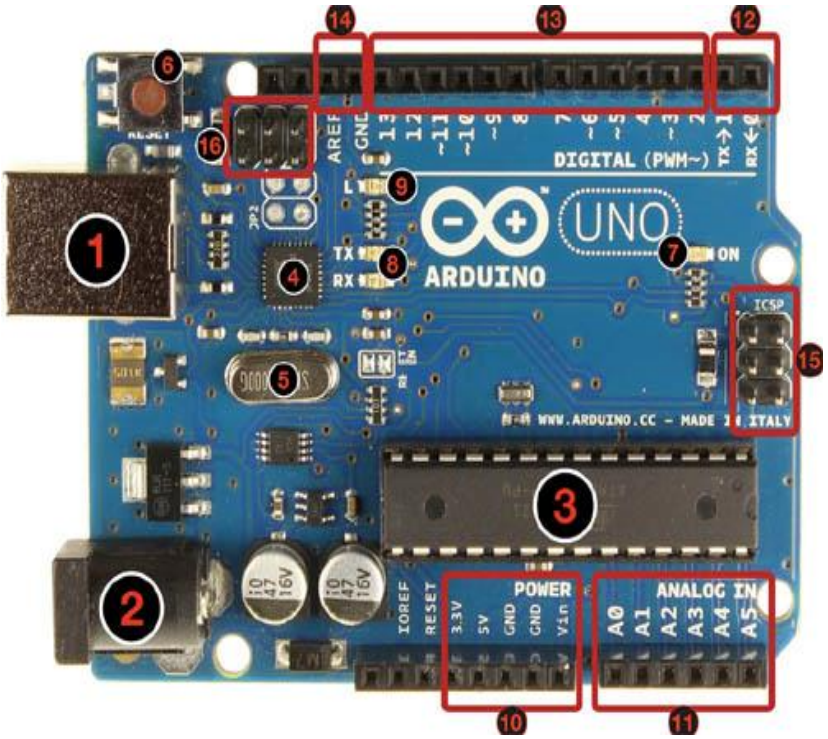
-2 adet DC motorun bağımsız kontrolü
-Her bir kanaldan sürekli 2A'e kadar verebilmektedir.
-Sensör bağlantıları için boş bırakılmış analog ve dijital giriş pinleri
-Ürün Boyutları: 68x55x30mm Ağırlık: 37g
-Dahili 5V voltaj regültörü
-5V-35V sürüş voltajı

Şekil c. L298N Motor Sürücü Kartı

2.4 ARDUİNO UNO

Arduino Uno 'nun 14 tane dijital giriş / çıkış pini vardır. Bunlardan 6 tanesi PWM çıkışı olarak kullanılabilir. Ayrıca 6 adet analog girişi, bir adet 16 MHz kristal osilatörü, USB bağlantısı, power jakı (2.1mm), ICSP başlığı ve reset butonu bulunmaktadır. Arduino Uno bir mikrodenetleyiciyi desteklemek için gerekli bileşenlerin hepsini içerir. Arduino Uno 'yu bir bilgisayara bağlayarak, bir adaptör ile ya da pil ile çalıştırabilirsiniz. Aşağıdaki resimde Arduino Uno R3 'ün kısımları gösterilmektedir.

1 : USB jakı
2 : Power jakı (7-12 V DC)
3 : Mikrodenetleyici ATmega328
4 : Haberleşme çipi
5 : 16 MHz kristal
6 : Reset butonu
7 : Power ledi
8 : TX / NX ledleri
9 : Led
10 : Power pinleri
11 : Analog girişler
12 : TX / RX pinleri
13 : Dijital giriş / çıkış pinleri (yanında ~ işareti olan pinler PWM çıkışı olarak kullanılabilir.)
14 : Ground ve AREF pinleri
15 : ATmega328 için ICSP
16 : USB arayüzü için ICSP



TEKNIK ÖZELLIKLER

- Mikrodenetleyici : ATmega328
- Çalışma gerilimi : +5 V DC
- Tavsiye edilen besleme gerilimi : 7 - 12 V DC
- Besleme gerilimi limitleri : 6 - 20 V
- Dijital giriş / çıkış pinleri : 14 tane (6 tanesi PWM çıkışını destekler)
- Analog giriş pinleri : 6 tane
- Giriş / çıkış pini başına düşen DC akım : 40 mA
- 3,3 V pini için akım : 50 mA
- Flash hafıza : 32 KB (0.5 KB bootloader için kullanılır)
- SRAM : 2 KB
- EEPROM : 1 KB
- Saat frekansı : 16 MHz

ARDUINO UNO GÜÇ

Arduino Uno bir USB kablosu ile bilgisayar bağlanarak çalıştırılabilir ya da harici bir güç kaynağından beslenebilir. Harici güç kaynağı bir AC-DC adaptör ya da bir pil / batarya olabilir. Adaptörün 2.1 mm jaklı ucunun merkezi pozitif olmalıdır ve Arduino Uno 'nun power girişine takılmalıdır. Pil veya bataryanın uçları ise power konnektörünün GND ve Vin pinlerine bağlanmalıdır.

VIN : Arduino Uno kartına harici bir güç kaynağı bağlandığında kullanılan voltaj girişidir.

5V : Bu pin Arduino kartındaki regülatörden 5 V çıkış sağlar. Kart DC power jakından (2 numaralı kısım) 7-12 V adaptör ile, USB jakından (1 numaralı kısım) 5 V ile ya da VIN pininden 7-12 V ile beslenebilir. 5V ve 3.3V pininden voltaj beslemesi regülatörü bertaraf eder ve karta zarar verir.

3.3V : Arduino kart üzerindeki regülatörden sağlanan 3,3V çıkışıdır. Maksimum 50 mA dir.

GND : Toprak pinidir.

IOREF : Arduino kartlar üzerindeki bu pin, mikrodenetleyicinin çalıştığı voltaj referansını sağlar. Uygun yapılandırılmış bir shield IOREF pin voltajını okuyabilir ve uygun güç kaynaklarını seçebilir ya da 3.3 V ve 5 V ile çalışmak için çıkışlarında gerilim dönüştürücülerini etkinleştirebilir.

ARDUINO UNO GIRIŞ VE ÇIKIŞLAR

Arduino Uno 'da bulunan 14 tane dijital giriş / çıkış pininin tamamı, pinMode(), digitalWrite() ve digitalRead() fonksiyonları ile giriş ya da çıkış olarak kullanılabilir. Bu pinler 5 V ile çalışır. Her pin maksimum 40 mA çekebilir ya da sağlayabilir ve 20-50 KOhm dahili pull - up dirençleri vardır. Ayrıca bazı pinlerin özel fonksiyonları vardır:

-Serial 0 (RX) ve 1 (TX) : Bu pinler TTL seri data almak (receive - RX) ve yaymak (transmit - TX) içindir.
-Harici kesmeler (2 ve 3) : Bu pinler bir kesmeyi tetiklemek için kullanılabilir.
-PWM: 3, 5, 6, 9, 10, ve 11 : Bu pinler analogWrite () fonksiyonu ile 8-bit PWM sinyali sağlar.
-SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) : Bu pinler SPI kütüphanesi ile SPI haberleşmeyi sağlar.
-LED 13 : Dijital pin 13 e bağlı bir leddir. Pinin değeri High olduğunda yanar, Low olduğunda söner.

Arduino Uno 'nun A0 dan A5 e kadar etiketlenmiş 6 adet analog girişi bulnur, her biri 10 bitlik çözünürlük destekler. Varsayılan ayarlarda topraktan 5 V a kadar ölçerler. Ancak, AREF pini ve analogReference() fonksiyonu kullanılarak üst limit ayarlanabilir.

-TWI : A4 ya da SDA pini ve A5 ya da SCL pini Wire kütüphanesini kullanarak TWI haberleşmesini destekler.
-AREF : Analog girişler için referans voltajıdır. analogReference() fonksiyonu ile kullanılır.
-RESET : Mikrodenetleyiciyi resetlemek içindir. Genellikle shield üzerine reset butonu eklemek için kullanılır.

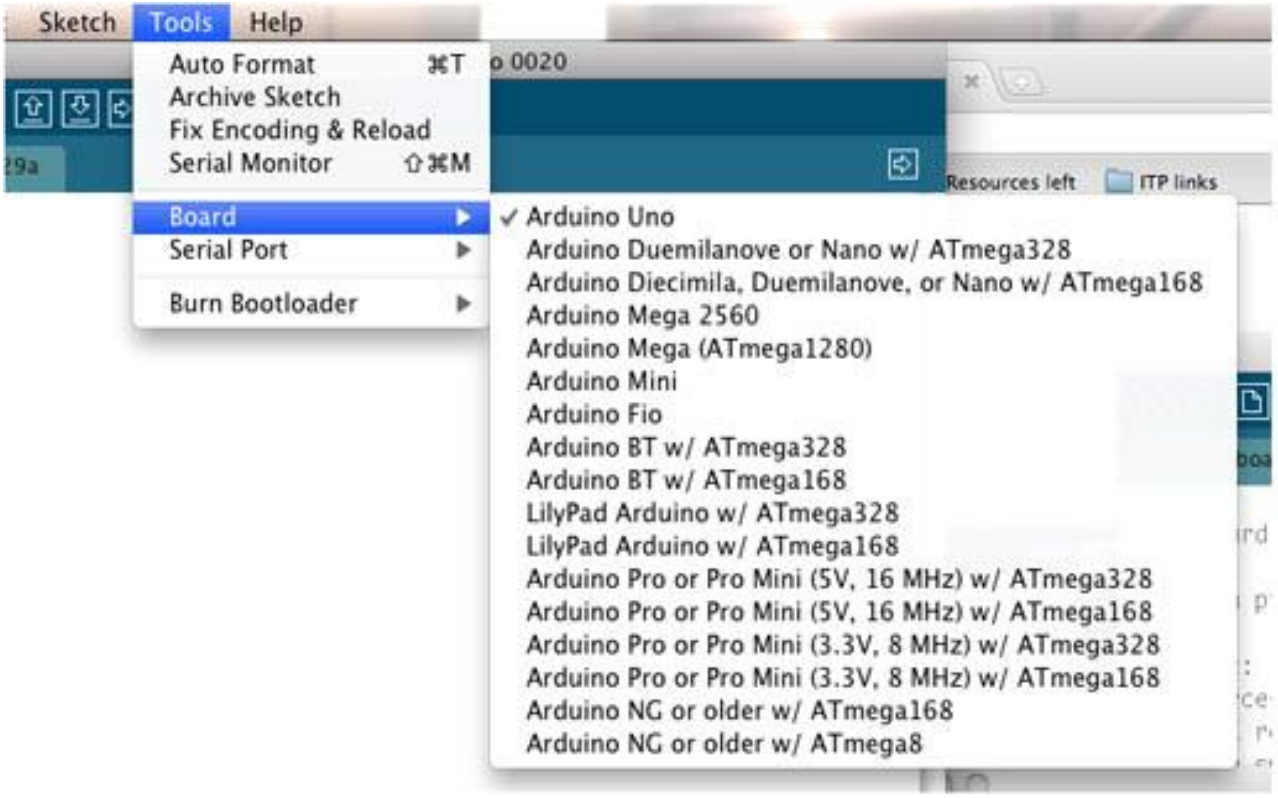
ARDUINO UNO HABERLEŞME

Arduino Uno bir bilgisayar ile, başka bir Arduino ile ya da diğer mikrodenetleyiciler ile haberleşme için çeşitli imkanlar sunar. ATmega328 mikrodenetleyici, RX ve TX pinlerinden erişilebilen UART TTL (5V) seri haberleşmeyi destekler. Kart üzerindeki bir ATmega16U2 seri haberleşmeyi USB üzerinden kanalize eder ve bilgisayardaki yazılıma sanal bir com portu olarak görünür. 16U2 standart USB com sürücülerini kullanır ve harici sürücü gerektirmez. Ancak, Windows 'ta bir .inf dosyası gereklidir. Kart üzerindeki RX ve TX ledleri USB den seri çipe ve USB den bilgisayara veri giderken yanıp söner.

SoftwareSerial kütüphanesi Arduino Uno 'nun digital pinlerinden herhangi biri üzerinden seri haberleşmeye imkan sağlar. Ayrıca ATmega328 I2C (TWI) ve SPI haberleşmelerini de destekler.

ARDUINO UNO PROGRAMLAMA

Arduino Uno 'yu programlamak için Arduino programını indirmeniz gerekir. Programı indirip açtıktan sonra Tools > Board menüsünden Arduino Uno 'yu seçiniz.



Arduino Uno üzerindeki ATmega328 e önceden bir bootloader yüklenmiştir. Bu bootloader sayesinde Arduino 'yu programlamanız için harici bir programlayıcı donanımına ihtiyacınız olmaz. Orjinal STK500 programını kullanarak haberleşir.

Ayrıca Arduino ISP kullanarak Arduino 'nun bootloader 'ını devre dışı bırakabilir ve mikrodenetleyiciyi ICSP (In Circuit Serial Programming) pini üzerinden programlayabilirsiniz.

ARDUINO UNO USB AŞIRI AKIM KORUMASI

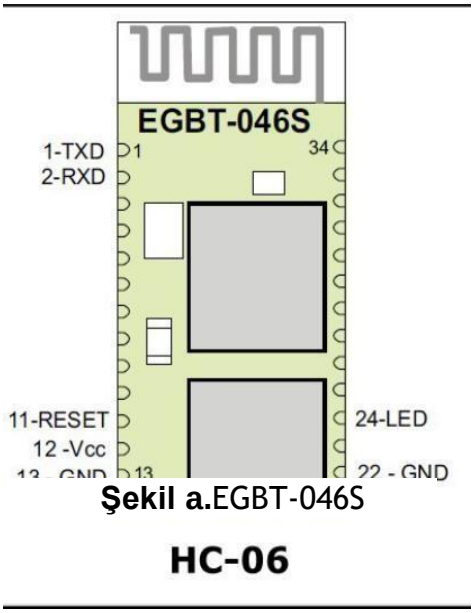
Arduino Uno, bilgisayarınızın USB portunu aşırı akım ve kısa devreden koruyan resetlenebilir bir çoklu sigortası bulunur. Çoğu bilgilsayarın portlar için kendi korumaları olmasına rağmen bu sigorta ekstra bir koruma katmanı sağlar. Eğer USB portuna 500 mA den fazla bir yük binerse, sigorta otomatik olarak bağlantıyı kısa devre veya aşırı akım durumu ortadan kalkana dek keser.

2.5 BLUETOOTH HABERLEŞME MODÜLÜ HC-06

Robot ile kumanda arasındaki seri haberleşmeyi bağlantısını kurabilmek için bluetooth modül kullanılmıştır. Kullanılan modül HC-06 bluetooth modülüdür. Bağlantı olarak TX, RX,+V ve GND olmak üzere 4 pini vardır.

ÖZELLIKLER

-Bluetooth Protokolü: Bluetooth 2.0+EDR (Gelişmiş Veri Hızı)
-2.4GHz haberleşme frekansı
-Hassasiyet: ≤-80 dBm
-Çıkış Gücü: ≤+4 dBm
-Asenkron Hız: 2.1 MBps/160 KBps
-Senkron Hız: 1 MBps/1 MBps
-Güvenlik: Kimlik Doğrulama ve Şifreleme
-Çalışma Gerilimi: 1.8-5V(Önerilen 3.3V)
-Akım: 50 mA
-Boyutları: 43x16x7mm



Şekil a.EGBT-046S

Şekil b. HC-06, bluetooth modülü.

2.6 Lİ-PO BATARYA

Lipo Pil, Nimh pillerden sonra bulunan ve sağladığı avatajlar sebebiyle büyük beğeni toplamıştır. Hafif olması, istenilen boyutlarda üretilebilmesi, yüksek kapasite ve güce sahip olması, hızlı şarj deşarj imkanı vermesi Lipo kullanımını arttıran nedenlerdir.

Bu kadar avantajın yanında dezavantajda barındırmıyor değil. Nimh pillere göre yüksek fiyatlarda olması, Şarj/deşarj ömrünün kısa olması, patlama riski taşıması, şarj ve deşarj edilirken yoğun talimatlara uyulması gibi dezavantajları bulunur.

Lipo pil hücreleri kullanılarak değişik kombinasyonlarda farklı amaçlar için farklı piller üretilir. Pilde "S" değeri pil içerisinde kaçtane hücrenin seri bağlandığını gösterir. Bir lipo hücresi 3.7V değerindedir. "S" değeri arttıkça pilin voltajı artmaktadır.

1S Lipo = 1 Lipo Hücresi = 3.7V

2S Lipo = 2 Lipo Hücresi = 3.7V+3.7V =7.4V

mAh değeri pilin kapasitesini göstermektedir (miliamper/saat). 2000 mAh kapasiteli bir pilden 1000 miliamper çekilirse 2 saatte pil tamamen deşarj olur. 2000 miliamper çekilirse 1 saatte tamamen deşarj olacaktır. mAh değeri arttıkça kapasite artacaktır ve aracınızı kullanma süreniz uzayacaktır.

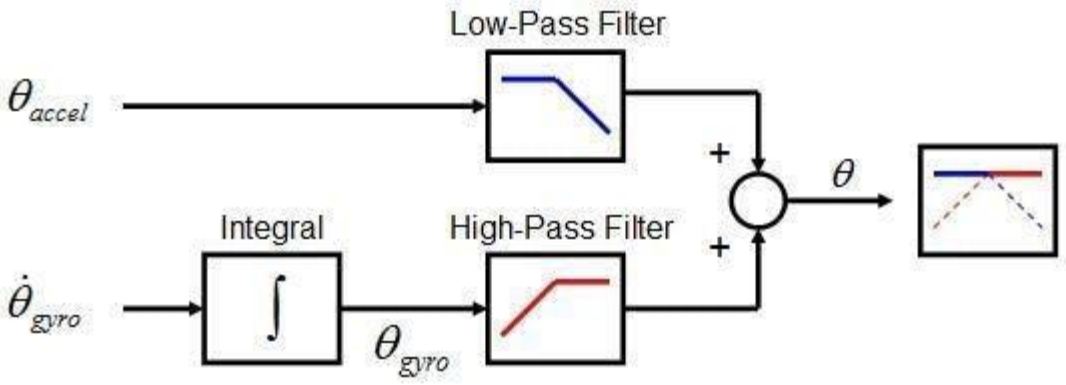
"C" değeri pilin deşarj hızını temsil eder. 10C değerine sahip bir pil kapasitesinin 10 katı kadar hızda deşarj edilebilir. 20C değerindeki pil 20 katı kadar, 30C değerindeki pil 30 katı kadar … bu şekilde devam eder. Örnekle açıklamak gerekirse, 2000mAh kapasiteli 10C bir pilden sürekli olarak en fazla 20 amper çekilebilir. 5000 mAh 25C bir pilden sürekli olarak en fazla 125 amper çekilebilir. Genelde 25-30C değerindeki piller işimizi görmektedir fakat imkan varsa daha yüksek C değerine sahip pilleri tercih etmekte fayda vardır.

Lipo pil kullanımında %80 kullanım kuralı vardır. 3000mAh kapasiteli bir pilin sadece %80 inini yani 2400mAh kadar kullanmanız gerekir. Voltaj olarak ele aldığımızda, bir lipo hücresinin en düşük voltajı 3.7V olmalıdır. Hücre voltajı 4.2V olduğunda hücrenin tamamen dolu olduğu anlamına gelir. 2S bir pil kullandığımızı farzedersek iki hücrenin toplam voltajı 8.4V olduğunda pil tamamen doludur. 7.4V olduğunda ise pil tamamen boştur ki bu voltajı görmenizi tavsiye etmiyoruz. %80 kuralını baz alırsak en fazla 3.74 voltaja kadar düşmesini gerekir.

Şekil 1. Li-Po Batary

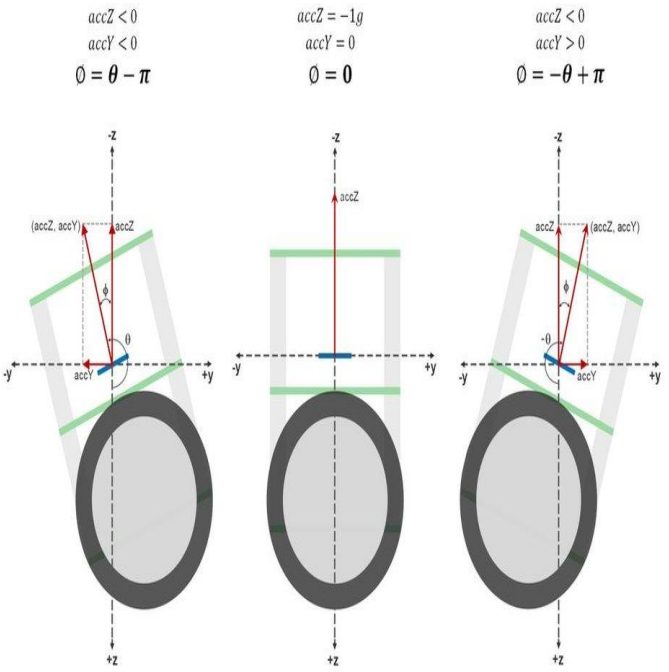
3. PROJENİN GENEL ÇALIŞMA MANTIĞI

Projemizde Arduino ile kendini dengeleyen robot üzerine çalışılmıştır. Yapmış bulunduğumuz denge robotu uzaktan kontrol edilebilmektedir. Robotumuzun kontrol sisteminden bahsedecek olursak dengede durması için gyro sensörden gelen açı analiz edilmektedir ve PID kontrol sistemiyle açıya göre motorlara ivmelenme vererek dengeyi sağlamaktadır.



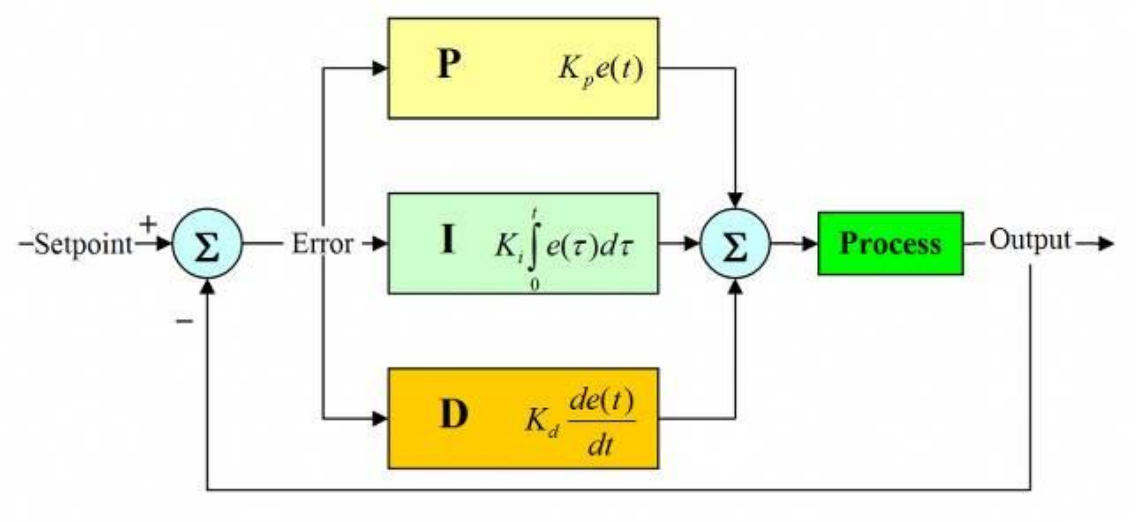
4. GYRO ÇALIŞMA PRENSİBİ

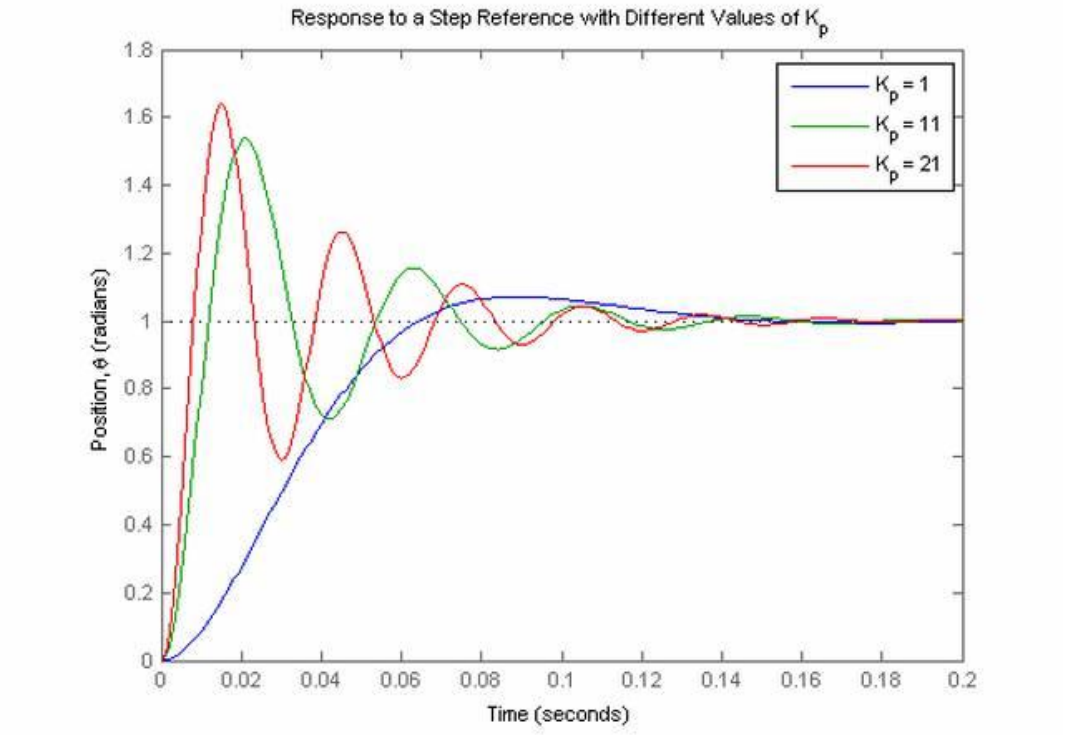
Gyro, (Jiroskop) yön ölçümü veya ayarlamasında kullanılan, açısal dengenin korunması ilkesiyle çalışan bir alettir. Jiroskopik hareketin temeli fizik kurallarına ve açısal momentumun korunumu ilkesine dayalıdır. Gyro, yön belirleme amaçlı kullanılmaktadır. Örneğin; Jiroskopa sahip telefonu yan çevirdiğinizde ekranında yan dönmesine olanak sağlamaktadır.



5. PID ÇALIŞMA PRENSİBİ

PID sık kullanılan geri besleme denetleyicisi yöntemidir. PID(proportional,Integral,Derivative) oransal-integral-türevsel denetleyici PID kontrol döngüsü yöntemi , yaygın endüstriyel kontrol sistemlerinde kullanılan genel bir kontrol döngüsü geribildirim mekanizmasıdır. Bir PID denetleyici ölçülü bir süreç içinde değişen ve istenilen ayar noktası ile arasındaki farkı olarak bir "hata" değerini hesaplar. Kontrolör proses kontrol girişini ayarlayarak hatayı en aza indirerek istenilen ayar değerine ulaşmak için çalışır. PID kontrolcüleri akış, sıcaklık, seviye, basınç ve diğer proses değişkenlerini düzenlemek, regüle etmek için kullanılır.





6. PROJENİN KODLARI

MustafaUnlu.DengeRobot §

```
#include "Wire.h"
#define MIN_ABS_SPEED 0
#include <SoftwareSerial.h>
MPU6050 mpu;
float veri;
char state;
float veri2;
bool dmpReady = false;
uint8_t mpuIntStatus;
uint8_t devStatus;
uint16_t packetSize;
uint16_t fifoCount;
uint8_t fifoBuffer[64];

Quaternion q;
VectorFloat gravity;
float ypr[3];

```

MustafaUnlu.DengeRobot §

```
double originalSetpoint = 177;  //173
double setpoint = originalSetpoint;
double movingAngleOffset = 3.3;  //0.3
double input, output;
int moveState=0;

double Kp = 35;   //40
double Kd = 0.7;  //1
double Ki = 70;   //50
PID pid(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);
double motorSpeedFactorLeft = 0.55;
double motorSpeedFactorRight = 0.45;
int ENA = 5;
int IN1 = 6;
int IN2 = 7;
int IN3 = 8;
int IN4 = 9;
int ENB = 10;
```

Dosya Düzenle Taslak Araçlar Yardım

MustafaUnlu.DengeRobot §

```
//MUSTAFA ÜNLÜ DENGE ROBOTU KODU 161202112
/*
BLUETOOTH İLE KONTROL İÇİN
DENGE HASSAS AYARI == (+),(-)
DENGE HASSAS İNCE AYARI == (a),(b)
İLERİ HIZLI == (1)
DUR == (2)
GERİ HIZLI == (3)
DUR NORMAL == (c)
İLERİ NORMAL == (d)
GERİ NORMAL == (e)
SAG DÖN == (f)
SOL DÖN == (g)
*/
#include <PID_v1.h>
#include <LMotorController.h>
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
```

MustafaUnlu.DengeRobot §

```
55 LMotorController motorController(ENA, IN1, IN2, ENB, IN3, IN4, motorSpeedFactorLeft, motorSpeedFactorRight);
56
57 volatile bool mpuInterrupt = false;
58 void dmpDataReady()
59 {
60 mpuInterrupt = true;
61 }
62
63
64 void setup()
65 {
66
67 Serial.begin(9600);
68 #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
69 Wire.begin();
70 TWBR = 24;
71 #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
72 Fastwire::setup(400, true);
```

MustafaUnlu.DengeRobot §

```
91 packetSize = mpu.dmpGetFIFOPacketSize();
92
93 pid.SetMode(AUTOMATIC);
94 pid.SetSampleTime(5);
95 pid.SetOutputLimits(-255, 255);
96 }
97 }
98 void loop()
99 {
100 moveBackForth();
101 while (!mpuInterrupt && fifoCount < packetSize)
102 {
103 if(moveState==3){                                    //Sağ taraf
104 setpoint = originalSetpoint - 1;
105 pid.Compute();
106 motorController.turnLeft(output, 120);
107 delay(1);
108 }
```

MustafaUnlu.DengeRobot §

```
73 #endif
74 while (!Serial);
75 mpu.initialize();
76
77 devStatus = mpu.dmpInitialize();
78
79 mpu.setXGyroOffset(128); //128
80 mpu.setYGyroOffset(50); //50
81 mpu.setZGyroOffset(-100); //-100
82 mpu.setZAccelOffset(1688); // 1688
83
84 if (devStatus == 0)
85 {
86 mpu.setDMPEnabled(true);
87
88 attachInterrupt(0, dmpDataReady, RISING);
89 mpuIntStatus = mpu.getIntStatus();
90 dmpReady = true;
```

MustafaUnlu.DengeRobot §

```
else if(moveState==4){                                  //Sol taraf
setpoint = originalSetpoint + 1;
pid.Compute();
motorController.turnRight(output, 120);
delay(1);
}
else if(moveState==0||moveState==1||moveState==2){
pid.Compute();
motorController.move(output, MIN_ABS_SPEED);
delay(1);
}
}
mpuInterrupt = false;
mpuIntStatus = mpu.getIntStatus();
fifoCount = mpu.getFIFOCount();
if ((mpuIntStatus & 0x10) || fifoCount == 1024)
{
mpu.resetFIFO();
```

MustafaUnlu.DengeRobot §

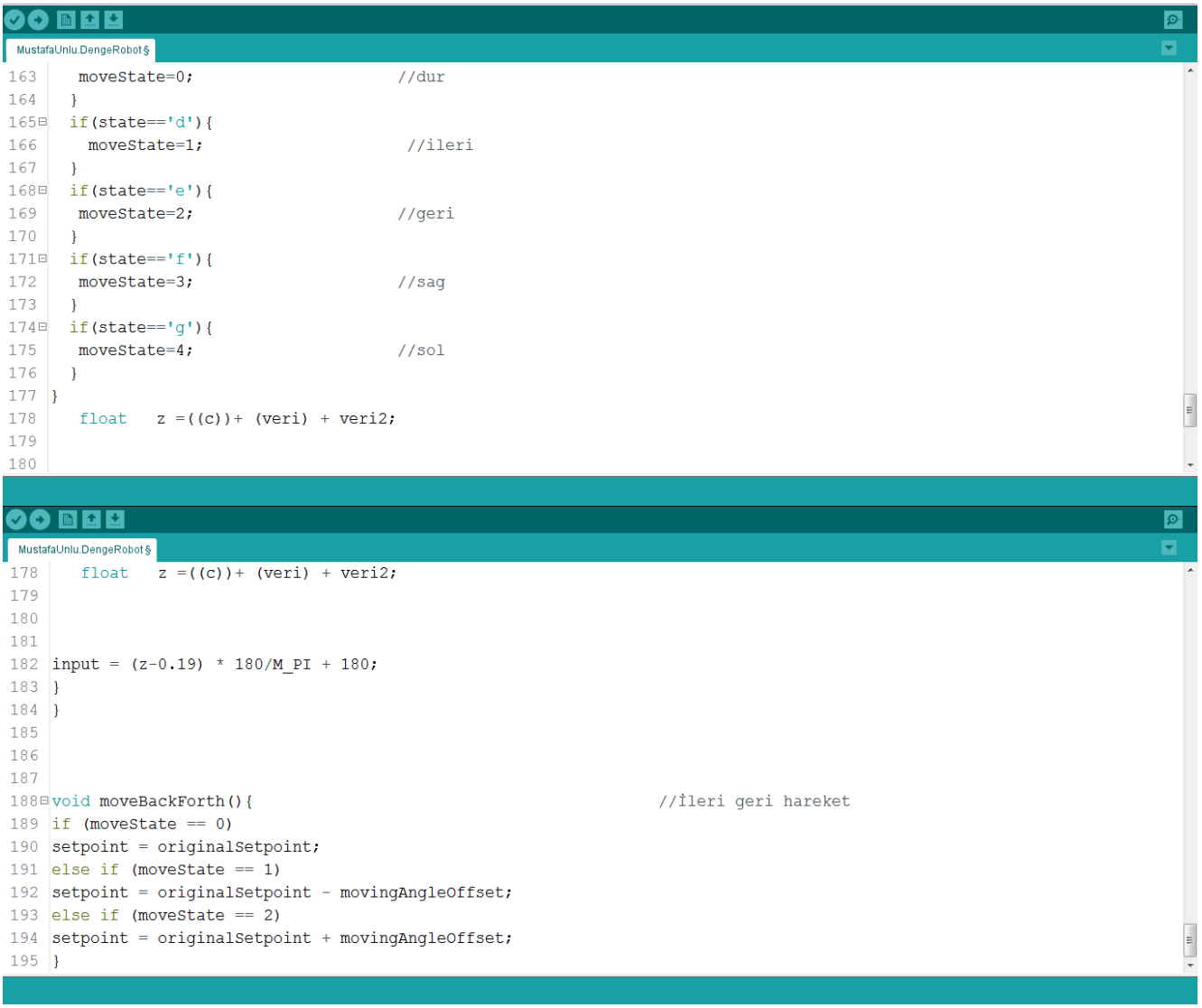
```

}
else if (mpuIntStatus & 0x02)
{
while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
mpu.getFIFOBytes(fifoBuffer, packetSize);
fifoCount -= packetSize;
mpu.dmpGetQuaternion(&q, fifoBuffer);
mpu.dmpGetGravity(&gravity, &q);
mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

float c=ypr[1];
if(Serial.available() > 0){
  state=Serial.read();
  if(state=='1'){
    veri2=0.04;                     //ileri
  }
  if(state=='2'){
```

MustafaUnlu.DengeRobot §

```
    veri2=0;                    //dur
  }
  if(state=='3'){
    veri2=-0.04;                    //geri
  }
  if(state=='+'){
    veri=veri-0.01;                   //denge ayarı
  }
  if(state=='-'){
    veri=veri+0.01;                   //denge ayarı
  }
  if(state=='a'){
    veri=veri-0.005;                   //hassas denge ayarı
  }
  if(state=='b'){
    veri=veri+0.005;                    //hassas denge ayarı
  }
  if(state=='c'){
```



```
MustafaUnlu.DengeRobot §
163     moveState=0;                    //dur
164    }
165    if(state=='d'){
166      moveState=1;                    //ileri
167    }
168    if(state=='e'){
169     moveState=2;                    //geri
170    }
171    if(state=='f'){
172     moveState=3;                    //sag
173    }
174    if(state=='g'){
175     moveState=4;                    //sol
176    }
177  }
178     float   z =((c))+ (veri) + veri2;
179
180
```

```
MustafaUnlu.DengeRobot §
178     float   z =((c))+ (veri) + veri2;
179
180
181
182  input = (z-0.19) * 180/M_PI + 180;
183  }
184  }
185
186
187
188  void moveBackForth(){                                    //İleri geri hareket
189  if (moveState == 0)
190  setpoint = originalSetpoint;
191  else if (moveState == 1)
192  setpoint = originalSetpoint - movingAngleOffset;
193  else if (moveState == 2)
194  setpoint = originalSetpoint + movingAngleOffset;
195  }
```

```
#//MUSTAFA ÜNLÜ DENGE ROBOTU KODU 161202112
/*
BLUETOOTH İLE KONTROL İÇİN
DENGE HASSAS AYARI == (+),(-)
DENGE HASSAS İNCE AYARI == (a),(b)
İLERİ HIZLI == (1)
DUR == (2)
GERİ HIZLI == (3)
DUR NORMAL == (c)
İLERİ NORMAL == (d)
GERİ NORMAL == (e)
SAG DÖN == (f)
SOL DÖN == (g)
*/
#include <PID_v1.h>
#include <LMotorController.h>
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include "Wire.h"
#define MIN_ABS_SPEED 0       // Minumum Baslangıc Hızı
#include <SoftwareSerial.h>
MPU6050 mpu;
```

```
float veri;
char state;
float veri2;
bool dmpReady = false;              // DMP init başarılı olmuşsa doğru olarak ayarlayın
uint8_t mpuIntStatus;               //MPU'dan gerçek kesme durumu baytı tutar
uint8_t devStatus;                  //Her cihaz çalışmasından sonra dönüş durumu (0 = başarı,! 0 = hata) (0 = success, !0 = error)
uint16_t packetSize;                //Beklenen DMP paket boyutu (varsayılan 42 bayt)
uint16_t fifoCount;                 //Şu anda FIFO'daki tüm baytların sayısı
uint8_t fifoBuffer[64];             //FIFO depolama arabelleği
Quaternion q;
VectorFloat gravity;
float ypr[3];
double originalSetpoint = 177;  //173
double setpoint = originalSetpoint;
double movingAngleOffset = 3.3;  //0.3
double input, output;
int moveState=0;
double Kp = 35;   //40
double Kd = 0.7;  //1
double Ki = 70;   //50
PID pid(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);
double motorSpeedFactorLeft = 0.55;
double motorSpeedFactorRight = 0.45;
// motor sürücü
int ENA = 5;
int IN1 = 6;
int IN2 = 7;
int IN3 = 8;
int IN4 = 9;
int ENB = 10;
LMotorController motorController(ENA, IN1, IN2, ENB, IN3, IN4, motorSpeedFactorLeft, motorSpeedFactorRight);
volatile bool mpuInterrupt = false;             //MPU kesme pininin yüksek olup olmadığını gösterir
void dmpDataReady()
{
mpuInterrupt = true;
}
void setup()
{
Serial.begin(9600);
//I2C veriyoluna katıl (I2Cdev kütüphanesi bunu otomatik olarak yapmaz)
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
Wire.begin();
TWBR = 24;
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
Fastwire::setup(400, true);
#endif
while (!Serial);
mpu.initialize();
devStatus = mpu.dmpInitialize();
//burada kendi gyro ofsetlerini, minimum hassasiyet için ölçeklendirdim
mpu.setXGyroOffset(128); //128
mpu.setYGyroOffset(50); //50
mpu.setZGyroOffset(-100); //-100
mpu.setZAccelOffset(1688); // 1688
//Çalıştığından emin ol (eğer öyleyse 0 döndürür)
if (devStatus == 0)
{
//Hazır olduğunda, DMP'yi aç
mpu.setDMPEnabled(true);
```

```
//Arduino kesme algılamasını etkinleştir
attachInterrupt(0, dmpDataReady, RISING);
mpuIntStatus = mpu.getIntStatus();
dmpReady = true;
packetSize = mpu.dmpGetFIFOPacketSize();
pid.SetMode(AUTOMATIC);
pid.SetSampleTime(5);
pid.SetOutputLimits(-255, 255);
}
}
void loop()
{
moveBackForth();
//MPU kesmesi veya mevcut ek paket (ler) i bekle
while (!mpuInterrupt && fifoCount < packetSize)
{
if(moveState==3){                                   //Sağ taraf
setpoint = originalSetpoint - 1;
pid.Compute();
motorController.turnLeft(output, 120);
delay(1);
}
else if(moveState==4){                              //Sol taraf
setpoint = originalSetpoint + 1;
pid.Compute();
motorController.turnRight(output, 120);
delay(1);
}
else if(moveState==0||moveState==1||moveState==2){
pid.Compute();
motorController.move(output, MIN_ABS_SPEED);
delay(1);
}
}
//kesme işaretini sıfırla ve INT_STATUS baytını al
mpuInterrupt = false;
mpuIntStatus = mpu.getIntStatus();
//mevcut FIFO sayısını al
fifoCount = mpu.getFIFOCount();
//taşma olup olmadığını kontrol et
if ((mpuIntStatus & 0x10) || fifoCount == 1024)
{
//sıfırla, böylece temiz bir şekilde devam edebiliriz
mpu.resetFIFO();
}
//Aksi takdirde, DMP veri hazır interrupt'ını kontrol edin (bu sık sık yapılmalıdır)
else if (mpuIntStatus & 0x02)
{

//Doğru kullanılabilir veri uzunluğunu bekle, ÇOK kısa bir bekleyiş olmalı
while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
//FIFO'dan bir paket oku
mpu.getFIFOBytes(fifoBuffer, packetSize);
fifoCount -= packetSize;
mpu.dmpGetQuaternion(&q, fifoBuffer);
mpu.dmpGetGravity(&gravity, &q);
mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
float c=ypr[1];
if(Serial.available() > 0){
```

```
  state=Serial.read();
  if(state=='1'){
    veri2=0.04;                    //ileri
  }
  if(state=='2'){
    veri2=0;                    //dur
  }
  if(state=='3'){
    veri2=-0.04;                    //geri
  }
  if(state=='+'){
    veri=veri-0.01;                    //denge ayarı
  }
  if(state=='-'){
    veri=veri+0.01;                    //denge ayarı
  }
  if(state=='a'){
    veri=veri-0.005;                    //hassas denge ayarı
  }
  if(state=='b'){
    veri=veri+0.005;                    //hassas denge ayarı
  }
  if(state=='c'){
   moveState=0;                    //dur
  }
  if(state=='d'){
    moveState=1;                    //ileri
  }
  if(state=='e'){
   moveState=2;                    //geri
  }
  if(state=='f'){
   moveState=3;                    //sag
  }
  if(state=='g'){
   moveState=4;                    //sol
  }
}
   float   z =((c))+ (veri) + veri2;
input = (z-0.19) * 180/M_PI + 180;
}
}
void moveBackForth(){                                    //İleri geri hareket
if (moveState == 0)
setpoint = originalSetpoint;
else if (moveState == 1)
setpoint = originalSetpoint - movingAngleOffset;
else if (moveState == 2)
setpoint = originalSetpoint + movingAngleOffset;
}
```