Experiment 2
Black Line Follower Bot

Group Number 3 – Monday Batch

OV Shashank 14D070021 : Ayush Agrawal 14D070042 : Pratik Brahma 14D070003

AIM

Design and implement a PID controller for the Spark V robot to make it follow continuous black lines on the ground using the sensors provided on the robot for this purpose.

ACCENTUATED PARTS OF THE BOT FOR THE EXPERIMENT

Three white line sensors

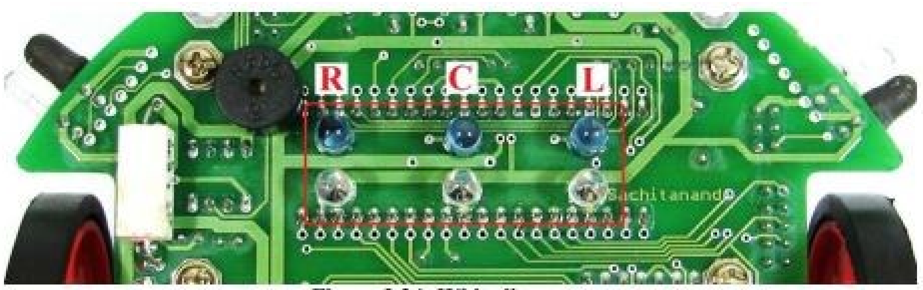


Figure 3.24: White line sensor

The white line sensors are used to extract the information of the orientation of the robot with respect to the black line. The main aim of the program is to orient the bot such that desired values of all the white line sensors are obtained which implies that the bot is following the black line with the centre sensor directly above the black line. The sensors give low value for indication of white colour and vice versa. More information will be given further below.

White line sensor potentiometers

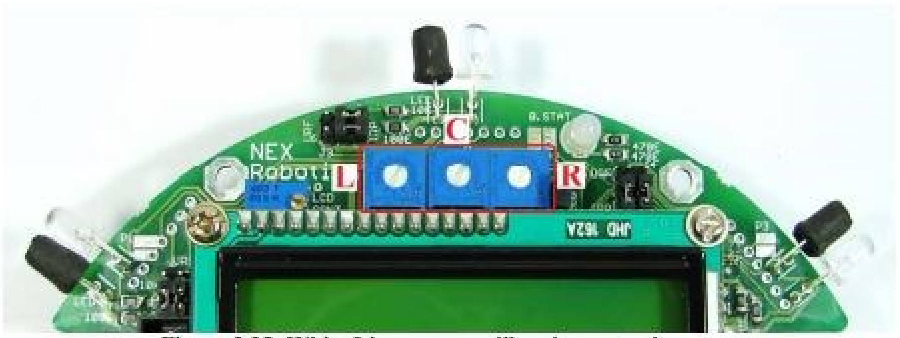


Figure 3.25: White Line sensor calibration potentiometers

The three potentiometers help to calibrate the corresponding white line sensors. This has to be done before programming the bot to follow the black line, so that appropriate threshold levels are coded in the program for the robot to differentiate between the black line and the white space.

Motion control

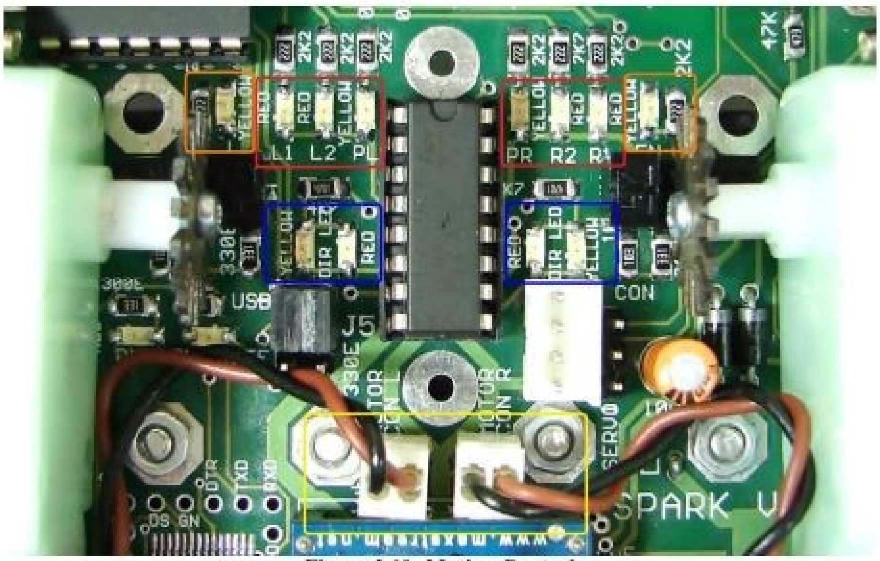


Figure 3.10: Motion Control

The motion control involves direction control and velocity control. The motors are being controlled by L293D dual motor driver. Direction control is being done using appropriate digital logics to the control pins of L293D. Velocity control is done using Pulse Width Modulation. In the program the PID algorithm is implemented to control the velocity of the motors for smooth transition between different curves of the black line. The logic levels for controlling the direction and velocity.

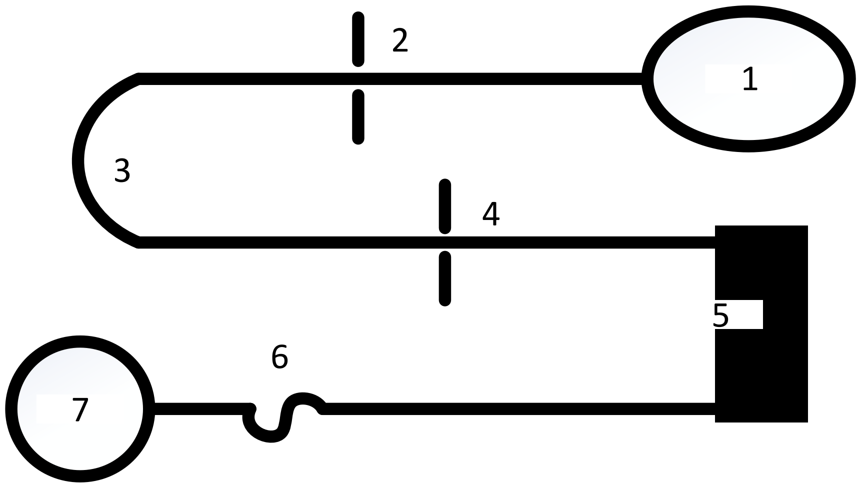
DIRECTION	LEFT BWD (LB) PB0	LEFT FWD(LF) PB1	RIGHT FWD(RF) PB2	RIGHT BWD(RB) PB3	PWM PD5 for left motor PD4 for right motor
FORWARD	0	1	1	0	As per velocity requirement
REVERSE	1	0	0	1	As per velocity requirement
RIGHT *(Left wheel forward, Right wheel backward)*	0	1	0	1	As per velocity requirement
LEFT*(Left wheel backward, Right wheel forward,)*	1	0	1	0	As per velocity requirement
SOFT RIGHT*(Left wheel forward,, Right wheel stop)*	0	1	0	0	As per velocity requirement
SOFT LEFT*(Left wheel stop, Right wheel forward,)*	0	0	1	0	As per velocity requirement
SOFT RIGHT 2 *(Left wheel stop, Right wheel backward)*	0	0	0	1	As per velocity requirement
SOFT LEFT 2 *(Left wheel backward, Right wheel stop)*	1	0	0	0	As per velocity requirement
HARD STOP	0	0	0	0	As per velocity requirement
SOFT STOP (Free running stop)	X	X	X	X	0

Table 3.4: Logic levels for setting direction and velocity

LCD interfacing

The LCD screen helps during debugging to read the values of sensors in real time and according modify the program to follow the black line. It is also used during the calibration of the white line sensors to set appropriate thresholds in the program.

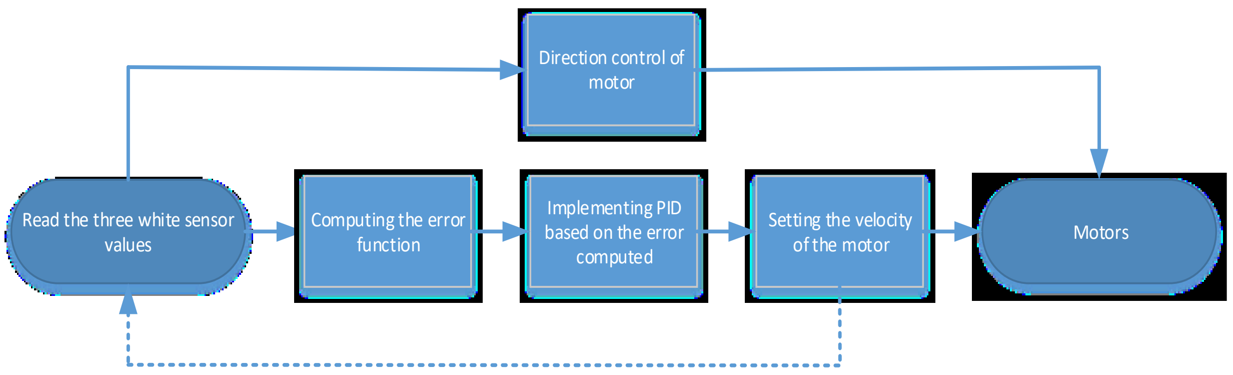
ROUGH SKETCH OF THE BLACK LINE TO BE FOLLOWED



The above path must be followed by the robot along the black line with the initial position from either circular path (1 or 7) to the other circular path at the end.

PROGRAM FLOW

The program flow can be understood by the below flow chart.



Direction control of motor
Read the three white sensor values
Computing the error function
Implementing PID based on the error computed
Setting the velocity of the motor
Motors

Computing error function

The error is calculated by finding out the deviation of the centre white line sensor from the maximum value attained by it (when the bot is moving on the black line with the centre white line sensor directly above it). This ensures that the bot is always moving aligned centre to the black line.

Implementing PID on the error function

The concept of the PID implemented is same as the previous experiment. The integral and derivate of error is computed and multiplied with appropriate constants and added up.

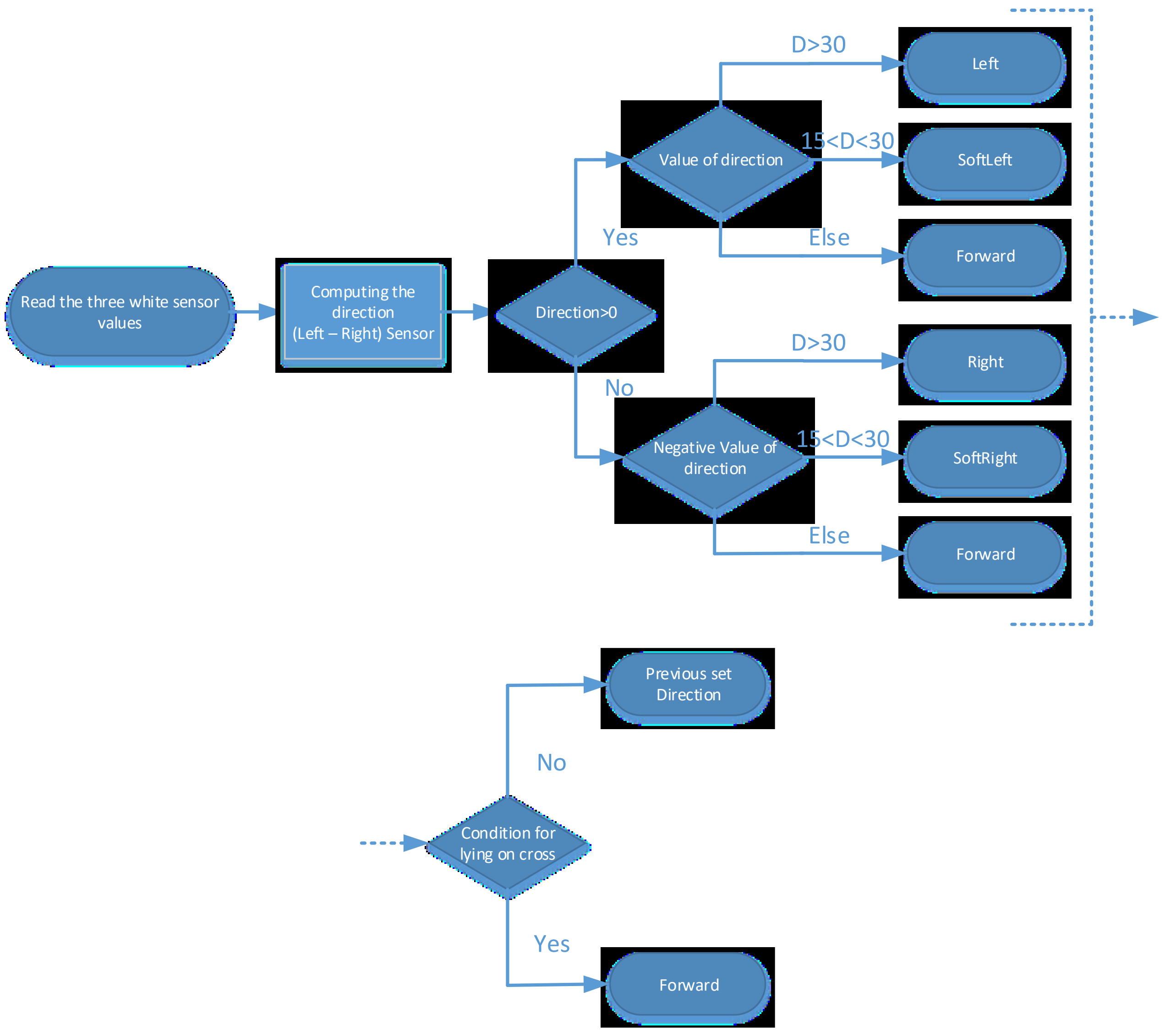
$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt},$$

1. **Proportional Error -** Accounts for the present value of the error. Higher current error, higher will be the proportionality error.
2. **Integral Error –** Accounts for higher precision of the error. Accumulates the error until the goal is not reached.
3. **Differential Error –** Takes care of any sudden change in error. Ensures smooth transition.

The resulting PID function is used to set the velocity of the motor. It is essential subtracted from the max value of the velocity attained by the bot.

Direction control of motor

The flow control is given for the direction control of motor.



PROBLEMS FACED IN THE EXPERIMENT

1. The bot would always turn to the left or right when the cross encountered. This was due to the non- centred alignment of the bot along the black line. This was solved by adjusting the threshold values of the centre, left and right sensor which introduced some tolerance in detecting the cross. Even if the bot is a little disoriented form the centre of the black line while encountering the cross, the threshold values were set such that it is considered to be a cross and the bot is programmed to move forward.
2. Similar problem was attained when the circular path end was reached. In this case if the bot is directly aligned to the centre line, then sometimes it goes straight without following the circular path. This was solved by similar method as above and adjusting the PID parameters.