# Fastest Line Follower Bot

## Introduction:

Rugved Systems decided to participate in the Fastest Line Follower competition to be held on 27th July 2023 in Indira Gandhi National Stadium as part of the Technoxian Robotics Competitions. We have constructed a bot for the same.

## Components:

1. RPI 4(model B):
    a. Compared with Arduino, Raspberry Pi provides high computational power and an in-built environment for C/CPP and Python.
    b. Arduino comes with a controller whereas Raspberry Pi comes with a 1.6 GHz Processor in the 4B variant of Raspberry Pi.
    c. RPI also provides better integration with electronics parts.
2. 8 Array IR Sensor
    a. IR sensors were decided to use for line detection as a black line on a white background creates a clear contrast which can be detected by IR sensors easily.
    b. IR sensors are also cheap, lightweight and easy to handle. Using an eight array IR sensor, we aim to improve the precision of our bot. It reduces possibility of bot going off-track and increases speed control.
3. 12V Lipo Battery
    a. Lipo batteries are extremely compact, lightweight and generally have a higher number of possible cycles, making it prone to lose shelf life.
    b. Lipo batteries are the best motorhome batteries. They do not need to be fully charged to give maximum output voltage, which makes it preferable, instead of lead-acid batteries.
4. Motor Driven L298N
    a. This motor driver can be used to control the speed and direction of two DC motors.
    b. It is also cheap, easily available and is one of the easiest ways to control DC motors.
    c. It is also good at regulating voltage drop corrections.
5. DC Motors (controlled by encoders)
    a. DC motor encoders are used for speed control feedback in DC motors where an armature or rotor with wound wires rotates inside a magnetic field created by a stator.
    b. The DC motor encoder provides a mechanism to measure the speed of the rotor and provide closed-loop feedback to the drive for precise speed control.
6. RunCam-New Swift Camera
    a. The camera is cheap, lightweight and provides a 25-30 fps video quality. It is suitable for mid-range line detection by the bot.
    b. It can also be used to calculate angle of bends in the track if time permits.
    c. Overall speed can be increased when using it as an optimizer alongside the IR sensors.
7. Wheels: Wheels will greatly affect bot stability and speed. Chances of slipping need to be accounted. Testing is required, as such they are not decided yet.

# Fastest Line Follower Bot

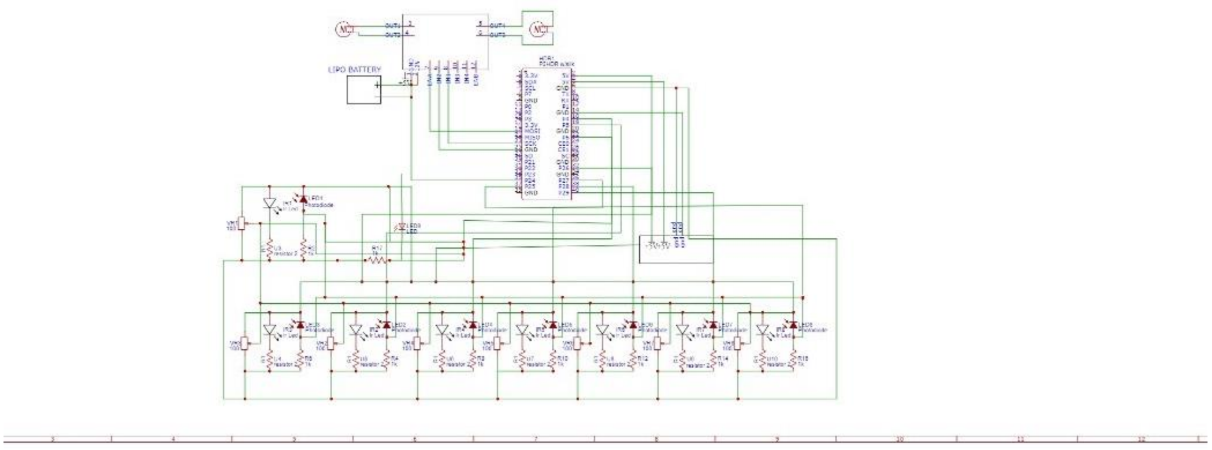
8. Bot Chasis
   a. The chasis is designed to focus on speed and provide decent stability. Having a frustum-like shape, we can position the sensors in the front for better reading.
   b. wheel position can be configured based on size and placing of components.

## Process of Making a chasis:

The body of the chassis has been 3d printed. Instead of going for conventional method, process of 3d printing has been chosen because of the following reasons:

a) Chances of failure in final product is greatly reduced.
b) during the process of prototyping major flaws of the design is already traced and thus helps to save time
c) being an additive manufacturing process, waste generated is minimal.
d)due to less waste generated, it helps to keep a check on cost of the overall build.
e) a great level of modularity can easily be achieved due to process of 3d printing
f) complex parts can easily be generated without any hassle helping to keep the parts intact, reducing the stress on joints, thus reducing the chances of failure
g) we have a huge range of variety of materials to choose from to achieve desirable property.

Below are the connections of the bot



**Choosing the material of build: ABS**

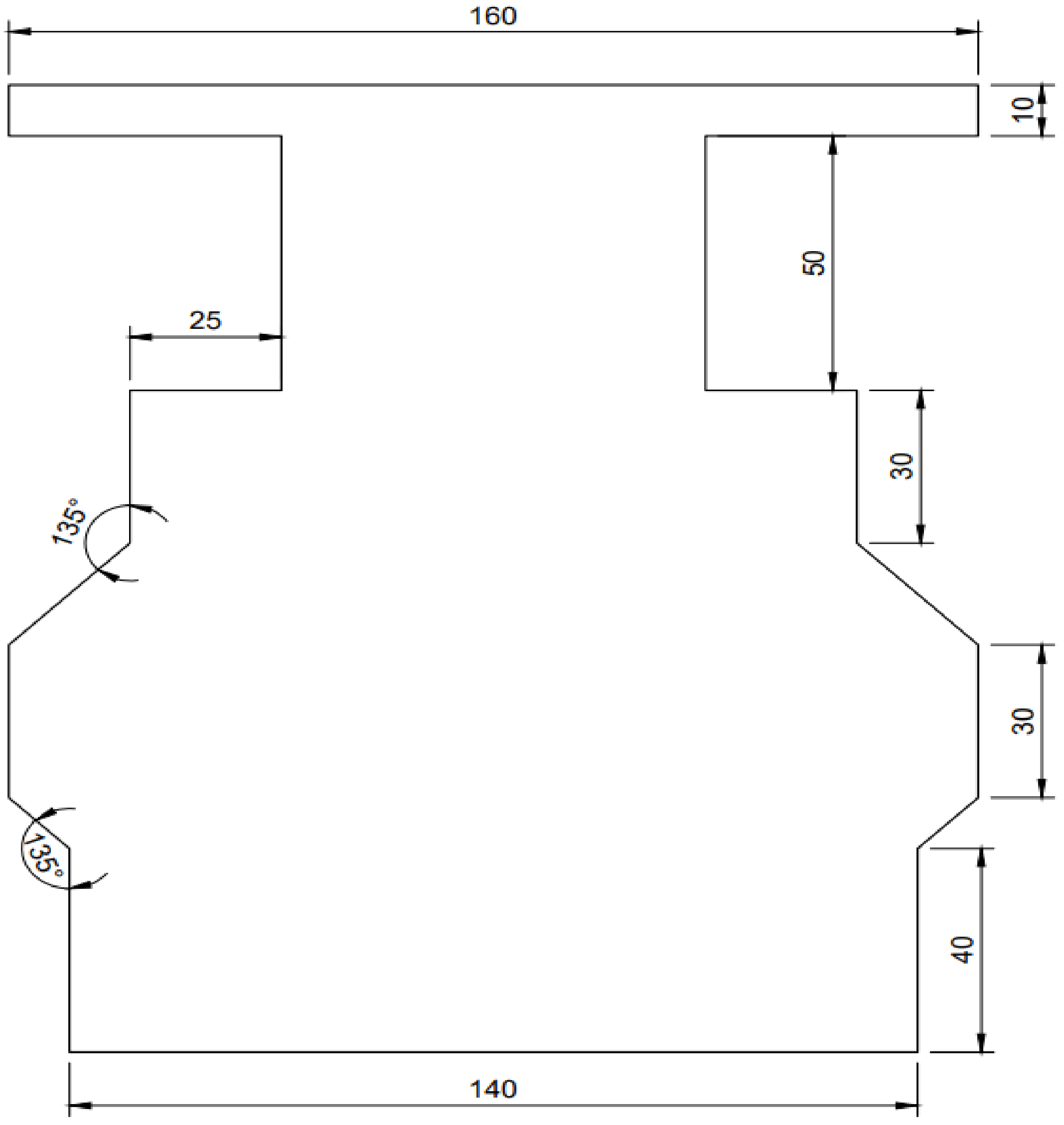
ABS stands for Acrylonitrile Butadiene Styrene. It is a common thermoplastic polymer that is widely used in various industries, including 3D printing. ABS offers a good balance of mechanical properties, durability, and affordability, making it a popular choice for many applications. Key features of ABS include:

1. Strength and Durability: ABS has excellent strength and impact resistance, making it suitable for parts that need to withstand mechanical stress or heavy use.

2. Heat Resistance: ABS has a relatively high melting point and can withstand moderately high temperatures without deforming or losing its structural integrity. This property makes ABS suitable for applications where exposure to heat is expected.
3. Chemical Resistance: ABS is resistant to many common chemicals, oils, and greases, making it suitable for parts that may encounter such substances.
4. Versatility: ABS can be easily molded, machined, and modified, allowing for flexibility in design and post-processing. It can be painted, sanded, glued, and welded, enabling further customization and finishing options.
5. Surface Finish: ABS typically has a smooth and glossy surface finish, which can be further enhanced through polishing or post-processing techniques.
6. Electrical Insulation: ABS is an electrical insulator, making it suitable for applications where electrical conductivity needs to be minimized or eliminated.

In the context of 3D printing, ABS filament is commonly used in Fused Deposition Modeling (FDM) printers. ABS filament is heated and extruded through a nozzle to create layers, which gradually build up to form a three-dimensional object. ABS has good adhesion between layers, allowing for strong and durable prints.

# Fastest Line Follower Bot



It is important to note that ABS emits potentially harmful fumes during the printing process. Adequate ventilation or the use of a dedicated enclosure is recommended to ensure safe operation and reduce exposure to these fumes.

## PID Algorithm:

PID stands for Proportion Internal Derivative. It is a widely used algorithm in line following completions and in low-level control systems. It is easy to implement, consumes less computational power and provides good accuracy for lines with good contrast with their background.
Its limitations include failure to detect line at sharp bends or consuming time in initial prediction due to quarter amplitude decay. PID also requires a high level of fine-tuning and parameters decided for a track may not be applicable in other tracks. Thus, PID cannot be used effectively in automated driving tasks.

Error(e): Error obtained between real value and theoretically desired value.

# Fastest Line Follower Bot

P_e(Proportional): This is the instantaneous error.
I_e(Integral): It is the integral of previous errors found.
D_e(Derivative): This represents the rate of change of error. It is sensitive to small changes.

Error= P_I*Kp + I_e*Ki + D_e*Kd
The K values represent constants that decide the weightage given to each part of the error. These values are decided based on testing.

Pseudocode for PID:

```
#include <SparkFun_TB6612.h>

//Define Motor 1 and Motor 2

int P, D, I, previousError, PIDvalue, error;
int lsp, rsp;
int lfspeed = 200;

float tKp = 0;
float tKd = 0;
float tKi = 0 ;

float sKp=0;
float sKd=0;
float sKi=0;

int maxs=0;

void linefollow(int s)
{
  int error = (analogRead(2) - analogRead(4));

  P = error;
  I = I + error;
  D = error - previousError;

  PIDvalue = (Kp * P) + (Ki * I) + (Kd * D);
  previousError = error;

  lsp = lfspeed - PIDvalue;
  rsp = lfspeed + PIDvalue;

  if (lsp > s) {
    lsp = 255;
  }
  if (lsp < 0) {
    lsp = 0;
  }
  if (rsp > s) {
    rsp = 255;
  }
  if (rsp < 0) {
    rsp = 0;
  }
  motor1.drive(lsp);
  motor2.drive(rsp);

}

float theta=0;
//get value of theta from openCV.
int s=0;
int minsp=0;
int l=0;//distance from next bend, get value from openCv.
void speedadjust(){
  error=l;
  P = error;
  I = I + error;
  D = error - previousError;

  PIDvalue = (Kp * P) + (Ki * I) + (Kd * D);
  previousError = error;

  s=s-PIDvalue;
}

void main(){
    int count=0;
    while(1){
        if(count%15==0)
            speedadjust();
        linefollow();
    }
}
```

Sources for PID:

1. https://medium.com/autonomous-robotics/pid-control-85596db59f35
2. https://towardinfinity.medium.com/pid-for-line-follower-11deb3a1a643

## Fuzzy Logic:

Fuzzy logic converts discrete values to a spectrum of values which can be used to account uncertainty in the model. Fuzzy logic is intuitive and easy to implement in Python. Using Fuzzy logic to calculate line length from the camera can help in controlling of speed and acceleration of the bot. Intersections can also be calculated. At most two lines are detected,

the primary line (one on which bot is travelling) and the second line to be followed at the intersection. The second line is decided based on a voting system.
Using fuzzy logic alongside PID can improve bot speed by a few seconds and prevent sudden stops during movement.

Sources for Fuzzy Logic:

1. https://www.mathworks.com/help/fuzzy/types-of-fuzzy-inference-systems.html
2. https://www.cse.unr.edu/~simon/fuzzy/fuzzylogic.htm

Pseudocode:

```
#include<SparkFun_TB6612.h>
#include<stdio.h>

//Define Motor 1 and Motor 2

int P, D, I, previousError, PIDvalue, error;
int lsp, rsp;
int lfspeed = 200;
// below values will be influenced by the fuzzy algorithm
float tKp = 0;
float tKd = 0;
float tKi = 0 ;

float sKp=0;
float sKd=0;
float sKi=0;

int maxs=0;

void linefollow(int s, fuzzy_output)
{
  int error = (analogRead(2) - analogRead(4));

  P = error;
  I = I + error;
  D = error - previousError;
  PIDvalue = (Kp * P) + (Ki * I) + (Kd * D);
  previousError = error;

  lsp = lfspeed - PIDvalue;
  rsp = lfspeed + PIDvalue;

  if (lsp > s) {
    lsp = 255;
  }
  if (lsp < 0) {
    lsp = 0;
  }
  if (rsp > s) {
    rsp = 255;
  }
  if (rsp < 0) {
    rsp = 0;
  }
  motor1.drive(lsp);
  motor2.drive(rsp);

}

float theta=0;
//get value of theta from openCV.
void fuzzy(int n)//n is num of iteration
{
   //max and min length of line
int min_l, max_l;// Values found from openCV
int n;//range used
int max_speed;//calculated from openCV
int x1,x2,x3,x4,y1,y2,y3,y4;//The two lines detected from camera
//Intersection defined.
int l_k=((x1y2 − y1x2) (y3 − y4))/(x1 −x2)(y3 −y4)−(y1 −y2)(x3 −x4)- ((y1 − y2) (x3y4 −
y3x4))/((x1 −x2)(y3 −y4)−(y1 −y2)(x3 −x4))
}
int b=(l_k*100)/max_l

// Converting the discrete values into a spectrum
int fuzzy_set[n];
//Defuzzifying the set.
int discrete_set[n];
//The discrete values will be used as input for constants used in PID
}


void main(){
   int count=0;
   while(1){
      if(count%15==0)
         speedadjust();
      linefollow();
      fuzzy()
   }
}
```