# Technical Design Report

# * Autonomous Robotics Challenge*

## 1. Introduction

This report presents the design, development, and testing of an autonomous robotic vehicle for the Autonomous Robotics Challenge. The project involves developing a tethered robotic car capable of handling three mission tasks: Autonomous Traffic Management, Automated Delivery Route, and Infrastructure Inspection and Repair. The aim is to simulate real-world smart city challenges by improving traffic safety, optimizing delivery systems, and ensuring infrastructure maintenance.

## 2.Design Process

The design process was structured in three phases: conceptualization, implementation, and iteration. During conceptualization, each mission task was broken down to understand the key technical requirements, including the use of sensors, actuators, and computer vision algorithms.

## 2.1. Conceptual Design

**Task 1:** Autonomous Traffic Management: The robot uses computer vision to detect road signs (Speed Limit, Stop, Yield) via a webcam and responds by controlling the motors and activating a buzzer.

**Task 2:** Automated Delivery Route: A line-following algorithm ensures the robot follows a predefined black line path on a white surface using line sensors.

**Task 3:** Infrastructure Inspection and Repair: The robot is equipped with a laser diode to illuminate pre-defined structural targets, simulating a maintenance process.

## 2.2. Hardware Design

**Chassis:** A modular platform is used to house the Arduino board, motors, and sensors. It was optimized for weight (under 5 kg) and size (30 cm x 40 cm x 30 cm).

Sensors: tcr5000 3 chanel line-following sensor were integrated to guide the robot along the delivery path. For the traffic task.

**Actuators:** The robot uses a buzzer for signaling (Task 1) and a laser diode for target illumination (Task 3).

## 2.3. Control Systems

**Motor Control:** The L298N motor driver board was used to drive the motors, allowing precise control over the robot's movements.

**Algorithm Development:**

**Task 1:** a tenserflow model were made from scratch and implemented for real-time road sign detection.

**Task 2:** a proportional-integral-derivative (PID) control algorithm was developed to ensure smooth navigation of the path.

**Task 3:** precise motor control was used to align the laser with the targets.

## 3. Technical Specifications

**Dimensions:** The robot adhered to competition constraints of 30 cm x 40 cm x 30 cm.

**Power Supply:** The robot operated using an external power source through a 5-meter tether.

**Components Used:**

**Arduino:** Primary microcontroller for coordinating sensor and motor inputs.

**Line Sensors:** For path-following in Task 2.

**Webcam:** For computer vision in Task 1.

**Laser Diode:** For target illumination in Task 3.

**Buzzer:** For auditory signals in Task 1.

## 4. Testing and Troubleshooting

Testing was conducted in stages for each task:

**Task 1 (Traffic Management):** The vision system was tested under different lighting conditions to ensure reliable sign detection. Adjustments were made to improve detection speed and accuracy.

**Task 2 (Delivery Route):** Path-following was tested using different route configurations. The PID controller was tuned to minimize deviations and reduce oscillation.

**Task 3 (Inspection and Repair):** The laser system was tested for accuracy in hitting the designated targets, with adjustments made for precision alignment.

## 5.tasks code overview:

### Task1:

ile Edit Sketch Tools Help

Arduino Uno

task.ino

```cpp
int slowSpeedValue;     // Speed for 30 Speed Limit sign

// Setup function to initialize pins and serial communication
void setup() {
    // Set motor control pins as outputs
    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);

    // Set enable (PWM) pins as outputs
    pinMode(EnPin1, OUTPUT);
    pinMode(EnPin2, OUTPUT);

    // Set buzzer pin as output
    pinMode(buzzerPin, OUTPUT);

    // Set initial motor speed (PWM values)
    analogWrite(EnPin1, speedValue);  // Right motor initial speed
    analogWrite(EnPin2, speedValue);  // Left motor initial speed

    // Start serial communication
    Serial.begin(9600);

    // Set slow speed to 30% of maximum speed
    slowSpeedValue = speedValue * 0.3;
}

// Main loop to control motors and buzzer based on detected signs
void loop() {
    // Always move forward by default
    moveForward();

```

Output

ile Edit Sketch Tools Help

Arduino Uno

task.ino

```cpp
    // Check if there is any data available on the serial port
    if (Serial.available()) {
        // Read the incoming command
        char command = Serial.read();

        // Perform action based on the received command
        switch (command) {
            case 's':  // Stop motors
                stopAndMoveBackward();
                break;

            case 'u':  // Turn buzzer on
                digitalWrite(buzzerPin, HIGH);  // Buzzer on
                break;

            case 'p':  // Turn buzzer off
                digitalWrite(buzzerPin, LOW);  // Buzzer off
                break;

            case 'r':  // Yield sign detected
                digitalWrite(buzzerPin, HIGH);  // Buzzer on for Yield
                // Logic to yield or pause can be added here
                break;

            case 'o':  // Stop sign detected
                digitalWrite(buzzerPin, HIGH);  // Buzzer on for Stop
                stopAndMoveBackward();
                break;

            case 't':  // 30 Speed Limit sign detected
                analogWrite(EnPin1, slowSpeedValue);  // Set right motor to slow speed
                analogWrite(EnPin2, slowSpeedValue);  // Set left motor to slow speed
                break;
```

Output

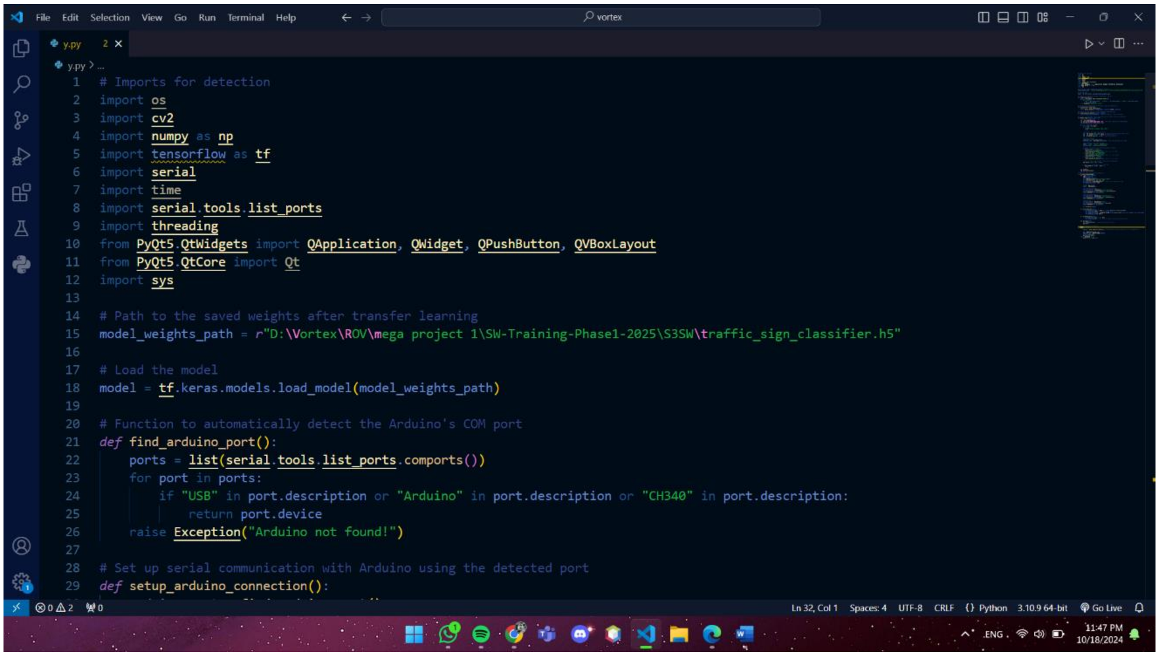
File Edit Sketch Tools Help

Arduino Uno

task.ino

```cpp
void moveForward() {
    digitalWrite(IN1, HIGH);  // Right motor forward
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);  // Left motor forward
    digitalWrite(IN4, LOW);
    analogWrite(EnPin1, speedValue);  // Set right motor speed
    analogWrite(EnPin2, speedValue);  // Set left motor speed
}

// Function to stop motors and move backward for 1 second
void stopAndMoveBackward() {
    stopMotors(); // Stop the motors
    delay(1000);  // Wait for 1 second

    // Move backward
    digitalWrite(IN1, LOW);   // Right motor backward
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW);   // Left motor backward
    digitalWrite(IN4, HIGH);
    analogWrite(EnPin1, speedValue);  // Set right motor speed
    analogWrite(EnPin2, speedValue);  // Set left motor speed
    delay(1000);  // Move backward for 1 second

    // Stop the motors again after moving backward
    stopMotors();
}

// Function to stop motors
void stopMotors() {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
```

Output


File Edit Selection View Go Run Terminal Help
vortex
y.py
# Imports for detection
import os
import cv2
import numpy as np
import tensorflow as tf
import serial
import time
import serial.tools.list_ports
import threading
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton, QVBoxLayout
from PyQt5.QtCore import Qt
import sys
# Path to the saved weights after transfer learning
model_weights_path = r"D:\Vortex\ROV\mega project 1\SW-Training-Phase1-2025\S3SW\traffic_sign_classifier.h5"
# Load the model
model = tf.keras.models.load_model(model_weights_path)
# Function to automatically detect the Arduino's COM port
def find_arduino_port():
ports = list(serial.tools.list_ports.comports())
for port in ports:
if "USB" in port.description or "Arduino" in port.description or "CH340" in port.description:
return port.device
raise Exception("Arduino not found!")
# Set up serial communication with Arduino using the detected port
def setup_arduino_connection():
Ln 32, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.9 64-bit Go Live
ENG
11:47 PM
10/18/2024

## **Task2:**

task.ino

```
#define IN_leftA 2
#define IN_leftB 8
#define IN_rightA 12
#define IN_rightB 13

#define RightMotor 9
#define LeftMotor 10

#define LeftSensor A2
#define RightSensor A0
#define MiddleSensor A1

int left_sensor_read, right_sensor_read, middle_sensor_read;
int prev_left_sensor_read, prev_right_sensor_read, prev_middle_sensor_read;

void setup() {
  pinMode(IN_leftA, OUTPUT);
  pinMode(IN_leftB, OUTPUT);
  pinMode(IN_rightA, OUTPUT);
  pinMode(IN_rightB, OUTPUT);

  pinMode(RightMotor, OUTPUT);
  pinMode(LeftMotor, OUTPUT);

  pinMode(LeftSensor, INPUT);
  pinMode(RightSensor, INPUT);
  pinMode(MiddleSensor, INPUT);

  Serial.begin(9600);
}
```

task.ino

```
void loop() {
  // Store previous sensor readings for recovery purposes
  prev_left_sensor_read = left_sensor_read;
  prev_right_sensor_read = right_sensor_read;
  prev_middle_sensor_read = middle_sensor_read;

  // Read current sensor values
  left_sensor_read = digitalRead(LeftSensor);
  right_sensor_read = digitalRead(RightSensor);
  middle_sensor_read = digitalRead(MiddleSensor);

  int speed = 50;       // Default motor speed
  int turn_speed = 30; // Slower speed when turning

  // Forward movement when the middle sensor detects the line
  if (middle_sensor_read == HIGH && left_sensor_read == LOW && right_sensor_read == LOW) {
    digitalWrite(IN_leftA, HIGH);
    digitalWrite(IN_leftB, LOW);
    analogWrite(LeftMotor, speed);

    digitalWrite(IN_rightA, HIGH);
    digitalWrite(IN_rightB, LOW);
    analogWrite(RightMotor, speed);

  }
  // Veering slightly right when the left sensor detects the line
  else if (middle_sensor_read == LOW && left_sensor_read == HIGH && right_sensor_read == LOW) {
    digitalWrite(IN_leftA, HIGH);
    digitalWrite(IN_leftB, LOW);
    analogWrite(LeftMotor, speed);

    digitalWrite(IN_rightA, HIGH);
    digitalWrite(IN_rightB, LOW);
    analogWrite(RightMotor, turn_speed);

  }
  // Veering slightly left when the right sensor detects the line
  else if (middle_sensor_read == LOW && left_sensor_read == LOW && right_sensor_read == HIGH) {
    digitalWrite(IN_leftA, HIGH);
    digitalWrite(IN_leftB, LOW);
    analogWrite(LeftMotor, turn_speed);

    digitalWrite(IN_rightA, HIGH);
    digitalWrite(IN_rightB, LOW);
    analogWrite(RightMotor, speed);

  }
  // 90-degree left turn when both middle and left sensors detect the line
  else if (middle_sensor_read == HIGH && left_sensor_read == HIGH && right_sensor_read == LOW) {
    digitalWrite(IN_leftA, LOW);  // Left motor backward
```

## **Task3:**

dit Sketch Tools Help

Arduino Uno

task.ino

```
#define IN_leftA 2
#define IN_leftB 8
#define IN_rightA 12
#define IN_rightB 13
#define pin_4 7
#define RightMotor 9
#define LeftMotor 10

#define LeftSensor A2
#define RightSensor A0
#define MiddleSensor A1
int left_sensor_read, right_sensor_read, middle_sensor_read;
int prev_left_sensor_read, prev_right_sensor_read, prev_middle_sensor_read;


void setup() {
  pinMode(IN_leftA,OUTPUT);
  pinMode(IN_leftB,OUTPUT);
  pinMode(IN_rightA,OUTPUT);
  pinMode(IN_rightB,OUTPUT);
  pinMode(pin_4, OUTPUT);

  pinMode(RightMotor,OUTPUT);
  pinMode(LeftMotor,OUTPUT);

  pinMode(LeftSensor,INPUT);
  pinMode(RightSensor,INPUT);
  pinMode(MiddleSensor,INPUT);

  Serial.begin(9600);

}

```

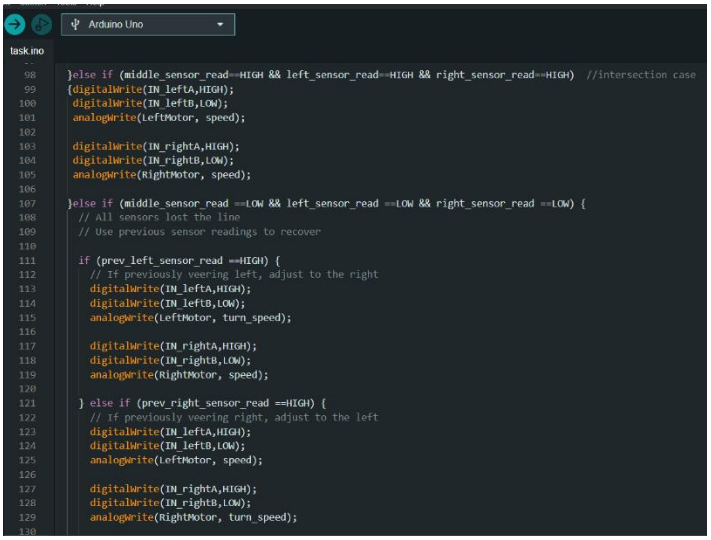
dit Sketch Tools Help

Arduino Uno

task.ino

```
void loop() {
  prev_left_sensor_read = left_sensor_read;
  prev_right_sensor_read = right_sensor_read;
  prev_middle_sensor_read = middle_sensor_read;

  int left_sensor_read   =digitalRead(LeftSensor);
  int right_sensor_read  =digitalRead(RightSensor);
  int middle_sensor_read =digitalRead(MiddleSensor);
  int speed       = 60;
  int turn_speed = 30;
  //int full_speed =255;

  if (middle_sensor_read==HIGH && left_sensor_read==LOW && right_sensor_read==LOW) //moving forward
{  digitalWrite(IN_leftA,HIGH);
   digitalWrite(IN_leftB,LOW);
   analogWrite(LeftMotor, speed);

   digitalWrite(IN_rightA,HIGH);
   digitalWrite(IN_rightB,LOW);
   analogWrite(RightMotor, speed);

  }else if (middle_sensor_read==LOW && left_sensor_read==HIGH && right_sensor_read==LOW)  //veering slightly right
  {digitalWrite(IN_leftA,HIGH);
   digitalWrite(IN_leftB,LOW);
   analogWrite(LeftMotor, speed);
   delay(50);


   digitalWrite(IN_rightA,HIGH);
   digitalWrite(IN_rightB,LOW);
   analogWrite(RightMotor, turn_speed);

```

```

  }else if (middle_sensor_read==LOW && left_sensor_read==LOW && right_sensor_read==HIGH)  //veering slightly left
   {digitalWrite(IN_leftA,HIGH);
    digitalWrite(IN_leftB,LOW);
    analogWrite(LeftMotor, turn_speed);

   digitalWrite(IN_rightA,HIGH);
   digitalWrite(IN_rightB,LOW);
   analogWrite(RightMotor, speed);
   delay(50);

  }else if (middle_sensor_read==HIGH && left_sensor_read==HIGH && right_sensor_read==LOW)  //90 degree turn to the left
  {
    digitalWrite(IN_leftA,LOW);
   digitalWrite(IN_leftB,HIGH);
   analogWrite(LeftMotor, 20);

   digitalWrite(IN_rightA,HIGH);
   digitalWrite(IN_rightB,LOW);
   analogWrite(RightMotor, speed);
   delay(50);

  }else if (middle_sensor_read==HIGH && left_sensor_read==LOW && right_sensor_read==HIGH)  //90 degree turn to the right
   {
   digitalWrite(IN_leftA,HIGH);
   digitalWrite(IN_leftB,LOW);
   analogWrite(LeftMotor, speed);

   digitalWrite(IN_rightA,LOW);
   digitalWrite(IN_rightB,HIGH);
   analogWrite(RightMotor, 20);
   delay(50);

```



```
Arduino Uno
task.ino
}else if (middle_sensor_read==HIGH && left_sensor_read==HIGH && right_sensor_read==HIGH)  //intersection case
{digitalWrite(IN_leftA,HIGH);
 digitalWrite(IN_leftB,LOW);
 analogWrite(LeftMotor, speed);

 digitalWrite(IN_rightA,HIGH);
 digitalWrite(IN_rightB,LOW);
 analogWrite(RightMotor, speed);

}else if (middle_sensor_read ==LOW && left_sensor_read ==LOW && right_sensor_read ==LOW) {
  // All sensors lost the line
  // Use previous sensor readings to recover

  if (prev_left_sensor_read ==HIGH) {
    // If previously veering left, adjust to the right
    digitalWrite(IN_leftA,HIGH);
    digitalWrite(IN_leftB,LOW);
    analogWrite(LeftMotor, turn_speed);

    digitalWrite(IN_rightA,HIGH);
    digitalWrite(IN_rightB,LOW);
    analogWrite(RightMotor, speed);

  } else if (prev_right_sensor_read ==HIGH) {
    // If previously veering right, adjust to the left
    digitalWrite(IN_leftA,HIGH);
    digitalWrite(IN_leftB,LOW);
    analogWrite(LeftMotor, speed);

    digitalWrite(IN_rightA,HIGH);
    digitalWrite(IN_rightB,LOW);
    analogWrite(RightMotor, turn_speed);
```

## 6 . Conclusion

The project successfully demonstrated the capability of an autonomous robotic vehicle to perform smart city-related tasks. The robot's design met all technical specifications, and the algorithms were fine-tuned to optimize task performance. Future enhancements could involve integrating more advanced sensors and improving obstacle avoidance for the delivery route.