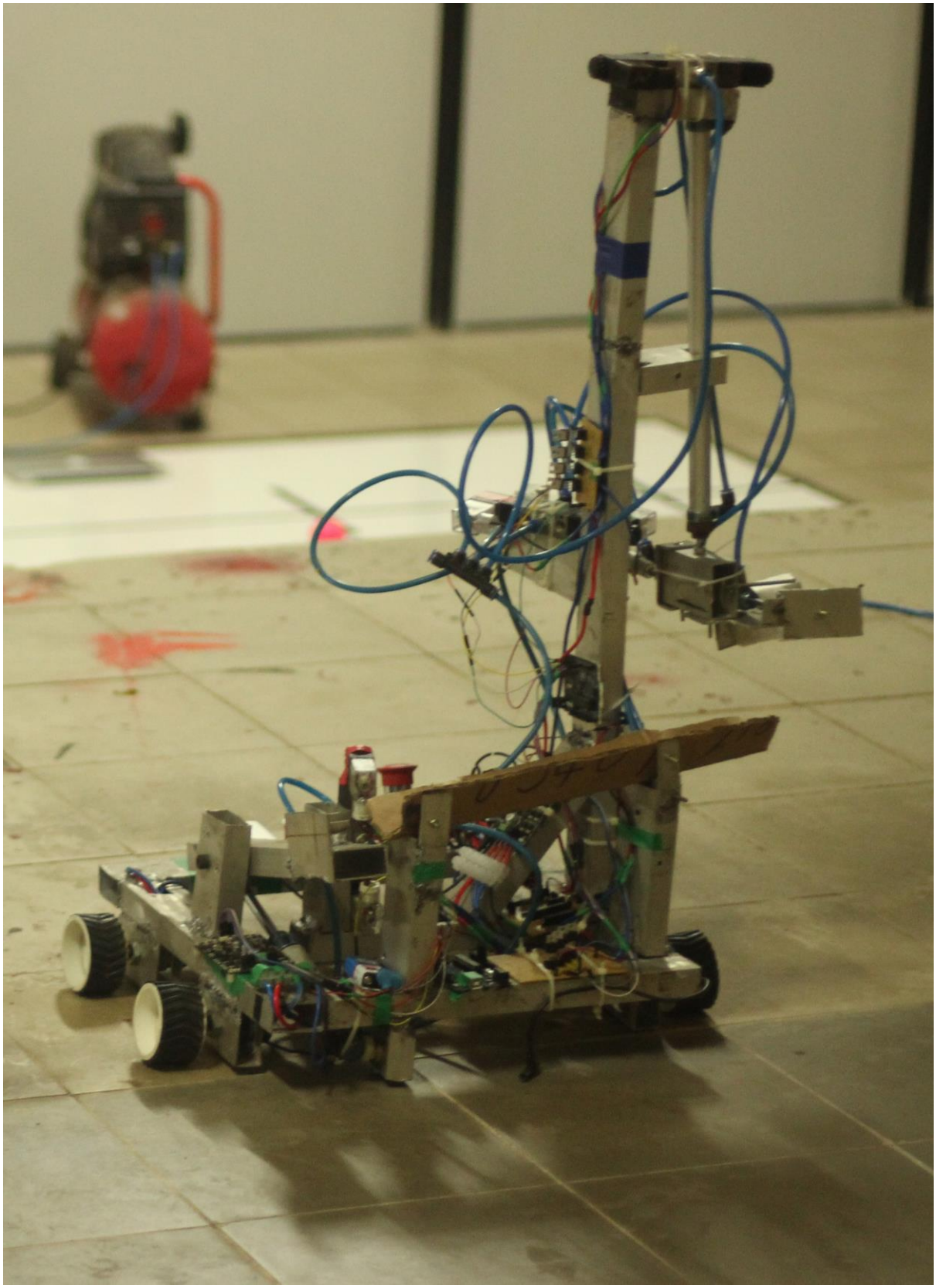
TASK 1 (Final Report)



Problem Statement:

1. The bot will have to follow the line and start from the autonomous start point.

2.The bot need to start just by pushing a button.

3. at first node the bot has to pick up the ball.

4.The bot need to identify the ball colour and then choose the path number that is assigned to the detected colour

Red -> Path 1

Yellow -> Path 2

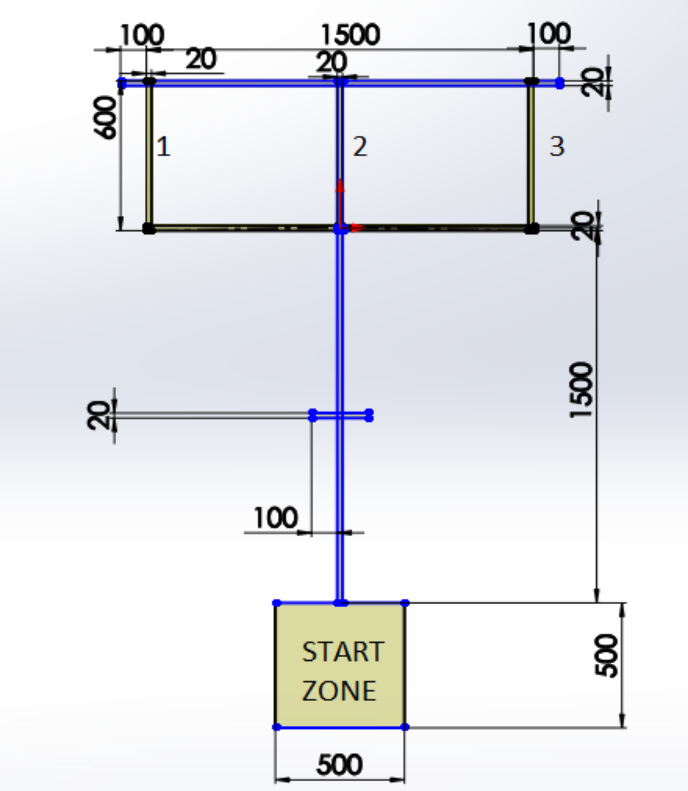
Green -> Path 3

5.At last the bot need to reach the last node and shoot the ball for 2m.

EOT

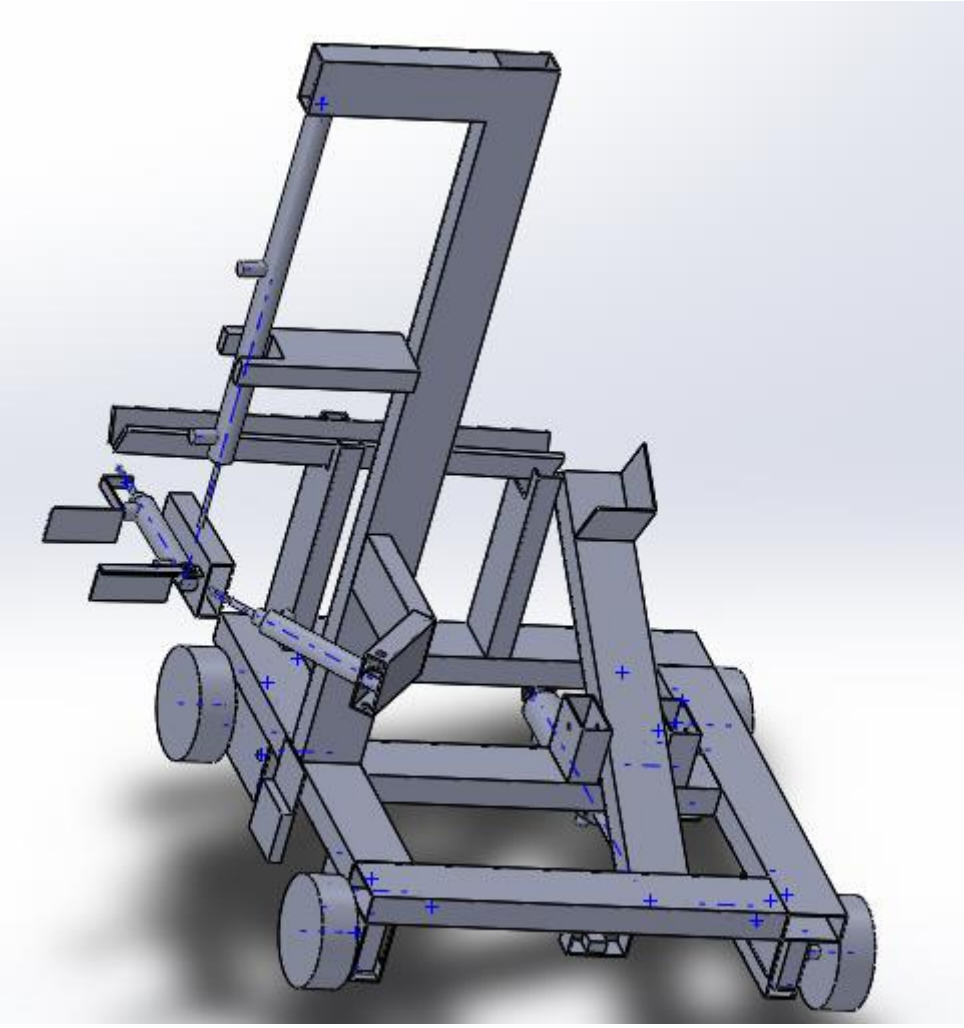
Rules and Regulation

1. The following task be completed in 3minutes
2. The bot should fit in the starting zone, ie the maximum dimension of the bot is 300*400*1000.



MECHANICAL DEPARTMENT

DESIGNING: The complete designing is done in solidworks.



BASE

The base is made by welding aluminium channels vertical to each other making a square base. A channel is welded parallel to two of its sides through the centre. In which the gripper pathway and catapult is mounted.

4-wheeled drive is been used.

GRIPPER

A pneumatic is attached to the horizontal support for up and down z axis movement of the gripper. An aluminium channel is attached to the pneumatic to support the holding mechanism. A pneumatic with grip at 2 ends is attached to the support using zip tags. This pneumatic works as the gripper mechanism.

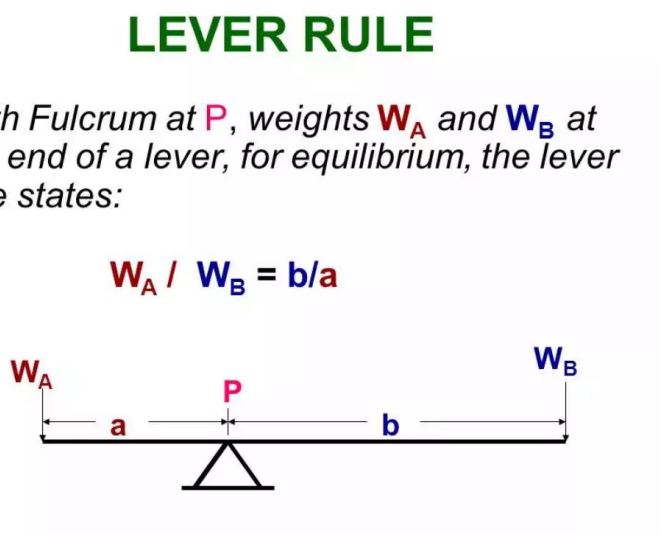
Another pneumatic is attached on an aluminium channel parallel to the gripper mechanism on the vertical support. This is used to change direction of the gripper.

PATHWAY

A pathway is made of cardboard from the gripper to the Catapult. The pathway is supported with vertically welded aluminium channels at definite intervals, which acts as a support.

CATAPULT PART

After the ball reaches the catapult which has a pneumatic connected to the other end to the one where the ball landed, the pneumatic actuates releasing the ball at some force. The catapult works on the lever rule.



All the pneumatics are actuated at a pressure of 4 bar.

After the manufacturing phase, the robot is handed over to the electrical department.

CHALLENGES FACED DURING FABRICATION

1. Initially all the four wheels were not aligned properly so we had to change the mounting a little to solve it.
2. After the completion of the gripper and the catapult, we noticed that the ball could not reach the catapult from the gripper as the plane of the catapult was higher than that of the gripper. So, the weld for the channel supporting the Z axis gripper was broken and re-welded with a longer aluminium channel and a longer pneumatic with more stroke length to overcome this issue.
3. The travelling of the ball from the gripper to the catapult had to be done with trial and error for precision. The horizontal gripper (used for hitting the gripper to turn it towards the pathway) had to be mounted with precision and with many trials. Also, determining the pressure required for accurate positioning of the gripper was also a difficult task.

ELECTRICAL DEPARTMENT

The motor calculations done in the initial stages during designing phase gave an estimated value of 7 to 8.5 kgcm motor with an rpm of around 350. On looking through the team inventory we found four 10kgcm motors (fully working) and four 5kgcm motors. On suggestions, we found that four 5kgcm motors were enough to drive the bot so we chose to use 5kgcm and 150 rpm motors as the time was still not an issue for the task. During the testing of the motors all the motors had an rpm of around 115 which was quite enough for the bot to drive.

Motor driver that we chose was cytron MDD10A for two reasons although there was always an option to use L298n

1)Better speed control and easier to use.

2)High current rating.

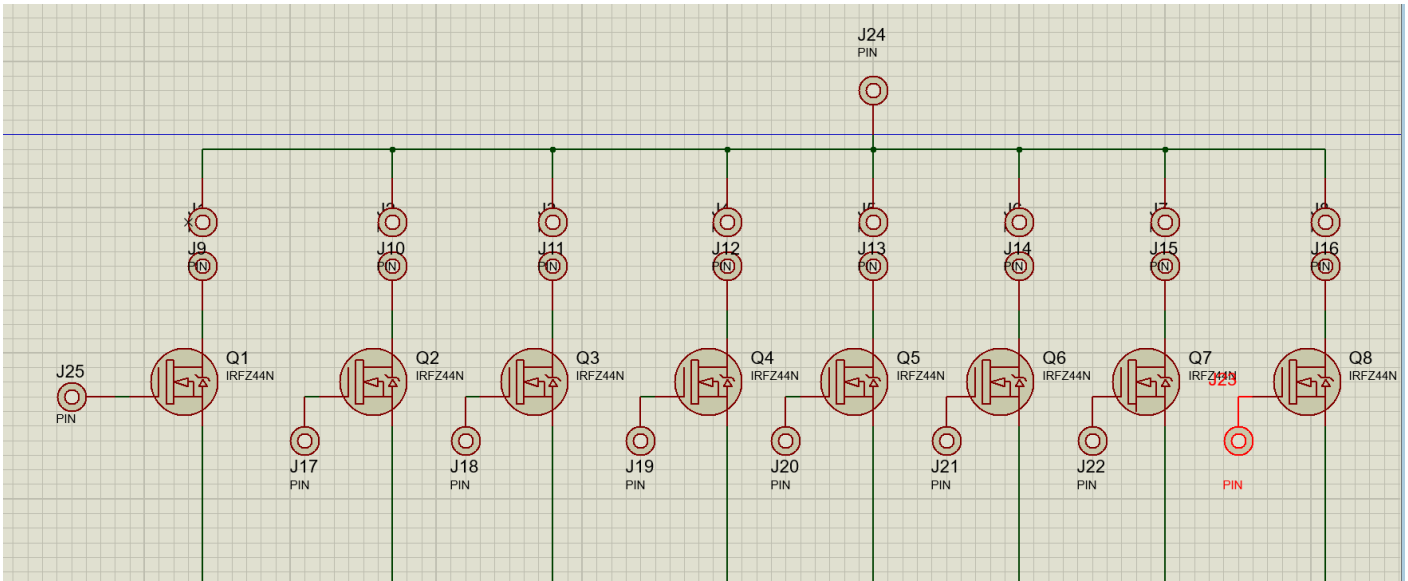
The testing of motors and motor drivers was success we decided to continue mounting of motors on Saturday. Till night we had everything mounted. The mechanical guys drill slots through the bot's body for the wires which was very handy during the wiring the bot in a clean and neat way. The motors were then tested through the buttons on the motor driver and it worked like a charm.

On the next day, there was some problem with the gripper mechanism of the bot which required a little bit of extra support and welding because of which we had to remove all the wires and motors again. The pins of the motor were too brittle to handle any more stress and because of this we had to solder wires again to a little exposed part of motors. It took a lot of concentration and time but at the end a little bit of mseal and soldering gave the desired results. We were again mounting the motors on Tuesday night we were back to saturday.

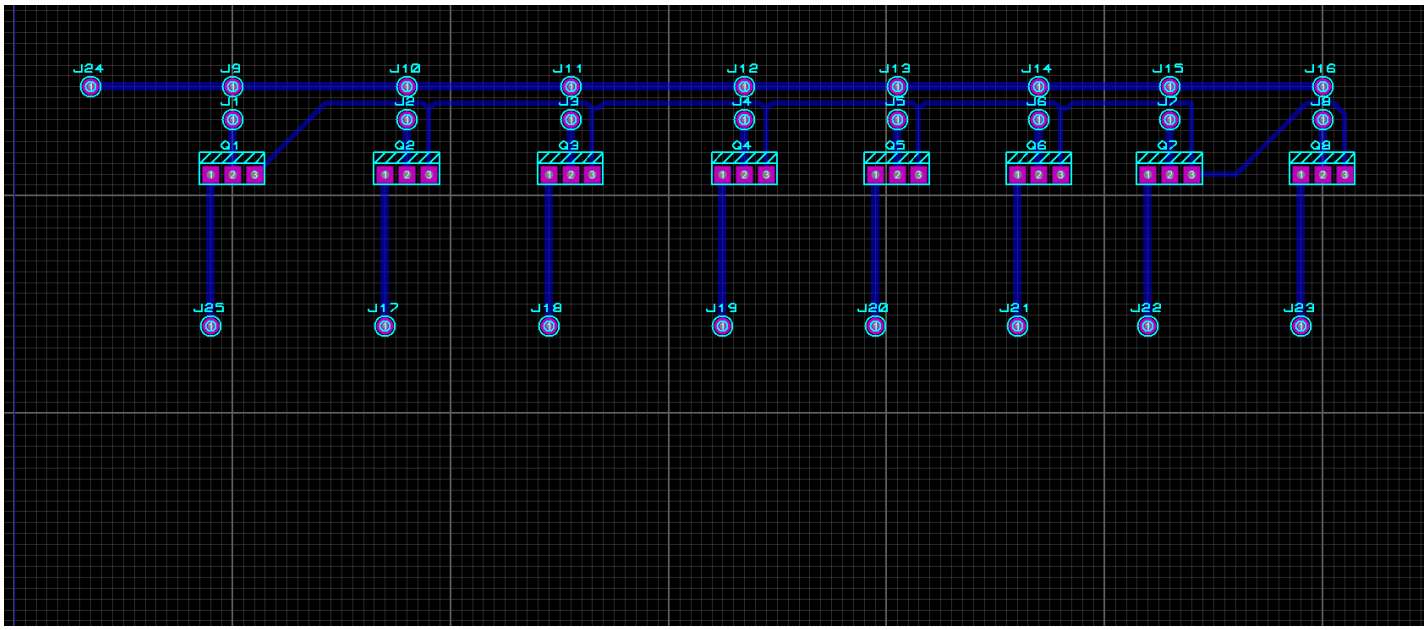
Mounting the mosfet boards for all the four pneumatics was the next important job. We had to make sure each wire was soldered properly insulated properly with heat shrinks and the power supply was proper with proper of each of the boards.

Using mosfets as a switch for controlling solenoid valve

Circuit Schema:



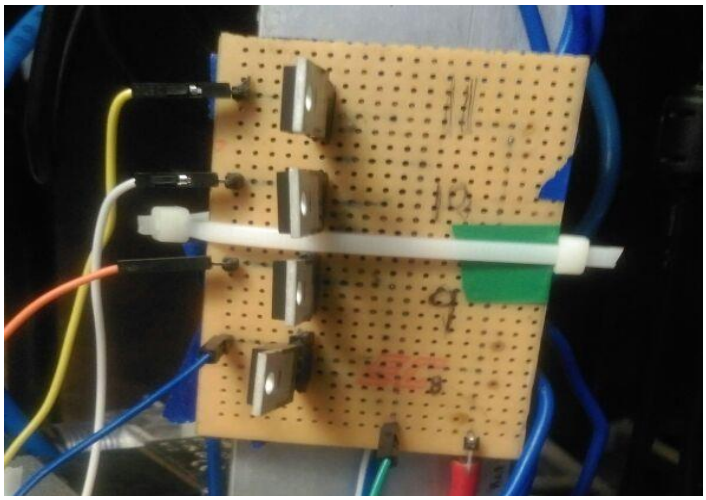
PCB layout:



- We have used IRFZ44N power mosfets to control solenoids using Arduino board(5v output).
- The digital pin of the Arduino is connected to gate of the mosfet.
- The solenoid is placed between the 12 v power supply and drain of the mosfet where positive terminal of the mosfet is connect to 12 v and negative terminal is connected to the drain.
- The negative terminal of the 12 v power supply and gnd pin are connect to the source of mosfet thus providing common ground.
- When the digital pin of the Arduino gives low signal the drain-source channel acts as an open circuit.
- When we the digital pin of Arduino gives high signal(5V) the drain source channel acts as a short circuit.

The circuit for the boards was simple the source was grounded the drain goes to the solenoids negative and the solenoids positive to the +12V supply. The board was already tested with solenoids and all the different DCVs. It was ready to be mounted on the bot. We mounted one of the boards

on the gripper extension tower and soldered the wires to it and power supply the other mosfet board to control the other half of the pneumatics system.



The mosfet board was tested next day and the following problem was seen, a broken jumper, a bad mosfet. We did the testing again and it was working like a anything. The mosfets were heating a little but turning them off solved the problem maybe it was because the mosfets were on which caused the heating.

Powering everything up

There were the following things to power up

1)Two motor drivers

2)2 MOSFET boards

3)2 Arduinos

4)A RASPBERRYPi

5)LSA08

For motor drivers we used a lipo(11.1V) to power it up the current rating of the battery was 2.2Ah which was sufficient to run the motors for required time. The Kill Switch was placed in the load and source for a single switch power control. The max current rating was 1A for the motors so for four motors the total maximum current that can be drawn was 4A and the other components like solenoids had a rating of 129mA which will not effect much to battery drainage. So this battery would be enough to run the bot for half an hour on extensive use. Mosfets were powered using the lipo only

Arduinos were powered from the 9V of the power boards .

This is a 4 channel powerboard to step down the 12v supply from the battery to smaller voltages like 5v, 9v etc required for driving the sensors.

Components used are7809,7805,header pins, 15 A Wires, Solder Lead.

Problems Faced:

1.While designing and selecting the component we realised that the max current ratings of a 7809 is 500Ma, and it can give out at max 1A for 10 seconds before blasting out,

Solution:

We decided to make a universal board with many regulators so that we can split up the current load on a regulator while wiring to avoid losses and also prevent burning off.

2. While making the PCB Schematic it was quite difficult to make 4 channel board .

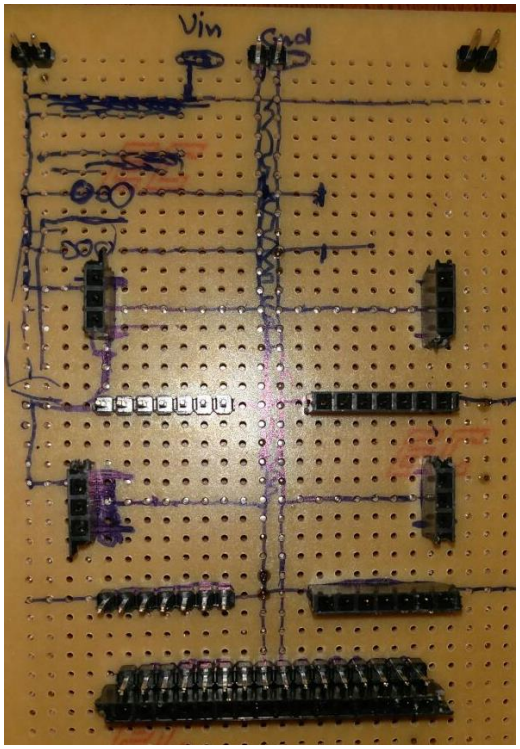
Solution:

We used Proteus to simplify the pcb schematic and make the circuit easy to make the breadboard and used some designing techniques as specified on the web to make a simplified board.

3.Solder Burning issue- When we apply a high voltage like 12v and it draws too much current the solder might burn off easily, so to avoid this we made a thicker line for the 12v and the ground to prevent the burning.

4.Common Grounding Issue-

While testing the mosfets and some of the sensors we were not getting the desired output when the power board was used so we used a common ground to the battery and the microcontrollers and sensor which solved the problem greatly and the same was the issue with our mosfet board.



Raspberry pi we used power bank.

LSA08 we used a 9V radio transistor battery.

CSE DEPARTMENT

The main components of our Code included :-

- PID Algorithm for line following.
- Colour Detection of ball via OpenCV Python.
- Using Pneumatics Control to pick up, place, throw the ball.

PID Line Following:

So what do we do with the error value to calculate how much the output be altered by? We would need to simply add the error value to the output to adjust the robot's motion. And this would work, and is known as proportional control (the P in PID). It is often necessary to scale the error value before adding it to the output by using the constant(Kp).

Proportional:

Difference = (Target Position) - (Measured Position)
Proportional = Kp*(Difference)

This approach would work, but it is found that if we want a quick response time, by using a large constant, or if the error is very large, the output may overshoot from the set value. Hence the change in output may turn out to be unpredictable and oscillating. In order to control this, derivative expression comes to limelight.

Derivative:

Derivative provides us the rate of change of error. This would help us know how quickly does the error change from time to time and accordingly we can set the output.

Rate of Change = ((Difference) – (Previous Difference))/time interval
Derivative= Kd *(Rate of Change)

The time interval can be obtained by using the timer of microcontroller.

The integral improves steady state performance, i.e. when the output is steady how far away is it from the setpoint. By adding together all previous errors it is possible to monitor if there are accumulating errors. For example- if the position is slightly to the right all the time, the error will always be positive so the sum of the errors will get bigger, the inverse is true if position is always to the left. This can be monitored and used to further improve the accuracy of line following.

Integral:

Integral = Integral + Difference
Integral = Ki*(Integral)

Summarizing "PID" control-

Term	Expression	Effect
Proportional	Kp x error	It reduces a large part of the error based on present time error.
Integral	error dt	Reduces the final error in a system. Cumulative of a small error over time would help us further reduce the error.
Derivative	Kd x derror / dt	Counteracts the Kp and Ki terms when the output changes quickly.

Therefore, Control value used to adjust the robot's motion=

(Proportional) + (Integral) + (Derivative)

This code is used for line following

```
int lsa= A0;

int setpoint=35;
```

```
int positionx;
int error;
int ki;
int kd;
int lf=0;
int lb=0;
int rf=0;
int rb=0;
int maxspeed=200;
int maxspeedl = 100;
int junc=13;
int junction = 0;
int dir_lf= 7;
int dir_rf= 2;
int dir_lb= 8;
int dir_rb= 4;
int pwm_lf= 6;
int pwm_rf= 3;
int pwm_lb= 9;
int pwm_rb= 5;


void setup() {
  pinMode(lsa,INPUT);
  Serial.begin(9600);
  pinMode(junc,INPUT);
}

void loop() {
```

```
  positionx=analogRead(lsa);
  positionx = ((float)positionx / 921) * 70;

  pidcalc();
  pidcalcturn();
  move_motor(lf,lb,rf,rb,0,0);
}

void pidcalc(){
  error = p*16 + d*5 + i*0.06;
  Serial.println(error);
}

void pidcalcturn(){
  if (error<-100){
    error=-100;
  }
  else if (error>100){
    error=100;
  }
  if (error<0){
    rf=maxspeed-error/2;
    rb=maxspeed-error/2;
    lf=maxspeedl+error/2;
    lb=maxspeedl+error/2;
  }
  else{
    rf=maxspeed-error/2;
    rb=maxspeed-error/2;
```

```
    lf=maxspeedl+error/2;
    lb=maxspeedl+error/2;
  }
}

void move_motor(int lf,int lb,int rf,int rb,int dl,int dr){
  digitalWrite(dir_lf,dl);
  analogWrite(pwm_lf,lf); // Left Forward

  digitalWrite(dir_rf,dr);
  analogWrite(pwm_rf,rf); // Right Forward

  digitalWrite(dir_lb,dl);
  analogWrite(pwm_lb,lb); // Left Backward

  digitalWrite(dir_rb,dr);
  analogWrite(pwm_rb,rb); // Right Backward
}
```

Colour Detection via OpenCV:

```
import cv2
import numpy as np
def something(cap):
    while True:
        _,frame=cap.read()
        frame = cv2.flip(frame,2)
        hsv=cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        #bgr=cv2.cvtColor(frame, cv2.COLOR_HSV2BGR)

        low_r=np.array([0,150,150])
        high_r=np.array([40,255,255])
        mask_r=cv2.inRange(hsv,low_r,high_r)

        low_p=np.array([140,120,150])
        high_p=np.array([200,230,255])
        mask_p=cv2.inRange(hsv,low_p,high_p)

        final_r=cv2.bitwise_and(frame,frame,mask=mask_r)
        final_p=cv2.bitwise_and(frame,frame,mask=mask_p)

        f_gray_r=cv2.cvtColor(final_r,cv2.COLOR_BGR2GRAY)
        f_gray_p=cv2.cvtColor(final_p,cv2.COLOR_BGR2GRAY)


        gray_r = cv2.cvtColor(final_r,cv2.COLOR_BGR2GRAY)
        ret, add_r = cv2.threshold(gray_r, 29, 255, cv2.THRESH_BINARY)
```

```
                gray_p = cv2.cvtColor(final_p,cv2.COLOR_BGR2GRAY)
                ret, add_p = cv2.threshold(gray_p, 29, 255, cv2.THRESH_BINARY)

                average_color_r = np.average(add_r, axis=0)
                average_color_r = np.average(average_color_r)

                average_color_p = np.average(add_p, axis=0)
                average_color_p = np.average(average_color_p)

                print(average_color_r,average_color_p)

                cv2.imshow('pink',final_p)
                cv2.imshow('red',final_r)
                #cv2.imshow('f',frame)
                if cv2.waitKey(1)&0xFF==ord('a'):
                        return
cap=cv2.VideoCapture(1)
something(cap)
cap.release()
```

Difficulties faced during the whole operation :-

- Firstly, the ball had to be checked for HSV values
  The HSV values varied along with different light conditions.

  Hence, it becomes difficult to include all the ranges.
- The LSA that was used in line following also had to be calibrated every time with different surfaces which took an enormous amount of time.
  Moreover finding the values of P, I, D was the difficult task.
- The motors were of different RPM Values hence it became a huge difficulty in finding the constants of PID.
- The solenoids were not working correctly as they were a bit old. Due to this the mosfet was unable to turn the switch.