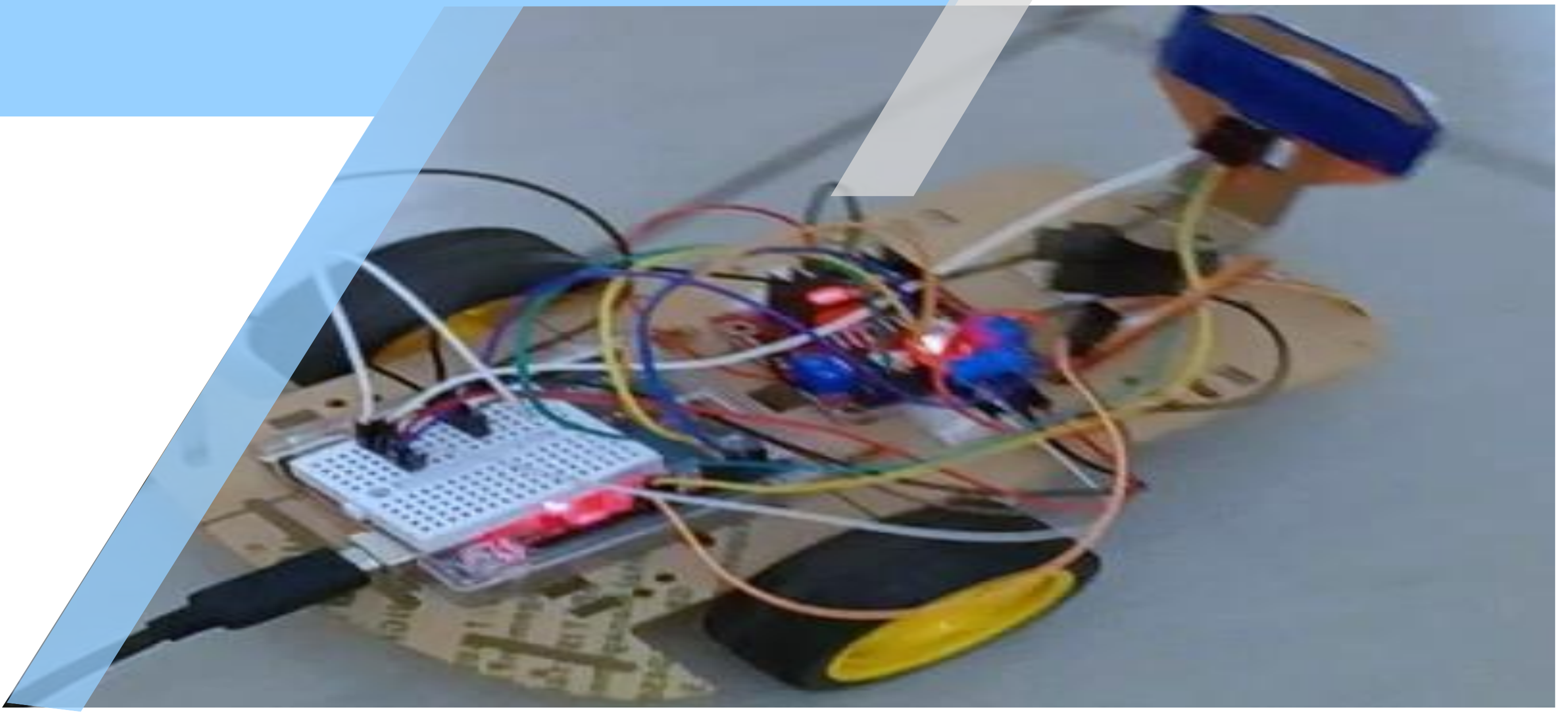
Yaovi Joel DJATASSIBA

REPORT

Automatic Obstacle Avoidance Car



E-mail
jdjatassiba@gmail.com

LinkedIn
www.linkedin.com/in/joeldjatassiba/

Site web
cyber-trinity.github.io/joeldjatassiba/

Introduction

The Automatic Obstacle Avoidance Car is a foundational robotics project built upon the Arduino platform, demonstrating principles of autonomous navigation and embedded control. The primary aim is to develop a mobile robotic platform capable of autonomous navigation in an unpredictable environment. The robot must identify obstacles and independently determine the optimal path for avoidance. This project is crucial as it serves as a practical exercise in integrating hardware (sensors, motors, actuators) and software (C++ control logic) to achieve intelligent, self-governing movement without human intervention.

Hardware Components (Bill of Materials)

The robot utilizes the following components:

Component	Role in the System
Arduino UNO R3	Microcontroller; executes all control logic.
Ultrasonic Sensor (HC-SR04)	Measures distance to obstacles using sound waves.
Standard Servo Motor	Sweeps the ultrasonic sensor to check side distances.
DC Geared Motors (x2)	Provides mechanical propulsion for the wheels.
L298N Motor Driver Module	Interface between Arduino and DC motors, controlling power and direction.
Car Chassis	Physical frame for mounting all components.
Power Supply	Dedicated battery pack (e.g., Li-ion or 9V) for motors and electronics.
Breadboard & Jumper Wires	For prototyping and making connections.

Software Dependencies and Libraries

The project relies on a minimal set of software tools:

Dependency	Type	Purpose in Code
Arduino IDE	Development Environment	For writing and uploading C++ code to the Arduino.
Servo.h	External Library	Simplifies control over the servo motor's angle (myservo.write()).
Standard Arduino Core	Built-in Functions	Provides all fundamental I/O

Implementation and Construction Guide

Implementation involves two main phases: Hardware Assembly and Wiring.

Mechanical Assembly

1. Mount Motors: Attach the DC geared motors to the chassis using clamps and screws. Mount the wheels onto the motor shafts.
2. Mount Caster: Attach the caster wheel (or ball caster) to the front or back of the chassis for stability.
3. Mount Arduino and Driver: Secure the Arduino UNO and the L298N motor driver to the chassis using standoffs or double-sided tape.
4. Assemble Sensor Mount: Attach the servo motor to a rigid part of the chassis. Mount the ultrasonic sensor onto the servo horn (using a small bracket or card) so that it can rotate smoothly.
5. Mount Power Supply: Secure the battery pack to the chassis.

Wiring Connections

Referencing the pin definitions in the provided code:

Component	**Component Pin**	**Connection to Arduino Pin**
DC Motors	Connect to Motor Driver Outputs	(No direct Arduino connection)
Motor Driver (L298N)	IN1 (Motor A)	**Digital Pin 3 (IN1)**
Motor Driver (L298N)	IN2 (Motor A)	**Digital Pin 2 (IN2)**
Motor Driver (L298N)	IN3 (Motor B)	**Digital Pin 4 (IN3)**
Motor Driver (L298N)	IN4 (Motor B)	**Digital Pin 5 (IN4)**
Ultrasonic Sensor	Trig Pin	**Digital Pin 8 (trig_pin)**
Ultrasonic Sensor	Echo Pin	**Digital Pin 9 (echo_pin)**
Servo Motor	Signal Pin	**Digital Pin 10**
Power Connections	Connect common GND and supply power to the motor driver (Vin and GND). The driver will supply power to the Arduino.	(Ensure all components share a common ground)

Code Analysis and Logic Flow

The Arduino code executes a continuous loop of sensing and decision-making.

1. **Initialization (setup):** Sets motor pins (IN1-IN4) and sensor pins (trig/echo) as outputs/inputs. Initializes the servo on pin 10 and Serial communication.

2. **Main Loop (loop)**:

- The servo is set to the center position (90°).
- The front distance is checked using checkDistance().
- If Distance >= 30 cm: The car moves forward (forward()).
- If Distance < 30 cm (Obstacle Detected):
    - The car stops (stop()).
    - Direction Scan (checkDirection()): The servo is rotated to 180° (left) and then 0° (right) to find the path with the greatest clearance (>= 30 cm).
    - Decision:
        - Case 1 (Left Clear): Turns left for 425 ms, then stops and proceeds forward.
        - Case 2 (Right Clear): Turns right for 415 ms, then stops and proceeds forward.
        - Case 4 (Blocked): Moves backward for 415 ms, stops, and runs recheckDirection() (which re-scans left/right after reversing) before turning and resuming movement.
        -

3. Utility Functions:

- checkDistance(): Sends a pulse, measures the return pulse duration using pulseIn(), and calculates the distance in centimeters.
- Movement functions (forward(), backward(), turnLeft(), turnRight(), stop()): Control the motor driver pins (IN1-IN4) to command the wheels.

Conclusion and Results

This project successfully implements a basic but robust autonomous control system. The resulting robot is able to navigate a confined space, dynamically sensing and reacting to obstacles by prioritizing clear paths (left or right) and utilizing a reliable reversing and re-scanning fallback when trapped. The system effectively validates the successful integration of digital sensors, mechanical actuation, and embedded programming logic into a single, functional robotic platform.