AIN SHAMS UNIVERSITY

FACULTY OF ENGINEERING

MECHATRONICS ENGINEERING DEPARTMENT

MCT211 : Automatic control

Project 1 (Angle)

Team 5

NAME	*ID*
Mohamed Emad Mohamed Hussien	***1802245***
Mohammed Abdulaziz Farrag Ahmed	***1805858***
Alaa Elsayed Metwally Mohamed	**1809361**
Ali Abdelhakem Mahmod Hussin	**1805347**
Mohamed Emad Mohamed Moslem	**1806770**
Ahmed Yasser ahmed abelhamed	**1804679**

Submitted By:

Sec: 2

1. Objective and Way of working:

• ***Objective is to make position control system through closed loop feedback system, understand effect of PID parameters on the performance of the system and how to tune them.***

 The system is divided into:

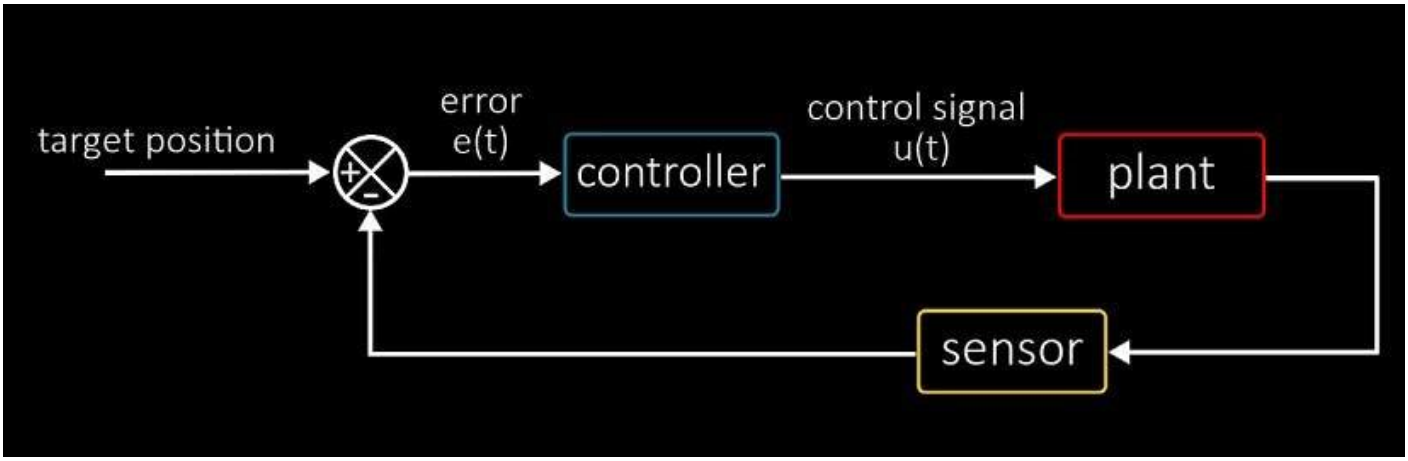
✓ ***Encoder:*** *used for speed control feedback provide mechanism to measure speed of rotor and provide closed feedback to drive precise speed control.*

✓ ***H-bridge:*** *arrange switching the polarity of the voltage applied to a DC motor, thus controlling its direction of rotation.*

✓ ***PWM:*** *speed of the motor by sending a series of pulses the width of the pulses are varied to control the motor speed, pulses with a narrow width will cause the motor to spin quite slowly. Increasing the pulse width will increase the speed of the motor and feeds it to the PWM block output to the motor driver to adjust the motor shaft to the desired position. Note that the direction of the motor is already known from the encoder feedback signals*

✓ ***PID:*** *sensor that compute the desired actuator output by calculating proportional, integral, and derivative responses and summing those three components to compute the output from the last position and set point as zero position which control desired angle and speed of motor.*

2. Block Diagram:



3. Control action, final control element, sensor and controlled variable:

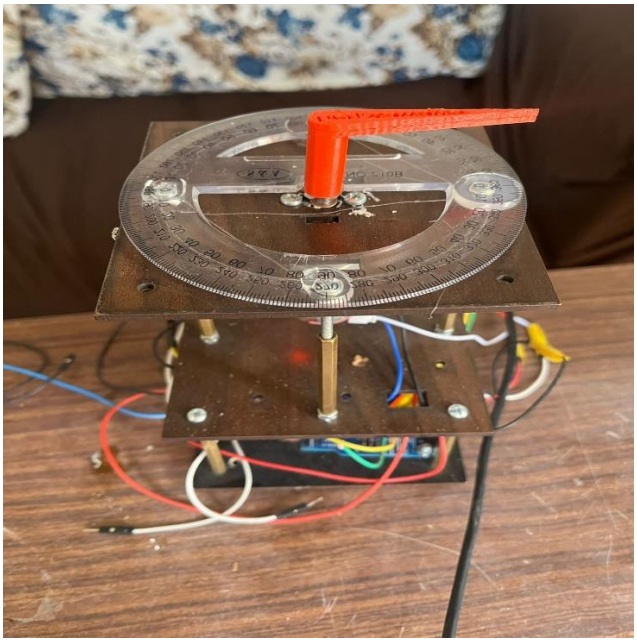
- ***The PID controller controls the system by decreasing the error that comes through the closed loop feedback mechanism using a magnetic encoder attached to the DC motor to achieve the desired the motor position and speed.***

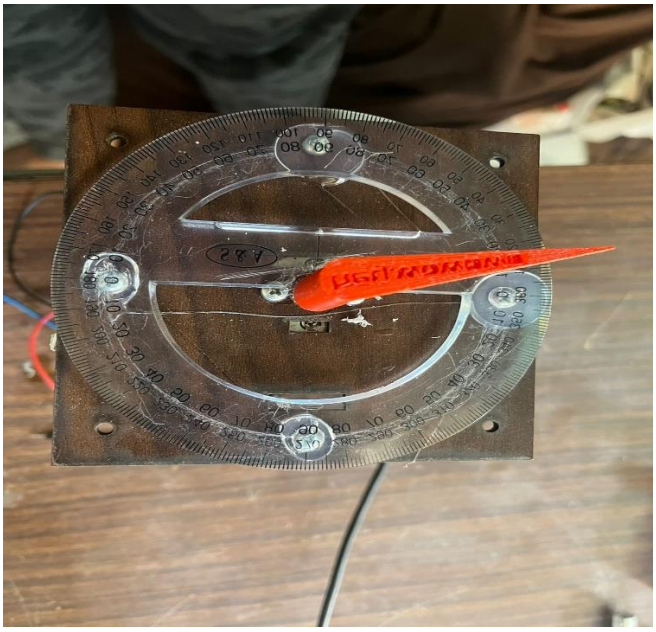
 The final control element is the DC motor.

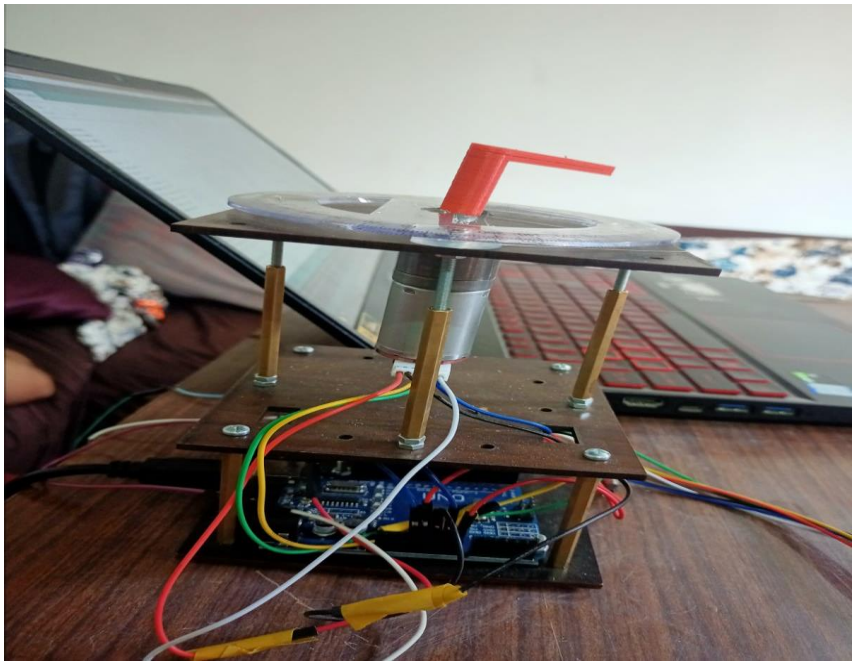
 Our sensor is a magnetic encoder attached to the DC motor *which detects rotational position information as changes of the magnetic field, converts them into electrical signals, and outputs them.*

 The controlled variables are Motor position and speed.

4. Overall Real Hardware System Picture:







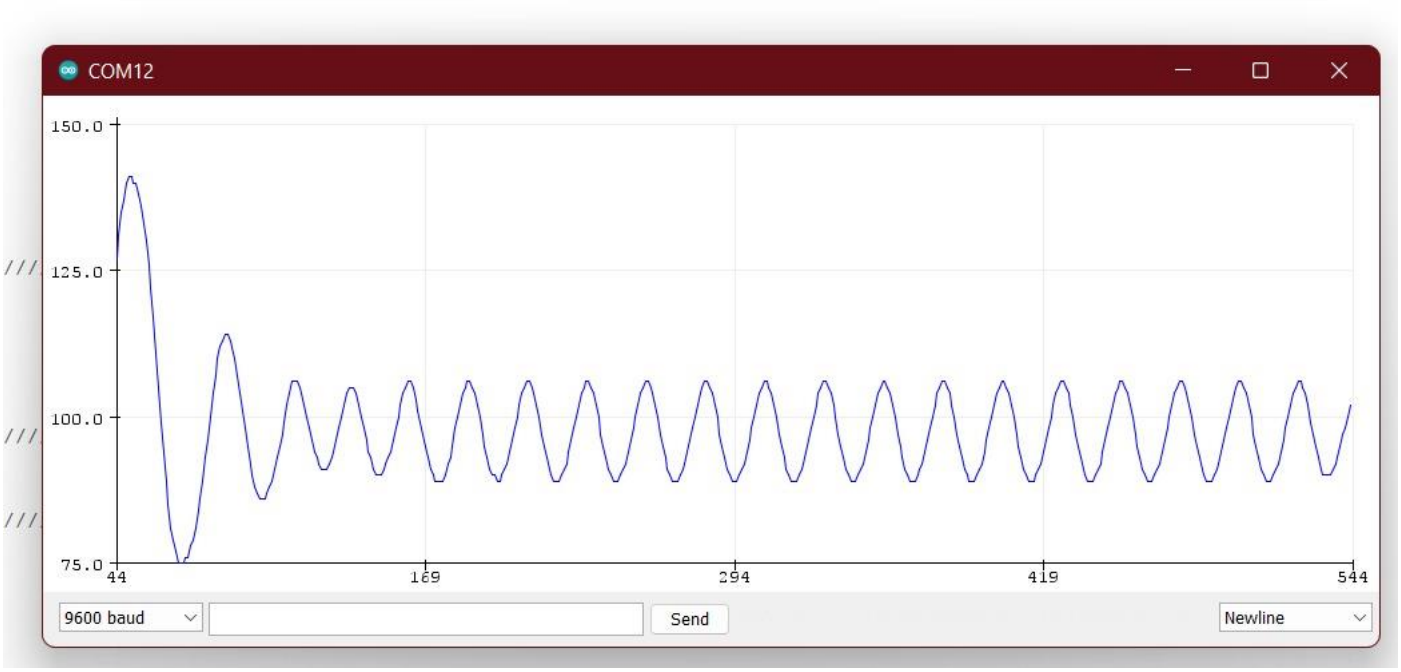
5. Components:

Its Name and function	Picture
DC motor magnetic encoder GM25-360CA *, the motor converts the received current into rotational motion ,while The encoder used for speed control feedback in DC motors where an armature or rotor with wound wires rotates inside a magnetic field created by a stator. The DC motor encoder provides a mechanism to measure the speed of the rotor and provide closed loop feedback to the drive for precise speed control.*	
H-bridge motor driver L298N *which allows speed and direction control of the DC motor.*	

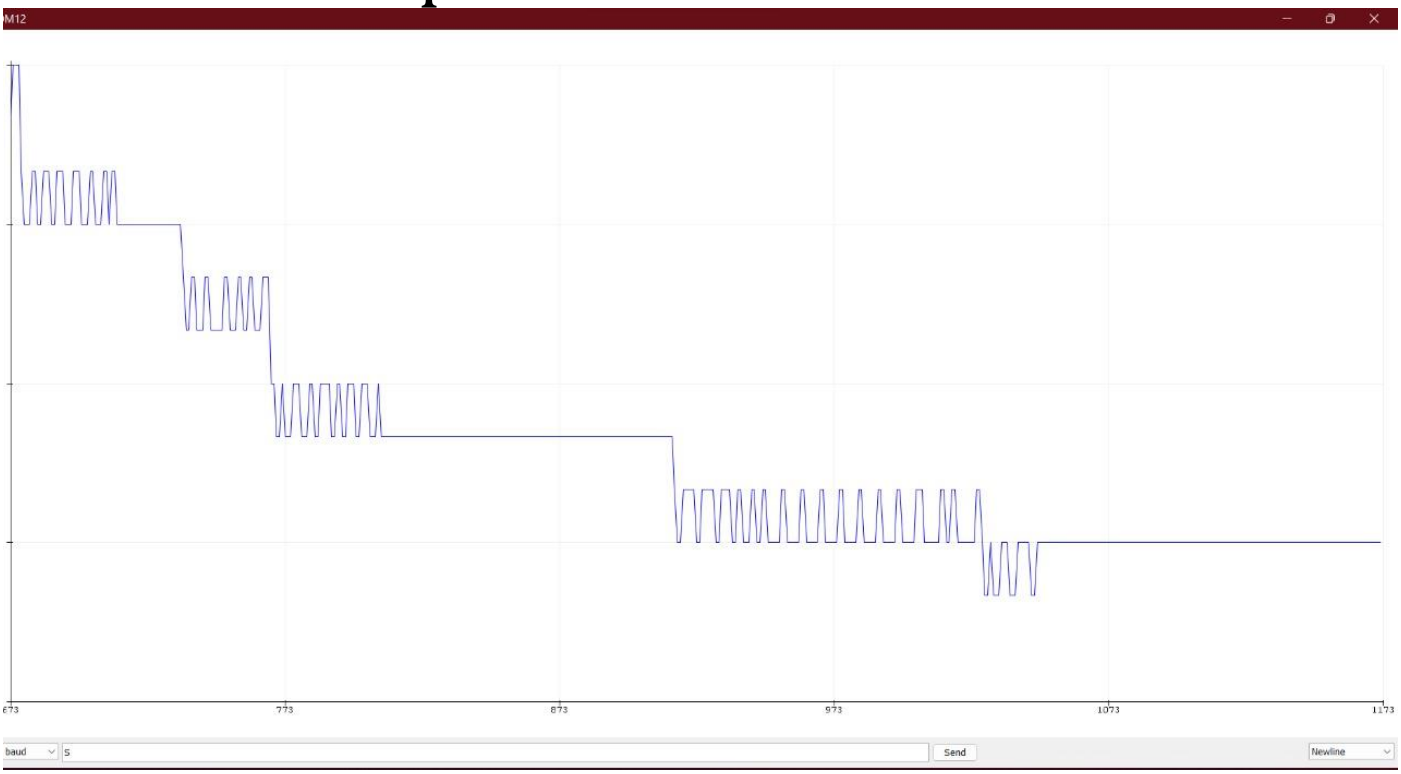
Arduino UNO *is able to read inputs-sensor and turn it into an output - activating a motor*	
Protractor and Pointer *used as visual indicator for the reached motor position.*	
Jumper wires *used to create the circuit of the system and connect the components with each other.*	

6. Trial Graphs: Motor Speed Control Graphs:

kp=15 kd=0 ki=0.05

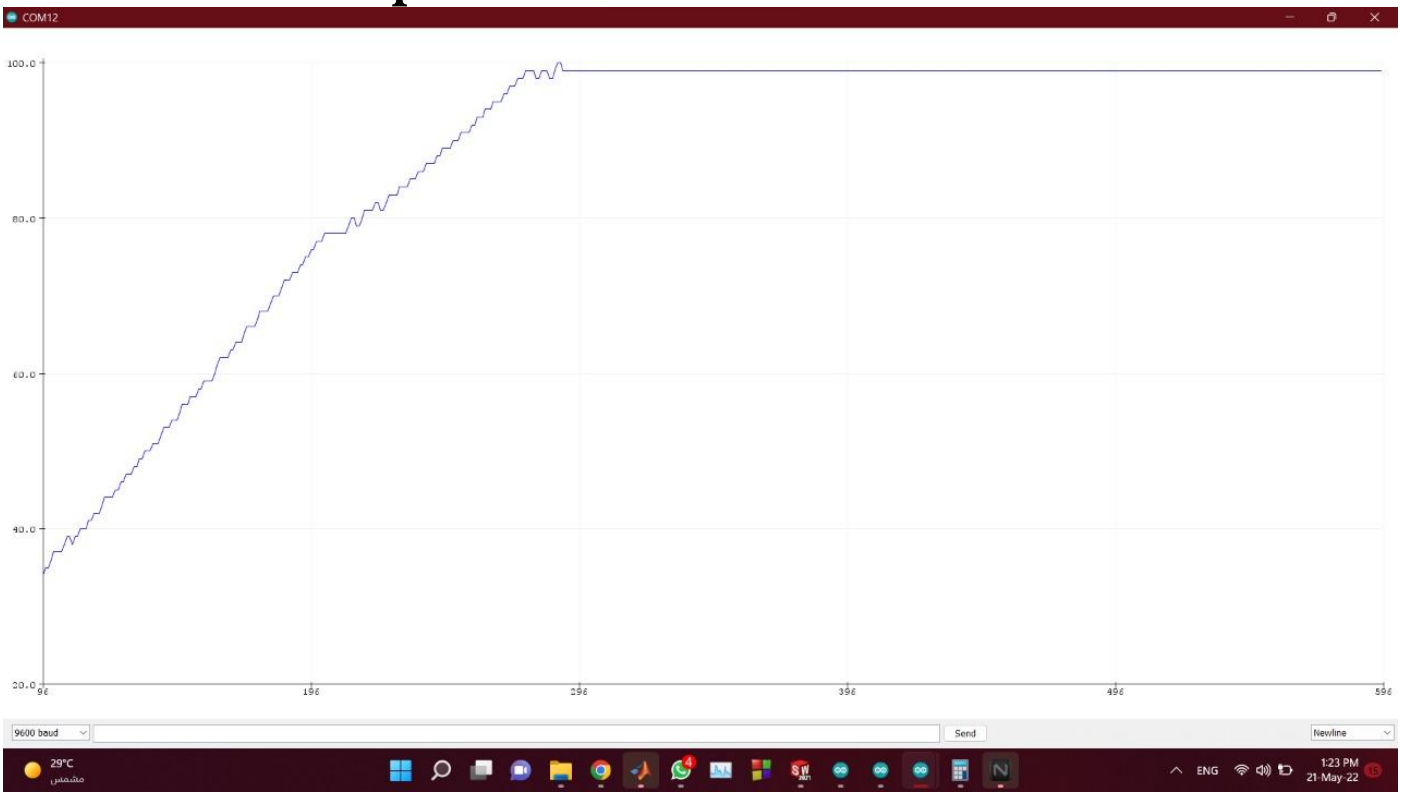


Kp=5 kd=5 ki= 5

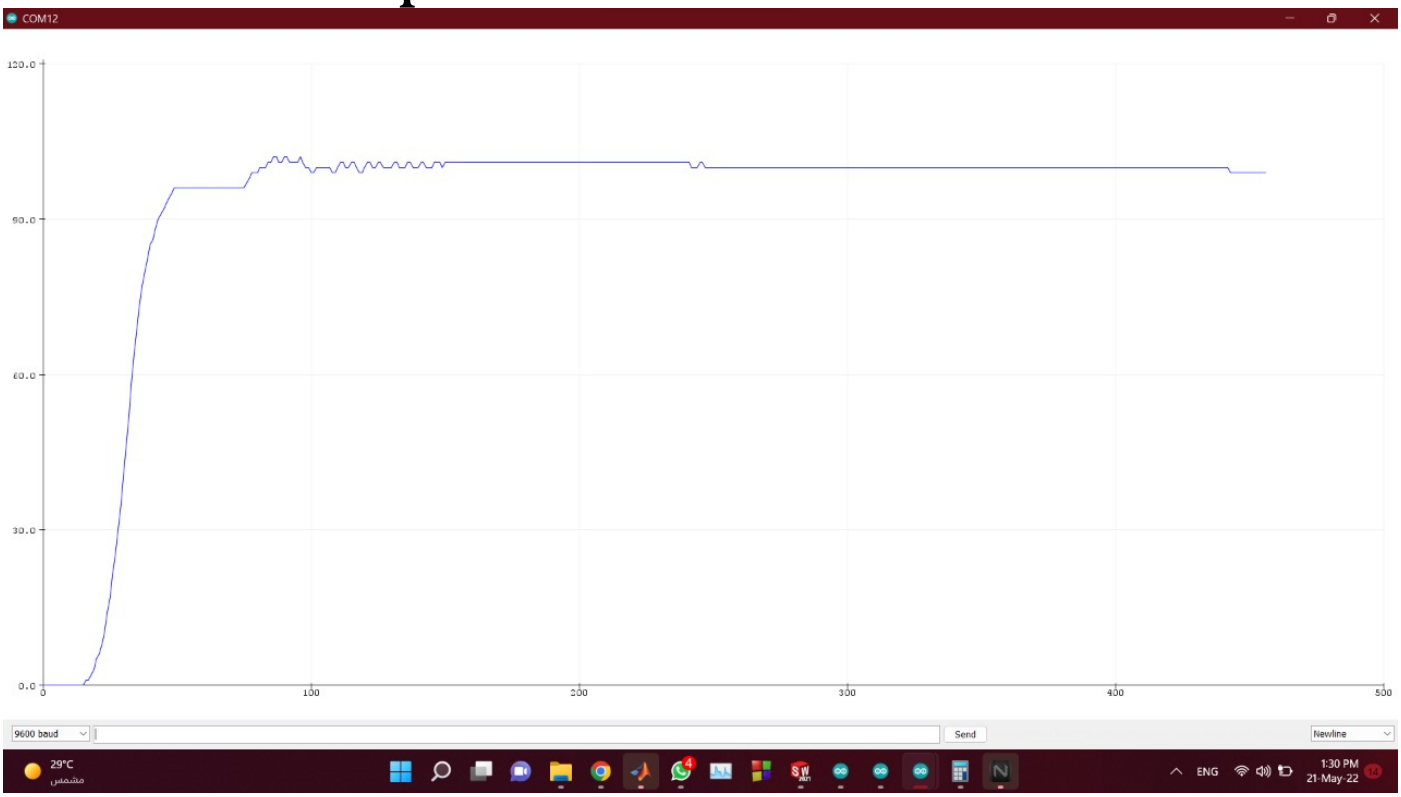


7. motor Position Control Graphs:

kp=10 kd=15 ki=1



kp=50 kd=2 ki =20



8. Final Code:

Position_angle

```
/*****************************************************************/
/* Team     : 5
 * Project  : Position Angle Control
*****************************************************************/

#define ENCA 3
#define ENCB 2
#define PWM 6
#define IN2 9
#define IN1 10

int pos = 0;
int last_pos=0;
long prevT = 0;
float eprev = 0;
float eintegral = 0;
long last_millis=0;
int currentMillis=0;
int interval=100;
int last_error=0;
int sum_error=0;

int control_mode=1;
void setup() {

Serial.begin(9600);
pinMode(ENCA,INPUT);
pinMode(ENCB,INPUT);
attachInterrupt(digitalPinToInterrupt(ENCA),readEncoder,RISING);

pinMode(PWM,OUTPUT);
pinMode(IN1,OUTPUT);
pinMode(IN2,OUTPUT);
last_millis=millis();
```

Done Saving.

```
Sketch uses 4388 bytes (13%) of program storage space. Maximum is 32256 bytes.
Global variables use 208 bytes (10%) of dynamic memory, leaving 1840 bytes for local variables. Maximum is 2048 bytes.
```

Position_angle

```
/*********************Function Definition***********************/

/******************Setmotor function**********************
*Function : configure speed By passing PWM and direction of motor
*/
void setMotor(int dir, int pwmVal, int pwm, int in1, int in2){
analogWrite(pwm,pwmVal);
if(dir == 1){
digitalWrite(in1,LOW);
digitalWrite(in2,HIGH);
   }
else if(dir == -1){
digitalWrite(in1,HIGH);
digitalWrite(in2,LOW);
   }
else{
digitalWrite(in1,LOW);
digitalWrite(in2,LOW);
   }
}


/***************readEncoder****************
/*Function : Read position of Encoder and setting the position to desired position
*/
void readEncoder(){
int b = digitalRead(ENCB);
if(b > 0){
pos++;
}
else{
pos--;
}
}
```

Position_angle

```
{
pwr = 180;
}else
{
  pwr=0;
}
int dir = 1; // Motor Direction
if(u<0){
dir = -1;
}

setMotor(dir,pwr,PWM,IN1,IN2); // Passing Parameter to Funtion

Serial.println(pos);
Serial.print(" ");
}
}

/*********************Function Definition***********************/

/******************Setmotor function***********************
*Function : configure speed By passing PWM and direction of motor
*/
void setMotor(int dir, int pwmVal, int pwm, int in1, int in2){
analogWrite(pwm,pwmVal);
if(dir == 1){
digitalWrite(in1,LOW);
digitalWrite(in2,HIGH);
   }
else if(dir == -1){
digitalWrite(in1,HIGH);
digitalWrite(in2,LOW);
   }
else{
```

```
}
void loop() {

if(control_mode==1)
{
int x=180;
int target_position=map(x,0,360,0,395); //Mapping to convert setangle to  no.of pulses
/*  Configure PID constants :
 *  Kp -> for Rising time
 *  Ki -> for Steady state
 *  Kd -> for Oscillation
*/
float kp = 50;
float kd = 2;
float ki = 15;

long currT = micros();
float deltaT = ((float) (currT - prevT))/( 1.0e6 );
prevT = currT;
int e = (target_position-pos);
float dedt = (e-eprev)/(deltaT);
eprev = e;
eintegral = eintegral + e*deltaT;
// control signal
float u = kp*e + kd*dedt + ki*eintegral; // Output  specific signal to power motor
float pwr = abs(u);
if( pwr > 255 ){
pwr = 255;
}else if(pwr>50&&pwr<250)
{
pwr = 180;
}else
```