PROJECT FINAL REPORT

Electrical and
Computer Engineering

ECE 556 - MECHATRONICS

Submitted by:

Manasi Kulkarni (Student ID:200483153)

Shitikantha Bagh (Student ID: 200482962)

Amarnath Shinde (Student ID: 200481609)

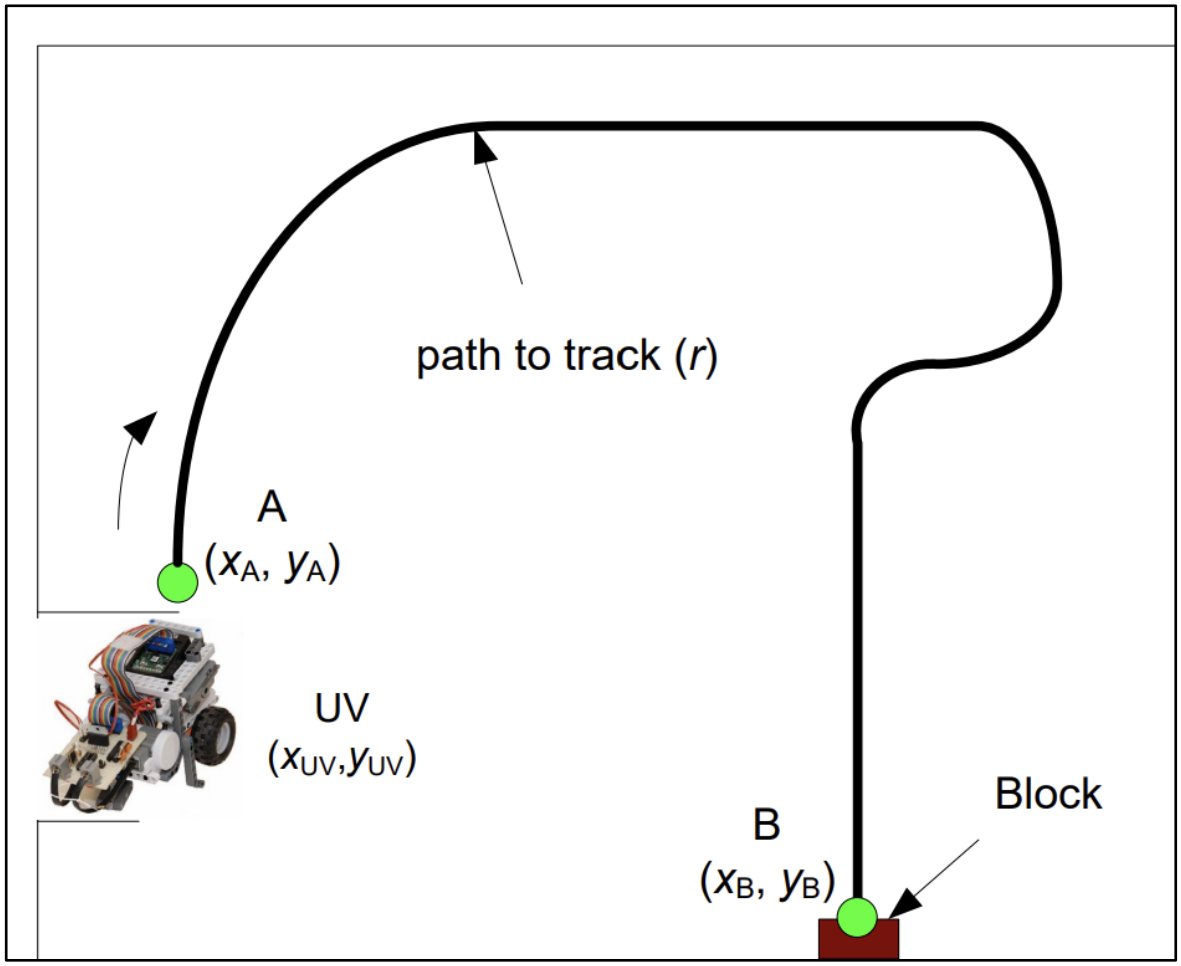
CONTENTS

1. INTRODUCTION:

The objective of this project is to use the sensors, actuators, and control knowledge learned in the Mechatronics course to perform the following tasks:

- **Obstacle Detection and Collision Avoidance:** Park as close as possible to the block (Point B) without touching it.
- **Platooning:** Implement a controller to regulate the UV speed to follow a moving UV in the front,maintain a constant distance and stop at a destined location.
- **Line following with Obstacle Detection:** Implement a PID control for the UV to move from point A with coordinate (xA, yA) to Point B (xB, yB) as fast as possible by tracking a given path r along with obstacle detection.

Fig. 1: Simple track layout for Robot Demonstration



2. TASK DESCRIPTION:

- **Obstacle Detection and Collision Avoidance:**
    a. In this task an object was placed at some distance.
    b. Robot needs to stop as close as possible to the obstacle and avoid the collision.
    c. If the robot collides with the obstacle then that run is counted as null.
    d. In this task we used an Ultrasonic sensor to detect the obstacle.
    e. Since it is like parking a car, it is also called Parking.

- **Platooning:**
    a. In this task there are two goals to be achieved.
    b. There is a TA robot which is placed at a particular distance , say 20 cm from our robot.
    c. Our task is to follow that robot which moves in a straight line.
    d. As our bot is following the TA robot , it should also maintain that distance of 20 cm throughout.
    e. This process is known as Platooning.
    f. For the second task , the robot had to stop at the red mark which was at the end of our path.
    g. For this part we used one of the color sensors which was mounted right at the bottom of the robot.
    h. Separate grading was given to platooning and for stopping at the red mark.

- **Line following with Obstacle Detection:**
    a. This was the last task and in this we have to follow a black path using the color sensors.
    b. In all, we were given three color sensors and it was up to our logic that how many will be used in line following.
    c. We used two color sensors which were at the edges of our Zero PCB.
    d. Line following was done on a curved path and a PID controller was implemented to control the movement of the bot.
    e. At the end of line following the bot needs to detect the obstacle and stop as close as possible for which ultrasonic sensor was used.

.

3. PROJECT SETUP:

The model makes use of the following setup:

1. Lego EV3 Brick (Microcontroller) with 2050 mAh DC battery.

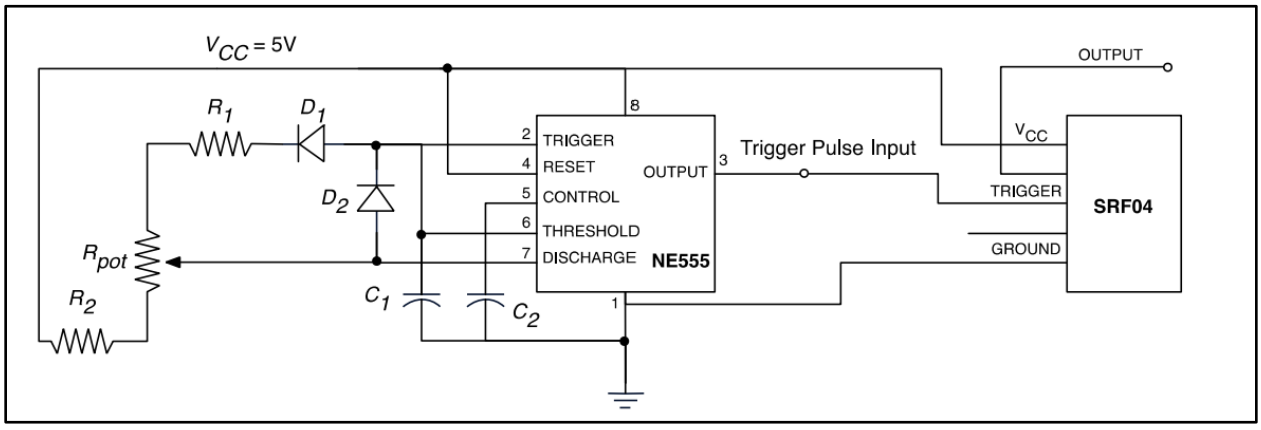
2. Lego EV3 large motors:

Max RPM	160-170
Running Torque	20 N cm
Stall Torque	40 N cm

The project was mainly based on developing our own Ultrasonic sensors and Light sensors and integrating these sensor circuits with EV3 software, to develop our Robot. The details of 2 circuits built are as follows:

3. Ultrasonic Sensor :

Fig 2: Ultrasonic sensor circuit



The Ultrasonic sensor used in the project was SRF04, for the purpose of Obstacle detection.The astable multi-vibrator circuit output (as shown in the above diagram),was used to trigger the ultrasonic sensor. The echo pulses obtained from the ultrasonic sensor were fed to the EV3 brick.

Dimension	45-20-15 mm
Working Voltage (DC)	5 V
Range	3-400 cm
Measuring Angle	15 degrees

In astable mode, the 555 timer acts as an oscillator that generates a square wave. The frequency of the wave can be adjusted by changing the values of two resistors and a capacitor connected to the chip. The formulas below helps us understand the length of the output's on and off cycles with different resistors and capacitors:

Fig 3: Formulas for Astable Multivibrator circuit

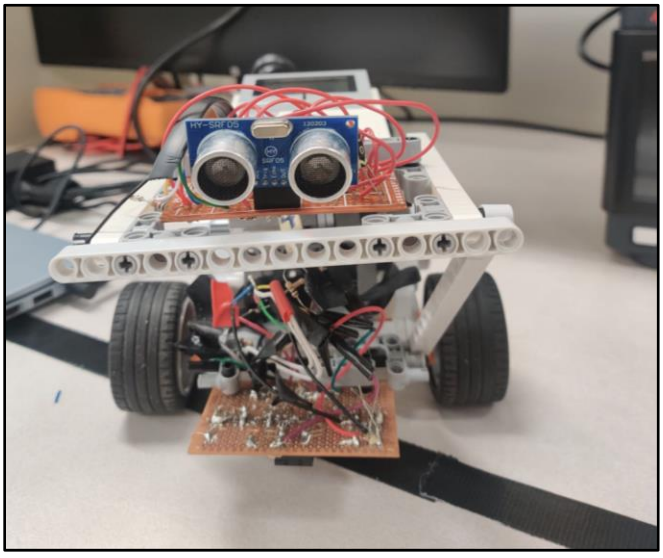
$$t_{on} = 0.69 \times C1 \times (R1 + R2)$$
$$t_{off} = 0.69 \times C1 \times R2$$

t_{on} : *Length of high output pulse in seconds*
t_{off} : *Length of low output pulse in seconds*
$R1$: *Resistance of* $R1$ *in Ohms*
$R2$: *Resistance of* $R2$ *in Ohms*
$C1$: *Capacitance of* $C1$ *in Farads*

We have used R2 as a 10K Ohm potentiometer, as it makes it easy to adjust the duty cycle according to our needs. Additionally, the ultrasonic sensor is connected to a low-pass filter (LPF) so that high frequency noise from the output signal can be removed. The Resistor and Capacitor values used were 10 kilo-ohm and 5µF. The cutoff frequency for the low pass filter was 5 Hz.

The placement of ultrasonic sensor circuit, for the purpose of obstacle detection on the robot was done as below:

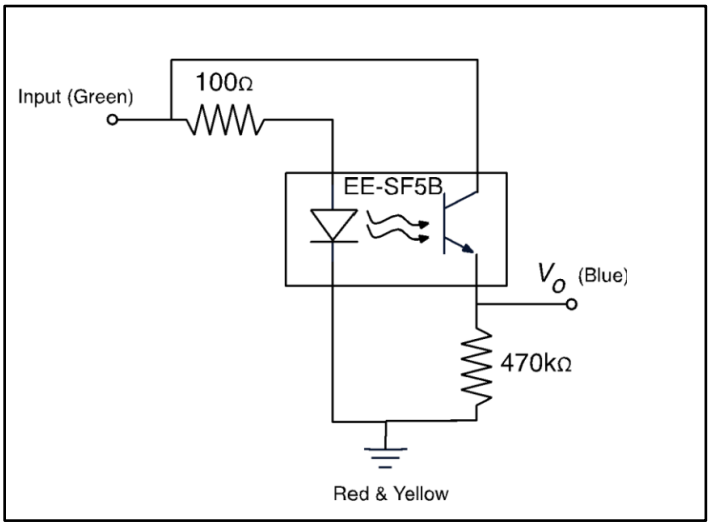
Fig 4: Placement of Ultrasonic sensor



4. Light sensor:

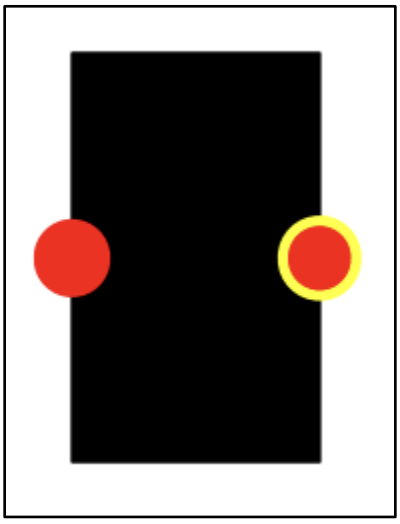
For the light sensor circuit an OMRON EE-SF5B sensor was utilized. Below figure. shows the circuit implementation.

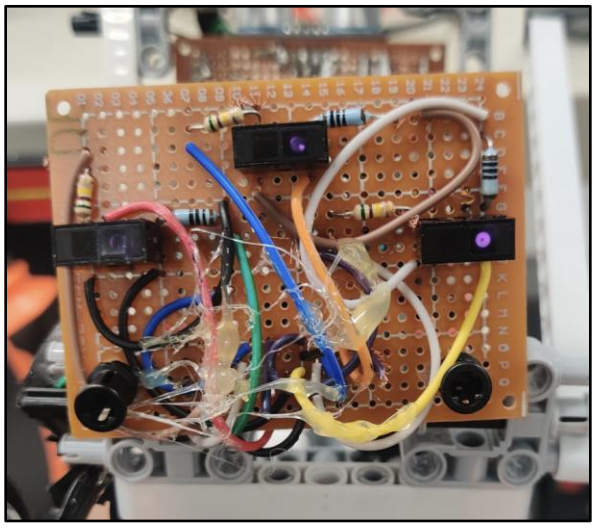
Fig 5 : Light sensor circuit



The light sensor readings are used to develop the path tracking/ line following algorithm. It helps the bot to be aligned in the desired black path on a given track.
The arrangement of light sensors was done to cover the edges of the black path, that would make it easy for the bot to navigate and correct itself when it moves towards the white path of the track. The arrangement of sensors on the bot is shown in the below figures:

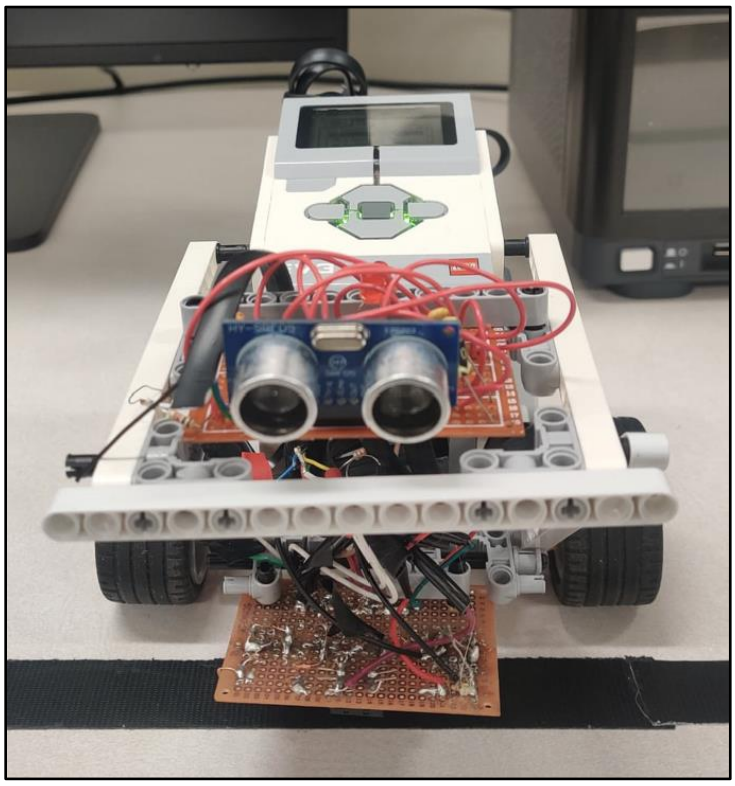
Fig 6: Placement of Light sensors





Final Bot construction:

Fig 7: Final Bot construction



4. ALGORITHM DESIGN AND DEVELOPMENT:

The goal of the project was to achieve 3 major tasks using our own-built UGV namely:

1. **Obstacle Detection and Collision Avoidance**
2. **Platooning**
3. **Path Tracking**

Each task had a unique problem statement to be addressed. Therefore, it required designing different algorithm strategies to complete the task. Additionally, each task demanded the data acquisition and processing from the sensors used for the task. In this section, we discuss in detail about the algorithm design and finally its implementation.

Software used:

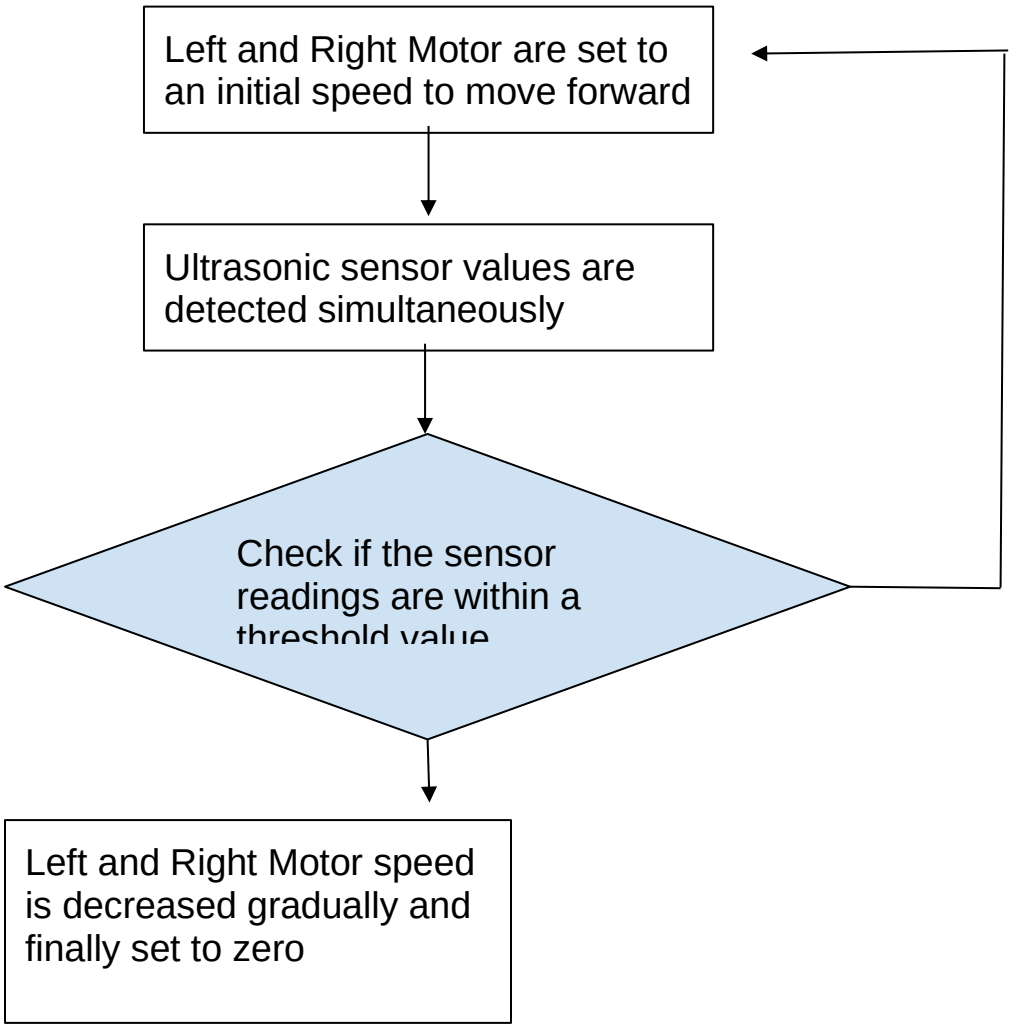
- LEGO EV3 MINDSTORMS

Alternatives:

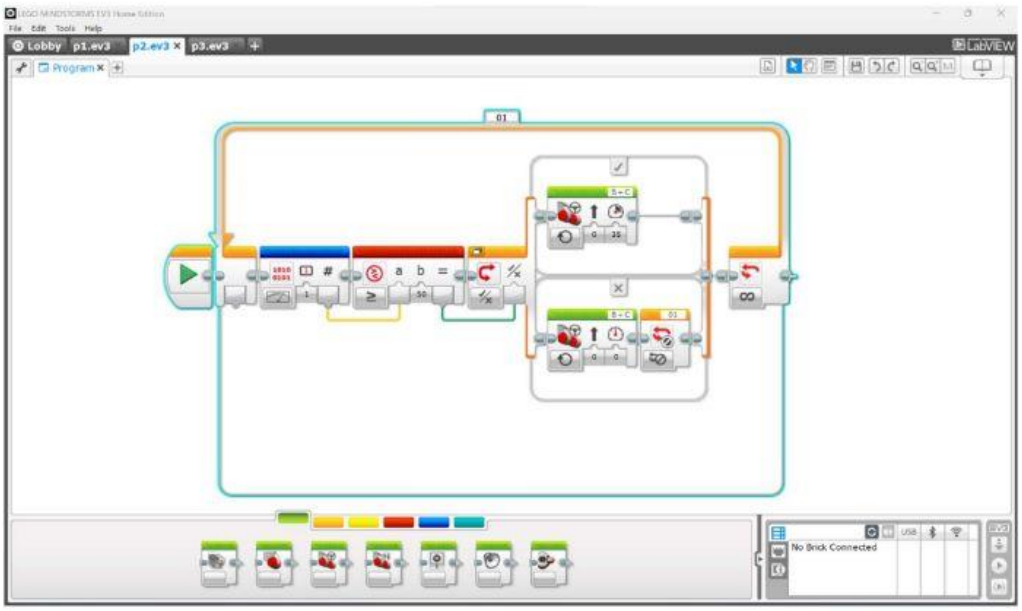
- MATLAB
- Simulink

Task 1 : Obstacle Detection and Collision Avoidance

Pseudo Code:

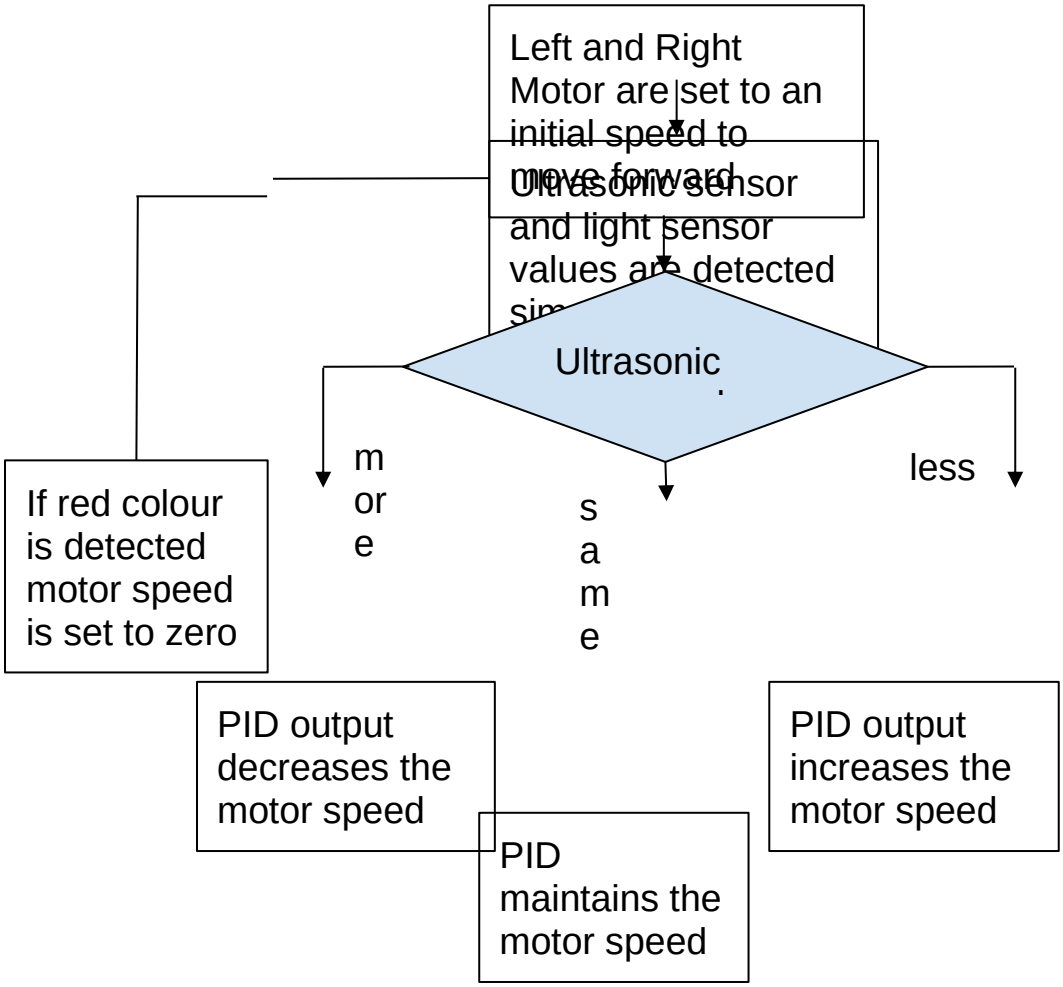


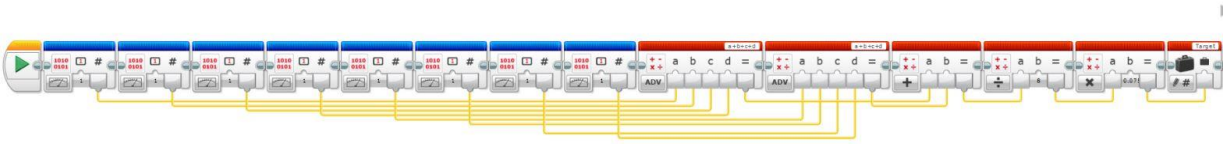
We conducted several experiments to detect the range of ultrasonic values for an obstacle near 3 cm and finally set a range so that whenever the values come within this, the bot stops moving and finally comes to a halt.



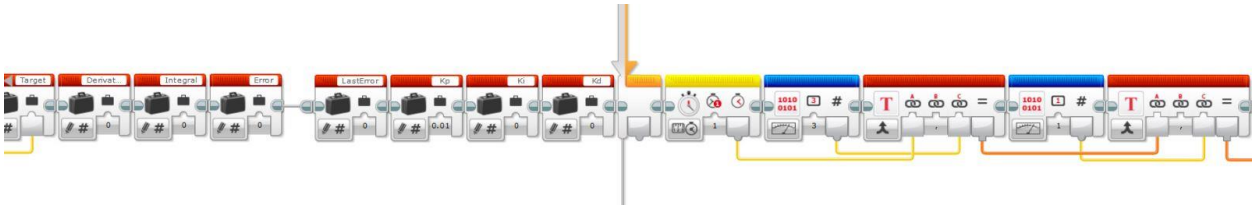
- The ultrasonic sensor output was connected to Port 1 of the EV3.
- The raw sensor value block (blue) was used to detect the sensor readings.
- This block's output was compared with a fixed value using the compare block (red).
- Finally the output of this block was given to the Switch Case block (orange), if its true the Move Steering Block (green) speed is set to an initial speed . If it's false, the speed is set to zero and a Loop Interrupt Block (red) terminates the loop.

Task 2: Platooning



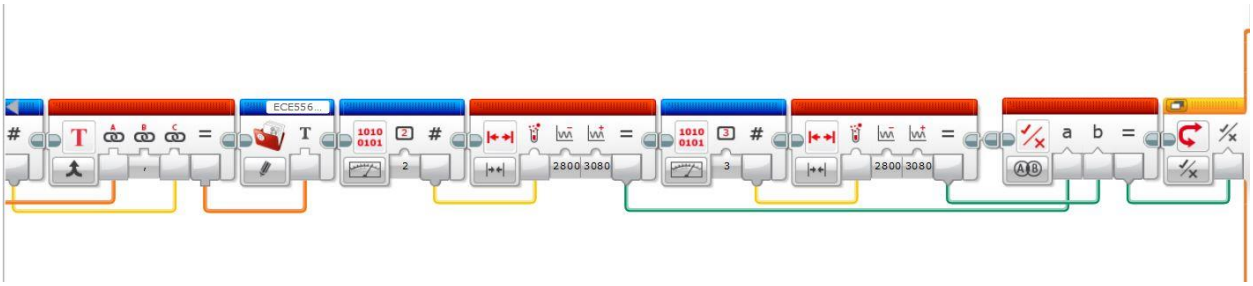


- The bot is programmed to take 8 readings of the sensor (blue)initially to detect the distance from the bot in front of it and the mean was calculated using the mathematical operation block (red) which was set as the target value.

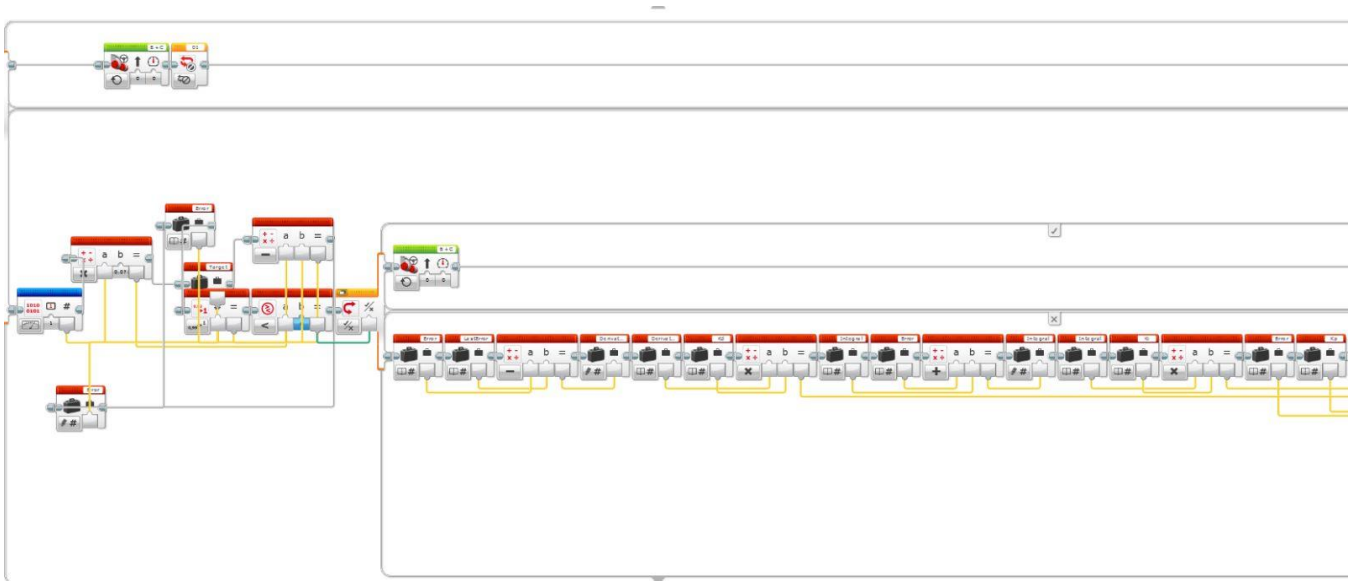


- The variables KP, Ki, Kd, Error and Last Error are initialized using the variable blocks (red). Now the loop starts where the ultrasonic sensor and light sensor readings are detected.

- Data is acquired from both the sensors along with the timestamp (yellow block) using the text block (red) and finally storing the data as a text file using the File Access block (blue).



- The light values are compared within a range and consequently the output is given to the switch block. If the values are within the range (suggesting the detection of red colour) then the motor speed is set to zero and the loop interrupt block is executed. If its false the ultrasonic sensor values are fed to the PID controller.



PID Equations in the code

- **Error = Target - Sensor value**

For Proportional:

- Proportional Output = Kp x Error

For Integral:

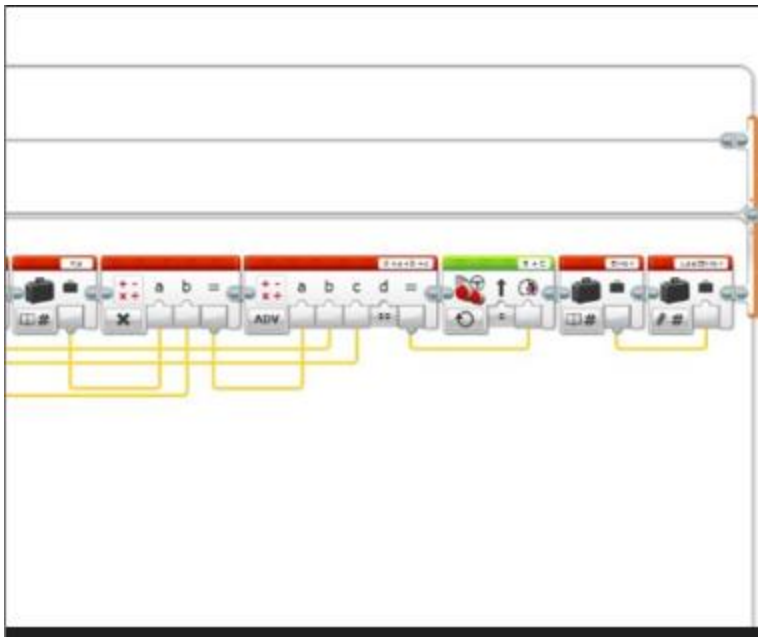
- Integral = Integral+Error
- Integral Output = Ki x Integral

For Derivative:

- Derivative = error - last error
- Derivative Output = Kd x Derivative

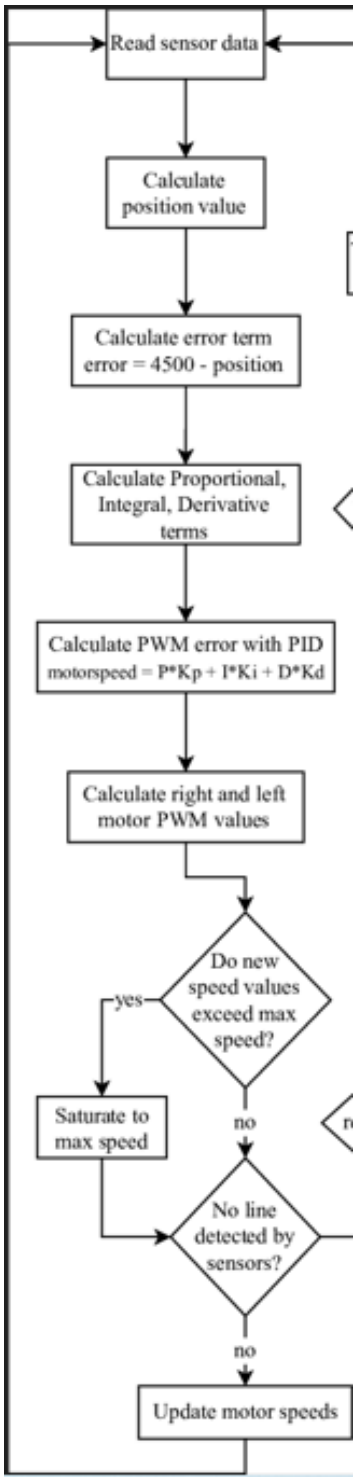
Final Output = Proportional Output + Integral Output + Derivative Output

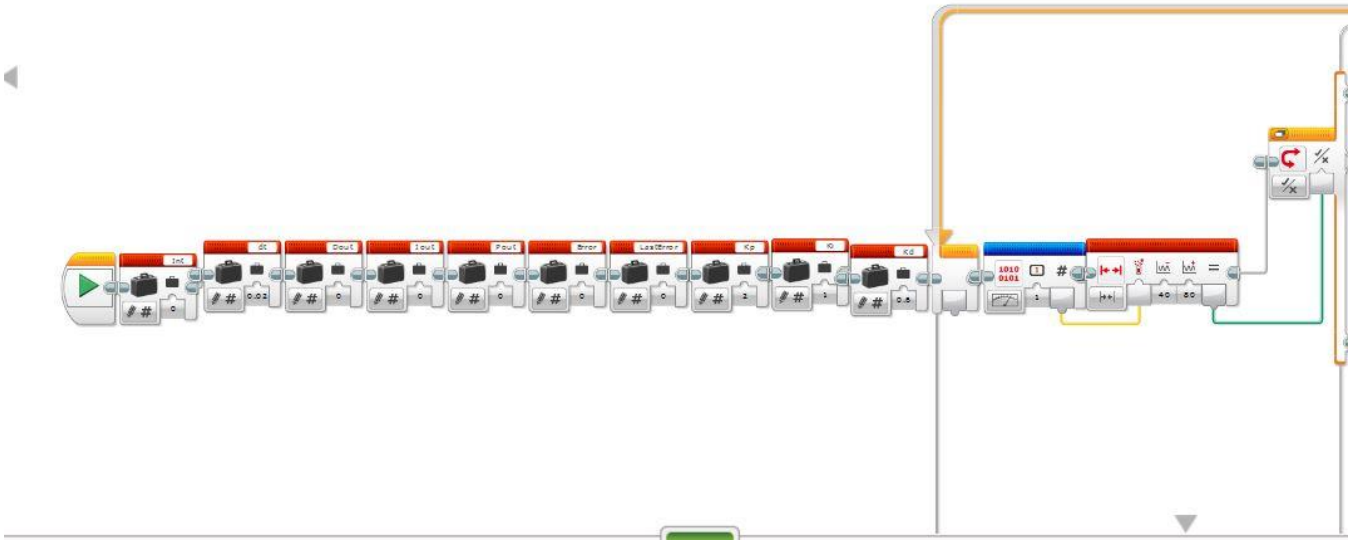
- This Output is added with a speed value and then fed to the Speed parameter of the Move Steering block. At the end the Error value is stored in the Last Error variable.



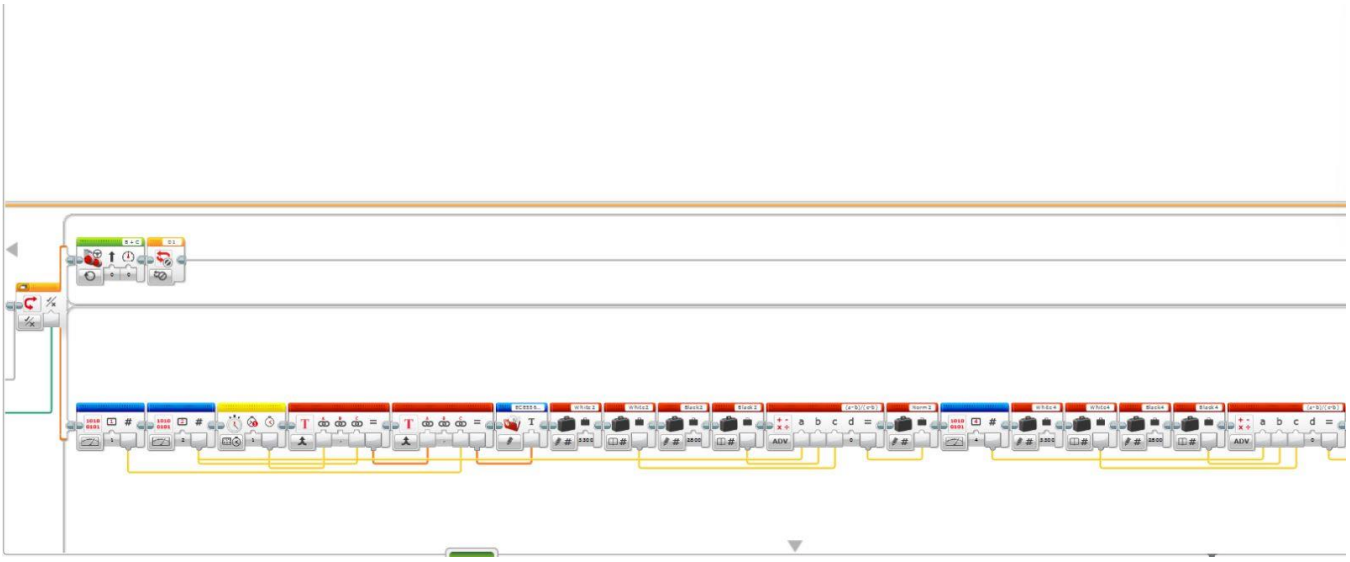
Task 3: Path Tracking

Pseudo Code


Read sensor data
Calculate position value
Calculate error term error = 4500 - position
Calculate Proportional, Integral, Derivative terms
Calculate PWM error with PID motorspeed = P*Kp + I*Ki + D*Kd
Calculate right and left motor PWM values
Do new speed values exceed max speed?
yes
Saturate to max speed
no
No line detected by sensors?
no
Update motor speeds

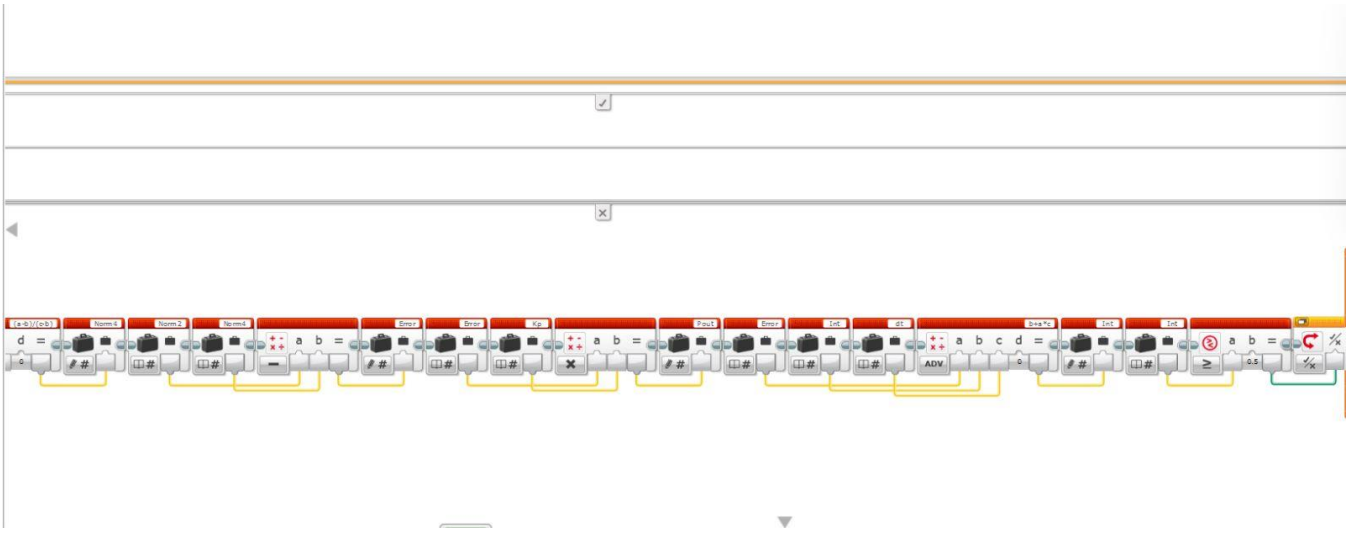


- Initializing the Kp, Ki , Kd , Error , Last Error variables. Then the loop starts and the ultrasonic sensor value is checked whether it falls within a range (suggesting the obstacle at the final point) .

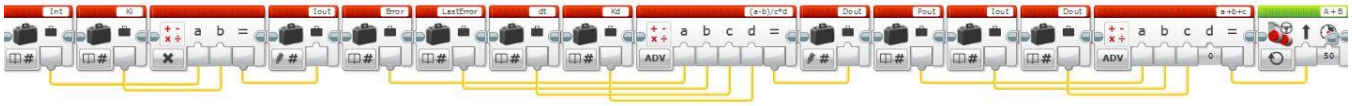


- If it falls within the range then motor speed is set to zero and a Loop Interrupt block gets executed. If it is false, then the timestamp, ultrasonic sensor and light sensor readings are acquired using the Text block and finally saved as a file using the File access block.

- Normalizing the two light sensor values by detecting the values at black color and white color.



- The Error value is calculated and fed to the PID controller which uses the equations mentioned earlier thereby calculating the individual outputs.



- Finally the outputs are summed up and are fed to the direction parameter of the Move Steering Block.

5. RESULTS:

Demo -1: The objective of Demo -1 was to develop a bot using Ultrasonic sensors for obstacle detection

Trial 1: The bot follows the straight path, detects obstacle and parks itself at a distance of 0.6 cm
Trial 2: The bot follows the straight path, detects obstacle and parks itself at a distance of 0.7 cm
Trial 3: The bot follows the straight path, detects the obstacle, but touches the obstacle before stopping.

Demo -2: The objective of Demo -2 was to develop a bot using Ultrasonic sensors and Light sensors for platooning with the TA bot and stopping after detecting the red tape at the end of the track.

Trial 1: The bot follows Platooning with TA bot but doesn't stop after detecting red tape at the end of the track.
Trial 2: The bot follows Platooning with TA bot but doesn't stop after detecting red tape at the end of the track.
Trial 3: We didn't get enough time to complete a third trial for this demo.

Demo -3: The objective of Demo -3 was to develop a line following, self correcting bot using Ultrasonic sensors and Light sensors. It had to follow a black path over a white surface and stop at the end of the path after detecting an obstacle.

Trial 1: The bot follows the desired black path smoothly , detects the obstacle and parks itself at a distance of 0.8 cm, in the time interval of 20 seconds.
Trial 2: The bot follows the desired black path smoothly , detects the obstacle and parks itself at a distance of 1 cm, in the time interval of 20 seconds.
Trial 3: The bot follows the desired black path smoothly , detects the obstacle and parks itself at a distance of 1 cm, in the time interval of 40 seconds.

6. DISCUSSIONS

Hardware:

Some important points to be focussed while choosing hardware components -

- The RC filter chosen for 555 timer output should have a cutoff frequency of around 3Hz or less to get better results and a stable output.
- The SRF04 sensor would still give high fluctuated readings but not a massive change in output, therefore it needs to be normalized as per the range of values detected.
- The placement of the light sensors should be chosen appropriately according to the algorithm design for instance - edge tracking, black color tracking or white color tracking etc.

Software:

Some important points to be focussed while designing algorithm -

- MATLAB & Simulink offer the best coding platform but the EV3 firmware should be updated and able to support such file types.
- The raw sensor values from ultrasonic and light sensors should be normalized to perform basic operations and get a desired output.
- LEGO EV3 MINDSTORMS offers a wide range of actuator controls like move tank block, move steering block, single motor block with rotation modes etc. This can be explored and a correct block with correct modes could be chosen as per requirements.
- PID Tuning can be done using the Zeiger-Nichols method but often the tuning is done by observing the robot motion and output. Few suggestions are listed below:

(a) The robot oscillates a lot when following the line :
High Kp values cause the system to become under-damped thus increasing the oscillations. Thus, we need to decrease the value of the proportional gain to reduce the oscillations about a straight line.

(b) The robot tracks the straight lines well but cannot track the line with sharp turns
Derivative gain determines the sensitivity of the PID controller to upcoming changes. In a sharp turn, the change in error is most significant. If the robot does not follow the turn properly, then it implies that the derivative gain is not enough to influence the movement of the robot when a sharp change in error is detected. Thus, we need to increase the derivative gain in this scenario.

(c) The robot is not centered along the line. It always seems to have a constant offset either to the right or the left.
The integral portion of the PID controller keeps track of the accumulated error. When the robot has a constant offset for a long time, the instantaneous error and the change in error may not be significant but the accumulated error increases along with the time. Thus, if the robot is not correcting its offset, then it implies that the integral gain is not enough. It is advisable to increase the integral gain in this scenario.

7. REFERENCES:

1. NC State University ECE 556 Lectures- Fall 2022 by Prof. F. Livingston
2. NC State University ECE 556 Mechatronics Lab 4&5 - Fall 2022
3. Yoshizawa, Koichi, et al. "Path tracking control of mobile robots using a quadratic curve." Intelligent Vehicles Symposium, 1996., Proceedings of the 1996 IEEE. IEEE, 1996.
4. MATLAB help, Web: https://www.mathworks.com/help/matlab/
5. SRF04 datasheet, Web: https://www.robot-electronics.co.uk/htm/srf04tech.htm
6. EE-SF5B datasheet, Web: http://www.mouser.com/ds/2/307/en-ee_sf5-1221359.pdf