Mobility Management for the Self-Driving Car

Contents

[1. Overview of Components 1](#_Toc179055200)

[2. Mobility Management System Design 1](#_Toc179055201)

[Step 1: Sensor Integration and Data Acquisition 1](#_Toc179055202)

[Step 2: Motor and Servo Control 1](#_Toc179055203)

[Step 3: Basic Movement Functions 2](#_Toc179055204)

[Step 4: Navigation and Decision Making 2](#_Toc179055205)

[Step 5: Programming the Arduino 2](#_Toc179055206)

[3. Testing and Iteration 2](#_Toc179055207)

[4. Wiring the Components 2](#_Toc179055208)

[5. Programming the Arduino 3](#_Toc179055209)

# 1. Overview of Components

* **Arduino Mega**: Acts as the brain of the self-driving car, processing data from sensors and controlling the motors accordingly.
* **GY-271 (Magnetometer)**: Provides heading information, which is crucial for navigation.
* **HC-SR04 Ultrasonic Sensor:** Provides navigation and obstacle detectionabilities.
* **L298N Motor Driver**: Controls the speed and direction of the car's motors.
* **Servo Motor**: Steers the front wheels to navigate the car.

# 2. Mobility Management System Design

## Step 1: Sensor Integration and Data Acquisition

* **Integrate the GY-271**: We connected the GY-271 to the Arduino Mega via I2C communication (using the Arduino SDA and SCL pins). We used the magnetometer to determine the car's current heading, which is essential for making navigation decisions.

## Step 2: Motor and Servo Control

* **L298N Motor Driver**: We interfaced the L298N motor driver with the Arduino Mega to control the speed and direction of the car's motors. The input pins of the L298N were connected to the digital pins on the Arduino and the motor to the output pins of the L298N.
* **Servo Motor for Steering**: We connected the servo motor to one of the PWM (Pulse Width Modulation) pins on the Arduino Mega. The servo will control the steering mechanism for the car.

## Step 3: Basic Movement Functions

* **Forward, Reverse, Stop**: We implemented functions to control the car's basic movements by manipulating the L298N's inputs from the Arduino. For example, to move forward, we ensure both motors spin in the same direction.
* **Steering**: We used the servo motor to steer. A function was taken that takes angle as input and adjusts the servo's position accordingly, allowing the car to turn.

## Step 4: Navigation and Decision Making

* **HC-SR04 Ultrasonic Sensor**: We used 3 ultrasonic sensors mounted on the front of the car to detect obstacles. These were connected to the trigger and echo pins to the Arduino Mega.
* **Implementing Compass Navigation**: We used the heading data from the GY-271 to navigate. For instance, if the car needs to head north and the current heading is east, the system will calculate the required turn to head in the right direction.
* **Basic Obstacle Avoidance**: If an obstacle is detected within a predefined safe distance, stop the motors, or adjust the direction to avoid the obstacle.
* **Advanced Navigation**: Combine distance measurements with sensor data from a GY-271 compass module follow a specific route and to avoid obstacle.
* **Simple Decision Making**: Our decision-making logic was simple, such as turning in a specific direction upon reaching a certain heading or stopping/moving around if an obstacle is detected.

## Step 5: Programming the Arduino

* **Code Structure**: We began by initializing all the components and setting up the necessary libraries for the GY-271, HC-SR04 and servo motor. Then, in the main loop, we continuously read the compass heading, decide on the movement based on the heading, and control the motors and servo accordingly.
* **Libraries and Functions**: We utilized the existing Arduino libraries for the GY-271 (such as Wire.h for I2C communication) and servo motor (Servo.h) and wrote specific functions for motor control that interact with the L298N.

# 3. Testing and Iteration

* **Initial Testing**: We started with simple straight-line movements and turns to ensure the car can move forward, reverse, and stop as expected.
* **Navigation Testing**: We tested the compass-based and ultrasonic sensor-based navigation by setting a target direction and having the car adjust its heading to match.
* **Iterate**: Based on the testing results, we refined the control algorithms, especially how the car decides to turn and adjust its speed.

# 4. Wiring the Components

1. **HC-SR04 Ultrasonic Sensor**:
   * VCC to 5V on the Arduino.
   * GND to GND.
   * TRIG to a digital pin (pin 2).
   * ECHO to another digital pin (pin 3).
2. **GY-271 Compass Module**:
   * VCC to 3.3V on the Arduino.
   * GND to GND.
   * SCL to SCL (pin 21 on Mega).
   * SDA to SDA (pin 20 on Mega).
3. **L298N Motor Driver**:
   * Motor outputs to your motors.
   * Input pins (IN1, IN2, IN3, IN4) to digital pins on the Arduino (pins 4, 5, 6, 7).
   * ENA and ENB to PWM-capable pins for speed control (pins 9 and 10).
   * VCC to battery pack +, GND to battery pack - and Arduino GND.
4. **Servo Motor**:
   * Control wire to a digital pin (pin 11).
   * VCC to 5V on the Arduino.
   * GND to GND.

# 5. Programming the Arduino

1. **Initialize Components:**
   * Include the necessary libraries (Servo.h for the servo, Wire.h for I2C communication, and any library needed for the GY-271).
   * Define pin numbers and initialize variables.
2. **Reading from the HC-SR04:**
   * Use the pulseIn() function to measure the duration of the echo from the HC-SR04, and calculate the distance to the nearest obstacle.
3. **Reading from the GY-271:**
   * Initialize the GY-271 and read the heading information. You might need to calibrate the compass for accurate readings.
4. **Controlling the Motors with the L298N:**
   * Write functions to control the motors' speed and direction based on the inputs from the sensors.
5. **Controlling the Servo Motor:**
   * Use the Servo.write() function to set the steering angle.
6. **Implementing Logic for Obstacle Avoidance and Navigation:**
   * Combine the sensor readings to make decisions about the car's movement. For example, if an obstacle is detected within a certain distance, stop or steer away from the obstacle.
   * Use the compass module to navigate in a desired direction.