# ME 2400

## Measurement, Instrumentation and Control

### Project Report

---

# - BALL BEAM BALANCING USING ARDUINO

---

*Author 1 :*
Veerendra Harshal
Roll.No. ME17B124

*Author 2:*
Harshal Katari
Roll.No. ME17B146

*Author 3:*
Sujay Bokil
Roll.No. ME17B120

*Author 4:*
Deepu Reneesh
Roll.No. ME17B138

*Instructor:*
Dr.Arunachalam

# Contents



# List of Figures



# List of Tables

# 1 Abstract

**The project Ball balancing bot follows the goal of developing a model and controller for the bot in order to balance a ball in the centre of a grooved track. If we push the ball away from the centre the bot automatically brings back and balances the ball at its mean position. Recognizing all these challenges involved in the problem statement, we have come up with various strategies to :-**

**1)Design the mechanical model and identify the manufacturing techniques required.**

**2)Instrument the system with a required sensor to track the ball movement. We have used ultrasonic sensors for detecting the position of the ball**

**3)Identify the actuator required to achieve the tasking taking torque, angular speed, etc. into account. We have used a servo motor for controlling the ball position.**

**4)Select the microcontroller in order to control the motor properties, we have spent time to learn controlling servo motors using Microcontroller.**

**5)Identify suitable signal conditioning for the sensor outputs. We have also gone through the basics of PID control algorithms in order to design the system required for achieving the problem statement**

# 2 Flowchart

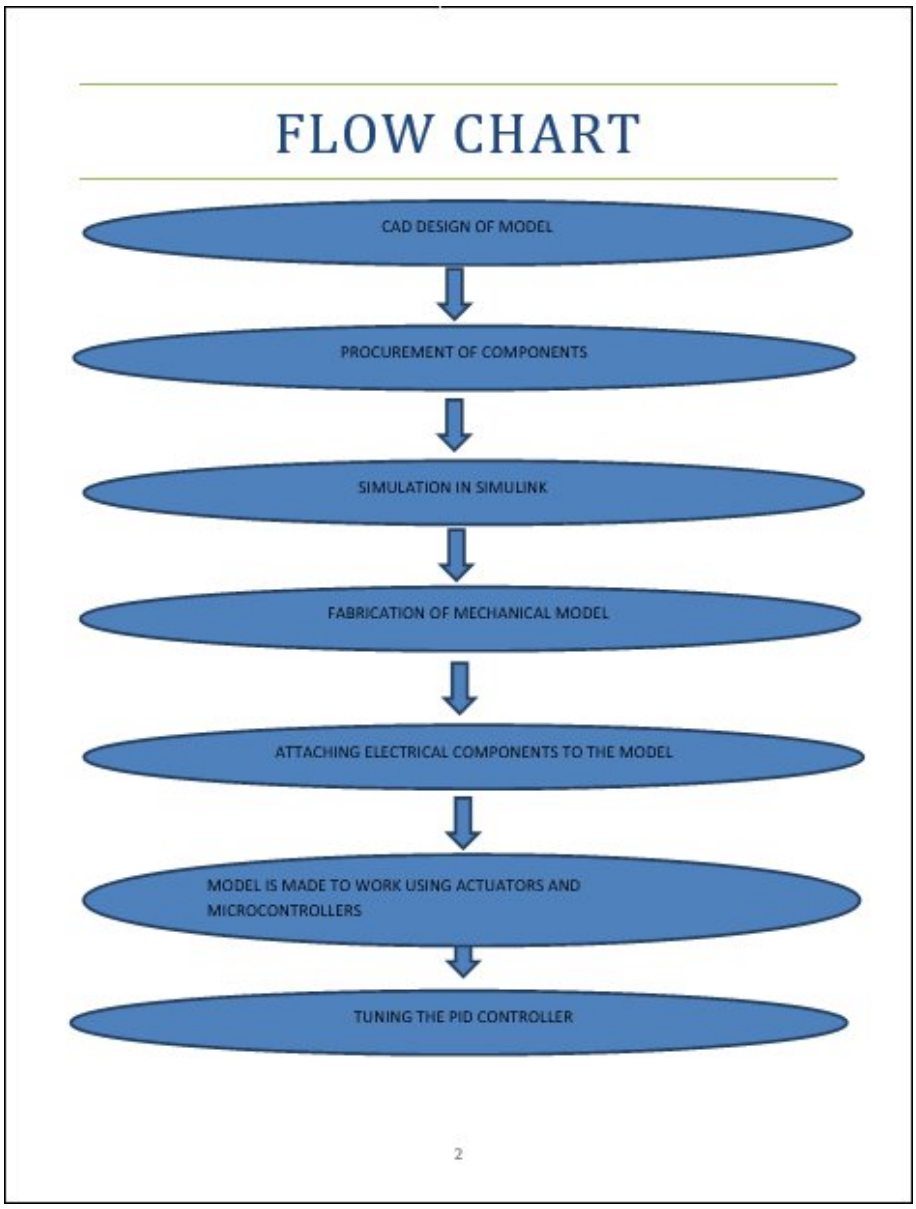


Figure 1: Flowchart

# 3 CAD model design (with drawings and proper dimensioning)

CAD model design (with drawings and proper dimensioning)

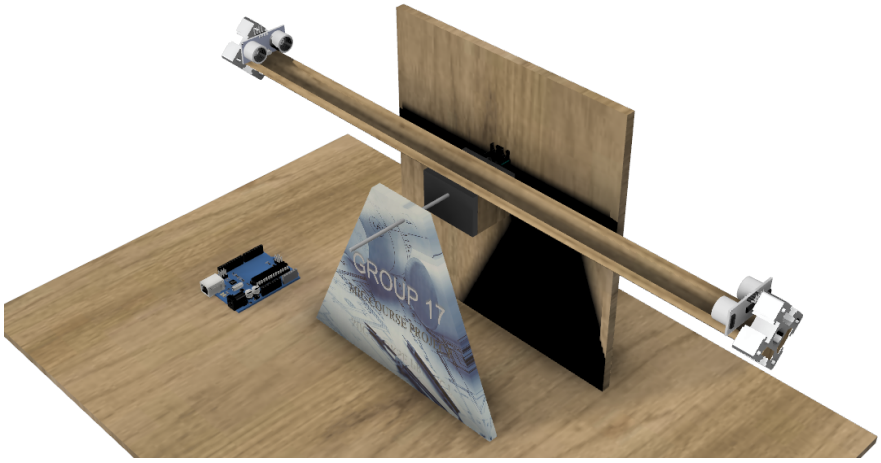


Figure 2: Model Rendering

The components to be manufactured are

- The base of the frame
- The back of the frame
- V- shape groove of 50 cm

The components to be purchased are the wood for the frame, the pieces of aluminum strut and the electronic components. The minimum track length was 30 cm so we decided to go for 50 cm making the torque calculations based on the maximum torque provided by the motor and the ball weight. The width of the groove was decided based on the ball diameter (around 4-5 cm) so that it would fit in perfectly and not be thrown out. The ball was chosen so that it was 50 cms which was the given ball weight and then the CAD model was designed based on it. The height of the frame was set up to ensure 60 degrees angle of rotation from the horizontal on either side. The aluminium struts were an ideal option for making a V-groove and we had

to just insert the wooden strips on two sides to make an exact 90 degrees V-groove. L-clamps were used to fix the parts of the frame because of the sizing of the frame and the fact that that glue would not be enough to keep it together. We used glue gun to fix the servo motor and added a triangular support on the other side to prevent bending of the servo motor axle. The attachment to connect groove with servo was designed to be as simple as possible.

# 4 Electrical Components

1. Generic Arduino Uno R3 ATMega328P and cable
2. Ultrasonic Sensor HC-SR04
3. MG-995 Servo Motor with attachments
4. Jumper wires
5. Single strand wires

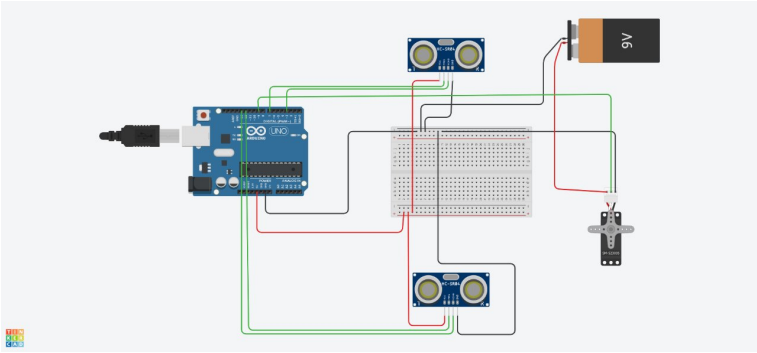


Figure 3: Circuit Diagram

**Correction elements** The correction element used by us is a MG995 Servo motor. In this application, we don't use a stepper motor as there is no feedback inbuilt and it's not necessary to give the output in steps. So, we use a servo motor instead. The Servo motor is sized based on the torque requirement. Based on the Fusion 360 model, we estimated

the maximum torque required using the ball weight and groove length. This torque should be less than the maximum torque provided by the servo. Also, the working voltage can be provided directly through the Arduino and for strength factor and reduce the cantilever bending of the axle of servo, we decided to go with metal gear type instead of the plastic gear type. The servo specifications are as follows.Weight : 55g Dimension : 40.7 × 19.7 × 42.9 mm Operating Speed : 1) 20sec / 60 deg (4.8V no load) 2) 16sec / 60 deg (6.0V no load) Stall Torque : 1) 10 kg-cm at 4.8V 2) 12 kg-cm at 6V Operation Voltage : 4.8 - 7.2Volts Gear Type: All Metal Gears Dead band width: 5 $\mu$s Stable and shock proof double ball bearing design Temperature range: 0 °C – 55 °C. Control System : Analog Operating Angle : 120degree Required Pulse : 900us-2100us

- Working Voltage: DC 5V
- Working Current: 15mA
- Working Frequency: 40Hz
- Max Range: 4m
- Min Range: 2cm
- Measuring Angle: 15 degree
- Trigger Input Signal: 10$\mu$S TTL pulse
- Echo Output Signal Input TTL lever signal and the range in proportion
- Dimension : 45 X 20 X 15 mm

# 5 Fabrication of CAD model and assembly of sensor, actuator and controller.

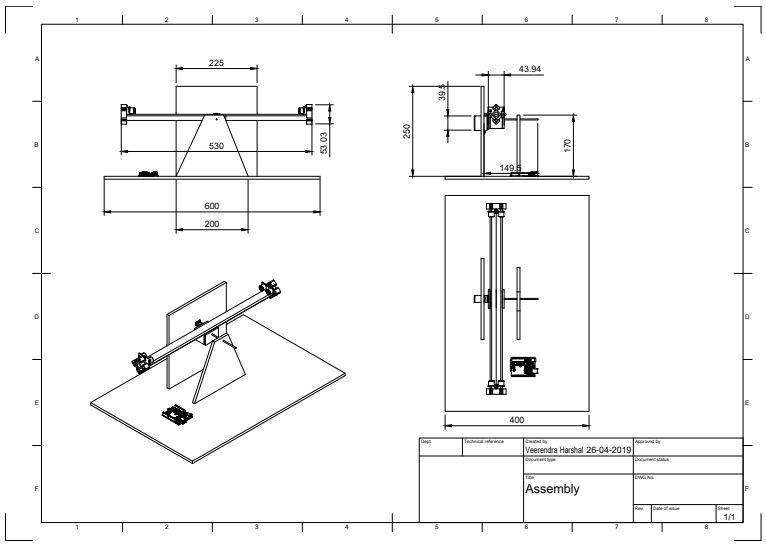


Figure 4: Assembly Drawing

The flowchart for the process is as follows.

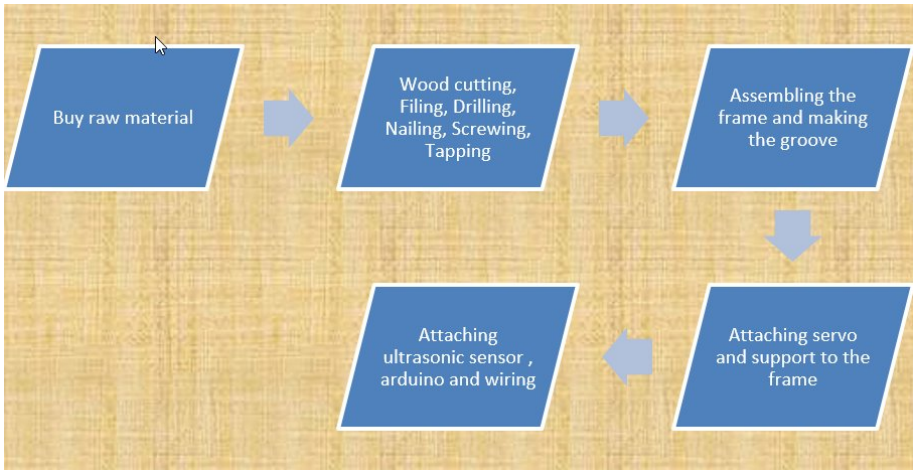


Figure 5: Fabrication Flowchart

The raw materials we brought for manufacturing mechanical model are

- 3 X 2 inches sheet of 8 mm thick wood
- Nails and screws
- L clamps Fevicol glue
- We have used a hacksaw, jigsaw and filer to cut the wooden sheets.
- The support and main frame are attached to the base using L-clamps and screws.
- The support is put up to ensure that there is no cantilever bending of the Servo axle and the groove doesn't fall out.
- The groove is made using wooden strips and inserting them into aluminium struts.
- The holes for attachment are drilled using a drill machine and its parts were attached using nails and hammer.
- The servo is attached to the frame by using a slot-insert mechanism and then screwing and gluing it to the back of the frame.
  itemThe servo attachment is glued to the rotating attachment and also nailed to it and that is attached to the servo using a screw.
- The Arduino is attached to the base using tape and the ultrasonic sensor is attached to the groove using double sided tape.

# 6 Arduino

**Arduino Model and specifications**

We used an Arduino Uno with the following specifications

- Microcontroller ATmega328P .6.
- Operating Voltage : 5V
- Digital I/O Pins
- 14 (of which 6 provide PWM output) PWM

- Digital I/O Pins .6 Analog Input Pins, 6 Flash Memory 32 KB (ATmega328P) of which 0.5 KB used by bootloader

We are using 2 ultrasonic sensors for measuring distance and a servo motor as our actuator. Sensors connections and pin numbers. We are using two standard libraries in our code namely PID and Servo. **Flowchart**
**The control flow in the process occurs in the following way**

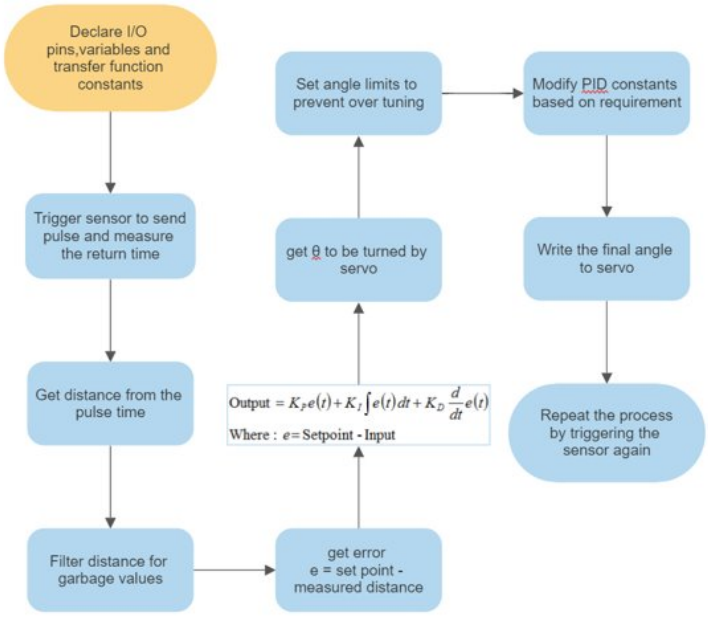


Figure 6: Flowchart of Block Diagram

**The code used for the Project**

```
// note: for my beam to be horizontal,
Servo Motor angle should be 18 degrees.


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

#define LEFT 0
```

```
#define RIGHT 1
#define TRIG 0
#define ECHO 1


const int maxAngle = 40;
float prevus[2];
int tolerance = 4;
const int servoPin = 9;
int pingPin[2][2];
const int eqbAngle = 18;
float currentAngle = eqbAngle;

float Kp = 0.357;
//Initial Proportional Gain
float Ki = 0.007  ;
//Initial Integral Gain
float Kd = 0.046;
//Intitial Derivative Gain

double Setpoint, Input, Output, ServoOutput;
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);
//Initialize PID object, which is in the class PID.

Servo myServo;
//Initialize Servo.

void setup() {

  prevus[LEFT]=prevus[RIGHT]=0.0;
  pingPin[LEFT][TRIG]=12;
  pingPin[LEFT][ECHO]=13;
  pingPin[RIGHT][TRIG]=7;
  pingPin[RIGHT][ECHO]=3;

  Serial.begin(9600);
//Begin Serial
  pinMode(pingPin[LEFT][TRIG], OUTPUT);
```

```
  pinMode(pingPin[RIGHT][TRIG], OUTPUT);
  pinMode(pingPin[LEFT][ECHO], INPUT);
  pinMode(pingPin[RIGHT][ECHO],INPUT);

  myServo.attach(servoPin);

position as the input to the PID algorithm
  long Angle_old = myServo.read();

  myPID.SetMode(AUTOMATIC);
//Set PID object myPID to AUTOMATIC
  myPID.SetOutputLimits(-80,80); //Set Output limits to -80
  and 80 degrees.
  float pos = 0;
}

void loop()
{
   Setpoint = 21.5;
  float leftDistance = readPosition(LEFT);
  float rightDistance = readPosition(RIGHT);
  prevus[LEFT] = leftDistance;
  prevus[RIGHT] = rightDistance;
//Serial.print("Left");
//Serial.println(leftDistance);
//Serial.print(",");
//Serial.println("Right");
Serial.println(rightDistance);
float minDistance = (leftDistance<rightDistance)?leftDistance:rightDista
//
  float error= fabs(minDistance);
  Input = error;
  myPID.Compute();

  Output = (fabs(Output)<maxAngle)?fabs(Output):maxAngle;
  ServoOutput = (leftDistance<rightDistance)?eqbAngle-Output:eqbAngle+O

//   if(currentAngle+4<ServoOutput){
```

```
//     myServo.write(++currentAngle);
//   }
//   else if(currentAngle>ServoOutput+4){
//     myServo.write(--currentAngle);
//   }

 // Serial.println(ServoOutput);
  myServo.write(ServoOutput );
}


float readPosition(int PIN)
{
  float duration;
  float cm;
  digitalWrite(pingPin[PIN][TRIG], LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin[PIN][TRIG], HIGH);
  delayMicroseconds(10);
  digitalWrite(pingPin[PIN][TRIG], LOW);

  duration = pulseIn(pingPin[PIN][ECHO], HIGH);
  cm = duration/(29*2);

  if (cm>25){
    cm = 25;
    prevus[PIN]=cm;
  }
//   else if(cm>4){
//     prevus[PIN]=cm;
//   }
//   else{
//     cm = prevus[PIN];
//   }
//   Serial.println(cm);
  return cm; //Returns distance value.
}
```

**Signal Conditioning:**
The sensor doesn't always give proper values. Beyond the limits of the sensor (too close and after maximum value) the sensor is giving garbage values. To overcome this issue, we are using two sensors. When the sensor gives value more than half the maximum length of the track, that input is ignored for computation.

**Control Design:** PID control is employed in the ball balancing system. The fundamental control that is employed is the proportional control. To reduce the amplitude of oscillations, integral control is employed. To ensure smoothness in control, differential control is used. Initially the values of parameters given by the Matlab Simulink were used, later we changed the values accordingly. The final values we got were as follows.

# 7 Simulink

We used a transfer function to simulate the plant in the model. We went for a PID controller and used the tuner function and subsequently iterated our transfer function to match the real model. Inspite of that we observed a significant deviation from the actual values.

We leant

- Transfer Function is less reliable than Simscape
- The PID parameters are offset due to the inability to incorporate real life constraints in the plant.
- The Scope function can be used to visualize the working of the system with reference to a reference signal.

**This is Our Simulink model used**

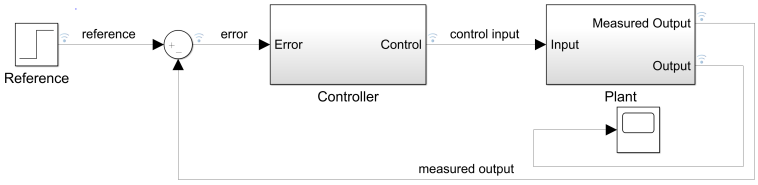


Figure 7: Simulink Model

**The values changed after iteration**

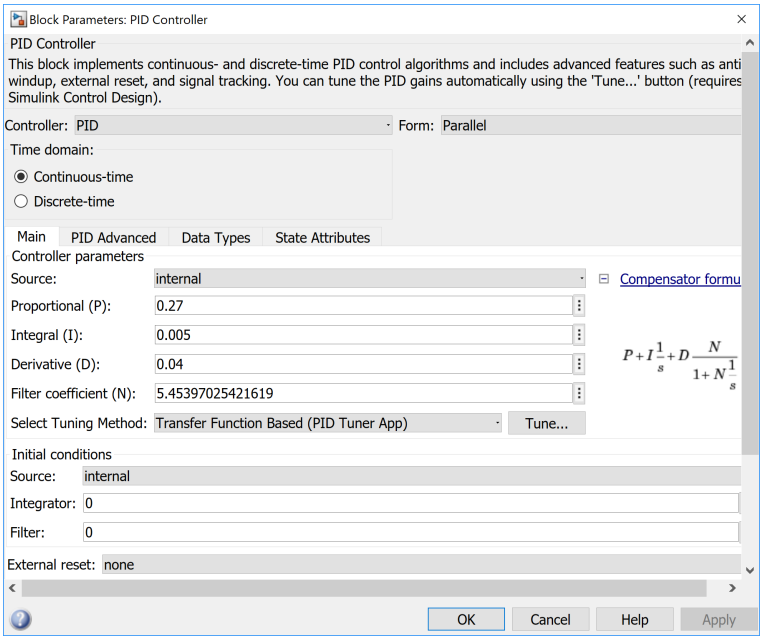


Figure 8: Simulink screenshot

**This is the simulink graph obtained on initial parameters**

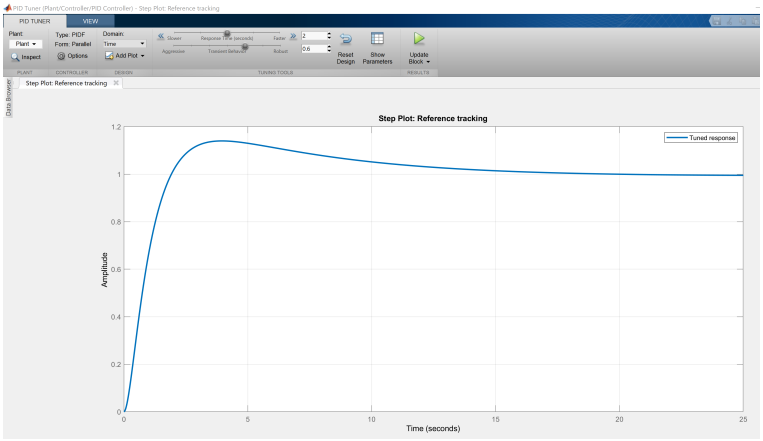


Figure 9: Simulink graph

# 8 Integration of Simulink and the model

The Simulink model was used to simulate the system and get Kp, Ki, Kd values for tuning of the the system. The values that we got from the simulation were used as the starting point for the tuning and then the tuning was done manually.

Table 1: PID Parameters

| | Value used | Value generated |
|---|---|---|
| Kp | 0.357 | 0.27 |
| Ki | 0.007 | 0.005 |
| Kd | 0.046 | 0.04 |

# 9 Any additional works done for BONUS marks

• The final effective track length for which the ball balances is 45 cms after fitting of the sensors.
• The ball is able to balance at only some multiple setpoints the beam but not in the vicinity of the sensors.

# 10 Challenges faced and how we solved them

• Taking out times from the schedules of everyone and working together was one of the challenges and so we split up in groups of 2 and we allotted works based on who was relatively free at the time.
• Ideation and making of the frame was the easiest part and everyone helped in it. The difficult part was to make the attachment connecting the servo to the groove since we changed our design a bit in between due to servo not working properly. We decide to go for the simplest design possible with which we can attach the servo to the groove with more durability.
• One of our servo motors started malfunctioning after working properly for some time and so we had to remove the servo and put everything back again, this time with a new one.
• Tuning was one of the most difficult jobs and after understanding it's mechanism we all took turns doing it.
• At one point the project became so delayed that we thought that we won't

be able to make it but then everyone thought that if they go down, the whole team goes down and we got through that phase.