BALL BALANCING BOT

May 4, 2019

GROUP B121
CHELLA THIYAGARAJAN (ME17B179)
MARIE RITTER (ME19F001)
NAVEEN RAPHAEL (ME17B178)
SHRUTHI (ME17B182)

May 4, 2019

Contents

1 ABSTRACT

This project aims at building our practical knowledge on the real-world application of control systems. The problem statement of the project was to demonstrate a mechatronic system that can balance a ball along a linear groove track by incorporating closed loop P/PI/PID control. We first started with ideating the mechanical design of the setup based on which we created a SolidWorks model. With assistance from the Central workshop, the mechanical model was fabricated. Simultaneously, the various components of the control system were purchased – Arduino, Servo motor, Ultrasonic sensors, bread board, adaptor, connecting wires and a potentiometer. After correctly placing all the components on the mechanical setup, we started working on the Arduino code and the Simulink model. We have incorporated a PID control in our project, for which the Kp, Ki, Kd values have been obtained from the Simulink model. All necessary connections were then made, and we proceeded to test our mechatronic system. There were several issues in the working of the system, some of them being, excessive noise in sensor output, no sensor output when ball touched the sensor, slight errors in Kp, Ki, Kd values etc. After overcoming these, we were successfully able to balance the ball at the centre of the groove track. With the help of a potentiometer, we were also able to balance the ball at a desired location on the groove.

2 FLOW CHART OF MANUFACTURING AND ASSEMBLY PROCESSES

Figure 1: flow chart

Firstly, we build the CAD assembly of the whole model. With this, we went to a carpenter and explain what we wanted to have. He built for us the frame out of wood. We went to the workshop to add the motor and the sensor to the frame. They helped us build the attachment for the motor. This required wood to be cut in the right dimensions and the hole to be drilled. In the end to assemble everything we used glue, screws and nails for the motor and glue and cello tape, for the sensor . The whole process is briefly shown in figure 5 as a flow diagram.

3 CAD MODEL DESIGN

Solid Works

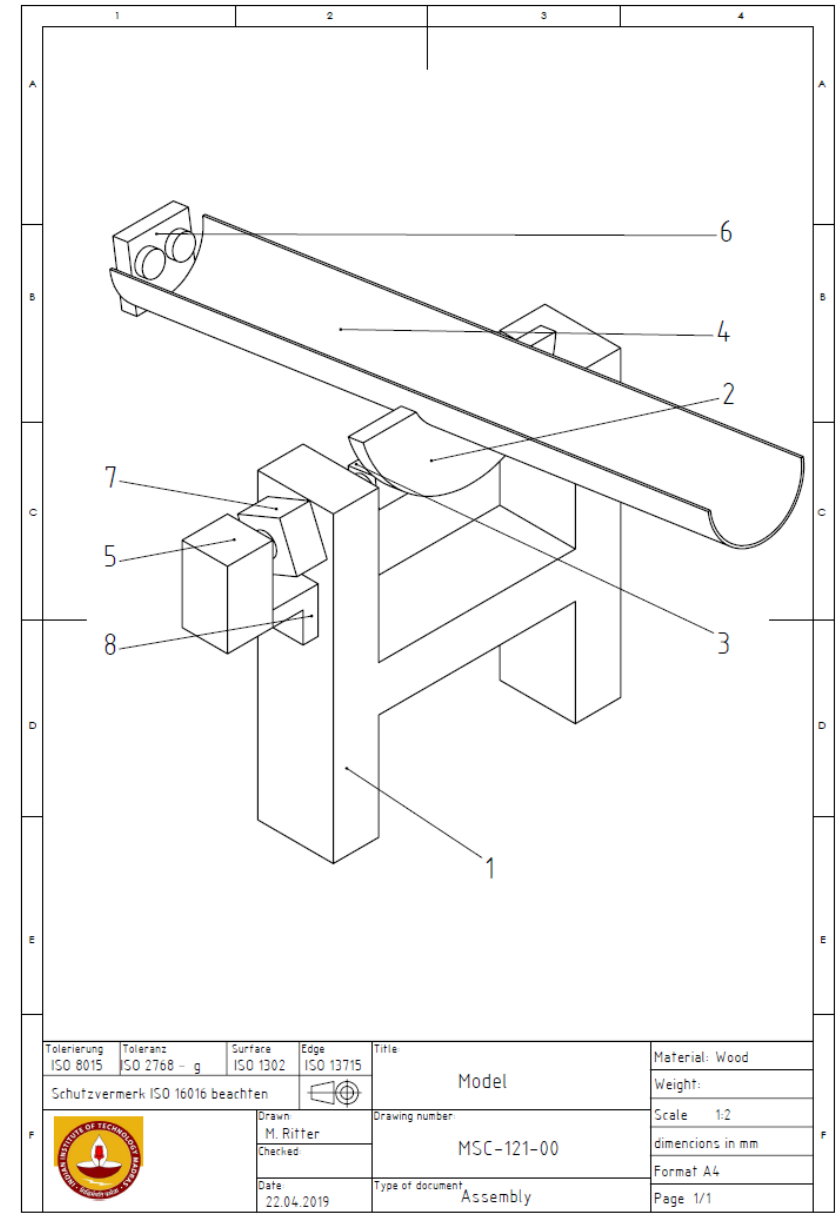


Figure 2: CAD model

4 COMPLETE COMPONENTS LIST

1	2	3	4	5	6
Pos.	Quanti ty	unit	Title	Part number	Note
1	1	pc	stand		manufactured
2	1	pc	connection joint to pipe		manufactured
3	1	pc	joint		manufactured
4	1	pc	half pipe		manufactured
5	1	pc	servo motor	TowerPro MG995	purchased
6	1	pc	ulatrasonic sensor	HC-SR04	purchased
7	1	pc	connection motor joint		manufactured
8	1	pc	attachment motor to stand		manufactured
9	1	pc	arduino uno r3	ATmega328P	purchased
10	1	pc	400 Tie-Points Solderless Breadboard	PRT-3300	purchased
11	7	pc	jumper wire male to male		purchased
12	11	pc	jumper wire male to female		purchased
13	1	pc	10 kohm potentiometer		purchased
14	1	pc	dc adapter	Max Power MP2019	purchased

Figure 3: Components

5 ELECTRONIC CIRCUIT DESIGN

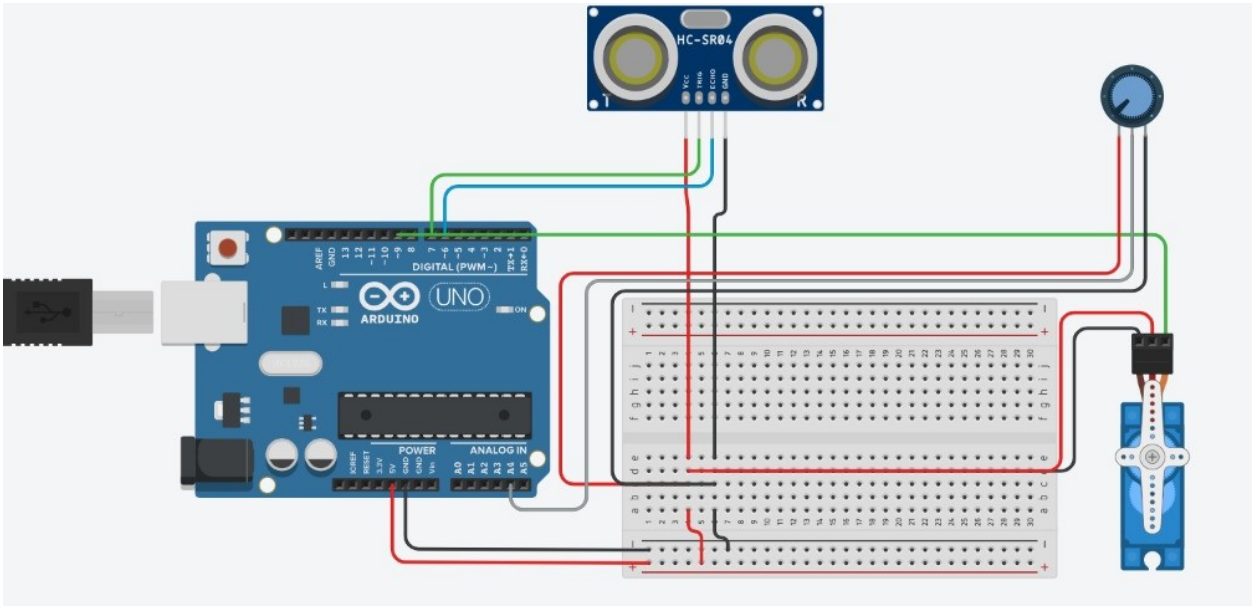


Figure 4: Circuit Design

Servomotor:

- Servomotor - It is a rotary actuator which helps in controlling angular or linear position, velocity and acceleration precisely. Therefore, Servomotor will be the perfect choice for the actuator.
- The MG995 Servo Motors comes as a set of two motor modules that boasts three pairs of blades which can rotate from 0 degrees to 180 degrees. We only need a range of 90 degrees rotation in our project
- The servo-motor produces torque at the joint of crank-rocker mechanism which is used to rotate the joint and so indirectly the groove.
- It is controlled by Arduino (microcontroller) which sends signals to change its angular position as per the feedback given by the ultrasonic sensor.
- 10 kg-cm torque means that the motor can pull a weight of 10kg when it is suspended at a distance of 1cm. That is more than enough for our system because the system that must be moved has a weight of circa 300g and our grove has a length of 20 cm from the middle.

Ultrasonic sensor:

- In the kinematic analysis, we inferred that the location of the ball on the groove is fundamental for the project
- The sensor should not interfere with the motion of the ball.

- The ultrasonic sensor is more economic compared to the more precise LIDAR sensors. The sensor (HC-SR04) we have identified has an accuracy of 1cm, and it can measure distances up to 400 cm. It returns a pulse output if an object is detected, or outputs 0 otherwise.

Bread Board:

To have a better overview and a common ground and voltage we decided to use a bread board. Because the electronic system is small a half bread board with 400 pins is enough for our system.

Potentiometer:

To balance the ball at any place at the grove, you need an input to the Arduino which you can change quite easy and gives you a linear change in the input as well. For that a potentiometer is useful. When you turn the potentiometer, the resistance is changing and so the voltage input to the Arduino as well.

DC Adapter:

The DC adapter need to be a voltage source with at least 9V up to 12V. To get the motor smoothly moving the adapter must provide at least 1A as well. That's why we decided to use the dc adapter max power MP2019. It has an output of 9V and 1A.

6 FABRICATION OF THE CAD MODEL AND ASSEMBLY OF SENSOR, ACTUATOR AND CONTROLLER

We started the sizing the parts after buying the ball. Because the ball must be balanced on the grove it is the main part of the mechanical design and everything has to be fitted to it. The ball has a diameter of 5.1cm

Part 1: The stand needed to be high enough so that the pipe could turn to at least 45 degrees without touching the ground. Our grove has a length of 40 cm, so that the stand needed to be higher than 10 cm. We decided to make it 20 cm high so that we have some clearance and some space for the hole for the joint.

The H (see Figure 5: drawing of stand) form of the stand provides a stable stand, so that nothing beside the pipe is moving with the motor.

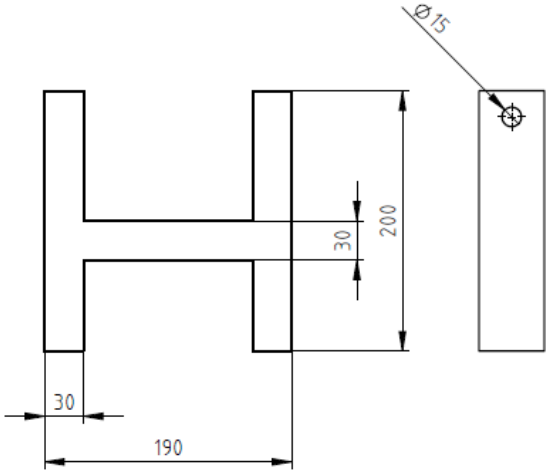


Figure 5: Drawing of Stand

Part 2: The connection between the pipe and the joint (see Figure 6) need to have a design so that it is not disturbing the ball to roll on the grove. That's why we decided to have half annuli which it is stuck via glue to the pipe and fixed with nails to the joint.

Figure 6: Drawing of Annuli Connection

Part 3: The joint (see Figure 7) need to be round at the end, so that it moves properly in the hole. The middle part is rectangular, so that you could easily fix the connection part (2). It is also bigger than the round ends, so that it stays in between the pillar of the stand and it is not moving outside.

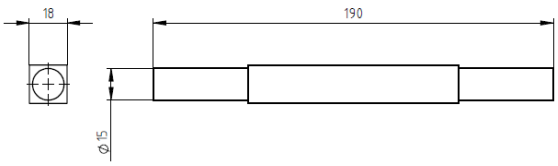


Figure 7: Drawing of Joint

Part 4: The half pipe (Figure 8) must have a bigger diameter then the ball, so that the ball can roll without friction. The diameter of the pipe also needed to be bigger then the sensor, so that sensor only detects the ball and not the wall of the pipe. That's why we decided to buy a pipe with an inner diameter of 6cm. The required length of the grove was 15 cm. To get some bonus point and to have enough space for the attachment of the sensor we decided to get a half pipe with a length of 40cm.

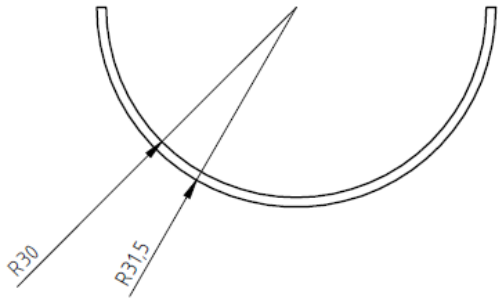


Figure 8: Drawing of Half-Pipe

Part 8: We did not want to use extra links to connect the motor with the pipe. Therefore, we decided to connect to motor directly to the joint. For that we need a part which increased the area where the motor could be connected.

Part 9: Through the L-form of the part for the attachment of the motor to the stand is the area where this part and the stand is connected increased so that a better transmission of the force to the stand is reached and you have more surface to fix it via nails and glue together. The dimension of this part resulting from the design of the other parts.

7 SIMULINK MODEL

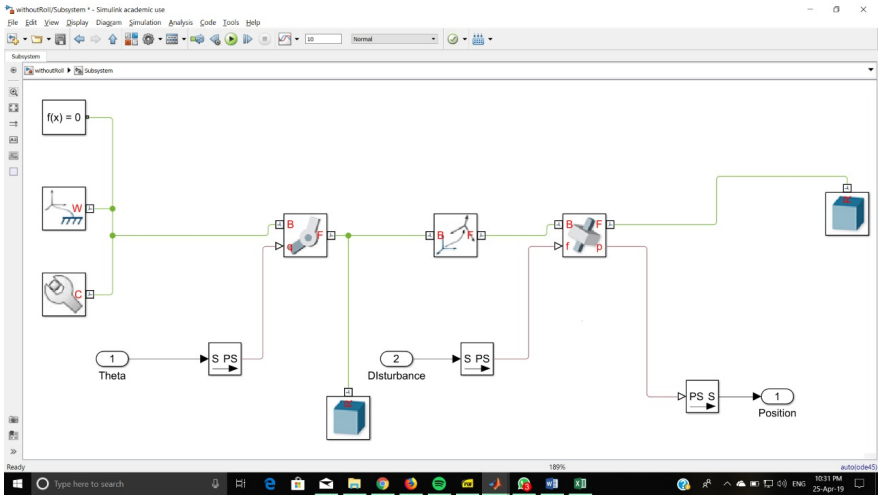


Figure 9

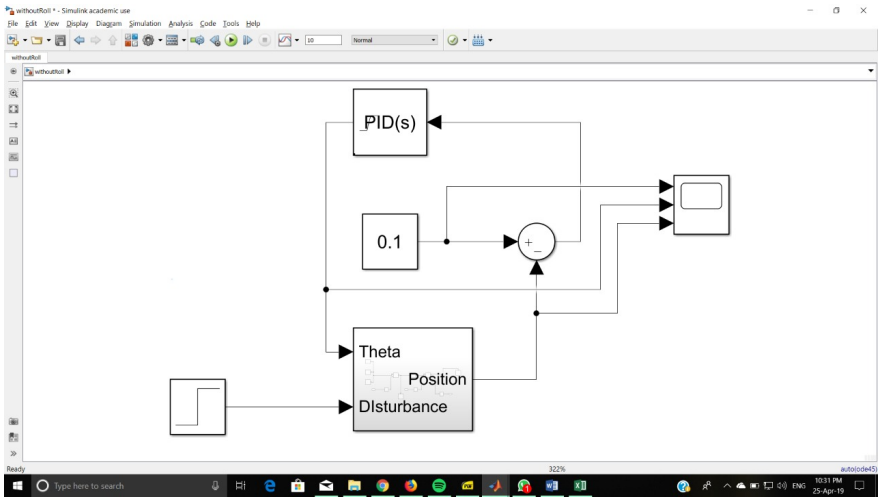


Figure 10

It must be noted that this SimScape diagram does not incorporate rolling, however, this can be approximated by changing the lass of the ball. The values for the PID controller was initially identified from MatLab, and then tuned a little to fit better.

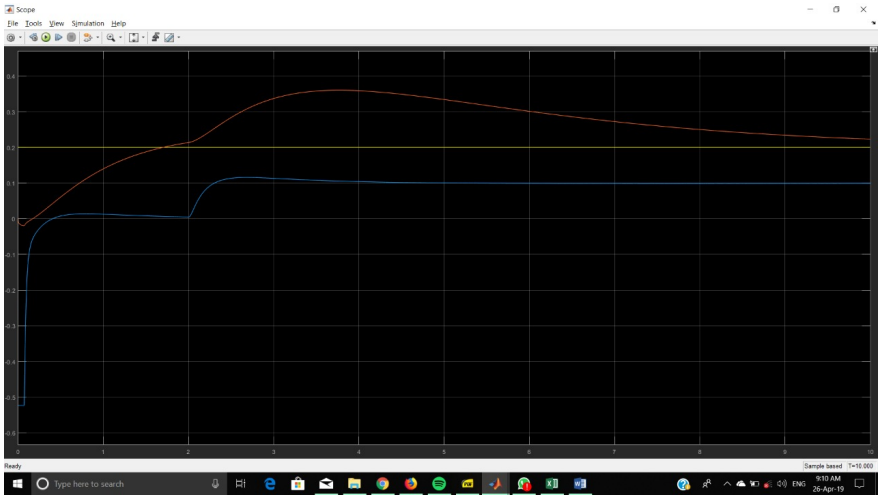


Figure 11: The Graph from the Scope: the red graph is the position of the ball, the yellow graph is the final position, and the blue line is the output theta in radians.

8 INTEGRATION OF SIMULINK MODEL AND ORIGINAL MODEL

A PID control is assumed for this project. Initially, since the mechanical model was built much earlier than expected, we tried to find the PID values through the ultimate cycle method. At the same time, we also built the simulink model. The Ultimate cycle method proved to be quite difficult, and the simulink model was made faster. So, we then took values from the simulink PID tuning block, which gave a system that is roughly tuned. Manual tuning was undertaken after this, tweaking the values taken from simulink, and thus the tuning was completed.

Working video of our model

Please click the links below to view it.

Footage 1

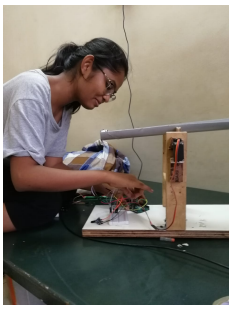
9 CHALLENGES FACED

Challenges faced on Mechanical domain

- ultrasonic sensor touching the ball, not giving proper reading
- Wooden groove too heavy, hence we used PVC pipe
- friction in the setup, continuously need to keep the wooden parts from rubbing against each other

Team Members

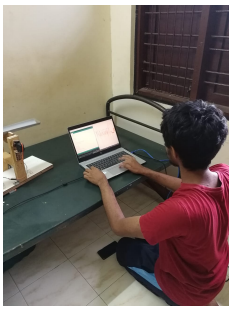
(a) Chella



(b) Shruthi

Figure 12

(a) Marie



(b) Naveen

Figure 13

10 ARDUINO PROGRAM

```
#include <PID_v1.h>
#include <Servo.h>

Servo myservo;

double Setpoint ; // chosen with Potentiometer
double Input; // from sensor
double Output ; // motor
//PID parameters
double Kp=0.605, Ki=0.31, Kd=0.075;
int range=35;
//create PID instance
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);

int val;
int potpin = A4; // analog pin used to connect the potentiometer
const int pingPin = 7; // Trigger Pin of Ultrasonic Sensor
const int echoPin = 6; // Echo Pin of Ultrasonic Sensor

int count=0;
int countSP=0;
double InitAngle=96;
void setup() {
  Serial.begin(9600);
  myservo.attach(9);
  myservo.write(InitAngle);
   Setpoint = (20);
  myPID.SetMode(AUTOMATIC);
  myPID.SetOutputLimits(-range, +range);
  myPID.SetSampleTime(10);
  myPID.SetTunings(Kp, Ki, Kd);
  delay(1000);       //A buffer time
}

void loop() {
   double duration, inches, cm;
   pinMode(pingPin, OUTPUT);
   digitalWrite(pingPin, LOW);
   delayMicroseconds(2);
   digitalWrite(pingPin, HIGH);
   delayMicroseconds(10);
   digitalWrite(pingPin, LOW);
   pinMode(echoPin, INPUT);
   duration = pulseIn(echoPin, HIGH);
   cm = microsecondsToCentimeters(duration);
```

```
  val=analogRead(potpin);
  if(val<350){
      val=12;
   }
  else{
      if(val<750){
          val=20;
       }
       else{
          val=27;
       }
   }

   if(Setpoint!=val){
       if(countSP>30){
          Setpoint=val;
          countSP=0;
       }
       else{
          countSP=countSP+1;
       }
   }
   else{
       countSP=0;
   }

   if(Input-cm>3||cm-Input>3){
       if(count<50){
          if(cm>Input){
              count=49;
              }
          cm=Input;
          count=count+1;
       }
       else{
          count=0;
       }
   }
   if(cm>3&&cm<40){
       Input = (cm);
       myPID.Compute();
   }
   else{
   Output=0;
   }
   myservo.write(Output+InitAngle);

 Serial.print(Input);
```

```
  Serial.print(" ");
  Serial.print(Output);
  Serial.print(" ");
  Serial.print (Setpoint);
  Serial.print(" ");
  Serial.println(myservo.read()-InitAngle);
}

long microsecondsToCentimeters(long microseconds) {
   return microseconds / 29 / 2;
}
```