FINAL REPORT

PROJECT: *LINE FOLLOWING BOT USING OPENCV*

Name: Aditya Aryam

Roll No.: BTECH/10420/20

Sub-Team: Embedded

Before proceeding, I would like to extend my sincere gratitude to the ROBOLUTION CLUB for providing me the opportunity to work upon this project.

I would also like to mention Shivam Shandilya Bhaiya, K19, Robolution who had been assigned as my mentor. He had been a great support over the course of this project.

PREREQUISITES REQUIRED:

- Knowledge of Open CV (A python module)
- Basics of Arduino UNO
- Knowledge of Arduino IDE (Similar to C/C++)
- Knowledge of PID controller
- Knowledge of L298N motor driver

INTRODUCTION:

Line follower is an autonomous robot which follows a black line in a high contrast surrounding. The robot detects the line on its own and keeps following it. In this project, the detection of the path has been done through image processing using Open CV.

BASIC IDEA OF THE ROBOT:

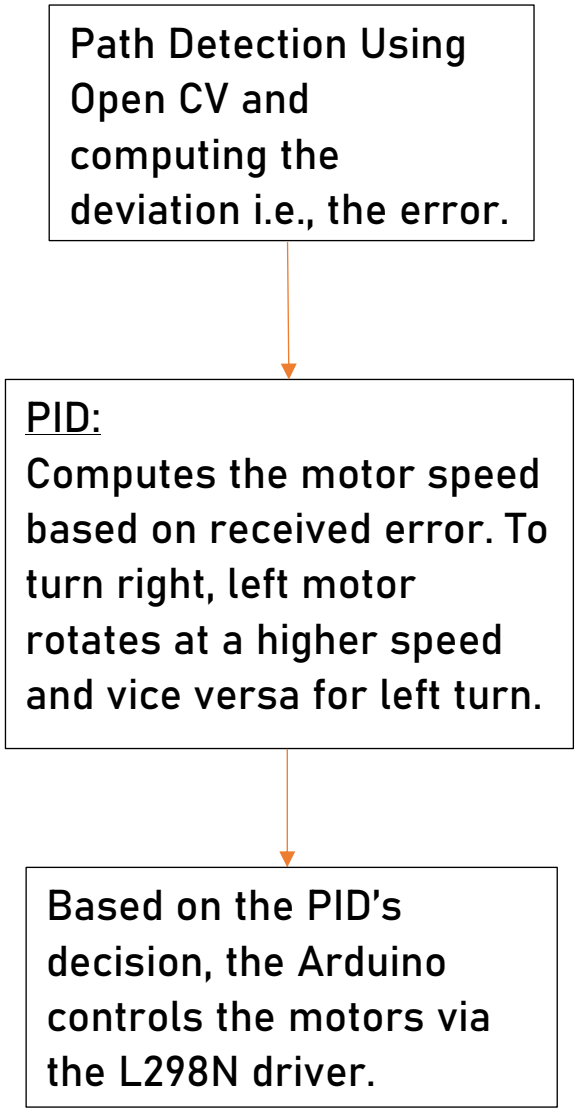


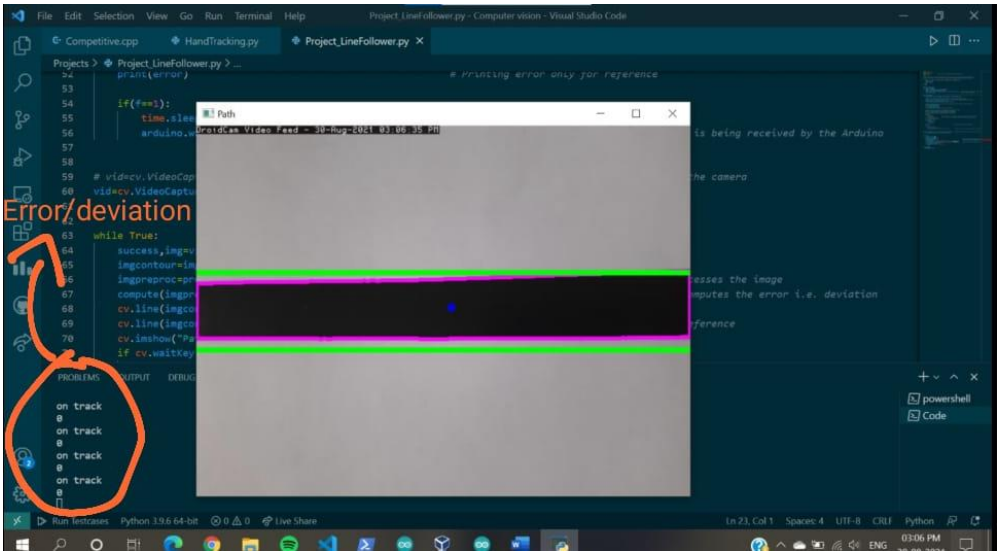
IMAGE PROCESSING:

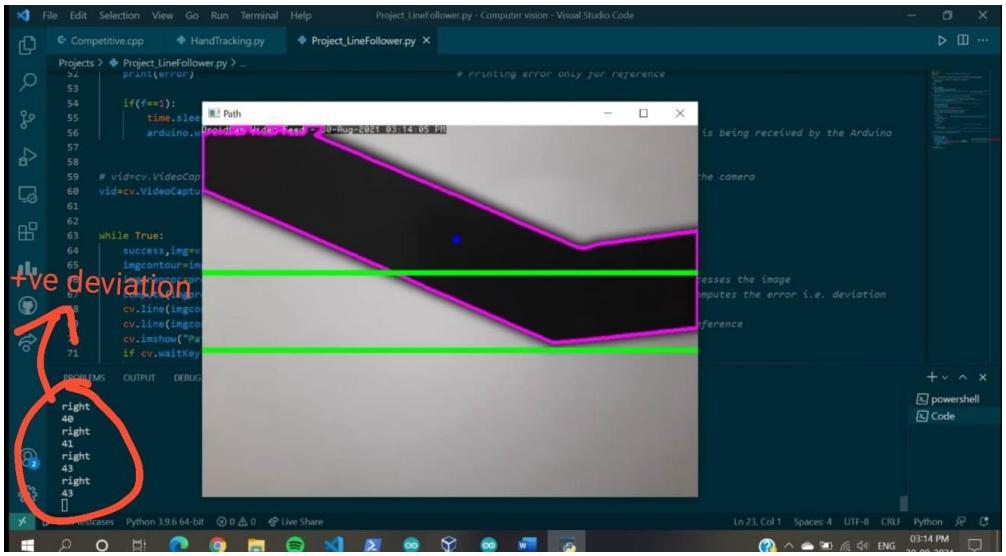
Why?

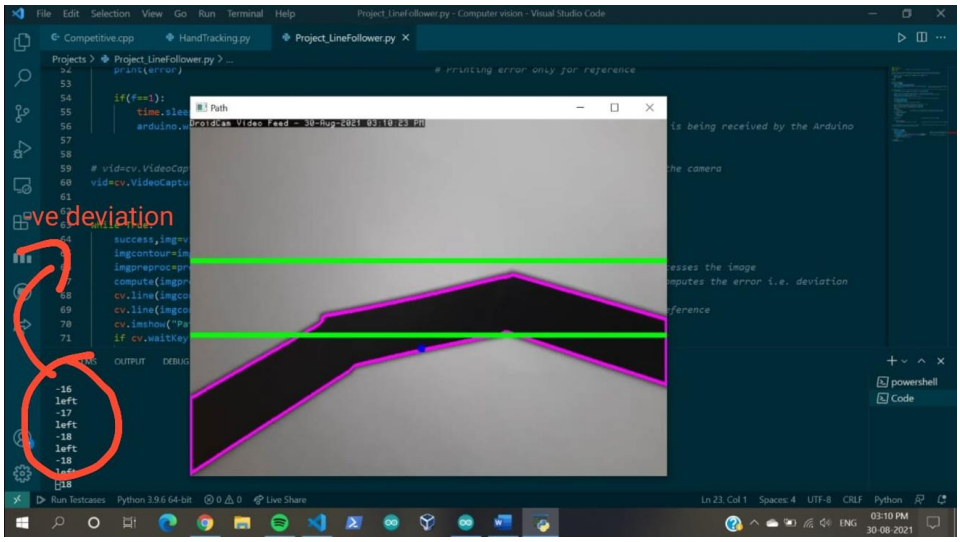
This is required to detect the path and calculate the deviation from the path.

Final result of Image Processing:

*(Bot moves from right side to left side of screen in following images)







How?

Firstly, we pre-process the image by Gray scaling, Gaussian Blurring, and Image Thresholding. These three processes help in better contour detection; thus, the edges of our path are recognized in a better manner.

```
def preprocess(image):
    imggray=cv.cvtColor(img,cv.COLOR_BGR2GRAY)
    imgblur=cv.GaussianBlur(imggray,(5,5),1)
    imgthreshed1=cv.threshold(imgblur,40,255,cv.THRESH_BINARY_INV)[1]
    return imgthreshed1
```

Now, we detect the contour of the threshold image. We also keep in mind that only our path is detected and no noise (some other black object in the camera's view). Therefore, only the contour of maximum area is taken into consideration.

```
contours,h=cv.findContours(img,cv.RETR_EXTERNAL,cv.CHAIN_APPROX_NONE)
biggestcnt=max(contours,key=cv.contourArea)
cv.drawContours(imgcontour,biggestcnt,-1,(255,0,255),3)
```

We also divide the screen into 3 parts by drawing 2 lines along the opposite edges. This helps in visualizing the boundary conditions for the edges.

```
cv.line(imgcontour, (0,190), (640,190), (0, 255, 0), thickness=5)
cv.line(imgcontour, (0,290), (640,290), (0, 255, 0), thickness=5)
```

Now, we find the centroid of our detected contour by the method of moments and draw a small indicating circle there.

```
mo=cv.moments(biggestcnt)
cx=int(mo["m10"]/mo["m00"])
cy=int(mo["m01"]/mo["m00"])
cv.circle(imgcontour,(cx,cy),5,(255,0,0),cv.FILLED)
```

The centroid and the indicating lines decide the extent of error. If the centroid is within the two lines, then the error would be 0 and would be the difference from the nearest line otherwise.

```
error=0
if(cy>290):
    print("left")
    error=290-cy
if( cy>=190  and cy<=290):
    print("on track")
if(cy<190):
    print("right")
    error=190-cy
```

The final step is to write this error on the Serial Port which would be read by the Arduino and would help in implementing the PID controller.

```
arduino.write(bytes(str(error), 'utf-8'))
```

Source Code of the Image Processing Section:

https://drive.google.com/drive/folders/1qLGhrKt9Eh_i1oAoV3tPtk13-FfZwi6m?usp=sharing

IMPLEMENTATION OF PID CONTROLLER IN ARDUINO:

To implement a PID controller in an Arduino Sketch, four parameters must be known, namely:

- Proportional Constant (Kp)
- Integral Constant (Ki)
- Derivative constant (Kd)
- Error

In this project, the error is being extracted from the Serial Port, and the rest three parameters are supposed to be tuned by trial and error.

The first part of the function should be determining the time lapsed.

```
current_time = millis();
elapsedTime = (current_time - timePrev) / 1000;
```

The integral of the error is the cumulative error over time. The derivative of the error is the rate of change of error

```
pid_p = error;
pid_i = pid_i + (error * elapsedTime);
pid_d = ((error - previous_error) / elapsedTime);
```

The computed output is

```
pid = (kp * pid_p) + (ki * pid_i) + (kd * pid_d);
```

Finally, time variables must be remembered for the next iteration.

```
previous_error = error;
timePrev = current_time;
```

Controlling Motor speed using the PID output

Initialize a base speed and maximum speed for both the motors, in short, the extremes.

```
const uint8_t maxspeed_A = 200;
const uint8_t maxspeed_B = 200;
const uint8_t basespeed_A = 130;
const uint8_t basespeed_B = 130;
```

Now, we simply add the output to the base speed and also saturate the overall speed over the base-speed and max-speed (if the extremities are crossed).

```
speed_A = basespeed_A + pid;
speed_B = basespeed_B - pid;

if (speed_A > maxspeed_A) {
  speed_A = maxspeed_A;
}
if (speed_B > maxspeed_B) {
  speed_B = maxspeed_B;
}

if (speed_A < 0) {
  speed_A = 0;
}
if (speed_B < 0) {
  speed_B = 0;
}
```

The following setups are essential:

```
void setup() {
  /* All the arduino pins used are outputs */
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(13, OUTPUT);

  current_time = millis();
  /* Begin the Serial Communication */
  Serial.begin(115200);
  Serial.setTimeout(1);
}
```

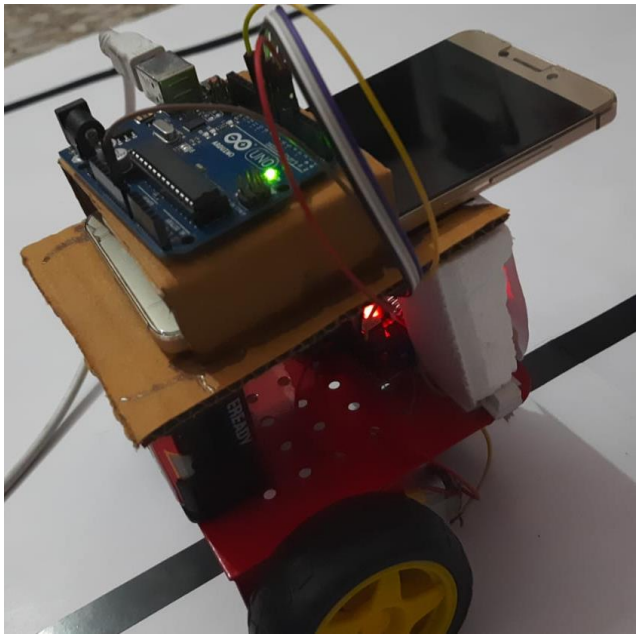
**It should be noted that the enable pins of the driver must be connected to the PWM pins of the Arduino UNO.

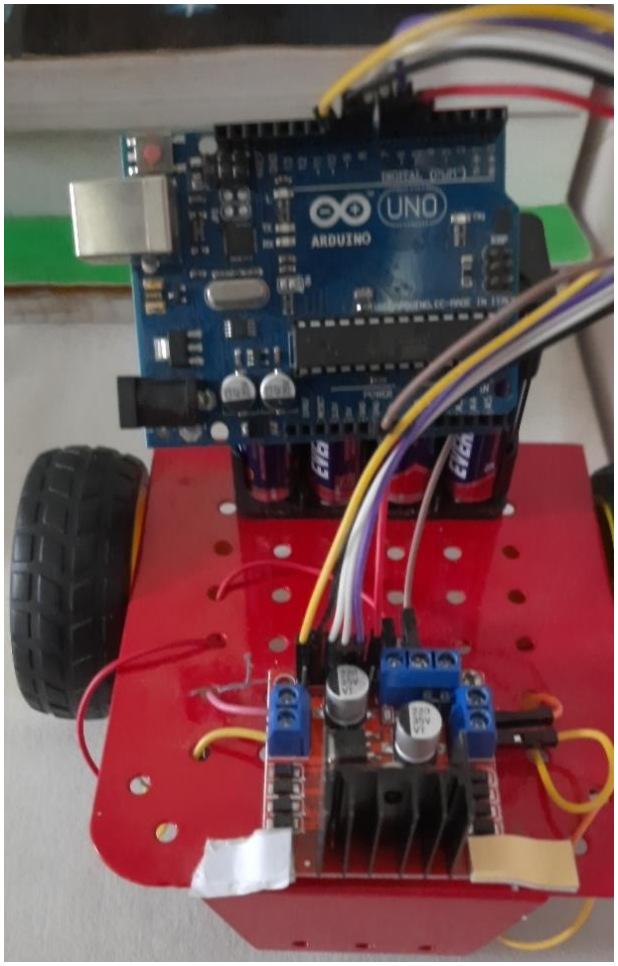
Source Code of the Arduino and PID Implementation:

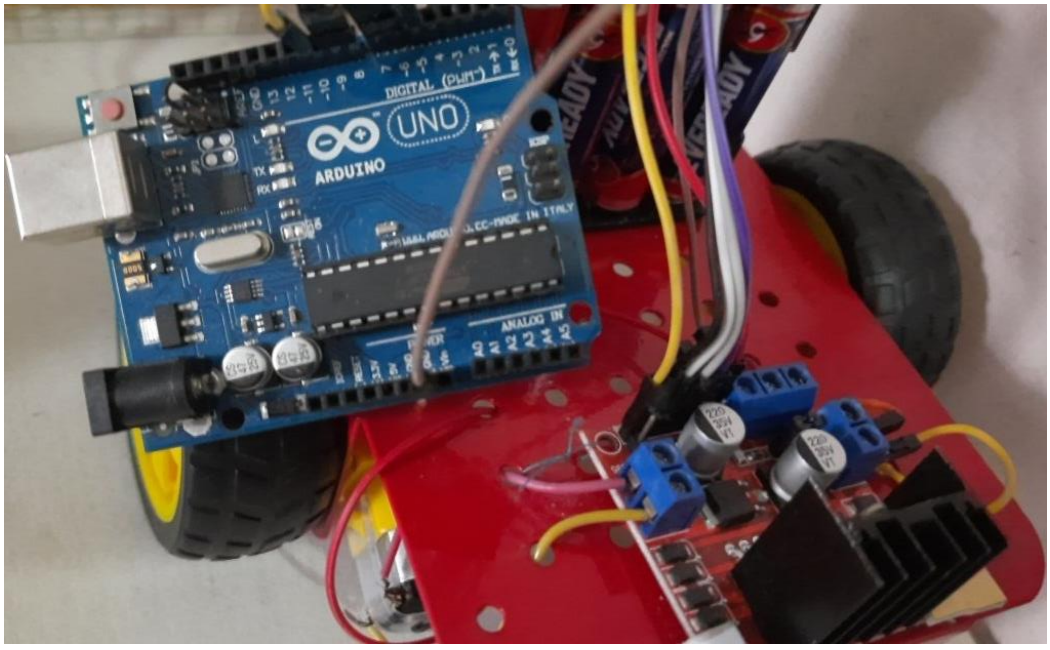
https://drive.google.com/drive/folders/1FOwcdnwmySA3lcMtbRCAeTNCfuQg4xBD?usp=sharing

STRUCTURES & CONNECTIONS:

**The L298N and the Arduino must be strictly connected to a common ground.







FINAL VIDEO OF THE LINE FOLLOWING BOT:

https://drive.google.com/drive/folders/1zob4_w_oZgTW6cU7Y99TCK5Uu427P_Nc?usp=sharing

Additional Project made during the learning process for the main project:

https://drive.google.com/drive/folders/1TLf2wJ1BRBSFi-Hfaq6U9xrm0IDTrbcJ?usp=sharing

TECHNICAL DIFFICULTIES FACED:

- Lighting Problem in image processing (resolved).
- PID tuning was a the most difficult task.
- Mounting the phone (used as a camera) was a challenge.
- The motors got burnt.
- Three sets of eight batteries have been (Two sets drained) used for powering L298N motor driver.

REFERENCES:

- https://www.geeksforgeeks.org/
- https://towardsdatascience.com/computer-vision-for-beginners-part-4-64a8d9856208
- https://stackoverflow.com/
- https://www.youtube.com/user/Mhproductionhouse
- https://www.youtube.com/channel/UC3x_n-h_n5eEaQvkc785fFQ
- https://www.youtube.com/channel/UC8butISFwT-Wl7EV0hUK0BQ
- https://docs.opencv.org/4.5.2/
- https://forum.arduino.cc/
- http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/
- https://create.arduino.cc/projecthub/ansh2919/serial-communication-between-python-and-arduino-e7cce0
- https://www.youtube.com/channel/UCfYfK0tzHZTpNFrc_NDKfTA