Ain Shams University

Faculty of Engineering

Mechatronics-Senior 1

MCT211s-Automatic Control

Project (3) Airflow Forced

Name	ID
Alaa Elsayed Metwally Mohamed	1809361
Ahmed Yasser Ahmed Abdelhamid	1804679
Ali Abdelhakem Mahmod Hussin	1805347
Mohamed Emad Mohamed Hussien	1802245
Mohamed Emad Mohamed Moslem	1806770
Mohamed Abdulaziz Farrag Ahmed	1805858

Contents

1. Description of Project and Objective

1.1. Objective:

- By using forced air flow which we can lift object (ball) and achieving desired position.
- Controlling height of the ball inside tube.

1.2. Overview:

- System Components:
  a) Ultrasonic sensor
  b) Plastic Pipe
  c) Ball
  d) Fan
  e) Motor Driver L293
  f) Microcontroller Arduino
- System Working:

- In order to achieve air stream capable of lifting balls with different masses, a fan with high air volume and static pressure was required.
- A 12V DC fan is inserted into a hole in a flat sheet of Plexiglass, blowing upwards. The fan speed is controlled by pulse width modulation (PWM) signal, and the dual full-bridge L298 driver chip is used in the fan drive circuit.
- The output voltage of the controller provides the input of a low-cost motor drive, where produces a pulse width modulated signal for fan rotation speed control.

2. Design Procedures

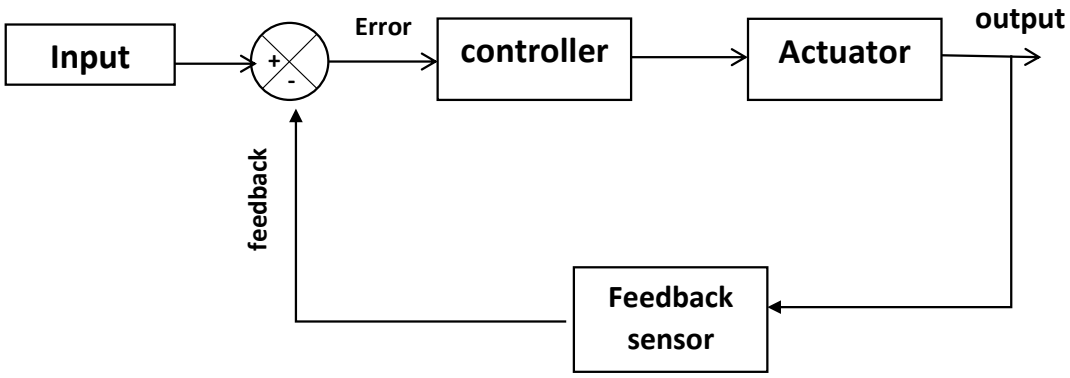
1) Brainstorming about the design, electronic components, and the circuit diagram.
2) Buying the materials and electronic components.
3) Assembly.
4) Coding.
5) Tuning PID parameters to achieve the best system response.

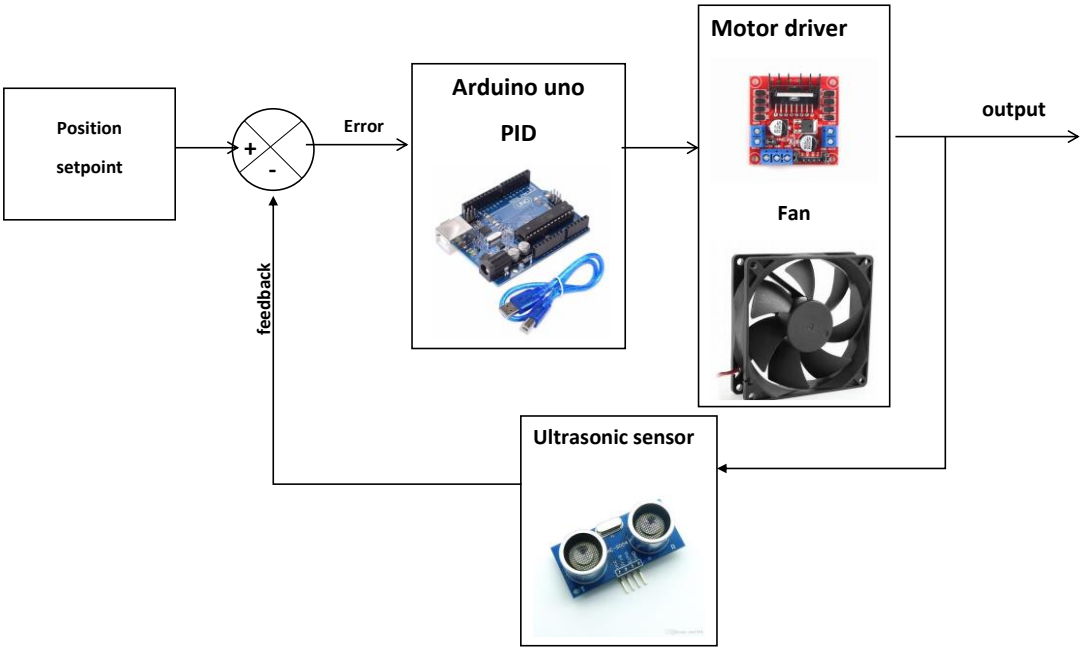
3. Closed loop Control system

3.1. Description of closed loop

- Passing the setpoint (desired position)
- Calculate Error between Output feedback from Ultrasonic and Setpoint
- Minimize error by PID from controller
- Send Suitable action to actuator
- Taking feedback from actuator by Ultrasonic
- Start loop again and achieving the desired objective (Ultrasonic)

3.2. Block Diagram



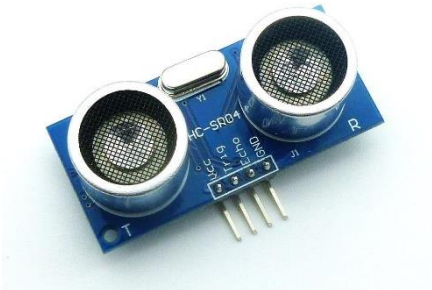


4. Components and its functions

4.1. Arduino UNO:

- Microcontroller that responsible for control the system and Minimize error by PID.

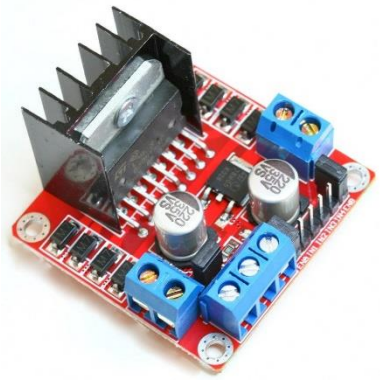
4.2. Ultrasonic Sensor



- It emits an ultrasound at 40 000 Hz which travels through the air and if there is an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance.
- Measuring Distance of Object

4.3. Motor Driver L298

- The L298N is a dual H-Bridge motor driver which allows speed and direction control. The module can drive Fan that have voltages between 0 and 12V.



4.4. Fan

- Used for achieving forced airflow to lift the object(ball)

5. System Assembly

6. Arduino code

Final_Air_Flow

```
#include <PID_v1.h>

#define echoPin 2
#define trigPin 3
#define fan 9
#define in1 6
#define in2 7
#define pwr 152
double e=0;
double Kp=22, Ki=0.5, Kd=0;
unsigned long T=0;
double Setpoint, Input,Output;

int pulsetime = 1000;
unsigned long Signaltime =0;

PID myfan(&Input, &Output, &Setpoint, Kp,Ki,Kd, DIRECT);

unsigned long SignalBegin;

void setup() {

  Signaltime = millis();
  Setpoint =15;
  pinMode(trigPin,OUTPUT);
  pinMode(echoPin,INPUT);
  pinMode(fan,OUTPUT);

  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);

  digitalWrite(in1,LOW);
  digitalWrite(in2,LOW);

  Serial.begin(9600);
  myfan.SetOutputLimits(0,pulsetime);

  myfan.SetMode(AUTOMATIC);
  T = millis();
,

void loop(){

  Input = readDistance();



  myfan.Compute();
  if((millis()-SignalBegin)>pulsetime){

    SignalBegin += pulsetime;
    }

    if((Output)>(millis()-SignalBegin)){
      digitalWrite(in1,LOW);
      digitalWrite(in2,HIGH);
  for(int i=pwr;i>0;i--)
      {
      analogWrite(fan,i);
      }



     }


     else if((Output)<(millis()-SignalBegin)){
      digitalWrite(in1,LOW);
     digitalWrite(in2,HIGH);
      for(int i=0;i<pwr;i++)
      {
      analogWrite(fan,i);
      }

      }
      else
      {
        digitalWrite(in1,LOW);
        digitalWrite(in2,LOW);

      for(int i=pwr;i>0;i--)
      {
      analogWrite(fan,i);
      }


        }


    Serial.println(readDistance());
  }

long readDistance(){

  long duration;
  long distance;
  digitalWrite (trigPin,LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin,HIGH);
    delayMicroseconds(2);
    digitalWrite(trigPin, LOW);
   duration = pulseIn(echoPin,HIGH);
   distance = duration * 0.034 / 2;
   return distance;

  }
```

7. Procedure to tune the PID

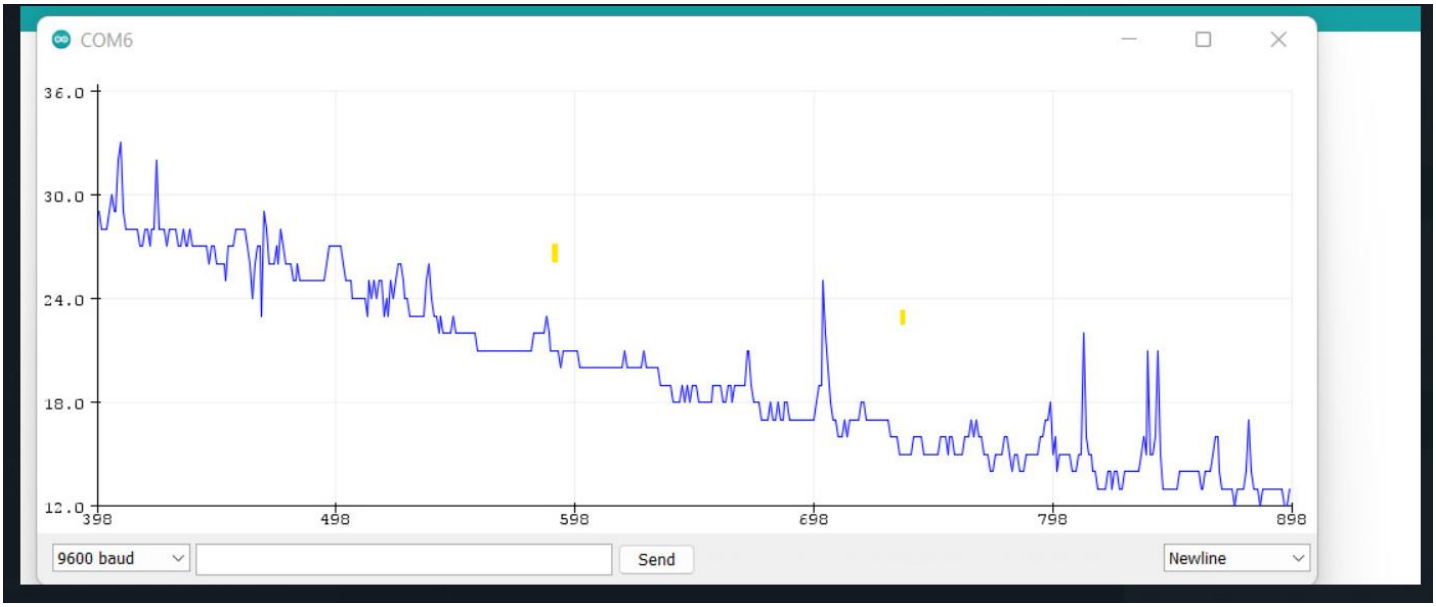
1. Increase P gain until you get the best response you can.
2. Decrease slightly the P term and add I term gradually (to improve steady state error). The I term will increase overshoot and oscillations.
3. Finally add D term to improve transient response.
4. Test stability and transient response at various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

8. Graphs

8.1. First trail

Kp →10 , Ki→1, Kd→0

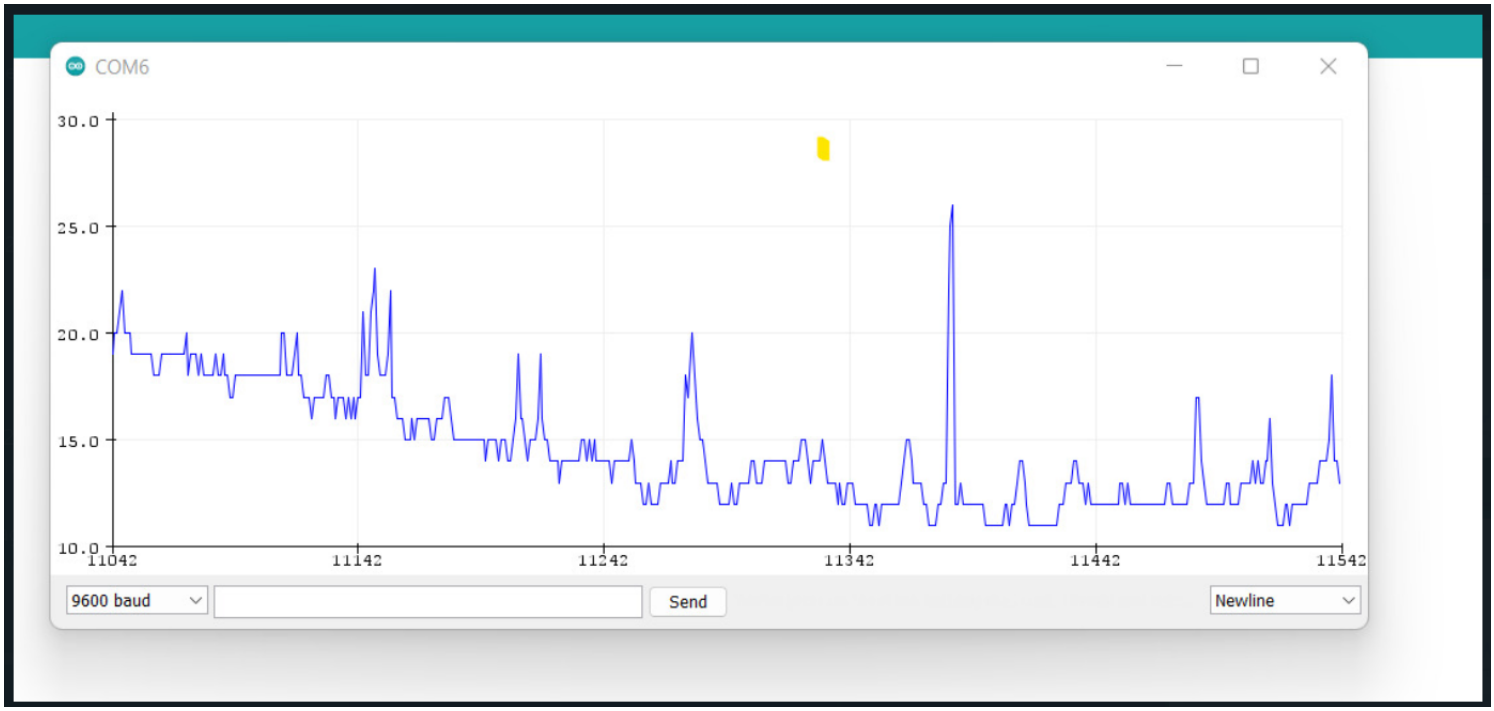
- Settling time = 790 microseconds
- Rising time = 698 microseconds
- Steady state error = +2,-2 (17,13)
- Overshoot = 25



7.2 Second trial

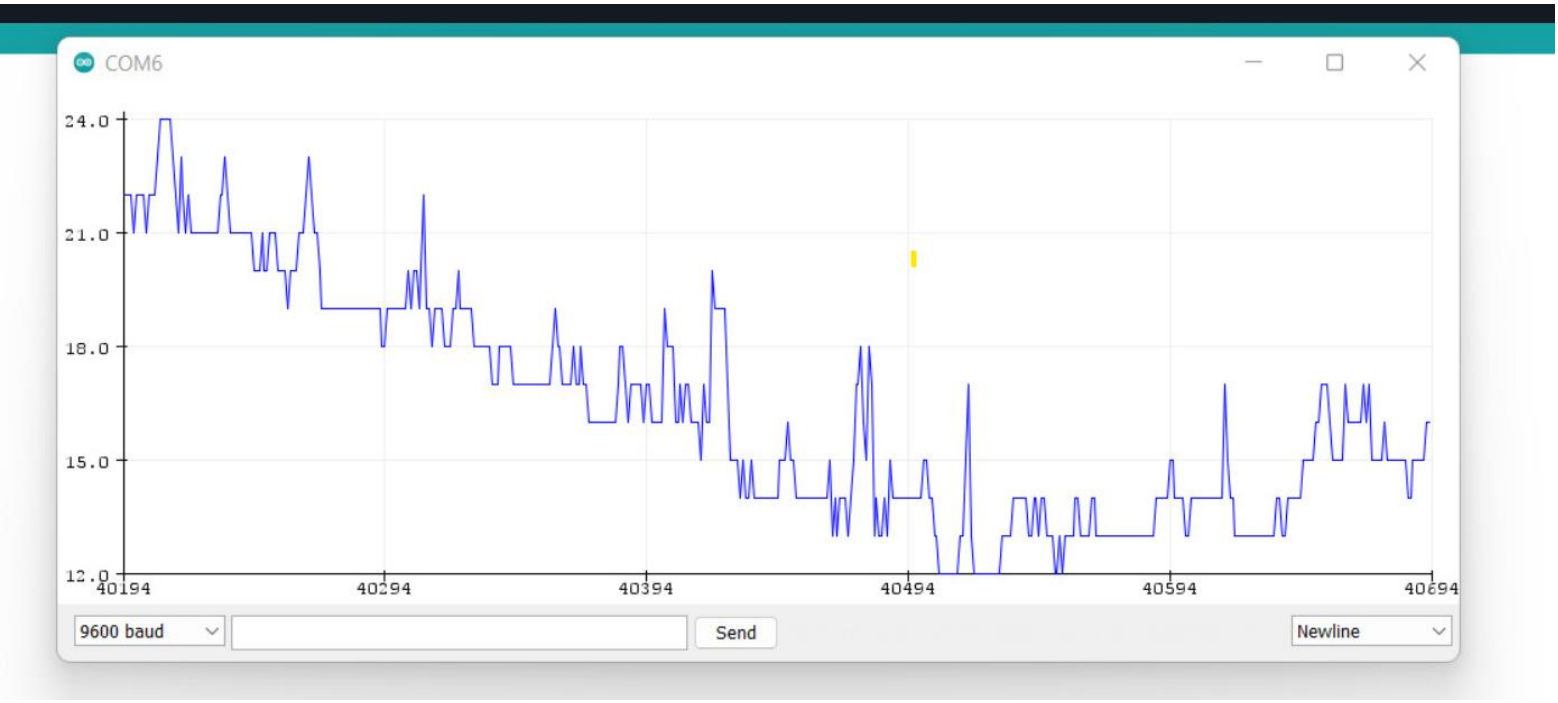
Kp →12 , Ki→0.5, Kd→0

- Settling time = 11442 microseconds
- Rising time = 11390 microseconds
- Steady state error = +2,-3(17,12)
- Overshoot = 25



7.3 Third trail

Kp →15 , Ki→0.5, Kd→0



7.4 Optimal trail

Kp →22 , Ki→0.5, Kd→0

- Settling time = 804 microseconds
- Rising time = 780 microseconds
- Steady state error = +2,-2(17,13)
- Overshoot = 25