ARDUINO BASED SMART OBSTACLE-AVOIDING ROBOT CAR

Avanthika Vuppala, Afia Rahman, Ella Bertolloti, Jaqueline Lopez

Main Goals/Specifications

The main goal of our project was to create a small smart car that could avoid obstacles on its own and also be controlled remotely through a remote device. We wanted the car to detect objects in front of it and automatically change direction to avoid crashing. It needed to decide which way to go based on the distance detected on each side. We also added manual control using Bluetooth so the user could drive the car with their phone when needed. To make this happen, we used motors, sensors, a microcontroller, and wireless communication modules. Everything was wired and programmed to work together as one system. The project had to demonstrate the six types of mechatronic subsystems we studied in class.

Mechatronic Design and Subsystems

1) Actuators

The car moves using four small DC motors, one for each wheel. These motors let the car drive forward, backward, and turn. There is also a small servo motor called an SG90 that is used to rotate the ultrasonic sensor on the front of the car. This rotation allows the sensor to scan both sides to help the car figure out where to go next. Both the wheel motors and the servo motor are what actually make the car move and react to its environment.

2) Sensors

We used two types of sensors in this project. The main one is an ultrasonic sensor that can measure how far away objects are in front of the car. It is mounted on the servo motor so it can turn and look around. This helps the car detect if something is blocking its path and choose a better direction. We also placed three infrared sensors under the car to help it follow a black line on a light surface. These line sensors can tell when the car is going off track and help it stay on course.

3) Input Signal Conditioning and Interfacing

To receive instructions from the user, the car has a Bluetooth module and a small infrared receiver. These parts take signals from a smartphone or a remote control and pass them to the Arduino board. The Arduino then figures out what the user wants the car to do, like move forward or stop. These inputs are what let us control the car from a distance without touching it directly.

4) Digital Control Architecture

At the center of the car is the Arduino Uno, which acts like the brain of the whole system. It runs the program that we wrote and makes decisions based on what the sensors are telling it or what the user is commanding. The Arduino checks if there are any obstacles, turns the wheels when needed, and controls how the car moves. It is also in charge of managing communication with the Bluetooth and IR modules. Everything runs through the Arduino to keep the system working smoothly.

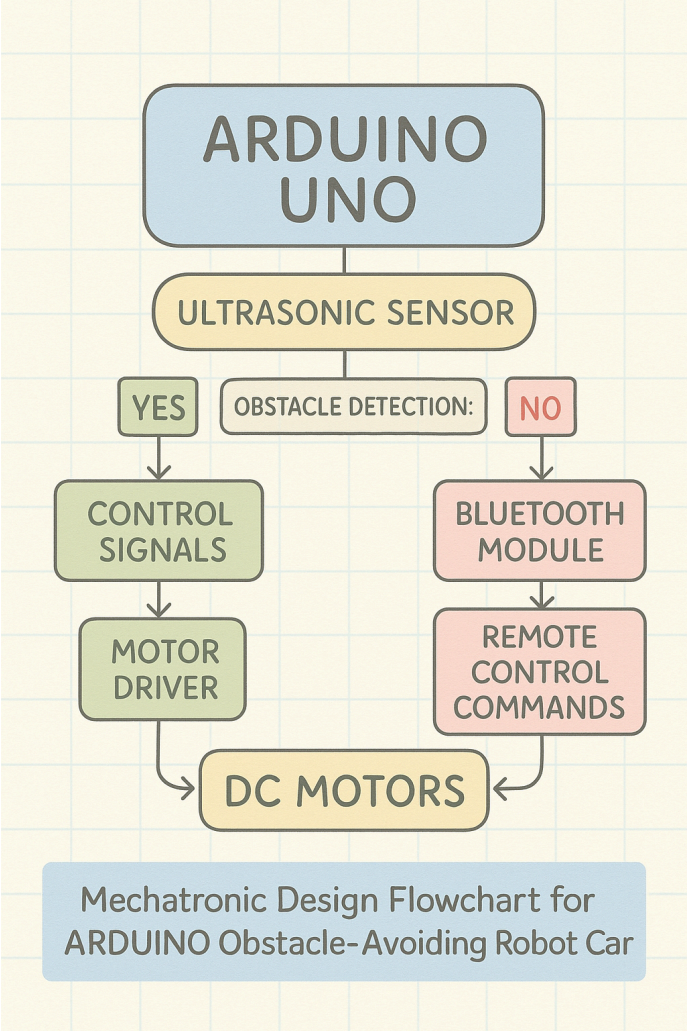
5) Output Signal Conditioning and Interfacing

The signals coming from the Arduino are not strong enough to power the motors directly. To fix that, we used an L298N motor driver board. This board boosts the signals and sends enough power to the motors so they can actually move the car. It also helps control the direction the motors spin, which is how the car knows when to turn or reverse. This part makes sure the motors get just the right amount of power at the right time.

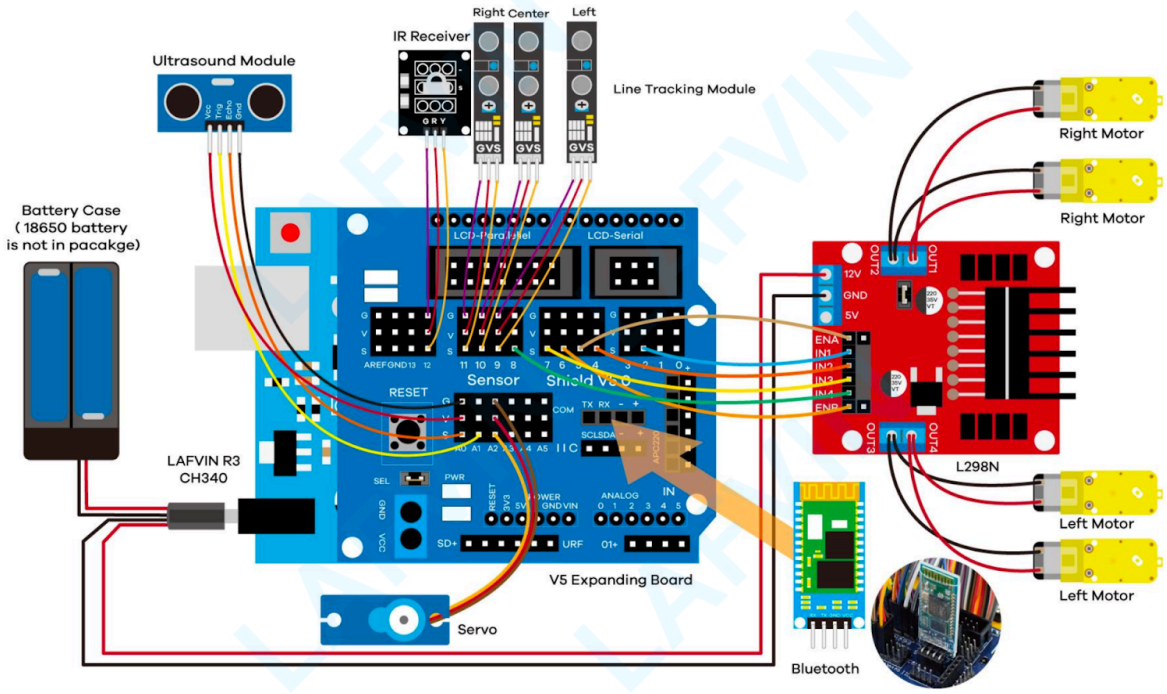
6) User Interface

The car can be controlled in two ways. The first is through Bluetooth using a phone app, and the second is by using an infrared remote. These tools let the user send commands to the car like go forward, backward, left, or right. The car responds by moving in real time based on those commands. The way the car moves and reacts gives feedback to the user, showing that the input was received and acted on. This is how we interact with the car and guide its actions when it's not in automatic mode.

FLOWCHART TO DESCRIBE DESIGN LOGIC:



SCHEMATIC TO SHOW MECHATRONIC/MICROCONTROLLER DESIGN:



Microcontroller and Main Components Used

The heart of the system is the Arduino Uno, which connects to everything and runs the program. The ultrasonic sensor (HC-SR04) is used to check how far away things are, and the SG90 servo motor helps it rotate to scan the area. The wheels are powered by four yellow TT DC motors, which get their energy and direction from the L298N motor driver. For wireless control, we used an HC-05 Bluetooth module, and for remote control testing, we used an infrared receiver. The bottom of the car has three TCRT5000 infrared sensors for line tracking. The whole system runs on a battery pack with four AA batteries, and all the parts are mounted on a plastic frame with wheels.

Main Results and Discussion

The robot car was successfully built after about 3–4 hours of assembly, including mounting the chassis, motors, sensors, and connecting the wiring. While the build itself went fine, most of our time was spent troubleshooting and adjusting the Arduino code through trial and error. Initially, all motor driver outputs were working, but somewhere along the way we accidentally damaged the OUT1 and OUT2 channels. We confirmed this because they worked earlier during testing. To work around it, we tried rewiring the motors to the remaining outputs, and although the motors did respond, they didn't behave the way we expected in terms of direction and control. Because of that, we decided to switch to a two-wheel drive setup instead of the original all-wheel drive plan. This change required modifying both the wiring and code, but we were still able to get the robot car running.

We also discovered that the car needs at least 9V to operate independently, but the batteries we had only supplied 3.7V. As a result, we had to keep the Arduino connected to the computer for it to receive enough power to function properly.

Conclusions and Summary

This project was a great way to apply what we learned in class. We got to work with real motors, sensors, and wiring, and we also learned how to write code that controls everything. Building something that actually moves and responds was fun and helped us understand how the different parts of a system work together. We also learned a lot about fixing problems and thinking through how to connect everything. If we had more time, we might have added more features like speed control or obstacle memory. But we were happy with what we built, and it worked just like we planned.

Video

IMG_6367.MOV