



NYU

**TANDON SCHOOL
OF ENGINEERING**

Advanced Mechatronics

ROB-GY-6103

Project 1 - Arduino Self Parking Robot

Team Members:

Rishabh Sivarajan(rs8426)

Kelvin Maliyakal(kpm8503)

Manan Malpani (mm11787)

Professor:

Dr. Vikram Kapila

Abstract

This master's project intends to build and implement a line-following robot with the ability to scan parking spaces and locate a free spot for parking, employing IR sensors and one ultrasonic sensor. The IR sensors will be used by the robot to navigate as it follows a black line. It must maintain track of the junctions and only scan the parking spaces at the odd intersections to find those that are occupied. The robot will pull into the first spot that opens before indicating how many occupied spaces it found on the way. The goal of this project is to create a line-following robot that can successfully move along a specified course and find open parking spaces, offering a novel solution for the autonomous vehicle industry.

| | |
|--|-----------|
| Advanced Mechatronics | 1 |
| Abstract | 2 |
| Introduction/Project Background | 4 |
| Apparatus | 5 |
| Bill of Materials(BOM) | 5 |
| Design Layout | 7 |
| Circuit diagram | 8 |
| Approach | 9 |
| Workflow | 10 |
| Solution for the Approach | 10 |
| Problems Faced | 11 |
| Robot Images | 13 |
| Robot: Back view | 13 |
| Scope of Improvement | 14 |
| Conclusion | 14 |

Introduction/Project Background


In this project, we aimed to design and build a robot that could follow a black line on a white surface autonomously using an Arduino microcontroller. Our goal was to develop a cost-effective and efficient solution that could be used in an autonomous parking interface using just Infrared (IR) and one Ultrasonic Sensor.




In this report, we will discuss the design and implementation of the robot, the programming logic used to enable it to follow the line, and the challenges we faced during the project. We will also share our insights into the potential applications and future improvements for this robot.

Apparatus

The project required us to build an autonomous self-parking robot, some of the parts were sourced from the previous semester's Mechatronics project and additionally, an Arduino kit and its components were purchased.

Bill of Materials(BOM)

| Serial No. | Part Name | Quantity | Part Image | Unit Cost(\$) | Total Cost(\$) |
|------------|--------------------------|----------|--|---------------|----------------|
| 1 | Parallax Screwdriver | 1 |  | 1.5 | 1.5 |
| 2 | Resistor(220Ω) | 3 |  | 0.33 | 0.99 |
| 3 | RGB LED | 1 |  | 2 | 2 |
| 4 | IR Sensor | 4 |  | 0.6 | 2.4 |
| 5 | Ultrasonic Sensor | 1 |  | 29.71 | 29.71 |
| 6 | Machine screws, Phillips | 8 |  | 0.25 | 2 |
| 7 | Hex Nuts | 4 |  | 0.14 | 0.56 |
| 8 | Tail wheel ball | 1 |  | 3.95 | 3.95 |
| 9 | Boe-Bot aluminum chassis | 1 |  | 29.99 | 29.99 |

| | | | | | |
|----|---|----|--|-------|-------|
| 10 | Battery holder with cable and barrel plug | 1 |  | 4.99 | 4.99 |
| 11 | Cotter Pin | 1 |  | 0.3 | 0.3 |
| 12 | Plastic wheels | 2 |  | 2 | 4 |
| 13 | Continuous rotation servo | 2 |  | 17.95 | 35.9 |
| 14 | AA Batteries | 4 |  | 1.33 | 5.32 |
| 15 | Arduino Board | 1 |  | 24.84 | 24.84 |
| 16 | Breadboard mini | 1 |  | 3.95 | 3.95 |
| 17 | Arduino Cable | 1 |  | 7.99 | 7.99 |
| 18 | Jumper Cables | 32 |  | 0.06 | 1.92 |

BOM: Table 1

Total cost: \$162.31

Design Layout

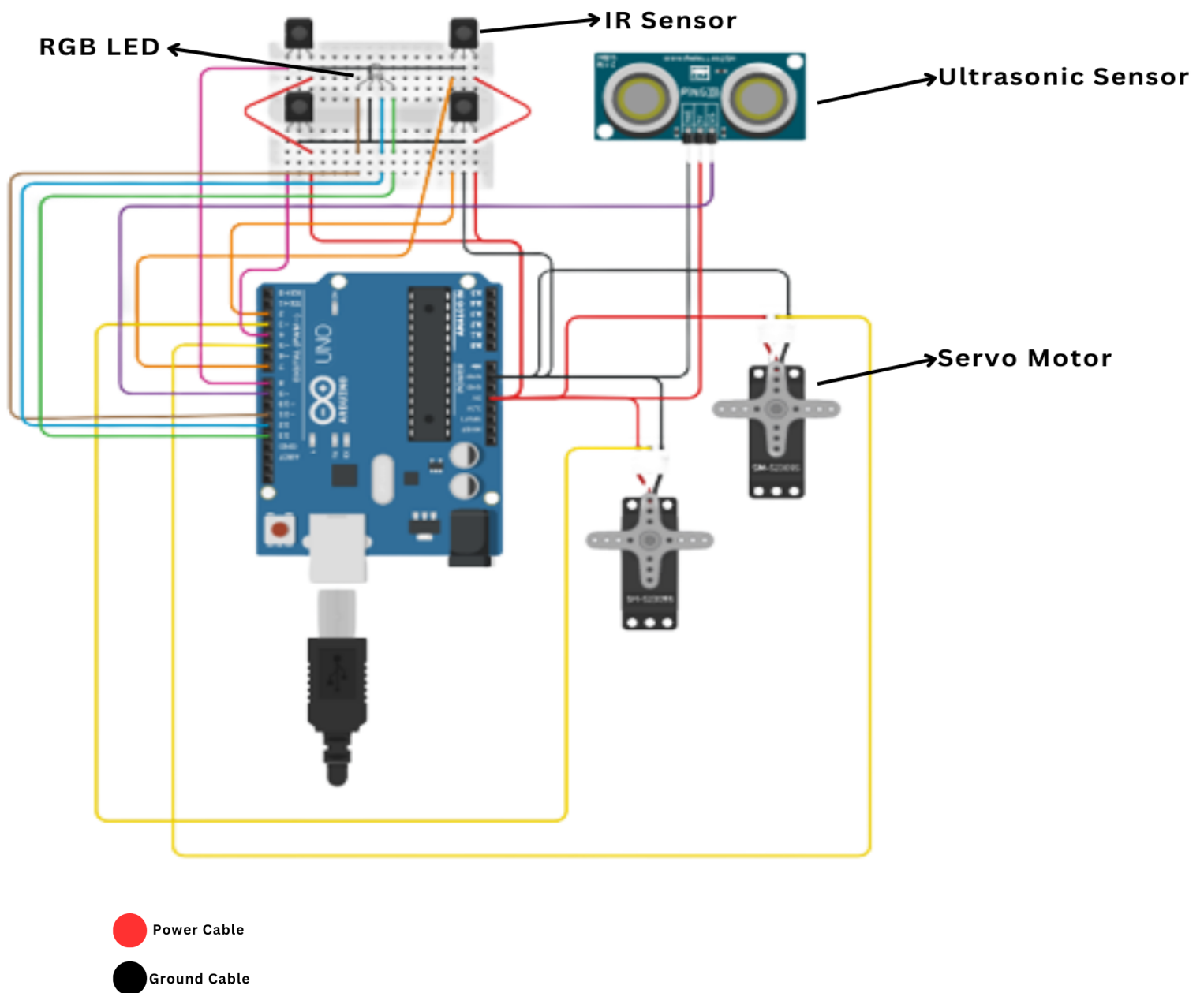
In this design layout, the robot is equipped with two infrared (IR) sensors in the front and two IR sensors in the back, as well as a fixed ultrasonic sensor on the left.

The front and back IR sensors detect the line on the ground. The two front sensors are used to help the robot maneuver when it is moving forward, the two rear sensors are used to allow the robot to come in the reverse direction when it encounters a parked car during its surveying process.

The fixed ultrasonic sensor on the left is used for obstacle detection(Parked space). Ultrasonic sensors emit high-frequency sound waves and measure the time it takes for the sound waves to bounce back after hitting an object. By using this information, the robot can detect if a parking slot is filled on its left after which it continues on its following movement.

The robot moves forward on the odd number intersections to determine if a parking space is full or not. Overall, this design layout allows the line-following robot to accurately follow a line on the ground while avoiding obstacles on its left side. The combination of front and back IR sensors provides redundancy and ensures that the robot stays on the line even if it moves slightly off-center. The fixed ultrasonic sensor offers an additional layer of safety by detecting obstacles and preventing collisions.

Circuit diagram



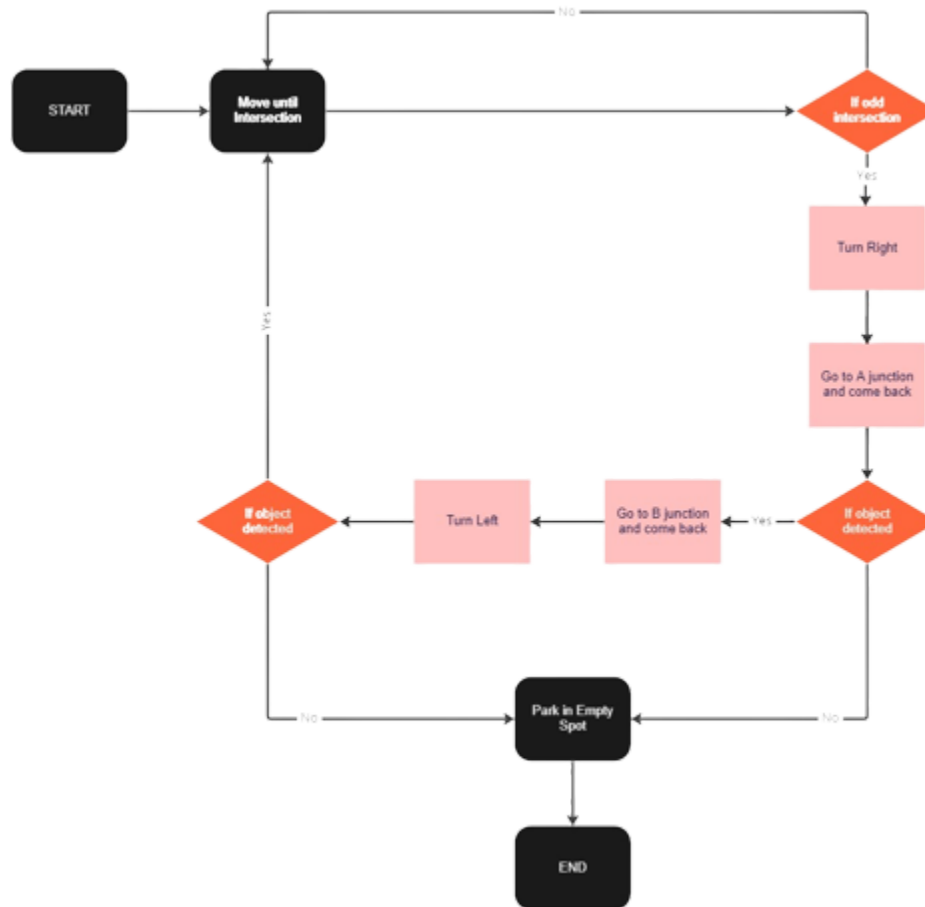
| Pin No. | Component | Color |
|---------|---------------------|---------|
| 2 | IR Sensor (Back 1) | Orange |
| 3 | Left Servo | Yellow |
| 4 | IR Sensor (Back 2) | Magenta |
| 5 | Right Servo | Yellow |
| 7 | IR Sensor (Front 2) | Orange |
| 8 | IR Sensor (Front 1) | Magenta |
| 9 | Ultrasonic | Purple |
| 11 | RGB (Red) | Brown |
| 12 | RGB (Blue) | Blue |
| 13 | RGB (Green) | Green |

Circuit Diagram Table:2

Approach

1. Initiate forward motion from the starting point.
2. Execute a right turn at the intersection.
3. Activate the ultrasonic sensor in conjunction with the line-following mechanism, and continuously scan for any stationary vehicles within range.
4. Terminate the loop upon detection of an object or upon reaching the end of the line junction in the right lane.
5. Upon encountering the center lane, reverse direction and verify the availability of the right parking lane. If vacant, park in that location. If occupied, continue reversing and assess the availability of the left parking lane. Terminate the loop upon detection of an object or the end of the line junction. Move forward until the center lane is detected.
6. Upon reaching the center lane, move forward and assess the availability of the left parking lane. If vacant, park in that location. If not, move forward and examine the next odd intersection. Repeat Steps 1-6 until the 9th intersection is reached.

Workflow



Solution for the Approach

1. To facilitate forward motion, we have implemented two IR sensors to detect lane markings. These sensors generate a high reading when black is detected and a low reading when white is detected. The sensors are placed such that they align with the width of the black line. The "motion" function is called to initiate forward movement when both sensors detect white. When either of the two sensors detects black, it indicates that the robot has deviated from the intended path, and a lane correction is executed accordingly. The servo motors are controlled using the Servo motor library, with one motor operating in the opposite direction to the other. We use the "writeMicroseconds" function to manipulate the servo motor's speed and direction of rotation. This function takes an argument in the range of 1000 to 2000, with 1500 being the midpoint or a halted position. Arguments below 1500 cause clockwise rotation, while arguments above 1500 cause counterclockwise rotation.
2. To make a right turn, the servos are given the same arguments for a certain duration.

3. After executing the right turn, the ultrasonic sensor is activated, and the distance is continuously measured on the lane.
4. When the distance measured is less than the set threshold, the parking variable is set to false, and the loop for checking the right lane is terminated.
5. The two IR sensors at the back are used for reverse movement, and this process is repeated until the IR sensors at the front detect black. At this point, the parking variable is checked, and if the lane is unoccupied, the robot turns left, moves forward, makes a right turn, moves forward again to park, and then exits. If the parking variable is false, the occupancy of the left lane is checked, and this process continues until an unoccupied spot is found.
6. The distance is continuously checked to ensure that it falls within the set threshold, and this process continues until the robot reaches lane B or the left lane. If the distance measured remains greater than the set value, the parking variable is set to true, and the loop is terminated. The robot moves forward using the two IR sensors at the front and then turns left at the center lane. The parking variable is checked, and if it is true, the robot moves forward and turns left at the intersection, followed by forward motion and an exit. If the parking variable is false, the process from step 1 is repeated while keeping track of the intersections. Once the robot reaches the 9th intersection, it exits.

Problems Faced

1) Blind right and left turns to calculated turning:

To address this issue, a technique was implemented to monitor the number of changes detected by an IR sensor located at the front of the robot. Specifically, when the sensor detects two changes in its input signal, it indicates that a turn has been completed, whether it be a right or left turn. To differentiate between a right and left turn, the front right IR and front left IR sensors were used to track the direction of the turn. This method allows the robot to accurately detect and complete turns during its operation.

2) Reverse using the IRs in the back:

At the initial stage, the detection of the center lane using the two IRs at the back of the robot led to a misinterpretation of the location of an object in the right lane as being in the left lane. This is because the robot came to a stop at the center lane upon detection by the back IRs, causing the majority of the robot to be positioned in the right lane along with the ultrasonic sensor. As a result, when the left lane parking function was called upon detection of the center lane, it wrongly identified the object in the right lane as being in the left lane, owing to the robot's positioning. To overcome this challenge, we programmed the robot to move in reverse until the front two IRs detected the center lane, thereby resolving the issue.

3) Problem with moving into restricted parking lanes:

At the outset, there was an issue with the counting of intersections. The function for forward motion brings the robot to a stop at every intersection, which updates a variable keeping track of the intersections. If the intersection is even, which corresponds to the lanes where parking is prohibited, the robot is supposed to skip the parking check. However, our initial approach led the robot to check at every intersection. This was because, after stopping at the intersection, the next stage of the code was to determine if the intersection was even or odd. The problem with this approach is that, after the first intersection, the robot comes to a halt at the next one, and since the intersection tracker variable is updated, it skips the main code for parking check on the left and right lanes, as intended. However, as soon as this iteration ends, forward motion is called, which is programmed to stop and end when both the front sensors detect black. This is problematic because the two sensors are already at the even intersection and halted, and yet the intersection variable is updated, leading to the robot checking the lane where parking is prohibited. To resolve this issue, we introduced a new function called `straight_correction()`, which is called at the end of each loop iteration. This function allows the robot to move forward for some additional time, thereby enabling it to cross the intersection and preventing the aforementioned problem.

4) Taking reverse by using the value of all 4 IR sensors:

To begin with, the program was designed to run the reverse function in a loop until both front IR sensors detect black, indicating the center lane has been reached.

while((outputir1 != 1) && (outputir2 != 1))

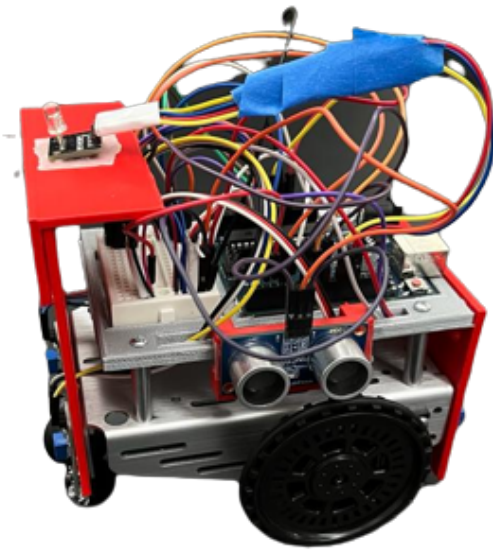
This condition was satisfactory until the point when one of the IR sensors detected black, leading to one of the separate conditions being false while the other one remained true. As a result, the overall AND condition became false, and the loop was terminated prematurely, leading to the execution of the subsequent lines of code.

To solve this problem we changed the line of code to

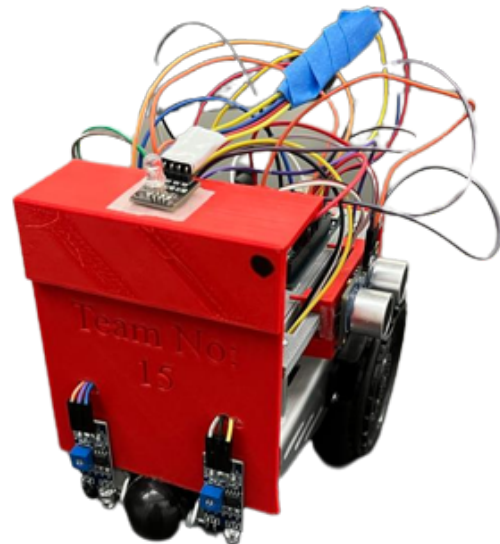
while(!((outputir1 == 1) && (outputir2 == 1)))

This line of code addresses the issue by correcting the logic in the event that one of the sensors detects black. The separate conditions, separated by the logical AND operator, will evaluate to false, and the overall NOT operator will change it to true when both the sensors give a white reading. In the event that one of the IR sensors detects black and the other detects white, one condition will evaluate to true and the other to false, resulting in a false overall evaluation, which is then inverted by the NOT operator to true which solves the problem. Only when both the separate conditions are true, and their overall '!' false, does the loop end.

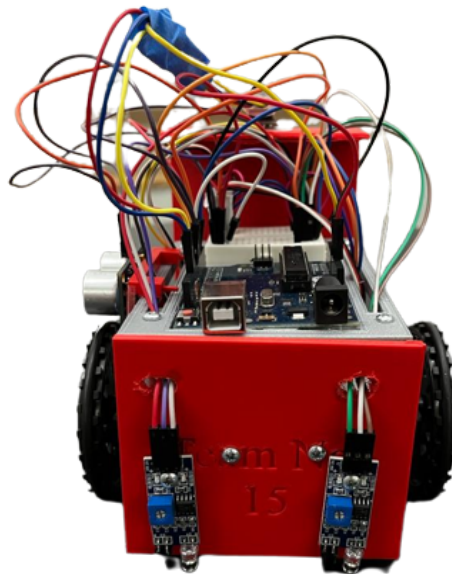
Robot Images



Robot: Side View (Ultrasonic Sensor)



Robot: Front View



Robot: Back view

Scope of Improvement

1. Tuning motor speed:

The motor speeds could be further fine-tuned to make the movement of the robot smoother and more precise, this is very difficult to achieve due to the limitations of the servo motor hardware.

2. Weight Balance:

The robot was slightly heavier towards the back than the servos and the Arduino weight. This caused the robot to jump a little when it moved from a standstill position to a forward position the front wheel may get suspended in the air for a short period of time.

3. Memory Optimisation:

Certain functions can be further optimized to be written with fewer lines of code to ensure it occupies lesser memory using port manipulation to consume fewer bits.

Conclusion

We developed a line-following robot that traces a black line. The line following robot has 4 IR sensors, 2 in the front and 2 at the back. The robot begins by turning right at every odd intersection where it travels to the T junction or until it spots a car parked in the adjacent parking using the ultrasonic sensor.

The team was able to successfully identify occupied parking spaces by flashing a RED LED. It was able to make its way to the empty parking space. The robot also flashes a BLUE LED at each intersection. Once parked the robot flashes the GREEN LED to indicate that it has been parked, followed by the PURPLE LED which blinks for the total number of occupied parking spots it encountered.

For the bonus case, the robot was able to identify when a vehicle is parked beyond the T-junction and indicate that by flashing a WHITE LED. It then proceeds to park in that parking spot.