AUTONOMOUS ROBOT WITH OBSTACLE AVOIDANCE AND AUTO PARKING



Embedded Systems

Dr. Mohamed Al-Habrouk

Team Members

|  |  |
| --- | --- |
| Name | ID |
| Basmala Akram | 2103159 |
| Bassant Mohamed | 20221376715 |
| Henar ElSayed | 2103131 |
| Noha Nael | 2103130 |
| Nureen Ehab | 20221465124 |
| Zeinab Mohamed | 20221310251 |

Table of content

Objective 2

Problem Description2

System Design3

Solution Implementation4

1. Chosen Algorithms4
2. Implementation Details5

Flowchart 11

Cost Analysis 12

**Objective**

Design and implement an autonomous robot capable of navigating its environment, avoiding obstacles, and autonomously parking itself.

**First: Problem Description**

Challenge: Navigating complex environments safely while autonomously parking vehicles poses significant challenges.

* Obstacle Avoidance: Requires real-time detection, decision-making, and path planning to avoid collisions with stationary and dynamic objects.
* Autonomous Parking: Involves precise maneuvering within confined spaces, often with limited sensor information.

Problem Statement : The objective of this project is to design and implement an embedded system that enables a car to detect and avoid obstacles effectively while performing autonomous parking maneuvers. This solution aims to enhance vehicle safety, convenience, and efficiency.

**Second: System Design**

Hardware Components:

Microcontroller/Processor: We used Arduino Uno: The brain of the system, responsible for sensor data processing, decision-making, and control algorithms.

Ultrasonic Sensors: We used one ultrasonic sensor to measure distances to avoid obstacles.

Motor Drive: The electronic device that controls the power supplied to the DC motor, ensuring it follows the commands from the control system based on feedback.

Servo Motor: Controls the steering mechanism of any other movable parts, and it adjusts the direction of the ultrasonic sensor for better obstacle detection.

Wheels: We used 4 rubber wheels , Each wheel is attached to a DC motor.

DC Motor (4): They Provide the driving force for the robot's wheels, allowing it to move forward, backward, and turn.

Power Supply: We used a battery pack with 2 batteries, 3.7V each , it supplies power to the Arduino, motors, and other electronic components.

Robot Platform: We used a 26 x 15 cm robot platform to provide a stable structure to mount all the components and support the overall integrity of the robot.

**Third: Solution Implementation**

***Chosen Algorithm:***

1. **Obstacle Detection and Response Algorithm:**

Measure the distance to the nearest obstacle using the ultrasonic sensor.

If an obstacle is detected within a threshold distance , the car stops.

The car moves backward for a short duration to create space.

The car stops and measures the distance to the right and left by rotating the servo.

The car turns in the direction where more space is available.

If no obstacle is detected within the threshold, the car continues moving forward.

1. **Distance Measurement Algorithm:**

-Distance Measurement using Ultrasonic Sensor:

Trigger the ultrasonic sensor to send a pulse.

Measure the time taken for the echo to return.

Convert the time into a distance measurement.

Return the measured distance. If no echo is received (distance is 0), return a default large distance (250 cm).

1. **Movement Control Algorithms**

- Gradual Speed Increase

When starting forward or backward movement, the speed of the motors is gradually increased to avoid sudden power surges that could drain the battery quickly.

Start the motors in the desired direction (forward or backward).

Gradually increase the motor speed in increments until the maximum speed is reached.

For a right turn, run the left motors forward and the right motors backward.

For a left turn, run the right motors forward and the left motors backward.

Maintain the turn for a fixed duration, then stop.

**4. Servo Positioning Algorithms**

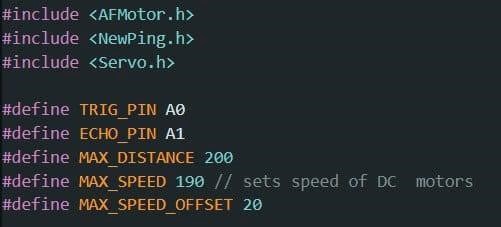
- Servo Positioning for Distance Measurement Move the servo to a predefined angle (right or left).

Wait for the servo to reach the position.

Measure the distance using the ultrasonic sensor.

Return the servo to the center position.

***Implementation Details:***



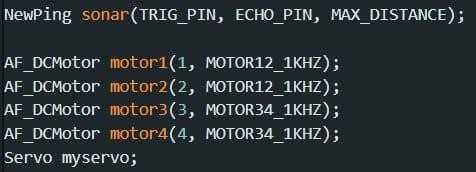
Firstly we initialize needed libraries for motor control (AFMotor), ultrasonic sensor (NewPing), and servo motor (Servo),then we define constants:

TRIG\_PIN and ECHO\_PIN for the ultrasonic sensor.

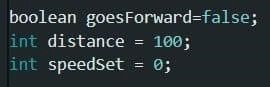
MAX\_DISTANCE for the maximum distance the ultrasonic sensor can measure.

MAX\_SPEED for the maximum speed of the motors.

MAX\_SPEED\_OFFSET for speed adjustment.



we create objects: sonar for ultrasonic distance measurements, motor1, motor2, motor3, motor4 for controlling our DC motors, myservo for controlling the servo motor.



Declaring some variables: goesForward to keep track of the car's movement direction, distance to store the distance measured by the ultrasonic sensor,speedSet for setting the speed of the motors.



we are intializing the car where the servo is given to pin 10,moving the servo to position 115 ,waits for 2 seconds, and then Takes initial distance measurements using the readPing function to stabilize the sensor readings.



in this part we are controlling the car behavior as If an obstacle is detected within 15 cm: we stop the car, move the car backward for 300 ms, stop the car, measure distance totheright and left, turn the direction with more space. Then If no obstacle is detected, it moves forward and updates the distance measurement.



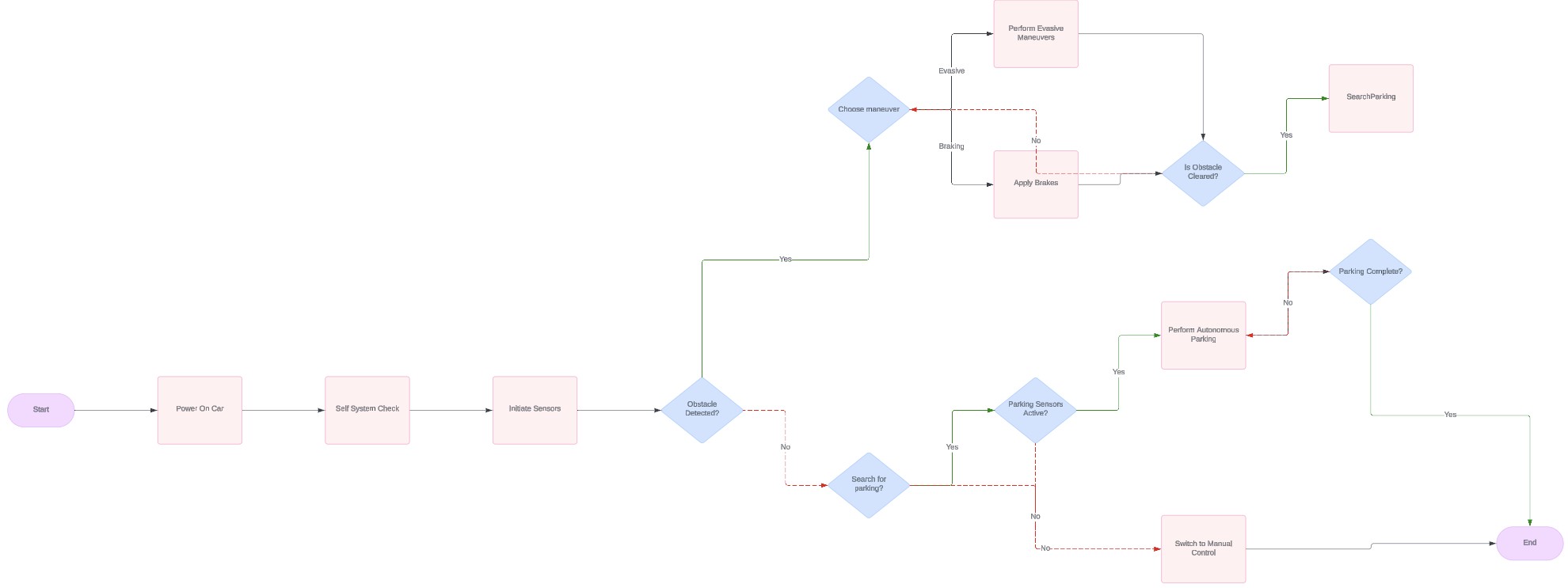
In the above part of code we define the look right and look left functions as these functions move the servo to the right or left,take a distance measurement,return the servo to the center position,and return the measured distance. We also defined the reading function that reads the distance from the ultrasonic sensor,Introduces a delay for stability,reads the distance in cm,If no reading, sets distance to 250 cm (out of range),and finally returns the distance.





In the above two images we define movement functions as follows moveStop: Stops all motors. moveForward: Moves the car forward and gradually increases speed. moveBackward: Moves the car backward and gradually increases speed. turnRight: Turns the car right by running motors in opposite directions. turnLeft: Turns the car left by running motors in opposite direction.

**Fourth: Flowchart**



**Fifth: Cost Analysis**

**Used components and their prices:**

|  |  |
| --- | --- |
| **Component** | **Price** |
| Robot platform (base of car) | 145 L.E |
| 4 DC geared motors with wheels | 55 L.E each with total of 220 L.E |
| Micro servo 180 degree | 110 L.E |
| L293 Motor driver shield | 115 L.E |
| **Total:** 590 L.E |

Reasons for Choosing Components:

1. Arduino Uno:

Arduino Uno is a widely used microcontroller board with a large community and extensive documentation, making it easy for beginners to get started. It provides sufficient processing power and I/O capabilities for controlling motors, sensors, and other peripherals.

1. Motor Drive Shield (L293):

The L293 motor driver shield is a cost-effective solution for controlling multiple DC motors with Arduino Uno.

It simplifies motor control wiring and provides protection against back EMF. 3. Ultrasonic Sensors (HC-SR04):

HC-SR04 ultrasonic sensors are chosen for their affordability and reliability. They provide accurate distance measurements, essential for obstacle detection and parking assistance.

1. Servo Motor (180 degrees):

The 180-degree servo motor offers precise angular control, ideal for steering the robot during auto parking maneuvers.

It is cost-effective compared to other servo motors with higher torque or continuous rotation capabilities.

1. DC Motors and Wheels:

These components are selected for their affordability and suitability for robot locomotion.

The DC motors provide adequate torque for driving the robot, while the wheels offer good traction and maneuverability.

1. Lithium Batteries:

Lithium batteries are lightweight, rechargeable, and offer high energy density, making them suitable for powering mobile robots.

They provide sufficient voltage and capacity to drive the motors and electronics for extended periods.

1. Miscellaneous:

Miscellaneous components include wires, connectors, chassis, and a switch button, necessary for assembling and operating the robot.

Cost-effective options are chosen to minimize the overall project cost.

Conclusion:

Choosing these components offers a balance between functionality and cost-effectiveness. By selecting affordable components without compromising essential features, you can build a capable autonomous robot within budget constraints.