Design and Implementation of a Line Following and Obstacle Avoidance Robot

Enea Robaj, Felix Diekenbrock, Moaz Bin Alamgir Chowdhury
*Systems Engineering and Prototyping, BSc Electronic Engineering
Hochschule Hamm-Lippstadt, Lippstadt, Germany
{enea.robaj, felix.diekenbrock, moaz-bin-alamgir.chowdhury}@stud.hshl.de

***Abstract*—This paper presents the following.**
***Index Terms*—Line follower robot,**

I. Introduction

This is the final report on the autonomous car project of **Team B3**. It presents all relevant aspects of the system, including hardware components, circuit design, programming, and Git collaboration.

II. System Design and Engineering

Our system uses a motor controller to drive two DC motors for mobility. Line-following is enabled by two IR sensors, while obstacle detection is handled by an ultrasonic sensor mounted on a servo motor. In addition, a color sensor identifies the color of the detected obstacles to determine which special maneuver should be performed. The system is powered by a LiPo battery and controlled by an Arduino UNO. A pre-made chassis was provided to us, and 3D-printed components designed in *SolidWorks* were used to mount and connect physical parts.

III. Prototyping and Design

A. Electronic Components

- Arduino UNO
- 11.1V LiPo battery
- SBC-MotoDriver2
- ”SEN-Color” Color sensor
- ”SEN-KY033LT” IR sensor
- ”SG-90” Servo motor
- ”HC-SR04” Ultrasonic sensor.

B. Virtual and Physical Prototypes

Tinkercad was used to design the virtual circuit prototype. For the physical connections, components were mounted using custom 3D-printed holders designed in *SolidWorks*. These include:

- Battery holder
- Breadboard holder
- IR sensor holder
- Motor driver holder
- DC Motor holder
- Ultrasonic holder
- Switch holder

After receiving the servo motor, we realized that our ultrasonic sensor holder was incompatible with it, so we decided to use the design provided by the professor. During assembly, we encountered minor dimensional issues with some 3D-printed parts, but these were quickly corrected.

C. Hardware Assembly

All components were mounted onto a pre-made chassis. The DC motors were attached using 3D-printed motor holders. The motor driver, breadboard, and battery were secured with their respective holders. Due to space constraints, the breadboard was placed above the battery, with the Arduino UNO and motor driver positioned in front. A switch for the battery was placed next to the battery. The IR sensors were fixed underneath the chassis. The ultrasonic sensor was mounted on the servo motor and placed at the front of the car. All components were connected using jumper wires on the breadboard, with the Arduino UNO serving as the central controller.

IV. Circuit Design

Due to some components missing in Tinkercad, we had to improvise in some areas. For more detail, please visit our github repository.

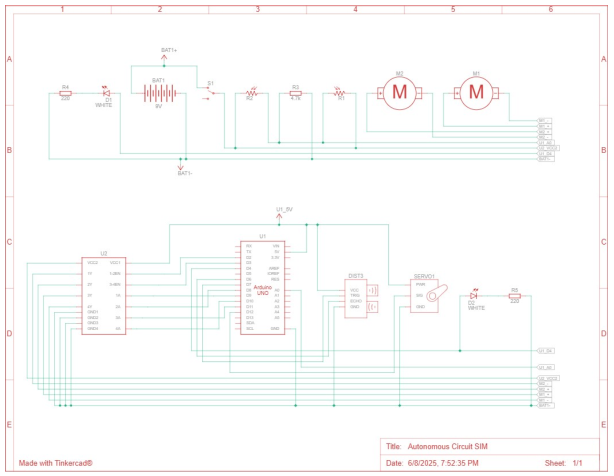

Fig. 1. Complete circuit layout of the robot

V. Software Implementation

A. Initialization of Sensors and Actuators

Pins for sensors were defined and initialized in the ‘setup()‘ function. Below is a snippet:

Listing 1. Sensor Initialization Code

```
void setup() {
 // IR Sensors
#define IR_SENSOR_RIGHT A0
#define IR_SENSOR_LEFT A1

// Motors
int enableRightMotor = 2;
int rightMotorPin1 = 7;
int rightMotorPin2 = 5;
int enableLeftMotor = 3;
int leftMotorPin1 = 9;
int leftMotorPin2 = 8;

// Ultrasonic Sensor
#define trigPin 13
#define echoPin 12

// Servo Motor
#define servoPin 11

// Color Sensor
#define S0 A2
#define S1 A3
#define S2 A4
#define S3 A5
#define sensorOut 10

}
```

B. Movement Control Algorithms

Below is a snippet showing line-following behavior using IR sensors:

Listing 2. Sensor Initialization Code

```
int rightIRSensorValue =
    digitalRead(IR_SENSOR_RIGHT);
int leftIRSensorValue =
    digitalRead(IR_SENSOR_LEFT);

if (rightIRSensorValue == LOW &&
    leftIRSensorValue == LOW) {
  rotateMotor(MOTOR_SPEED, MOTOR_SPEED);  //
      Move forward
}
else if (rightIRSensorValue == HIGH &&
    leftIRSensorValue == LOW) {
  rotateMotor(-MOTOR_SPEED, MOTOR_SPEED);  //
      Turn right
}
else if (rightIRSensorValue == LOW &&
    leftIRSensorValue == HIGH) {
  rotateMotor(MOTOR_SPEED, -MOTOR_SPEED);  //
      Turn left
}
else {
  rotateMotor(0, 0);  // Stop
}
```

C. Behavioral Decision Logic

Behavioral decisions were made based on sensor input (distance, color, line). Below is a snippet showing obstacle and color-based logic:

Listing 3. Sensor Initialization Code

```
if (distance > 0 && distance <= 10) {
  rotateMotor(0, 0);  // Stop for obstacle

  if (!ambientChecked) {
    // Measure ambient red
    // (Red detection loop here)
  }

  // Detect red color
  if (red) {
    Serial.println("Red␣detected");
    // Special maneuver logic here
  } else {
    Serial.println("Green␣detected");
    // Continue or different action
  }
}
```

VI. Git Usage and Collaboration

Project development was tracked using GitHub. Each member contributed by working on something for every task. Version control ensured smooth collaboration.

Team Member	**Number of Commits**
Enea Robaj	47
Felix Diekenbrock	18
Moaz Chowdhury	46
Total	**111**

TABLE I
GitHub Commits by Team Members

VII. Key Achievements

- Physical connections through custom 3D-printed parts
- Successful integration of sensors and actuators
- Reliable line-following with IR sensors
- Obstacle detection using ultrasonic sensor and servo
- Special maneuvers through color sensor
- Battery integrated with on/off switch for safe power management
- Effective teamwork through Git collaboration

Acknowledgment

We thank our professor and the assistants for their guidance and support throughout the project. Appreciation to team members for their dedication and collaboration.

References

[1] Arduino Documentation. https://www.arduino.cc/en/Guide
[2] Tinkercad Simulation. https://bit.ly/4kpkwXQ
[3] Uppaal Model Checker. http://www.uppaal.org/

Declaration of Originality

We hereby declare that this report is our own work and that we have acknowledged all sources used.