TASK – 1 REPORT CSE

Hardik Ahuja
Abhit Pahwa
Atishi Miral
Aryan

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.