Infrastructure Inspection and Repair Robot

Technical Report

Asmaa Aboushady

October 17, 2024

Contents

1 Introduction

1.1 Project Overview

This project focuses on developing an autonomous robot to simulate the inspection and maintenance of city infrastructure, such as bridges and tunnels. The robot identifies structural faults and simulates minor repairs by illuminating targets using a mounted laser diode.

1.2 Mission Objectives

- Use a mounted laser diode to illuminate four designated targets representing structural faults or maintenance points.
- Targets must be illuminated in a specific order to simulate prioritized repair schedules.
- Ensure precise control and positioning of the robot to align the laser diode with each target accurately.

2 Design Process

2.1 Methodology

The design process involved multiple stages, including assembling the robot platform, integrating the laser diode, and developing control algorithms for movement and laser activation. The robot is controlled using an Arduino board, and a Python-based GUI is developed for user interaction.

2.2 Decision-Making Process

The use of Arduino was decided based on its compatibility with motor drivers and ease of serial communication. Python was chosen for its simplicity and versatility in creating both a control script and a graphical user interface (GUI).

3 Technical Details

3.1 Components

The robot consists of the following components:

- **Arduino Uno**: Microcontroller used to control the motors and the laser diode.
- **L298N Motor Driver**: Drives the motors based on signals from the Arduino.
- **Laser Diode**: Simulates the repair action by illuminating the targets.

3.2 Tkinter GUI Code

To provide a graphical interface for controlling the robot, we used the Tkinter library. The following code creates a GUI for issuing movement commands and controlling the laser diode:

```
import tkinter as tk
import serial
import time
import keyboard
import threading

# Initialize serial connection
ser = serial.Serial('COM6', 9600)

# Robot control functions
def move_forward():
    ser.write(b'w')
    update_status("Moving Forward")

def move_backward():
    ser.write(b's')
    update_status("Moving Backward")

def turn_left():
    ser.write(b'd')
    update_status("Turning Left")

def turn_right():
    ser.write(b'a')
    update_status("Turning Right")

def stop():
    ser.write(b'm')
    update_status("Stopped")

def turn_on_laser():
    ser.write(b'u')
    update_status("Laser On")

def turn_off_laser():
    ser.write(b'p')
    update_status("Laser Off")

def update_status(action):
    status_label.config(text=f"Status: {action}")

# Function to control robot using keyboard input
def control_robot_with_keyboard():
    while True:
        if keyboard.is_pressed('w'):
            move_forward()
        elif keyboard.is_pressed('s'):
            move_backward()
        elif keyboard.is_pressed('d'):
            turn_left()
        elif keyboard.is_pressed('a'):
            turn_right()
```

```
        elif keyboard.is_pressed('u'):
            turn_on_laser()
        elif keyboard.is_pressed('p'):
            turn_off_laser()
        elif keyboard.is_pressed('space'):
            stop()
        time.sleep(0.1)  # Delay to prevent CPU overload

# Create the main window
window = tk.Tk()
window.title("Robot Controller")

# Create buttons
forward_button = tk.Button(window, text="Move Forward", command=
    move_forward, width=15, height=2)
backward_button = tk.Button(window, text="Move Backward", command=
    move_backward, width=15, height=2)
left_button = tk.Button(window, text="Turn Left", command=turn_left,
    width=15, height=2)
right_button = tk.Button(window, text="Turn Right", command=turn_right,
     width=15, height=2)
stop_button = tk.Button(window, text="Stop", command=stop, width=15,
    height=2)
laser_on_button = tk.Button(window, text="Laser On", command=
    turn_on_laser, width=15, height=2)
laser_off_button = tk.Button(window, text="Laser Off", command=
    turn_off_laser, width=15, height=2)

# Status display
status_label = tk.Label(window, text="Status: ", font=('Arial', 14))

# Arrange buttons on grid
forward_button.grid(row=0, column=1, padx=5, pady=5)
backward_button.grid(row=2, column=1, padx=5, pady=5)
left_button.grid(row=1, column=0, padx=5, pady=5)
right_button.grid(row=1, column=2, padx=5, pady=5)
stop_button.grid(row=1, column=1, padx=5, pady=5)
laser_on_button.grid(row=3, column=0, padx=5, pady=5)
laser_off_button.grid(row=3, column=2, padx=5, pady=5);

# Status label positioning
status_label.grid(row=4, column=0, columnspan=3, pady=10)

# Start a thread to control the robot using keyboard input
keyboard_thread = threading.Thread(target=control_robot_with_keyboard,
    daemon=True)
keyboard_thread.start()

# Run the application
window.mainloop()
```

Listing 1: Tkinter GUI script

3.3 Arduino Code

The Arduino script processes the serial commands received from the Python script to control the motors and laser diode. Below is the updated Arduino code:

```
int IN1 = 2;
int IN2 = 8;
int IN3 = 12;
int IN4 = 13;
int EnPin1 = 9;
int EnPin2 = 10;
int laserPin = 11; // Updated laser pin

void setup() {
    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);
    pinMode(EnPin1, OUTPUT);
    pinMode(EnPin2, OUTPUT);
    pinMode(laserPin, OUTPUT); // Set laser pin as output
    Serial.begin(9600);
}

void loop() {
    if (Serial.available()) {
        char command = Serial.read();

        switch (command) {
            case 'w': // Move Forward
                digitalWrite(IN1, HIGH);
                digitalWrite(IN3, HIGH);
                break;
            case 's': // Move Backward
                digitalWrite(IN2, HIGH);
                digitalWrite(IN4, HIGH);
                break;
            case 'd': // Turn Left
                digitalWrite(IN2, HIGH);
                digitalWrite(IN3, HIGH);
                break;
            case 'a': // Turn Right
                digitalWrite(IN1, HIGH);
                digitalWrite(IN4, HIGH);
                break;
            case 'm': // Stop
                digitalWrite(IN1, LOW);
                digitalWrite(IN2, LOW);
                digitalWrite(IN3, LOW);
                digitalWrite(IN4, LOW);
                break;
            case 'u': // Activate laser
                digitalWrite(laserPin, HIGH);
                break;
            case 'p': // Deactivate laser
                digitalWrite(laserPin, LOW);
                break;
        }
    }
}
```

Listing 2: Arduino script

4 Testing and Troubleshooting

4.1 Testing Strategy

The robot was tested for movement, laser alignment, and communication between the Python script and Arduino. Each test involved issuing commands from the keyboard or GUI to move the robot and activate the laser diode.

4.2 Troubleshooting Process

- Verified the serial connection by ensuring the correct COM port was specified in the Python script.
- Checked the wiring for the motor driver and laser diode.
- Ensured the Arduino code correctly processed the received commands.

5 Conclusion

The project successfully demonstrated the robot's ability to navigate and illuminate targets. Future work will focus on enhancing the robot's sensing capabilities for better target detection and automation.