EE 324: Experiment 3 Lab Report

Sravan K Suresh	22B3936
Swarup Dasharath Patil	22B3953
Amol Milind Pagare	22B3971

October 10, 2024
(Thursday Batch)

Experiment 3: Line Follower Bot

Contents

1 Objective **2**

2 Control Algorithm **2**

3 Challenges Encountered **4**

4 Results **4**

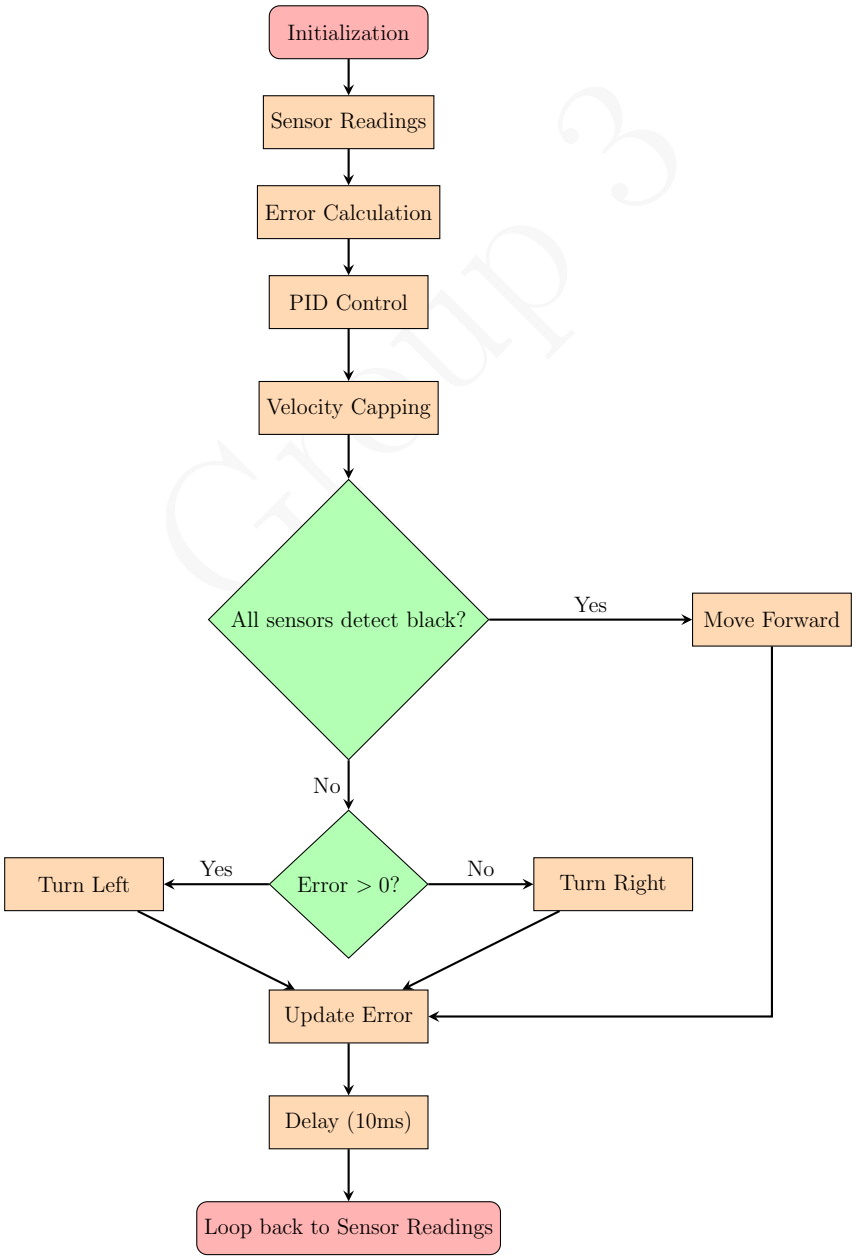
5 Observations and Inferences **4**

1 Objective

Design and implement a PID controller for the `Spark V` Robot, enabling it to follow a continuous track using infrared (IR) sensors within a time frame of 30 seconds.

2 Control Algorithm

Our algorithm implements a basic Proportional-Derivative (PD) controller to manage the robot's movement based on the readings from the three sensors (left, center, and right). The PD control computes the error based on the difference between the left and right sensor readings, adjusting the robot's velocity to ensure it follows a line. To avoid overshooting due to inertia, a limit on velocity is imposed. The robot moves forward when the center sensor detects a black strip, and turns left or right depending on the error.



The C code implementing this algorithm is given below:

```
unsigned char ADC_Conversion(unsigned char);
unsigned char l = 0, r = 0;

int error = 0, sum = 0;
float vel = 0;

int Kp = 2;
float Kd = 0.5;
float prev_error = 0;

// Main Function
int main(void) {
    init_devices();
    timer1_init();
    lcd_set_4bit();
    lcd_init();

    while (1) {
        l = ADC_Conversion(3);
        c = ADC_Conversion(4);
        r = ADC_Conversion(5);
        lcd_print(1, 1, l, 3);
        lcd_print(1, 5, c, 3);
        lcd_print(1, 9, r, 3);
        error = l - r;
        sum += error;

        vel = abs(Kp * error + Kd * (error - prev_error) / 10);
        lcd_print(2, 1, vel, 3);

        // To reduce the overshooting problem due to Inertia of Robot
        if (vel > 100)
            vel = 100;
        // Condition to move forward when all the sensors face black strip
        if ((l > 9) && (c > 9) && (r > 9))
            velocity(70, 70);

        if (abs(error) < 3) {
            // This decides the forward velocity
            if (c > 9) {
                forward();
                velocity(70, 70);
            }
        } else if (error > 0) {
            left();
            velocity(30, vel > 30 ? vel : 30);
        } else {
            right();
            velocity(vel > 30 ? vel : 30, 30);
        }
        prev_error = error;
        _delay_ms(10);
    }
}
```

The sensors return values between 0 and 255, which represent the brightness intensity detected by each sensor. A higher value indicates detection of a black surface, whereas a lower value represents white or lighter surfaces. Based on the error computed from sensor values, the robot adjusts its direction to maintain the desired course.

3 Challenges Encountered

Initially, we encountered difficulties in setting the appropriate threshold values for the left and right sensors to trigger a turn. Through multiple iterations of trial and error, we eventually found a suitable threshold.
In the first design, we implemented a proportional controller such that the speed of the robot was proportional to the center sensor's value. However, we observed that the center sensor value dropped significantly during turns, causing the robot to stop at those points. To address this, we introduced a minimum threshold speed, ensuring that the robot would maintain a baseline speed even when the controller suggested a lower value. This successfully resolved the issue.
Subsequently, we faced another challenge at sharp turns on the track. The sensor values for the left and right sensors did not change rapidly enough for the robot to detect and respond to the turn. This was compounded by the robot's speed being too high for effective detection. To fix this, we established a threshold speed for turns, ensuring the robot slowed down sufficiently to detect and take the turn appropriately.

4 Results

The control coefficients used by us are:

$$K_p = 2, \quad K_d = 0.5, \quad K_i \approx 0$$

The traversal time averaged across 4 runs was 27.5 seconds, which is within the time constraint of 30 seconds.
Threshold value for the left and right sensor = 7

5 Observations and Inferences

The video of our bot can be accessed via this link: Video Link
We followed a step-by-step approach, initially designing a proportional controller. While it functioned correctly, the bot exhibited shaky turns. To mitigate this, we introduced a differential controller, which helped smooth out the shakiness during turns. The TA pointed out that our bot was using PID only for speed control not for direction control. We controlled the direction using the left and right sensor values