German University in Cairo

Mechatronics Engineering (MCTR601)

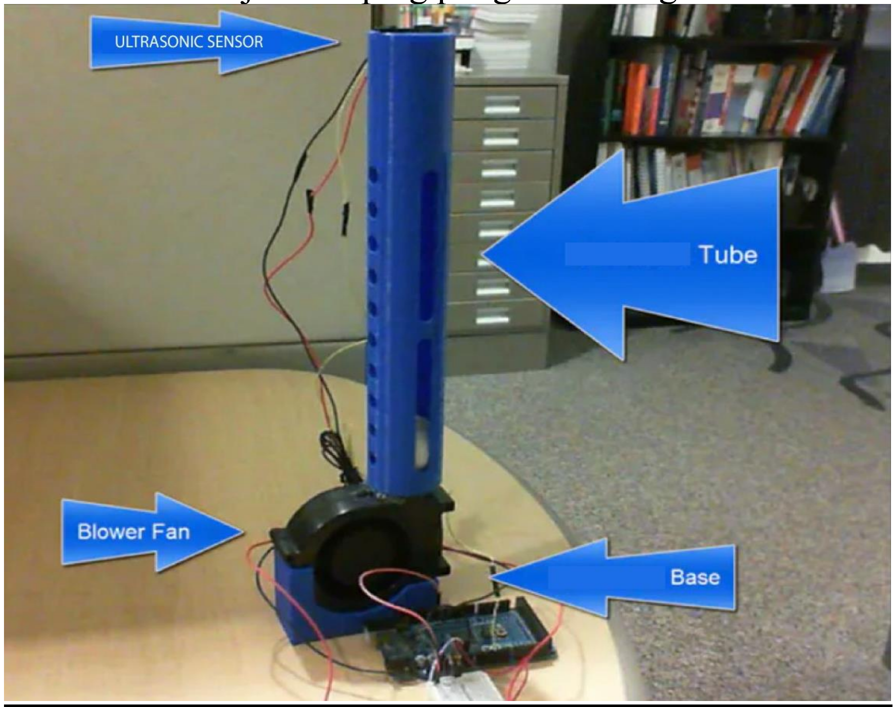
Ping Pong(Styrofoam) Levitation

Name	ID	Lab Number
Marwan Sallam	46-3401	T-53
Youssef Almahdi	46-4979	T-67
Mohammed Magdy	46-3493	T-55

Table of Contents

1. Project Description 3
2. Methodology 4
2.1 Mechanical design 4
2.2 Electrical design 5
2.3 Control 6
    2.3.1. Modeling 6
    2.3.2. Analysis 9
    2.3.3. Controller Design 9
2.4 Programming 11
3. Design Evaluation 20
4. Appendix 21

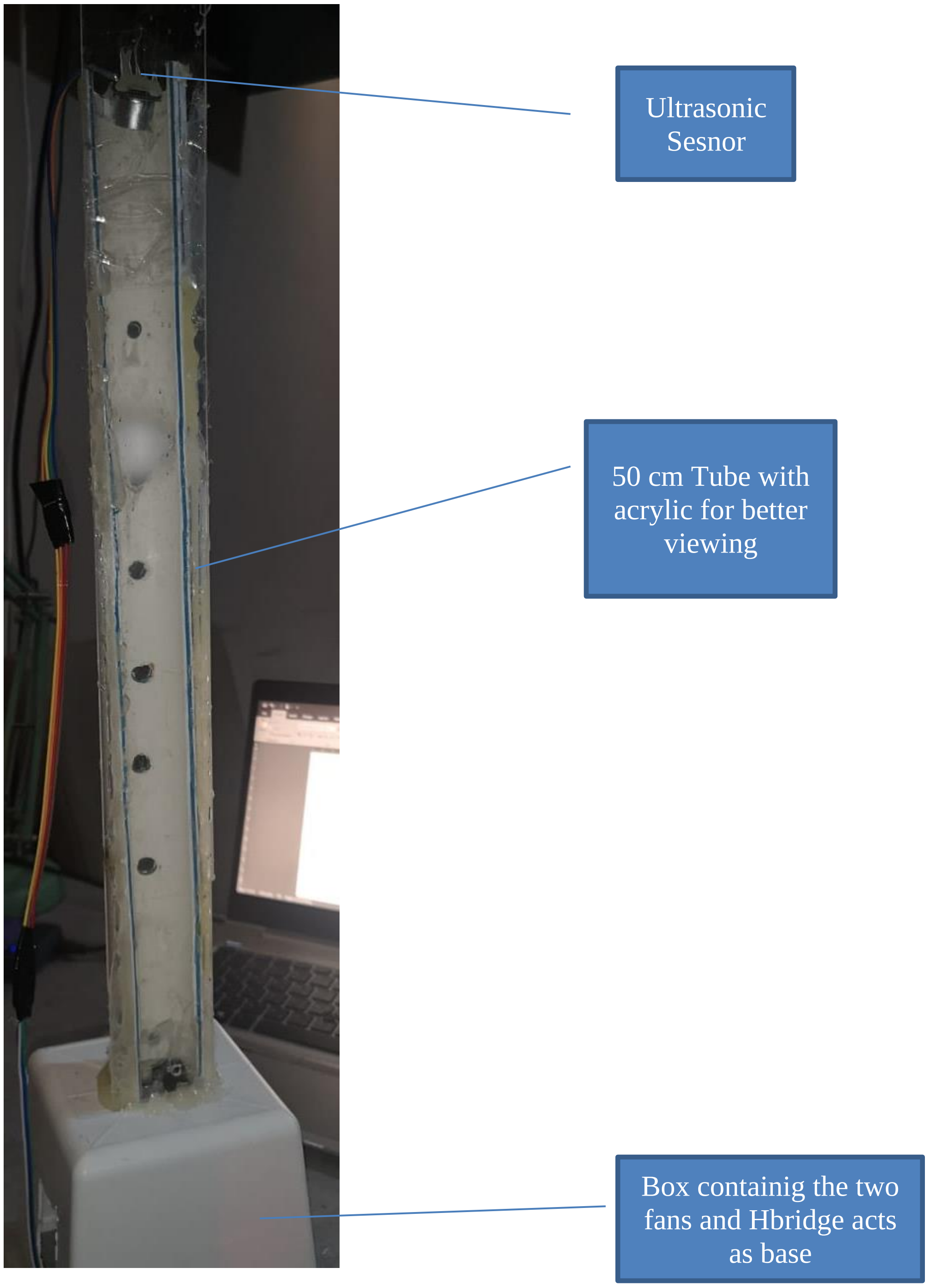
1. Project Description

- This project utilizes Arduino to control a blower fan, and levitate a ping pong ball to a specific height. The variable resistor will work as the human interface with the Arduino controlling the height and the Arduino will in return adjust the ping pong ball's height.
- 

Part number	Name
1	Arduino Uno
2	L289 H Bridge
3	HC SR04 Ultrasonic Sensor
4	Cooling fan
5	Potentiometer
6	Power Supply 12V

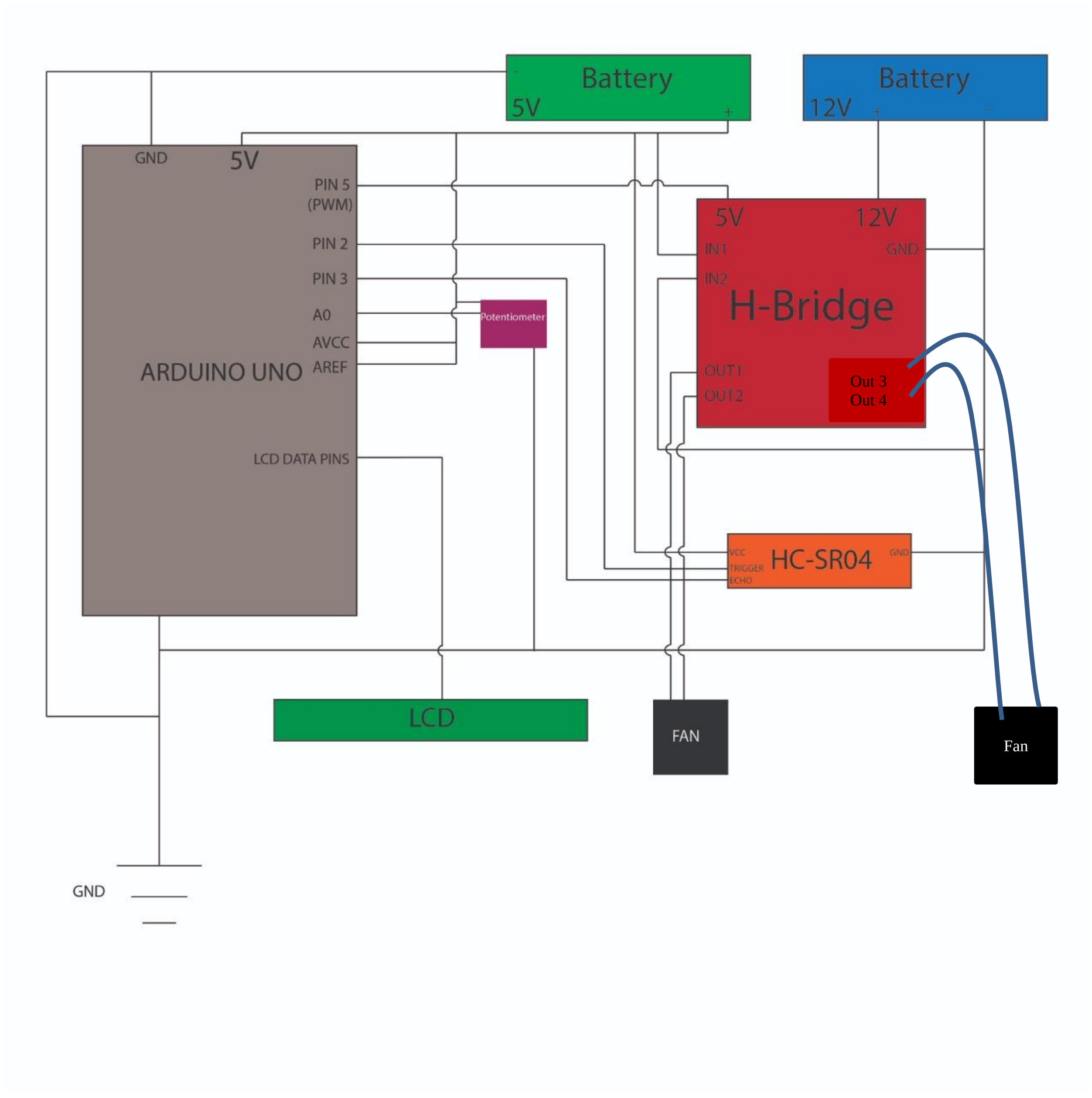
2. Methodology

2.1 Mechanical design


Ultrasonic Sesnor
50 cm Tube with acrylic for better viewing
Box containig the two fans and Hbridge acts as base

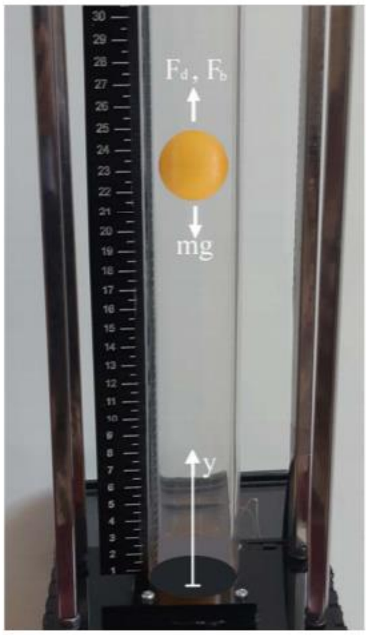
2.2 Electrical design

Include schematics for the electrical systems showing all connections



2.3 Control

2.3.1. Modeling



$$m\frac{d^2y}{dt^2} = F_b + F_d - F_g$$

$$F_b = \rho g V_b$$

$$F_d = f(\Delta p, F_f)$$

$$F_g = mg$$

[illegible]

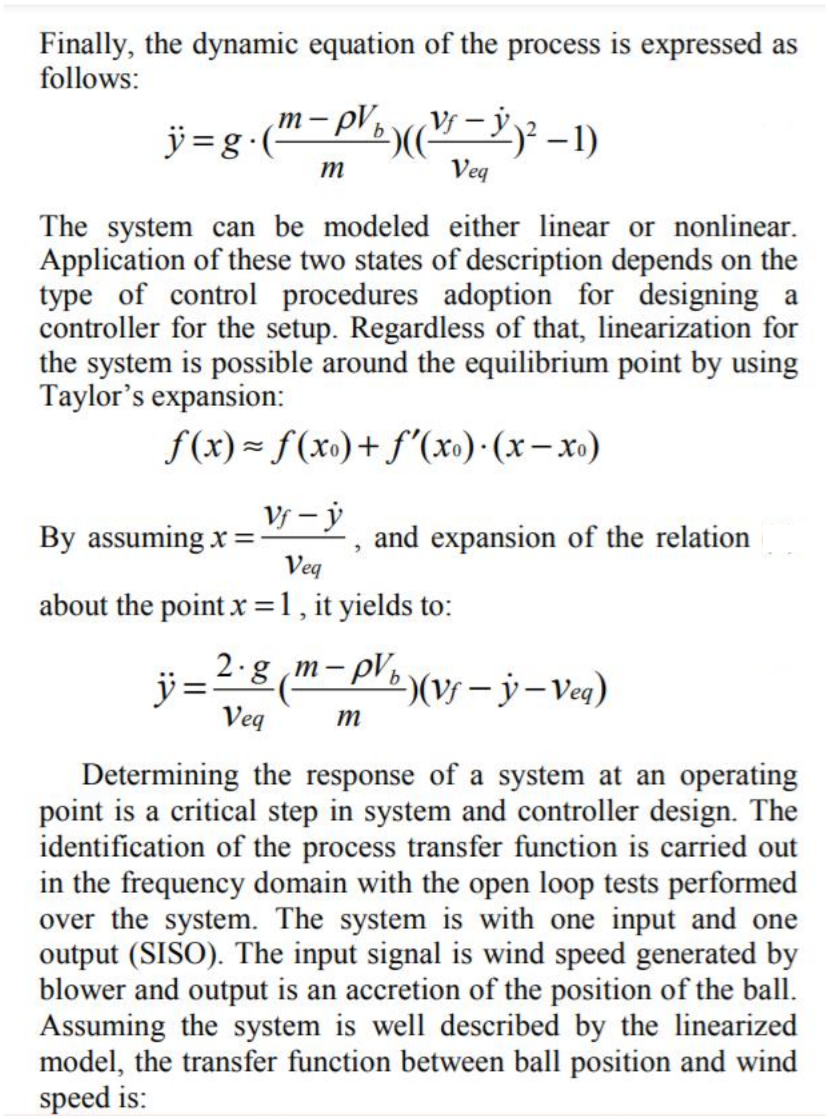
Where Δp is the air pressure difference, ρ is the density of air, g is the gravitational acceleration, m is the mass of the ball, V_b is the ball's volume and F_f is the friction force caused by airflow. The drag coefficient is a term that depends on Reynold's number, which, in turn, depends on the relative velocity of the ball that moves inside the flow, and the velocity of the flow.

$$F_d(\rho, c_d, A, v_f, y) = \frac{1}{2} \cdot C_d \cdot \rho \cdot A \cdot (v_f - \dot{y})^2$$

Where, v_f is the velocity of the air inside the tube, A_b is the ball's area, C_d is the so-called drag coefficient, and y is the position of the ball in the tube. By summarizing and reforming the relations mentioned above, the system's dynamic equations can be obtained by exploiting net force between the airflow force and gravitational force as:

$$m\Delta\ddot{y} = -mg + \frac{1}{2} \cdot C_d \cdot \rho \cdot A \cdot (v_f - \dot{y})^2 + \rho g V_b$$

In this study, it is assumed that c_d is constant due to the small velocity of flow. The levitating ball will be in a steady state when it does not move ($\ddot{y} = \dot{y} = 0$). The IR sensor measures the distance between the top of the pipe and the ball. In this way, the sensor provides information about the position of the ball. In the following, v_{eq} defined as airspeed at the equilibrium point ($v_{eq} = v_f - \dot{y}$).



Finally, the dynamic equation of the process is expressed as follows:

$$\ddot{y} = g \cdot \left(\frac{m - \rho V_b}{m}\right)\left(\left(\frac{v_f - \dot{y}}{v_{eq}}\right)^2 - 1\right)$$

The system can be modeled either linear or nonlinear. Application of these two states of description depends on the type of control procedures adoption for designing a controller for the setup. Regardless of that, linearization for the system is possible around the equilibrium point by using Taylor's expansion:

$$f(x) \approx f(x_0) + f'(x_0) \cdot (x - x_0)$$

By assuming $x = \frac{v_f - \dot{y}}{v_{eq}}$, and expansion of the relation about the point $x = 1$, it yields to:

$$\ddot{y} = \frac{2 \cdot g}{v_{eq}}\left(\frac{m - \rho V_b}{m}\right)(v_f - \dot{y} - v_{eq})$$

Determining the response of a system at an operating point is a critical step in system and controller design. The identification of the process transfer function is carried out in the frequency domain with the open loop tests performed over the system. The system is with one input and one output (SISO). The input signal is wind speed generated by blower and output is an accretion of the position of the ball. Assuming the system is well described by the linearized model, the transfer function between ball position and wind speed is:

$$\frac{y(s)}{v(s)} = \frac{1}{s}\frac{b}{(s+b)}$$

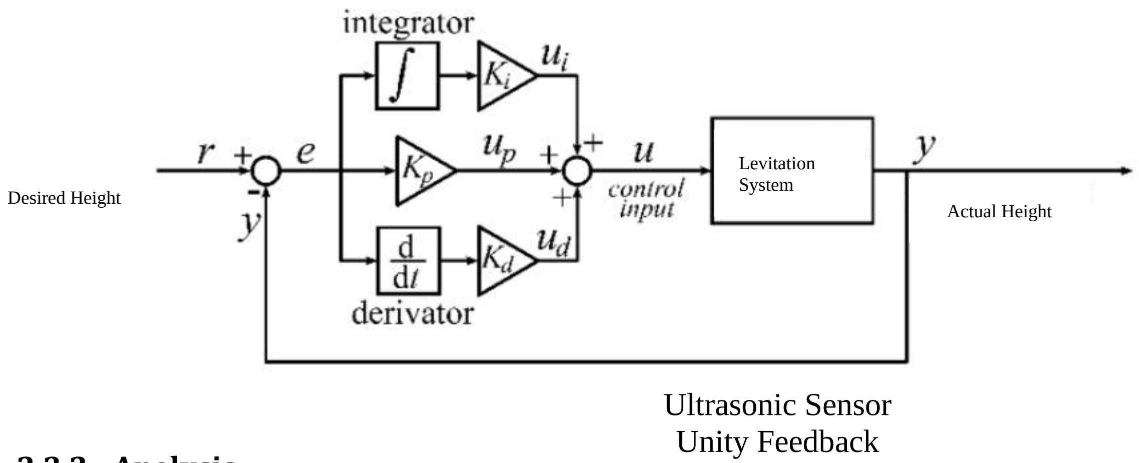
Where $v(s)$ and $y(s)$ are wind speed and increment of ball's position about the equilibrium point, respectively and $b = 2g(m - \rho V_b)/mv_{eq}$. Considering the fan can be modeled as a first-order process, the transfer function between the input voltage and the wind speed is represented as:

$$\frac{v(s)}{u(s)} = \frac{k_v}{\tau s + 1}$$

where *v(s)*, *u(s)*, are wind speed and input voltage, respectively. In addition, k_v is the sensitivity gain that relates the input voltage to the wind speed at steady state, and τ is the time constant of the fan.

Finally, the transfer function of the entire system defines as follows:

$$G(s) = \frac{y(s)}{u(s)} = \frac{b.k_v.}{s(s+b)(\tau s+1)}$$



2.3.2. Analysis

- **Solve equations and get the open loop response**

2.3.3. Controller Design

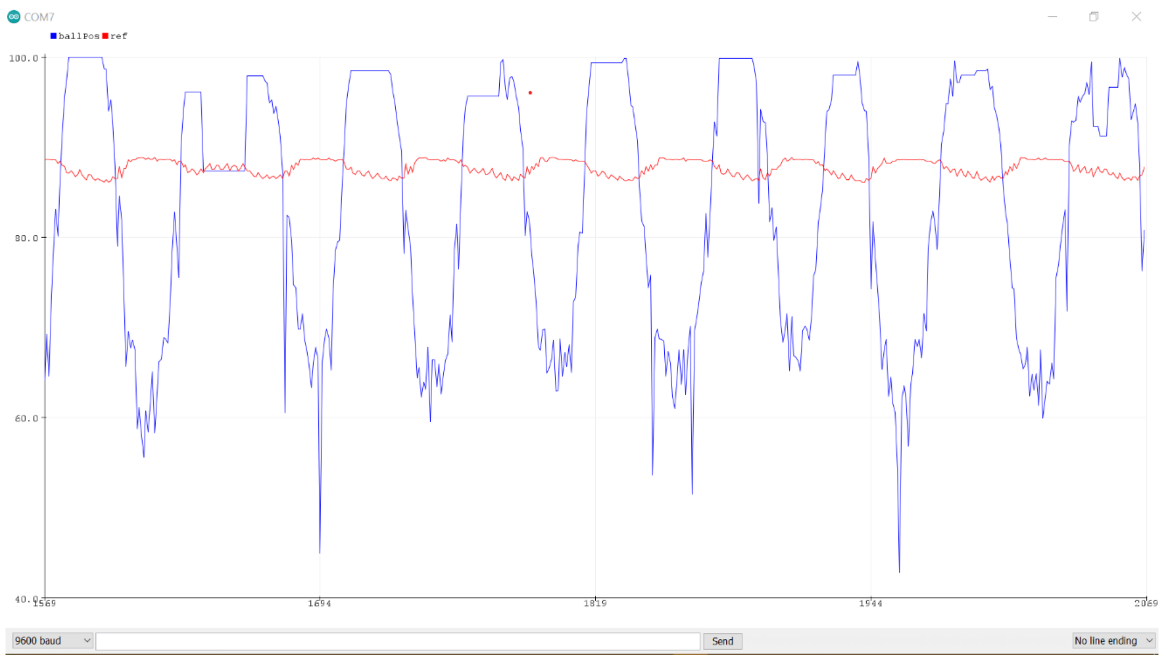
To fulfil all the requested points, we implemented the feedback control law with three parallel actions: proportional, integral and derivative. It is known from system control theory that integral action is needed if one wants to have zero error in the steady state. Also, derivative action is sometimes useful to stabilize the system if it happens to have oscillatory behaviour that cannot be controlled in other ways. Proportional action is obviously the most intuitive one and it is important to drive the ball in the early stage of

transient, when error is significant. We will give the explicit expressions in the following lines. Firstly, let us define (r)reference (desired ball position), (y) ball position, both as numbers between 0 and 100%
(Scaled in correspondence with the allowed region of ball position in tube). The error is defined as follows $e = r - y$. Total control u, i.e., voltage provided to the DC motor, is defined as: $u = P + I + D$

$$P = K_p \cdot e(t),$$

$$I = I_0 + K_i \int_0^t e(\tau)\mathrm{d}\tau,$$

$$D = K_d \frac{\mathrm{d}e(t)}{\mathrm{dt}},$$



2.4 Programming

```
const int pm = A0; //potenmeter

const int trigPinBall = 6; //ball position ultrasonic sensor Trig

const int echoPinBall = 5; //ball position ultrasonic sensor Echo

const int fanPin     = 3;// PWM output for motor control

// defines variables

unsigned long durationB;

double distanceB;

double ref = 0; //position of hand

double ballPos = 0; //position of the ball in the tube

//Geometrical parameters of the system

const double upperGap = 10; //upper gap in cm

const double lowerGap = 10; //lower gap in cm

const double columnL = 30; //effective length of the tube in cm

#define PREV_REF 70;

double previousRef = PREV_REF; // the initial desired reference is PREV_REF [%] of the effective length of the tube

const double ballDiam = 4; //diameter of the ball in cm

const int maxWaitTime = (int) (columnL * 3.5 / 0.034); //maximal waiting time for the ultrasonic sensors to receive the reflected wave

//variables for the control loop

double controlP = 0;

double controlI = 0;

double controlD = 0;

double control = 0;

double err = 0;
double prevErr = 0;

double previousBallPos = 0; //the initial ball position, then updates

double prevControlI = 0; //initial condition for integrator


//PID constants

const double samplingTime = 0.0625; //sampling time in seconds


const double intTimea = 0.4 * 2.0;

const double difTimea = 0.02 * 0.77;

const double Kpa = 0.7 * 3;

const double Kia = Kpa / intTimea;

const double Kda = Kpa * difTimea;


const double intTimeb = 0.50 * 3.2;

const double difTimeb = 0.012 * 2.2;

const double Kpb = 0.35 * 2.0;

const double Kib = Kpb / intTimeb;

const double Kdb = Kpb * difTimeb;


const double intTimec = 0.5 * 3.2;

const double difTimec = 0.014 * 1.1;

const double Kpc = 0.45 * 1.65;

const double Kic = Kpc / intTimec;

const double Kdc = Kpc * difTimec;

//flags
```

```
bool flagRefErr = 0; //true if there is an error reading reference
bool flagBallPosErr = 0; //true if there is an error reading ball position
char region = 'a'; //region 'a', 'b' or 'c' - division of the tube into three regions
//Limits for regions
const int limitAB = 18; //in % of the effective length of the tube
const int limitBC = 64; //in % of the effective length of the tube

void setup() {
  pinMode(trigPinBall, OUTPUT); // Sets the trigPin as an Output for Ball position sensor
  pinMode(echoPinBall, INPUT); // Sets the echoPin as an Input for Ball position sensor
  pinMode(pm, INPUT); // Sets the potentiometer as an Input for ref position sensor
  pinMode(fanPin, OUTPUT); //Sets the fanPin to an output mode
  Serial.begin(9600); // Starts the serial communication
  Serial.println("ballPos,ref,control");
}

void loop() {
region = 'a';
    //____________REF___________________
      ref = analogRead(pm);
      ref = ref/1023;
      ref = ref*100;

    //_____________BALL_________________
    digitalWrite(trigPinBall, LOW); // Clears the trigPin
    delayMicroseconds(2);

    digitalWrite(trigPinBall, HIGH);// Sets the trigPin on HIGH state for 10 microseconds

    delayMicroseconds(10);

    digitalWrite(trigPinBall, LOW);

    durationB = pulseIn(echoPinBall, HIGH, maxWaitTime);// Reads the echoPin, returns
the sound wave travel time in microseconds

    // Calculating the distance

    distanceB = durationB * 0.034 / 2;


    if (distanceB == 0) //Handling the error

    {

      flagBallPosErr = true;

      durationB = maxWaitTime;

    }

    else flagBallPosErr = false;

    ballPos = (columnL + upperGap) - (distanceB + ballDiam / 2); // scales the ball
position value in the reference system

    delay((int)(25 - durationB * 0.001));

    //scale the values from 0 to 100

    ballPos = ballPos * (100 / columnL);

    if ((ballPos < 0) || (ballPos > 100) || flagBallPosErr) ballPos = previousBallPo

    //Anti wind-up solution for transitions between different subregions of the tube

    if (ballPos > limitAB && ballPos <= limitBC && previousBallPos <= limitAB)
//transition from region A to B

    {

      prevControlI = control - Kpb * (ref - ballPos) - Kib * (ref - ballPos) * samplingTime -
Kdb * (-ballPos + previousBallPos) / samplingTime; //bumpless control
```

```
      region = 'b';

    }

    else if (ballPos > limitBC && previousBallPos > limitAB && previousBallPos <=
limitBC) //transition from region B to C

    {

      prevControlI = control - Kpc * (ref - ballPos) - Kic * (ref - ballPos) * samplingTime -
Kdc * (-ballPos + previousBallPos) / samplingTime; //bumpless control

      region = 'c';

    }

    else if (ballPos > limitAB && ballPos <= limitBC && previousBallPos > limitBC)
//transition from region C to B

    {

      prevControlI = control - Kpb * (ref - ballPos) - Kib * (ref - ballPos) * samplingTime -
Kdb * (-ballPos + previousBallPos) / samplingTime; //bumpless control

      region = 'b';

    }

    else if (ballPos <= limitAB && previousBallPos > limitAB && previousBallPos <=
limitBC) //transition from region B to A

    {

      prevControlI = control - Kpa * (ref - ballPos) - Kia * (ref - ballPos) * samplingTime -
Kda * (-ballPos + previousBallPos) / samplingTime; //bumpless control

      region = 'a';

    }

    else if (ballPos > limitBC && previousBallPos <= limitAB) //transition from region A
to C

    {

      prevControlI = control - Kpc*(ref-ballPos) - Kic*(ref-ballPos)*samplingTime -
Kdc*(-ballPos+previousBallPos)/samplingTime; //bumpless control
```

```
      region = 'c';

    }

    else if (ballPos <= limitAB && previousBallPos > limitBC) //transition from region C
to A

    {

      prevControlI = control - Kpa*(ref-ballPos) - Kia*(ref-ballPos)*samplingTime -
Kda*(-ballPos+previousBallPos)/samplingTime; //bumpless control

      region = 'a';

    }


    PID(region); //calculates the control parameters for the current sampling interval

    //uses the PID to set the value of the power to the motor according to the position and
speed

      motorPower(fanPin, control); //Control voltage sent to motor

      previousBallPos = ballPos; //stores prevouos value of ref in memory for next iteration

 Serial.print(ballPos);

 Serial.print(",");

 Serial.print(ref);

 Serial.print(",");

 Serial.println(control);

  delay(5);


}


//**********************************************************************
**********************************************************************
****
```

```
double getDistance (int trigPin , int echoPin) {
  // this function get the distance and save the value as output the inputs are the pin of the US sensor
  unsigned long duration = 0;
  double distance = 0;
  digitalWrite(trigPin, LOW); // Clears the trigPin
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);// Sets the trigPin on HIGH state for 10 micro seconds
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);// Reads the echoPin, returns the sound wave travel time in microseconds
  // Calculating the distance
  distance = duration * 0.034 / 2;
  return distance;
}



void motorPower (int motorPin , double power) {
  //Function that writes the values of the desired power on the pin of the motor.
  //The inputs are the pin to which the motor is attached and the desired power (number 0-100).



  double powerOut = (int)((power / 100) * 255); //gets the fraction of power in input, number between 0 and 1
  analogWrite(motorPin, powerOut); //gives the voltage to the motor, an integer 0-255,
proportional to the fraction of the power in input, 20 mV each step


}





void PID (char region) {
  //This function calculates the control parameters of the feedback.
  //No inputs. Calculates and stores variables.
  double Kp, Ki, Kd;
  if (region == 'a') //controller parameters for the lower part of tube
  {
    Kp = Kpa;
    Ki = Kia;
    Kd = Kda;
  }
  else if (region == 'b') //controller parameters for the middle part of tube
  {
    Kp = Kpb;
    Ki = Kib;
    Kd = Kdb;
  }
  else  if (region == 'c') //controller parameters for the upper part of tube
  {
    Kp = Kpc;
    Ki = Kic;
    Kd = Kdc;
  }


  err = ref - ballPos; //current error
  //calculates the proportional part
  controlP = Kp * err;
  //calculates the integral part
  controlI = prevControlI + Ki * err * samplingTime; //integral-->area
  //calculates the derivative part
  controlD = Kd * (-ballPos + previousBallPos) / samplingTime; //neglecting the
reference, because of infinitive derivatives on step changes


  //calculates the total control
  control = controlP + controlI + controlD;
  if (control > 100) {//Technically impossible to actuate, but necessary to saturate in
program...
    control = 100;
    controlI = prevControlI; //...to prevent integrator wind-up
  }
  else if (control < 0) {//Technically impossible to actuate, but necessary to saturate in
program...
    control = 0;
    controlI = prevControlI; //...to prevent integrator wind-up in opposite direction
  }
  prevControlI = controlI; //stores the integrator value for the next iteration
  prevErr = err; //stores the value of the error for the next iteration
}
```

3. Design Evaluation

Briefly describe the success of the hardware by comparing the experimental results to the model simulations.
Comment on the obtained results of the system performance and the graphs.

Our attempt to assemble and control the system was not successful due to mulitple factors.

First: The proper fan was not availabe in the market(i.e a blower fan with strong rpm and large cfm) and thus we had to comprimise and use not only one but two cooling fans.

Second: The cooling fans combined where not enough to lift a ping pong ball and thus styrofoam ball was used which reduced the reliability and predictability of system and yet the two fans had difficulty lifting the styrofoam ball.

Third: Due to insufficient power by fans we were unable to make holes/slits in the tube for better height control.

Fourth: Ultrasonic sensor does not work well as it doesn’t measure well styrofoam material and on that the wind from the fan cause a lot of disturbance to the sensor that can be challenging to filter.

In our code we attempted to divide the tube into regions to help linearize the system as airflow in tube is inconsistent in tube. Using PID control the pwm signal is adjusted and thus control the fan speed.

These factors mentioned earlier resulted in the following to happen: System only worked for about 70 to 100% of the desired height to be controlled with almost good accuracy but still oscillates around equlibrium point.

4. Appendix

Append detailed wiring diagrams (if details are not included in earlier figures), anything else supporting the system details section.