Faculty of Engineering Ain Shams University

Spring 2024

Automatic Control Final Project

Team (9)

Name	ID	Section
Peter Munir Azmy	1901934	2
Fares Ahmed Mohamed	19P6135	2
Hassan Ahmed Mohamden	2002008	2
Ahmed Hawary Abdelrahem Ali	2000135	2
Mahmoud Mohamed Ibrahim Ali	2001481	3
Mohamed Hany Farouk	2001266	3

Project Objective:

Main objective

- To make a position control system through closed loop feedback system.
- understand the effect of PID parameters on the system performance.
- How to make the tuning of the controller parameters.

Project Objective

- convert DC motor to servo motor through a position control system.

The aim of this project is to design a position control system for a geared DC motor. The system should be able to achieve any angular position set point in the range from 0 to 180.

How the project work:

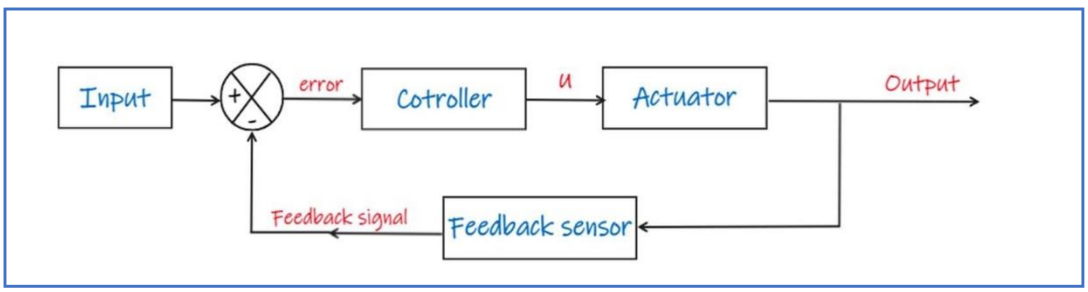
- First, the value of the angle changed by 45 degrees from the Arduino code.
- The driver receives the signal and transfers it to the motor as a voltagefrom the outsource power.
- The motor rotates the encoder with a certain no. Of steps upon the value ofthe Arduino and send it back to the Arduino to compare it with the required value and correct it upon the value of K_p, K_i, K_d.

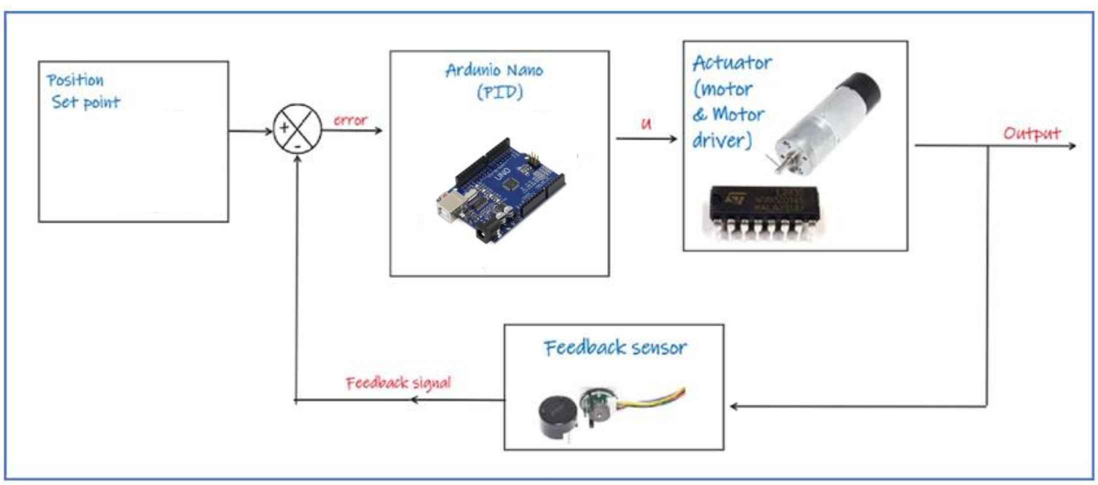
Design Procedures:

- Brainstorming about the design, electronic components, and the circuitdiagram.
- Buying the materials and electronic components.
- Assembly.
- Coding.
- Tuning PID parameters to achieve the best system response.

Block Diagram:

Closed loop control system





Description of the closed loop control system:

- It begins with setting the set point of the position (input).
- We have two Readings (position set point and the actual position)
- Magnetic sensor that fixed with motor acts as a feedback sensor as itgives the actual Position.
- They pass across the comparator and get the error difference (Set point –Actual).
- This error signal enters the controller and based on it the PID generatesoutput signal which goes to the channel relay module (Actuator).

Description with details

Control action: the Arduino (controller) compares the output value (angle) with the reference input by user, determines the deviation and the difference between the two values and produce a control action by signals to reduce this deviation into smaller value till reach to zero error the controlled variable (angle)measured and detected by the magnetic encoder.

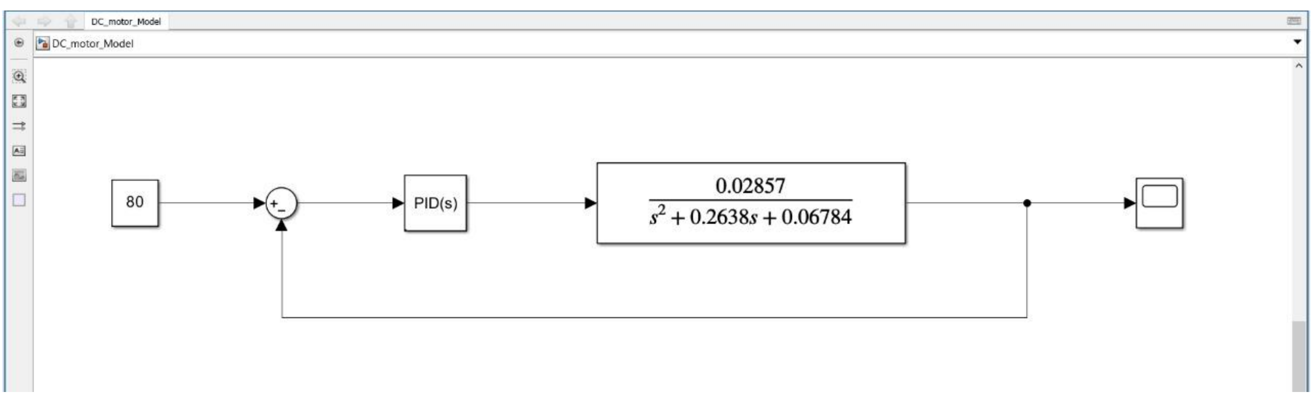
Final control element (actuator): the (Dc motor) that receives signal from thecontroller and H-bridge and rotate by certain angle by the control action.

Sensor: the (Magnetic encoder) detects the rotational position information of themotor as changes in the magnetic field and convert them into electrical signals then outputs them -Controlled variable: the angle that is input by the user and compared with output by the controller till reach to the desired input value

Procedure to tune the PID Controller:

- Increase P gain until you get the best response you can.
- Decrease slightly the P term and add I term gradually (to improve steadystate error). The I term will increase overshot and oscillations.
- Finally add D term to improve transient response.
- Test stability and transient response at various operating points(nonlinearities, varying system gain/sensitivity).
- Check response to set point step changes.
- Check response to major disturbances.
- Record your system control output in various steady state operationmodes.

The System Model



Position control Trials:

First Trial

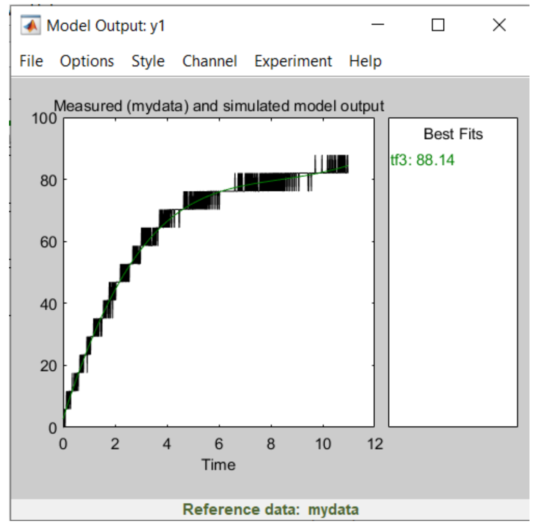
```
tf1 =
  From input "pulse" to output "speed":
         -0.004432
  -----------------------
  s^2 + 0.4632 s + 0.1497

Name: tf1
Continuous-time identified transfer function.

Parameterization:
   Number of poles: 2   Number of zeros: 0
   Number of free coefficients: 3
   Use "tfdata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using TFEST on time domain data "mydatad".
Fit to estimation data: 70.76% (stability enforced)
FPE: 39.3, MSE: 38.95
```

Second Trial

```
tf2 =
  From input "pulse" to output "speed":
          0.4871 s + 0.01627
  ----------------------------------
  s^3 + 1.068 s^2 + 1.523 s + 0.5393

Name: tf2
Continuous-time identified transfer function.

Parameterization:
   Number of poles: 3   Number of zeros: 1
   Number of free coefficients: 5
   Use "tfdata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using TFEST on time domain data "mydatad".
Fit to estimation data: 88.24% (stability enforced)
FPE: 6.393, MSE: 6.3
```

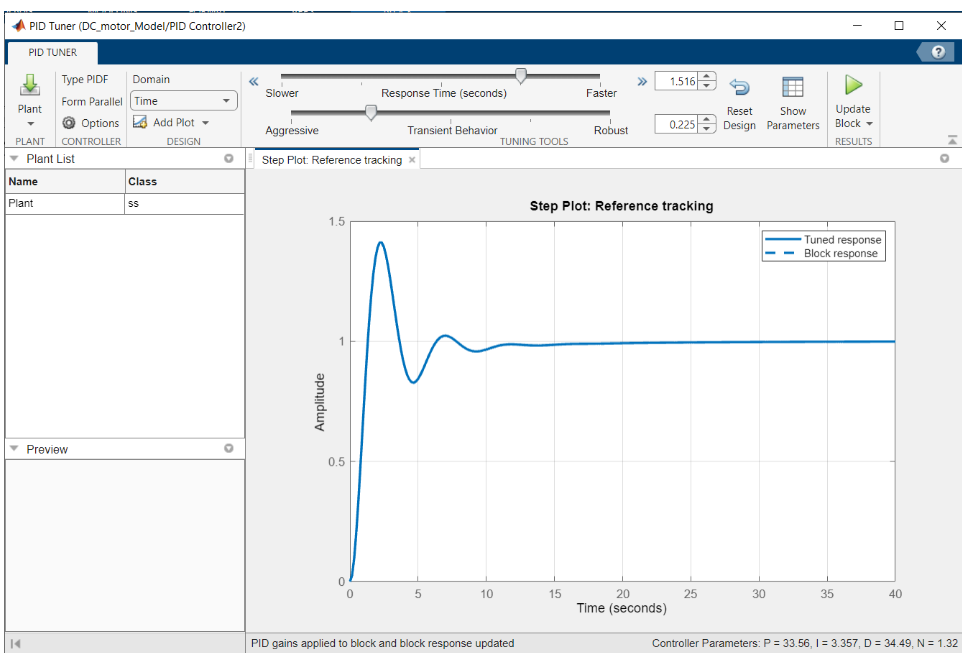
Third Trial (used in the model)

```
Name: tf3
Continuous-time identified transfer function.

Parameterization:
   Number of poles: 2   Number of zeros: 0
   Number of free coefficients: 3
   Use "tfdata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using TFEST on time domain data "mydata".
Fit to estimation data: 88.14% (stability enforced)
FPE: 6.465, MSE: 6.407

Model Properties
```

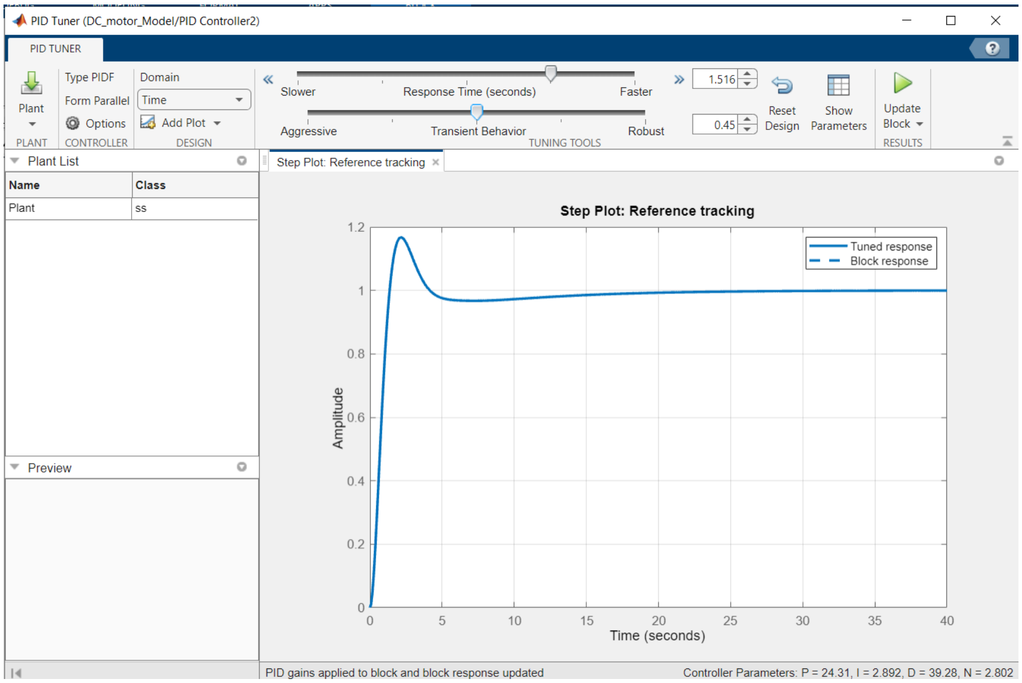


ID Parameters after tuning:

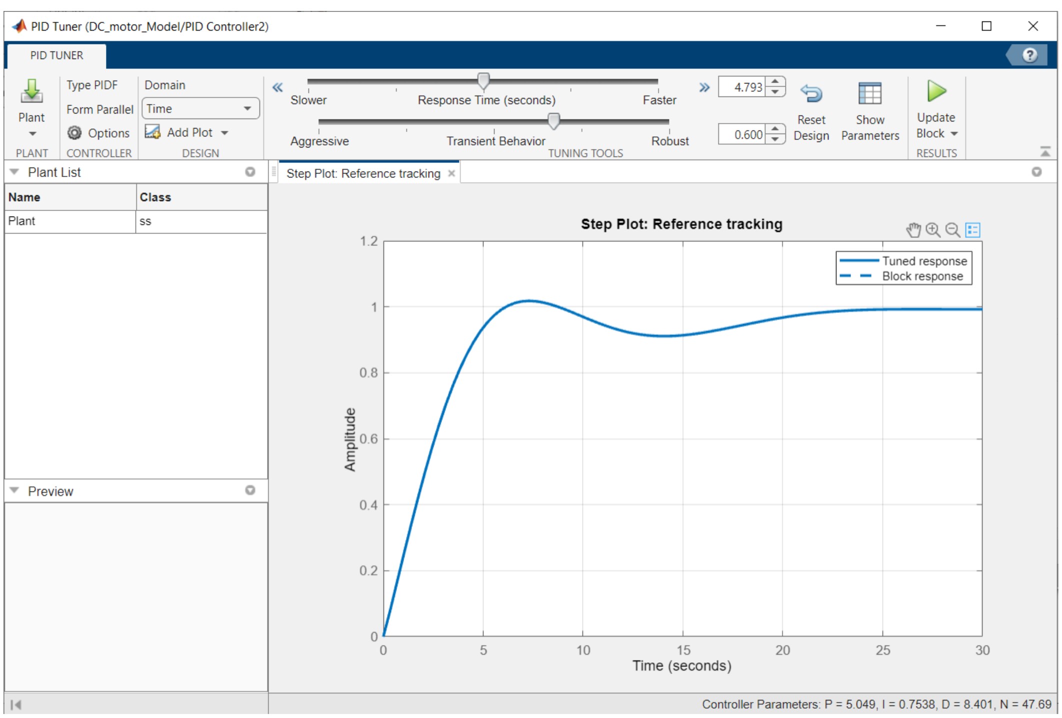
First tune:

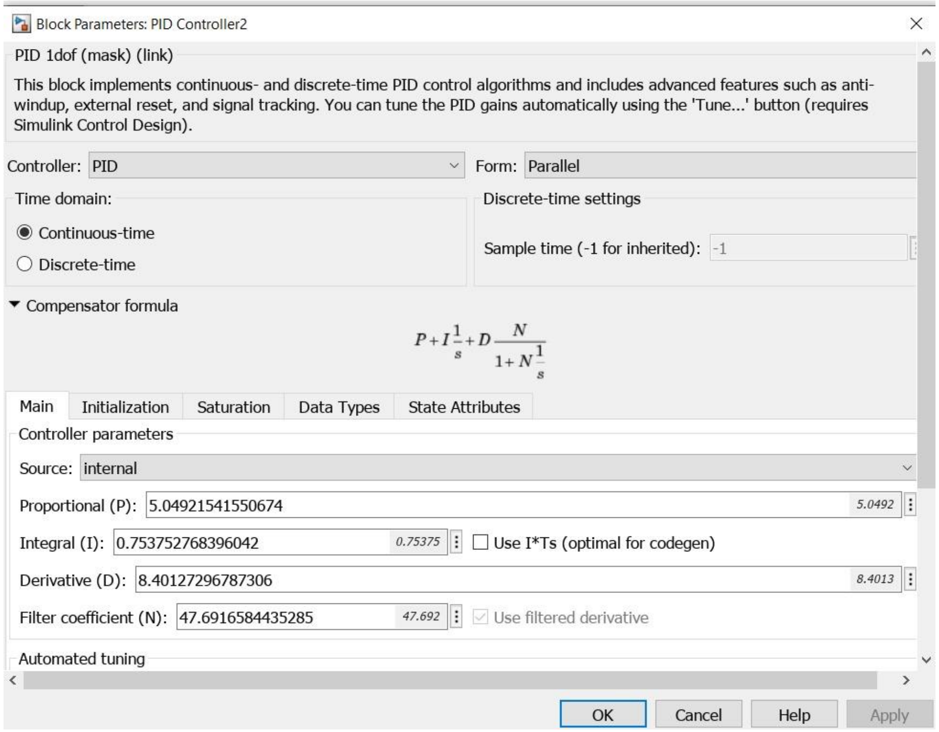


Second tune:

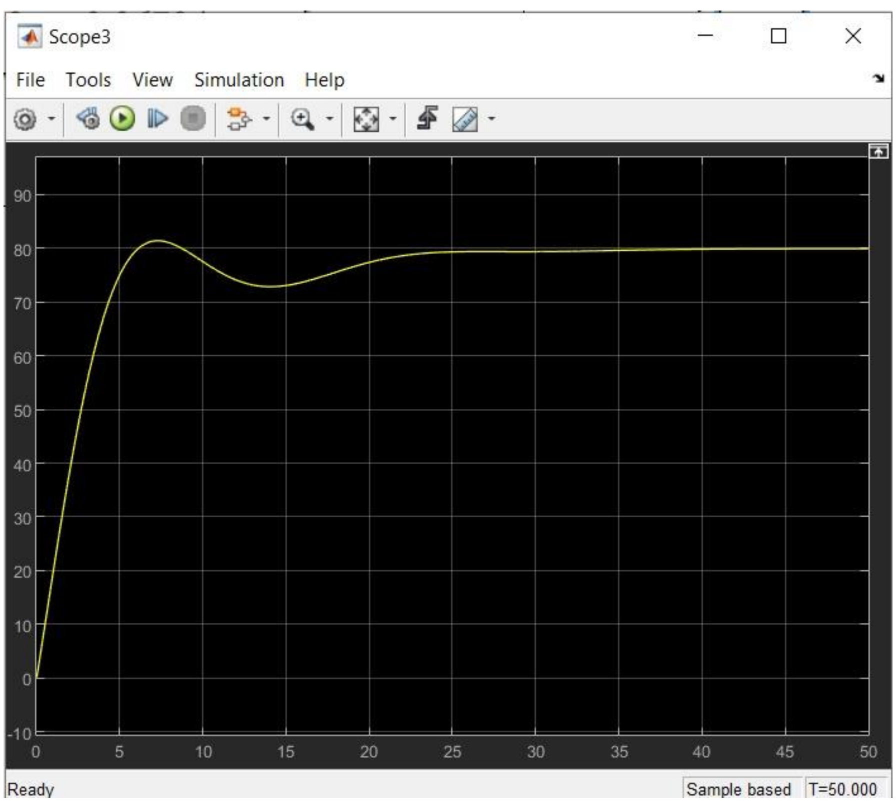


Third tune:





Output of set point =80



Arduino code:

```
//variable for encoder
#define outputencA 2   //yellow
#define outputencB 3  //green
//define pin num for motor
#define input1 8
#define input2 9
#define PWM 10
//variable of postion
int postion = 0;

float angle = 0;
float increment = (360.0 / 390);
long time1 = 0;
//vaeiable fpr pid control
long previoustime = 0;
float eprevious = 0;
float eintegral = 0;
int target = 0;
void setup() {
  Serial.begin(9600);
  pinMode(outputencA, INPUT_PULLUP);
  pinMode(outputencB, INPUT_PULLUP);
  pinMode(input1, OUTPUT);
  pinMode(input2, OUTPUT);
  pinMode(PWM, OUTPUT);
  //INterrupt for encoder
  attachInterrupt(digitalPinToInterrupt(outputencA), readEncoder, RISING);
}
void loop() {
  while (1) {  //pid variable
```

```
// PID variables
    float kp = 5.2;  // Proportional gain
    float ki = 0.001;   // Integral gain
    float kd = 1;  // Derivative gain
  float u = 1;
    eintegral = 0;
    while (u != 0) {
      u = pidcontrol(target, kp, kd, ki);
   //motor power
      float pwr = fabs(u);
      if (pwr > 235) {
        pwr = 235;
      } else if (pwr < 55) {
        pwr = 55;
      }
      //motor direction
      int dir = 1;
      if (u == 0) {
        dir = 0;
      } else if (u < 0) {
        dir = -1;
      }
      movemotor(dir, pwr, PWM, input1, input2);
      Serial.println(target);
      Serial.print(", ");
      Serial.println(angle);
      delay(20);
    }
    delay (3000);
    target += 45;
  }
```

```
}
void readEncoder() {
  int b = digitalRead(outputencB);
  if (b > 0) {
    postion++;
    angle += increment;
  } else {
    postion--;
    angle -= increment;
  }}
//function to move motor
void movemotor(int dir, int pwmvalue, int pwm, int in1, int in2) {
  analogWrite(pwm, pwmvalue);
  if (dir == 1) {
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
  } else if (dir == -1) {
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
  } else if (dir == 0) {
    digitalWrite(pwm, HIGH);
    digitalWrite(in1, HIGH);
    digitalWrite(in2, HIGH);
  }
}
float pidcontrol(int target, float kp, float kd, float ki) {
  float u;
  //get delta t
  long currenttime = micros() - time1;
  float deltaT = ((float)(abs(currenttime - previoustime))) / 1.0e6;
  //calculate error ,derivative ,integral
```

```
int e = (int)angle - target;
if (abs(e) < increment) {
  return 0;
}
float edrivative = (e - eprevious) / deltaT;
eintegral = eintegral + e * deltaT;

// contrpal signal
u = (kp * e) + (kd * edrivative) + (ki * eintegral);
//update variable for next iteration
previoustime = currenttime;
eprevious = e;
return u; }
```