Princess Sumaya University for Technology

King Abdullah II School of Engineering
كليـــة الملك عبد الله الثاني للهندســة

Final Project Report

Embedded Systems Project

Autonomous Robot Car challenge

Team Members:

Yazan Al-Naimat 20200702

Ayham Ismail 20220039

Rakan Alshishani 20220030

Instructor: **Dr. Belal Sababha**

Contents

1. Abstract

For our final project, we built an autonomous robot car using the PIC16F877A microcontroller. The idea was to make a robot that could drive itself along a black track, dodge obstacles, and recognize dark environments (like tunnels). We used a simple sensor layout: two IR sensors for line following, another one in front for obstacle detection, and two more on the sides for wall-following. We also included an LDR to detect low-light situations and a servo motor that raises a flag when the course is completed. Everything runs automatically based on what the sensors read in real time.

2. Introduction

Autonomous robots have become a big deal in automation and intelligent transport. For us, as engineering students, building a line-following robot gave us hands-on experience in integrating sensors, motors, and decision-making logic.

Our robot needed to do three main things:

1. **Follow a Line** using two IR sensors underneath.
2. **Avoid Obstacles** with a front IR sensor and side sensors.
3. **Detect Tunnels** using a light sensor (LDR) and make a sound.

We chose the PIC16F877A microcontroller because it had just the right number of ports for all the sensors and motor control we needed.

3. System Overview

The robot runs in a loop of sensing and acting. Here's the general process:

- **Sensors** read the environment — line, walls, obstacles, and light.
- The **PIC microcontroller** decides what "mode" the robot should be in (Line Tracking, Obstacle Avoidance, Tunnel).
- It then tells the **motors or buzzer** what to do.
- Everything is powered by a compact battery pack.

4. Mechanical Design

We went with a basic two-wheel drive chassis — reliable and easy to build on.

- **Drive System**: Two rear DC motors.
- **Stability**: A front caster wheel balances the robot.
- **Sensor Mounts**:
    - Line sensors under the chassis, spaced to detect black line edges.
    - A front IR sensor at bumper level to stop when something’s in front.
    - Side IRs on the left and right for tight spaces and wall detection.

We learned early on that even small vibrations mess up readings, so we made sure everything was mounted tightly.

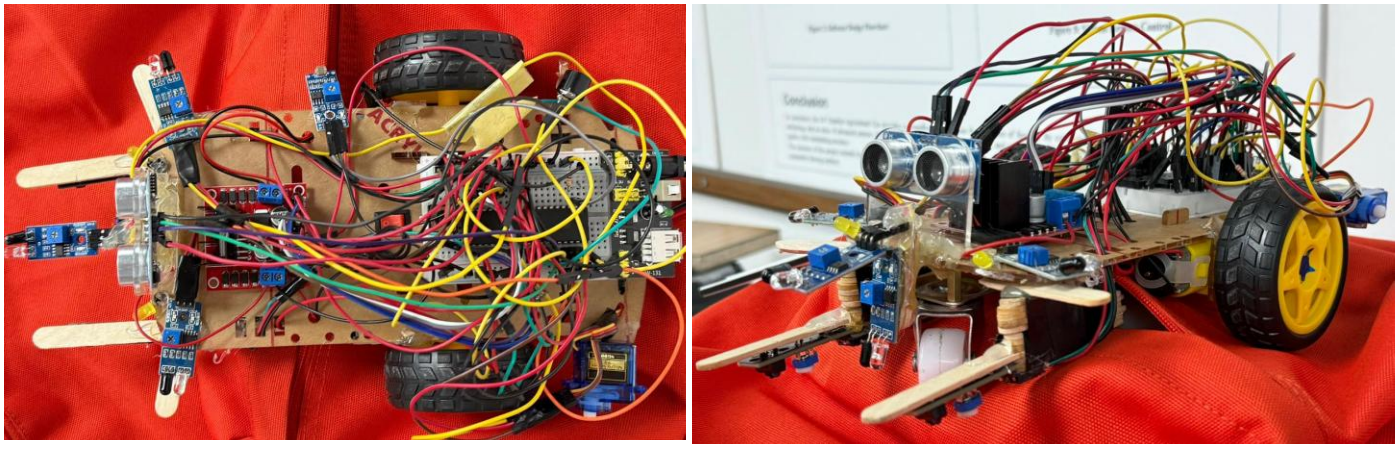

Figure 1: Top and side view of the autonomous robot showing sensor placement.

5. Hardware Design

5.1 Microcontroller

We used the PIC16F877A as the controller. It reads signals from IR sensors and the LDR, and then controls the motors and outputs. All logic — like whether to turn or stop — runs from here.

5.2 Sensors

- **Line Sensors** (connected to RB3 and RB4): These detect whether the robot is veering off the line.
- **Front Obstacle Sensor** (RB5): Sends a "stop" signal if it sees something too close.
- **Side IR Sensors** (RB6 and RB7): Help steer the robot around walls or tight corners.
- **LDR Sensor** (RA0): Checks for darkness. When light levels drop (e.g., entering a tunnel), the robot activates a buzzer.

5.3 Actuators

- **DC Motors**: Controlled through an L298N driver connected to PORTD.
- **Buzzer** (RC0): Goes off in dark areas to signal "Tunnel Mode."

- **Servo Motor** (RD5): Raises a flag when the robot finishes the track.

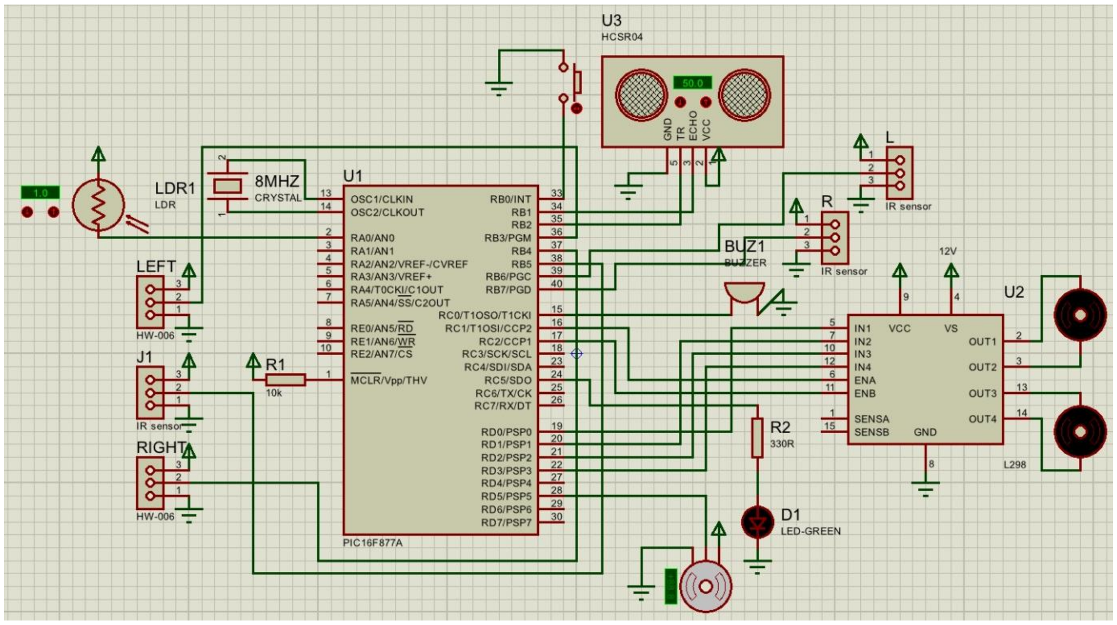

Figure 2: circuit diagram.

6. Software Design

We programmed the robot in Embedded C using mikroC. To keep things efficient, we used direct port masking instead of relying on heavy libraries.

Here's the priority logic in the software:

1. **Obstacle Check**: First, it checks if there's something ahead using the front IR sensor.
2. **Side Check**: If blocked, the robot looks at the side IRs to figure out the best escape path.
3. **Tunnel Detection**: The LDR is read via ADC — if it's dark enough, the buzzer turns on.
4. **Line Following**: If no obstacles or tunnel, the robot sticks to the line using the left and right IR sensors.

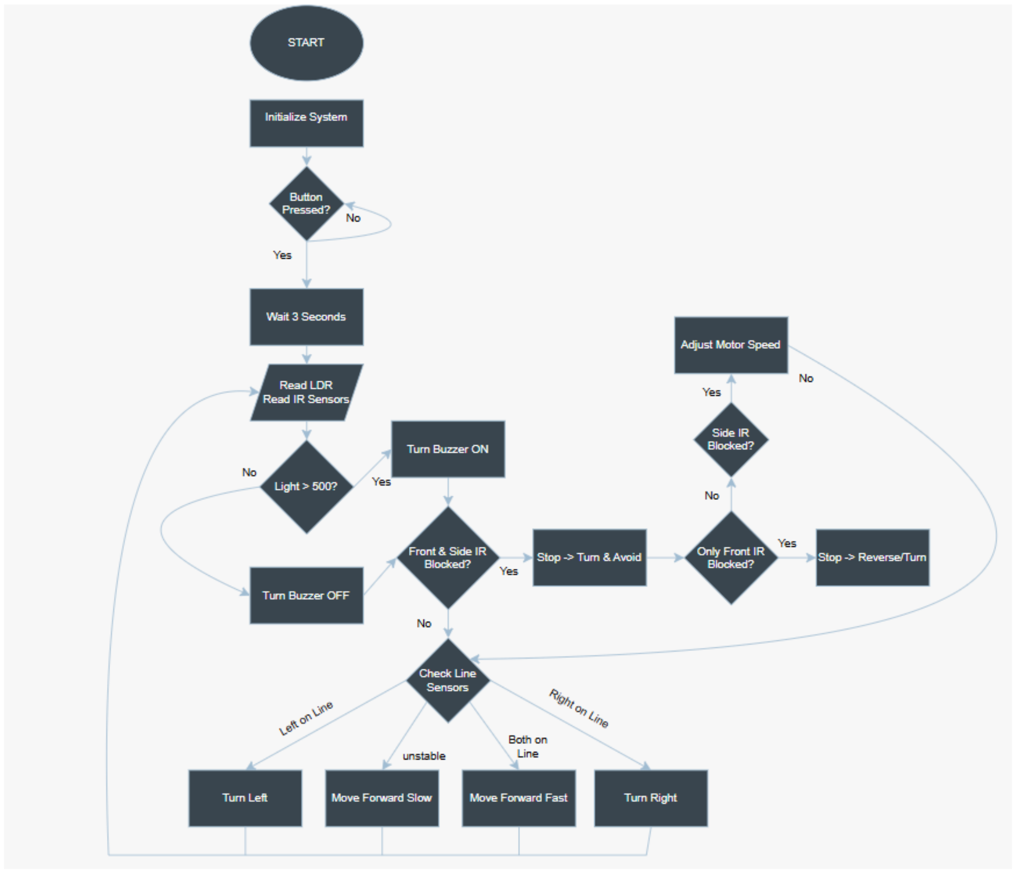

Figure 3: Software flowchart illustrating the control logic.

7. Testing and Results

We tested the bot on a hand-made course with turns, a box in the middle, and a tunnel made from cardboard.

Some notes from our testing phase:

- **Sensor Swap**: We initially tried an ultrasonic sensor for the front, but it was too slow and unreliable on shiny surfaces. Swapping it out for an IR sensor made the reaction time faster.
- **Line Following**: The robot kept itself pretty well aligned on the track with the two IR sensors.
- **Obstacle Avoidance**: It correctly stopped when sensing an object and turned away using side IRs.
- **Tunnel Mode**: In low light, the buzzer beeped as expected — a small win that felt pretty satisfying.

8. Conclusion and Future Work

This project was a great learning experience — from soldering wires to tweaking the code on the fly. Our robot successfully followed the line, avoided obstacles, and responded to dark environments.

What we'd improve if we had more time:

- **PID Control**: Would make the line-following smoother and reduce wobbling.
- **Custom PCB**: Would eliminate the messy wires and make the build more stable.

All in all, we're proud of how far this little robot has come!