Number: 2015-1
FEN VE TEKNOLOJİ BİLGİ PAYLAŞIMI
SCIENCE AND TECHNOLOGY INFORMATION SHARING
Article Web Page: www.ibrahimcayiroglu.com

İki Tekerlekli Dengede Durabilen Robotlar

(Two Wheel Balance Robots)

Erdoğan KURŞUNCU* , Aytül BOZKURT**

*Karabük Üniversitesi, Mühendislik Fakültesi, Mekatronik Mühendisliği, 78050, Karabük, k_kursuncu@hotmail.com
**Karabük Üniversitesi, Mühendislik Fakültesi, Mekatronik Mühendisliği, 78050, Karabük, aytulbozkurt@karabuk.edu.tr

Anahtar Kelimeler:
İki Tekerlekli Dengede Durabilen Robotlar

Özet: Bu makalede İki Tekerlekli Dengede Durabilen Robotların genel bir tanıtımı yapıldıktan sonra uygulamalı bir örneği verilmiştir. İki tekerlekli dengede durabilen robotlar günümüzde birçok alanda kullanılmaya başlanmış bir robottur. Verilen örnek direk uygulanıp denenebilir. Kavramların daha iyi anlaşılması için hem Türkçe alternatif anlatımları hem de İngilizce karşılıkları verilmiştir.

Keywords:
Two Wheel Balance Robots

Abstract: In this article, given an practical example of Two Wheel Balance Robots after a general introduction. This robot is that had been used in many fields.The examples given are applied directly. Alternative explanations for better understanding of concepts in both Turkish and English equivalents are given.

1. Giriş

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

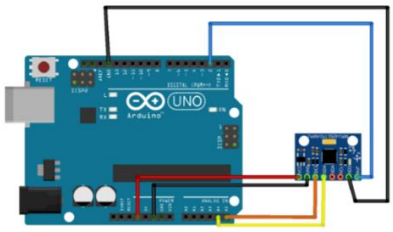
Çapı: 70mm
Kalınlık: 30mm
Ağırlık: 45gr

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.

Özellikleri

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 Kartı

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
1 adet step 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

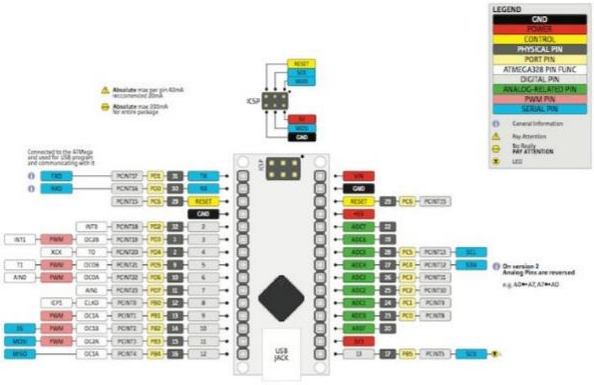
Şekil c. L298N Motor Sürücü Kartı

2.4. Arduino Nano

Arduino Nano; Atmega328 temelli bir mikrodenetleyici kartıdır. Üzerinde 14 adet dijital giriş/çıkış pini (6 tanesi PWM çıkışı olarak kullanılabilir), 8 analog giriş, 16Mhz kristal, usb soketi, ICSP konektörü ve reset tuşu bulundurmaktadır. Kart üzerinde mikrodenetleyicinin çalışması için gerekli olan her şey bulunmaktadır. Kolayca usb kablosu üzerinden bilgisayara bağlanabilir, adaptör veya pil ile çalıştırılabilir.

Teknik Özellikler

Mikrodenetleyici ATmega328
Çalışma Gerilimi 5V
Giriş Gerilimi (önerilen) 7-12V
Giriş Gerilimi (limit) 6-20V
Dijital I/O Pinleri 14 (6 tanesi PWM çıkışı)
Analog Giriş Pinleri 8
Her I/O için Akım 40 mA
3V Çıkış için Akım 50 mA
Flash Hafıza 32 KB (ATmega328) 2 KB kadarı bootloader
tarafından kullanılmaktadır -SRAM 2 KB (ATmega328)
EEPROM 1 KB (ATmega328)
Saat Hızı 16 MHz
Uzunluk 45 mm
Genişlik 18 mm
Ağırlık 5 g



Şekil d. Arduino Nano ve Çalışma Prensibi

Güç:

Arduino Nano gücünü usb üzerinden veya harici güç kaynağından alabilir. Harici güç kaynağı AC-DC adaptör olabileceği gibi bataryada olabilir. Adaptör ve batarya kart üzerindeki GND ve Vin pinleri üzerinden bağlanabilir. Kartın çalışması için sürekli olarak usb'nin bağlı olması şart değildir. Kart sadece adaptör veya batarya ile çalıştırılabilir. Bu sayede kart bilgisayardan bağımsız olarak

çalıştırılabilir. Harici güç kaynağı olarak 6-20V arası kullanılabilir. Ancak bu değerler limit değerleridir. Kart için önerilen harici besleme 7-12V arasıdır. Çünkü kart üzerinde bulunan regülatör 7V altındaki değerlerde stabil çalışmayabilir. 12V üstündeki değerlerde de aşırı ısınabilir. Nano kartının üzerindeki mikrodenetleyicinin çalışma gerilimi 5V'dur. Vin pini veya güç soketi üzerinden verilen 7-12V arası gerilim kart üzerinde bulunan voltaj regülatörü ile 5V'a düşürülerek karta dağılır.

Güç pinleri aşağıdaki gibidir:

-VIN: Harici güç kaynağı kullanılırken 7-12V arası gerilim giriş pini.
-5V: Bu pin regülatörden çıkan 5V çıkışı verir. Eğer kart sadece usb (5V) üzerinden çalışıyor ise usb üzerinden gelen 5V doğrudan bu pin üzerinden çıkış olarak verilir. Aynı zamanda bu pin üzerinden 5V girişi yapılabilir. Eğer karta güç Vin (7-12V) üzerinden veriliyorsa regülatörden çıkan 5V doğrudan bu pin üzerinden çıkış olarak verilir.
-3V3: Kart üzerinde bulunan 3.3V regülatörü çıkış pinidir. Maks. 50mA çıkış verebilir.
-GND: Toprak pinleridir.

Hafıza:

Atmega328 32 KB'lık flash belleğe sahiptir (2 KB kadarı bootloader tarafından kullanılmaktadır). 2 KB SRAM ve 1 KB EEPROM'u bulunmaktadır.

Giriş ve Çıkış:

Nano üzerindeki 14 adet dijital pinin hepsi giriş veya çıkış olarak kullanılabilir. 8 tane analog giriş pinide bulunmaktadır. Bu analog giriş pinleride aynı şekilde dijital giriş ve çıkış olarak kullanılabilir. Yani kart üzerinde toplam 20 tane dijital giriş çıkış pini vardır. Bu pinlerin tamamının lojik seviyesi 5V'dur. Her pin maks. 40mA giriş ve çıkış akımı ile çalışır. Ek olarak, bazı pinlerin farklı özellikleri bulunmaktadır. Özel pinler aşağıda belirtildiği gibidir.

-Seri Haberleşme, 0 (RX) ve 1 (TX): TTL Seri veri alıp (RX), vermek (TX) için kullanılır. Bu pinler doğrudan kart üzerinde bulunan FT232 usb-seri dönüştürücüsüne bağlıdır. Yani bilgisayardan karta kod yüklerken veya bilgisayar-nano arasında karşılıklı haberleşme yapılırkende bu pinler kullanılır. O yüzden karta kod yüklerken veya haberleşme yapılırken hata olmaması için mecbur kalınmadıkça bu pinlerin kullanılmamasında fayda vardır.

-Harici Kesme, 2 (interrupt 0) ve 3 (interrupt 1): Bu pinler yükselen kenar, düşen kenar veya değişiklik kesmesi pinleri olarak kullanılabilir. Ayrıntılı bilgi için attachInterrupt() fonksiyon sayfasını inceleyebilirsiniz.

-PWM, 3,5,6,9,10 ve 11: 8-bit çözünürülükte PWM çıkış pinleri olarak kullanılabilir.

-SPI, 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK): Bu pinler SPI haberleşmesi için kullanılır.

-LED, 13: Nano üzerinden 13. pine bağlı olan dahili bir led bulunmaktadır. Pin HIGH yapıldığında led yanacak, LOW yapıldığında led sönecektir.

-Analog, A0-A7: Nano 8 tane 10-bit çözünürlüğünde analog giriş pinine sahiptir. Bu pinler dijital giriş ve çıkış içinde kullanılabilir. Pinlerin ölçüm aralığı 0-5V'dur. AREF pini ve analogReference() foksiyonu kullanılarak alt limit yükseltilip, üst limit düşürülebilir.

-I2C, A4 veya SDA pini ve A5 veya SCL pini: Bu pinler I2C haberleşmesi için kullanılır.

-AREF: Analog giriş için referans pini.

-Reset: Mikrodenetleyici resetlenmek istendiğinde bu pin LOW yapılır. Reset işlemi kart üzerinde bulunan Reset Butonu ile de yapılabilir.

Haberleşme:

Arduino Nano'nun bilgisayarla, başka bir arduino veya mikrodenetleyici ile haberleşmesi için bir kaç farklı seçenek vardır. Atmega328, 0 (RX) ve 1 (TX) pinleri üzerinden UART TTL (5V) seri haberleşme imkanı sunar. Kart üzerinde bulunan FT232 usb-seri dönüştürücüde bilgisayarda sanal bir com port açarak Atmega328 ile bilgisayar arasında bir köprü kurar. Arduino bilgisayar programı içerisinde barındırdığı seri monitör ile arduino ile bilgisayar arasında text temelli bilgilerin gönderilip alınmasını sağlar. Usb-seri dönüştürücü ile bilgisayar arasında usb üzerinden haberleşme olduğu zaman kart üzerinde bulunan RX ve TX ledleri yanacaktır. Nano üzerinde donanımsal olarak bir adet seri port bulunmaktadır. Ancak SoftwareSerial kütüphanesi ile bu sayı yazılımsal olarak arttırılabilir. Atmega328 aynı şekilde I2C ve SPI portlarıda sağlamaktadır. Arduino bilgisayar programı ile gelen Wire kütüphanesi I2C kullanımını, SPI kütüphanesi de SPI haberleşmesini sağlamak için kullanılır.

Programlama:

Arduino Nano kartı Arduino bilgisayar programı (Arduino IDE) ile programlanır. Programda Tools > Board sekmesi altında Arduino Nano'yu seçip programlamaya başlayabilirsiniz. Arduino Nano üzerindeki Atmega328 üzerine bootloader denilen özel bir yazılım yüklü gelir. Bu sayede kartı programlarken ekstra bir programlayıcı kullanmanıza gerek yoktur.Haberleşme orjinal STK500 protokolü ile sağlanır. Bootloader yazılımı bypass edilerek kart doğrudan mikrodenetleyicinin ICSP header'i üzerinden ISP programlayıcı ile programlanabilir.

3.Projenin Arduino Kodları

```
#include "I2Cdev.h"

#include "MPU6050_6Axis_MotionApps20.h"

#if I2CDEV_IMPLEMENTATION ==

I2CDEV_ARDUINO_WIRE

#include "Wire.h"

#endif

MPU6050 mpu;

#define
OUTPUT_READABLE_YAWPITCHROLL

#define LED_PIN 13

bool blinkState = false;

double P = 300 , D = 28000 , I = 1.2 ;

#define AIL LOW

#define AILPIN 2

#define AILMID 1500

#define ROLL HIGH

#define ROLLPIN 3

#define ROLLMID 1500

volatile int AILin = AILMID; volatile unsigned
long StartAIL = 0; volatile boolean NewAIL =
false;

volatile int ROLLin = ROLLMID; volatile
unsigned long StartROLL = 0; volatile boolean
NewROLL = false;

unsigned long sondurum; double Ihata, sonhata,
hata, output, ref, Iout; float roll, aileron,
kumandaail, pitch, elevator, kumandaelev, in;

double motor1 = 0 , motor2 = 0 ; double output1,
output2, sagsol;

const int pwm_sol = 5; const int solmotor_in1 = 6;
const int solmotor_in2 = 4;

const int sagmotor_in1 = 8; const int sagmotor_in2
= 7; const int pwm_sag = 9;

// MPU control/status vars bool dmpReady = false;
uint8_t mpuIntStatus; uint8_t devStatus; uint16_t
packetSize; uint16_t fifoCount; uint8_t
fifoBuffer[64];

// orientation/motion vars

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

int something;

volatile bool mpuInterrupt = false; void
dmpDataReady() { mpuInterrupt = true;

}

void setup() { // Serial.begin(9600);

pinMode(solmotor_in1, OUTPUT);
pinMode(solmotor_in2, OUTPUT);
pinMode(sagmotor_in1, OUTPUT);

pinMode(sagmotor_in2, OUTPUT);
pinMode(pwm_sol, OUTPUT);
pinMode(pwm_sag, OUTPUT); pinMode(13,
OUTPUT);

#if I2CDEV_IMPLEMENTATION ==

I2CDEV_ARDUINO_WIRE

Wire.begin();

TWBR = 24; // 400kHz I2C clock (200kHz if CPU
is
```

```
8MHz)

#elif I2CDEV_IMPLEMENTATION ==

I2CDEV_BUILTIN_FASTWIRE

Fastwire::setup(400, true);

#endif

while (!Serial);

Serial.println(F("Initializing I2C devices..."));
mpu.initialize();

Serial.println(F("Testing device connections..."));
Serial.println(mpu.testConnection() ? F("MPU6050
connection successful") : F("MPU6050 connection
failed"));

Serial.println(F("\nSend any character to begin
DMP programming and demo: "));

Serial.println(F("Initializing DMP...")); devStatus =
mpu.dmpInitialize();

mpu.setXGyroOffset(220);
mpu.setYGyroOffset(76); mpu.setZGyroOffset(-
85); mpu.setZAccelOffset(1788);

if (devStatus == 0) {

Serial.println(F("Enabling DMP..."));
mpu.setDMPEnabled(true);

Serial.println(F("Enabling interrupt detection
(Arduino

external interrupt 0)...")); attachInterrupt(0,
dmpDataReady, RISING); mpuIntStatus =
mpu.getIntStatus();

Serial.println(F("DMP ready! Waiting for first

interrupt...")); dmpReady = true;

packetSize = mpu.dmpGetFIFOPacketSize();

}

{

attachInterrupt(AIL, AILhesapla, CHANGE);
attachInterrupt(ROLL, ROLLhesapla, CHANGE);

}

}

void loop() { digitalWrite(13,HIGH);

bekle:

//////////**************************************
********

*************************

if (NewAIL)

{

// Serial.println(AILin);

ref = map(AILin, 1064, 1948, -35, 35); //MIDLE
1460 ref = ref / 100;

// if ((ref>-0.1)&&(ref<0.1)) ref=0;

NewAIL = false;

}

if (NewROLL)

{

// Serial.println(ROLLin);

sagsol = map(ROLLin, 1056, 1920, -100, 100);
//MIDLE

1460

if ((sagsol>-10)&&(sagsol<10)) sagsol=0;

NewROLL = false;

}

///////**************************************
********

*************************

mpuInterrupt = false; mpuIntStatus =
mpu.getIntStatus();

fifoCount = mpu.getFIFOCount();

if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
```

```
mpu.resetFIFO();

} 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); //
Serial.println(ypr[1] * 180/M_PI); something =
(ypr[1] * 180 / M_PI);

}

/////*****************************************
*******

*****************************************
********

if (ypr[1]<-1) { dur(); Ihata=0; Iout=0;

Serial.println("robotu dik konuma getirin"); goto
bekle;

}

if (ypr[1]>1)

{ dur(); Ihata=0; Iout=0;

Serial.println("robotu dik konuma getirin"); goto
bekle;

}

PID();

motorayar();

ekranpid();

}

//////////*************************************
********

*****************************************
******

void PID()

{

unsigned long simdi = millis(); double
zamandegisimi = (double)(simdi - sondurum);

double hata = -0.07- ypr[1]+ref;

Ihata += (hata * zamandegisimi); double Lim =
250; double K=Lim/I; if (Ihata<-K) Ihata=-K; if
(Ihata>K) Ihata=K; Iout = I * Ihata; if (Iout < -Lim)
Iout = -Lim; if (Iout > Lim) Iout = Lim;

double dhata = (hata - sonhata) / zamandegisimi;

output = ( P * hata + Iout + D * dhata);

if (output > 250) output = 250; if (output < -250)
output = -250;

sonhata = hata; sondurum = simdi;

}

////////////////////*****************************
*******

*****************************************
********

** void motorayar()

{

motor1=output-sagsol; motor2=output+sagsol; if
(motor1<-250) {motor1=-250;} if (motor1>250){
motor1=250;}

if (motor2<-250) {motor2=-250;} if (motor2>250){
motor2=250;}

/////*************************************
*** if (motor1 <= 0)

{

motor1 = -motor1;

sol_geri(); analogWrite(pwm_sol, motor1);

} else if (motor1 > 0)

{

motor1 =motor1;

sol_ileri(); analogWrite(pwm_sol, motor1 );
```

```
}

/////*****************************************
***

if (motor2 <= 0) {

motor2 = -motor2;

sag_geri(); analogWrite(pwm_sag, motor2);

} else if (motor2 > 0)

{

motor2 =motor2;

sag_ileri(); analogWrite(pwm_sag, motor2 );

}

////*****************************************
******

}

void motorkontrol()

{

if (output > 250) output = 250; if (output < -250)
output = -250;

if (output <= 0) { output2 = 0; output1 = -output;

duz(); analogWrite(pwm_sol, output1);
analogWrite(pwm_sag, output1);

} if (output > 0) { output1 = 0; output2 = output;

geri(); analogWrite(pwm_sol, output2 );
analogWrite(pwm_sag, output2);

}

}

//////////*************************************
*********

*****************************************
******** ***

void ekranpid() {

Serial.print(" pitch = "); Serial.print(ypr[1]);

Serial.print(" AILAin = "); Serial.print(AILin);

// Serial.print(" output2= "); Serial.print(output2);

Serial.print(" ref = "); Serial.print(ref);

Serial.print(" ROLLin = "); Serial.print(ROLLin);

Serial.print(" sagsol = "); Serial.println(sagsol);

Serial.print(" output = "); Serial.print(output);

Serial.print(" motor1 = "); Serial.print(motor1);

Serial.print(" motor2 = "); Serial.print(motor2);

// Serial.println();

}

//////*****************************************
*********

*****************************************
*********

**********

void duz() { sag_ileri(); sol_ileri(); } void geri() {
sag_geri(); sol_geri(); } void sag_ileri() {
digitalWrite(sagmotor_in1, LOW);
digitalWrite(sagmotor_in2, HIGH);

} void sag_geri() { digitalWrite(sagmotor_in1,
HIGH); digitalWrite(sagmotor_in2, LOW);

} void sol_ileri() { digitalWrite(solmotor_in1,
LOW); digitalWrite(solmotor_in2, HIGH);

} void sol_geri() { digitalWrite(solmotor_in1,
HIGH); digitalWrite(solmotor_in2, LOW);

} void dur() { analogWrite(pwm_sol, 0);
analogWrite(pwm_sag, 0);

}

//////*************************************
*********
*****************************************
******

void AILhesapla()
```

```
{
if (digitalRead(AILPIN) == HIGH)
{
StartAIL = micros(); }
else {
if (StartAIL && (NewAIL == false))
{
AILin = (int)(micros() - StartAIL);
StartAIL = 0;
NewAIL = true;
}
}
}
//////////**************************************
********
**********************************
void ROLLhesapla()
{
if (digitalRead(ROLLPIN) == HIGH)
{
StartROLL = micros();
} else {
if (StartROLL && (NewROLL == false))
{
ROLLin = (int)(micros() - StartROLL);
StartROLL = 0;
NewROLL = true;
    }
  }
 }
```

4. Proje'nin Genel Çalışma Mantığı

Projemizde Arduino ile kendini dengeleyen robot üzerine çalışılmıştır. Yapmış bulunduğumuz denge robotu uzaktan Kumanda ile 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.

5. Gyro Çalışma Prensibi

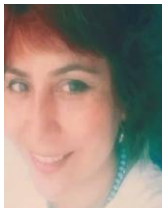
Gyro, (Jiroskop) yön ölçümü veya ayarlamasında kullanılan, açısal dengenin korunması ilkesiyle çalışan bir alettir. Jiroskopik hareketin temeli fizik kurallarına ve açısal momentumun korunumu ilkesine dayalıdır. Gyro, yön belirleme amaçlı kullanılmaktadır. Örneğin; Jiroskopa sahip telefonu yan çevirdiğinizde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

6. PID Çalışma Prensibi

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.

The Author

Erdoğan Kurşuncu *is a student in Mechatronic Engineering at Karabuk University, Turkey. He is born in Yıldırım / BURSA. Autocad,Solid, Ansys, Matlab, Visual Studio, Plc, C++, Arduino and working on smart home systems.*



Aytül BOZKURT is an currently Assistant Prof. in Mechatronic Engineering at Karabuk University, Turkey. She received her M.Sc. from Gazi University in 2002 and Ph.D. in Computer Engineering from Trakya University. in 2011, respectively. She has done her post-doctoral research study on "Video Transmission over Wireless Networks",from 11/12/2011 in Electrical and Computer Engineering at Stony Brook University, Stony Brook, Newyork, USA. Her research interests include mobile computing, wireless networking and systems, heteregeneous.