Control Systems Lab
Experiment 2: Line Follower Bot

Soham Rahul Inamdar (210100149)
Satush Parikh (21D070062)
Rohit Rajwansi (18D070025)

September 2023

1 Introduction

Spark V is a low-cost robot designed for robotics hobbyists and enthusiasts based on ATMEGA16 microcontroller. The speed of the Spark V bot is controlled using PWM. Below its base, there are three IR sensors that give a reading in the range [0,255] depending on the reflectivity of the surface they are facing.

2 Objectives

The objectives of the experiment were as follows:

(a) Design and implement a **PID speed controller** for the Spark V robot to follow a continuous black track, using the IR sensors provided on the robot for this very purpose.

(b) To ensure that the bot traces the entire track within **30s**.

3 Procedure

1. We studied the provided software and hardware lab manuals to gain a sound understanding of the working of the bot.

2. We used the sample code provided to us to get an idea of the values displayed by the IR sensors and tune the potentiometers corresponding to them to obtain accurate and relevant values from all sensors. We noticed that the values provided by the sensors increase when they detect a black surface.

4 PID Control Algorithm

The time-domain equation of a PID controller is given by:

$$y(t) = K_p \cdot e(t) + K_i \cdot \int e(t)dt + K_d \cdot \frac{de(t)}{dt}$$

The error signal($e(t)$) we considered was the **difference between values of right and left sensors**($r - l$). The value of the PID signal was calculated using the following code snippet:

```
    pid_val=(kp*err)+(ki*int_err)+((kd*(err-prev_err))/dt);
    int_err+=err;
    prev_err=err;
```

We have used a time step value(dt) of 1 in the above computation. Now, in order to change the velocity of the motors driving the left and right wheels, we used the following function which was given in the lab manual:

```
void velocity(unsigned char left_motor,unsigned char right_motor){
    //To set the speed of both motors using pwm

    OCR1AL=left_motor;
    OCR1BL=right_motor;
}
```

After, designing this function we had to figure out the required conditions and appropriate inputs to the **velocity()** function which is as described in the following lines of code:

```
//turn right
if (pid_val>5){
    velocity(base_speed+pid_val,base_speed-pid_val);
    right();
}
//turn left
else if (pid_val<-5){
    velocity(base_speed+pid_val,base_speed-pid_val);
    left();
}
//go straight
else{
    forward();
}
```

So, in the above snippet depending on the turning direction we add or subtract the PID value from a base speed(taken as 125). Here the direction depends only on the PID value and not on whether the bot is at a turn or at the square intersection hence, we **did not need** any extra conditional block to deal with the intersection. ± 5 is taken as a threshold here. The bot goes straight if the PID value is between $[-5, 5]$ and turns accordingly otherwise.

5 Observations and Inference

- The proportional gain(K_p) plays a significant role in controlling the speed of the bot on a straight path. A larger value of K_p indicates a higher speed except at boundaries and turns where K_d will play an instrumental role.
- The role of K_d is to ensure smooth turns and to ensure that the bot does not go off the track.

6 Results

We have used the following values of proportional gain(K_p), integral gain(K_i) and derivative gain(K_d) for our PID controller:

K_p	2.00
K_i	0.00
K_d	2.00

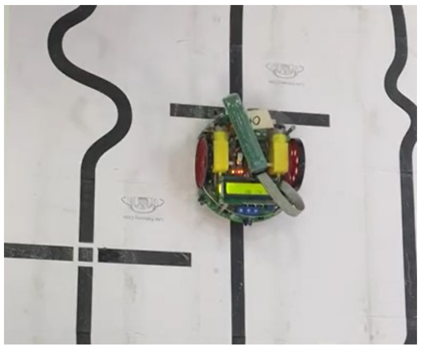
Table 1: PID parameters

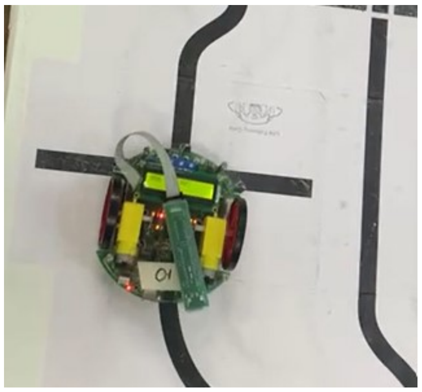
We were able to complete the whole track in **24.11 seconds**.

7 Challenges Faced

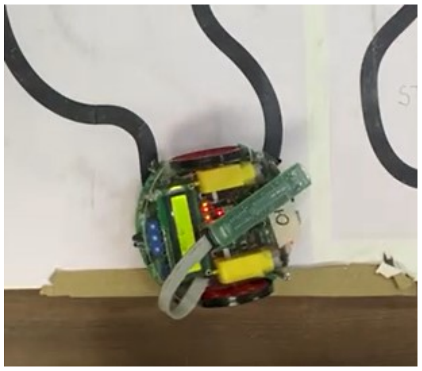
- Initially, we controlled the line follower using a conditional process(if-else statements). However, this resulted in sudden and jerky turns which is not feasible in a practical environment.
- Initially we were using a high value of K_i and not resetting the integral error after every turn which resulted in unnecessary accumulation of the integral error and our bot was thus performing rotational motion. We fixed this by making the value of K_i insignificant with respect to values of K_p and K_d.

8 Images


(a) Bot on straight path


(b) Bot at the intersection

(a) Bot on soft turn


(b) Bot on sharp turn