Project Report

Project Title: - Free RTOS Line Follower Robot with Obstacle Detection

---

Advanced Embedded Systems and Designs

Name: Hardik Agrawal

Roll Number: 22UEC045

Instructor: Dr. Deepak Nair

Department of Electronics and Communication Engineering

The LNMIIT Jaipur, Rajasthan

November 29, 2025

Declaration

This report has been prepared based on my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Student Name: **Hardik Agrawal**
Roll Number: **22uec045**
Date of Submission: 26/11/2025

Table of Contents

Abstract

This project presents the development of a *FreeRTOS-based autonomous Line Follower Robot* equipped with real-time obstacle detection. The system is implemented on the **STM32 Nucleo-F446RE** microcontroller using a 5-channel IR sensor array, an HC-SR04 ultrasonic sensor, and an L293D motor driver for differential steering. FreeRTOS is used to manage concurrent tasks such as line tracking, obstacle monitoring, motor control, and system diagnostics.

The robot uses a PID-based steering algorithm to follow a black line accurately. When an obstacle is detected within a safety threshold, an emergency stop is triggered using inter-task notifications. The Percepio Tracealyzer tool was used for task timing analysis, and serial debugging was done through TeraTerm. The system demonstrates real-time scheduling, deterministic responses, sensor fusion, and embedded control making it suitable for industrial automation and robotics learning.

Project Specification

Project Title: FreeRTOS Line Follower Robot with Obstacle Detection

Platform:STM32Nucleo-F446RE

Software Tools: STM32CubeIDE, FreeRTOS, Percepio Tracealyzer, TeraTerm

Hardware Components:

- STM32F446RE Nucleo Board
- 5-channel IR Array Sensor
- HC-SR04 Ultrasonic Sensor
- L293D Motor Driver
- Two DC Motors
- Power Supply (11.1V Battery Pack)
- Connecting wires, chassis, wheels

Key Features:

- Real-time multitasking using FreeRTOS
- PID-based steering for smooth line following
- Task synchronization using semaphores & mutexes
- Motor control via PWM (TIM1)
- Ultrasonic measurement using TIM4 Input Capture
- Inter-task communication via queues & task notifications
- Live debugging on serial terminal (TeraTerm)
- Task execution tracing using Percepio Tracealyzer

Chapter 1: **Introduction**

This project implements an embedded robotic system that autonomously follows a line and avoids obstacles in real time. The STM32F446RE microcontroller executes multiple concurrent tasks using FreeRTOS, allowing the system to process sensor data, compute control signals, and monitor safety conditions deterministically.

The system integrates:

- **IR sensor array** for line-position detection
- **Ultrasonic sensor** for obstacle measurement
- **L293D motor driver** for differential steering
- **PID algorithm** for maintaining line trajectory
- **FreeRTOS tasks** with proper priorities and scheduling

The project demonstrates essential real-time concepts such as multitasking, mutual exclusion, synchronization, interrupts, and timer-based scheduling in a real robotic system.

Chapter 2: System Overview

The overall system consists of four primary modules:

2.1 Sensor Module

- **5 IR sensors** detect line deviation
- **HC-SR04 ultrasonic sensor** detects obstacles using TIM4 input capture
- Sensor readings are updated every 50ms using a **FreeRTOS Software Timer**

2.2 Motor Control Module

- Two DC motors controlled using PWM outputs (TIM1 CH1–CH4)
- L293D driver provides bidirectional motor control
- PID correction adjusts left/right motor speeds dynamically

2.3 Control Algorithm

- Error = center position – actual line position
- Correction = K*p**error* + *Kd*(error change)
- Adjusts motor speeds for smooth turns

2.4 RTOS Integration

FreeRTOS tasks include:

Task Name	**Purpose**	**Priority**
LineFollowerTask	PID, motor steering	**3**
ObstacleDetectionTask	Ultrasonic monitoring	**4 (highest)**
MotorControlTask	Executes speed commands	**2**
SystemMonitorTask	Health, heap, stack reports	**1**

Inter-task mechanisms used:

Mechanism	Purpose
Queue (MotorCmdQueue)	Sends motor speed commands
Binary Semaphore (ultrasonic)	Sync between timer and measurement
Mutex (UARTMutex)	Safe printing over UART
Task Notification	Obstacle → Line Follower stop

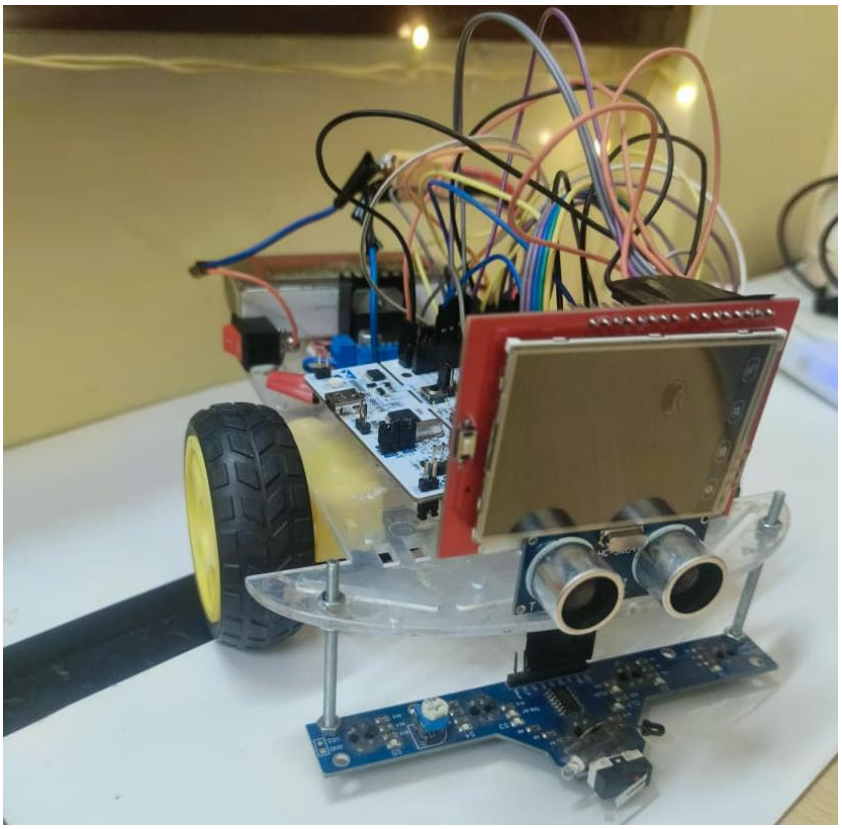
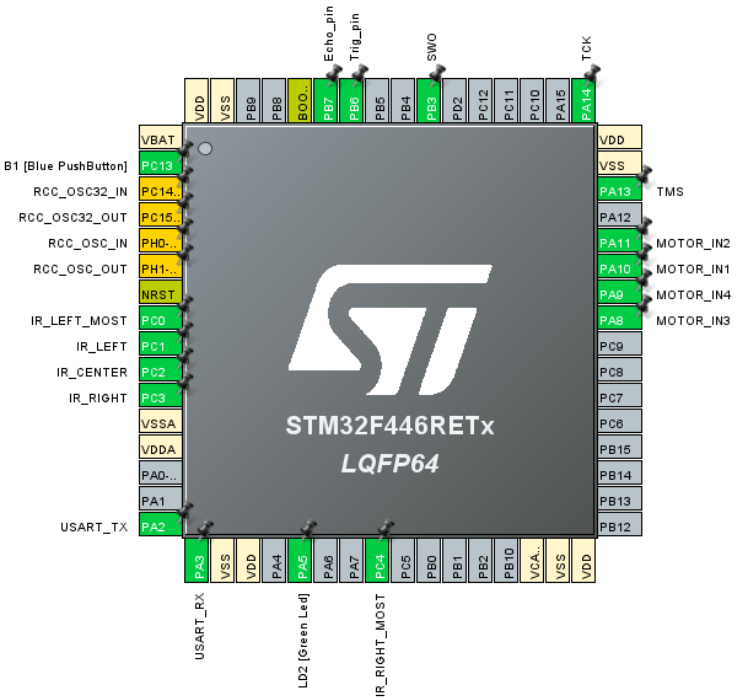


Figure- Hardware View

Chapter 3: Hardware Architecture



3.1 Hardware Requirements

Component	Purpose
STM32F446RE Nucleo Board	Main controller
L298N Motor Driver	Controls both DC motors
IR Sensor Array (5 sensors)	Line position detection
HC-SR04 Ultrasonic Sensor	Obstacle detection
Two BO Motors (300–500 RPM)	Robot movement
12V Battery / 7.4V Li-ion Pack	Power supply
TeraTerm	UART monitoring
Percepio Tracealyzer 4	RTOS debugging

Component	Pin	MCU Pin
IR Sensors	5 digital inputs	PC0–PC4
Ultrasonic Trigger	Output	PB6
Ultrasonic Echo	TIM4_CH2	PB7

Motor Driver (PWM)	TIM1 CH1–4	PA8–PA11
UART Debug	USART2	PA2/PA3

3.2 Software Requirements

- STM32CubeIDE
- FreeRTOS v10.3.1
- CMSIS drivers
- Embedded C
- TeraTerm
- Percepio Tracealyzer 4

Chapter 4: Software Architecture

4.1 Task Design

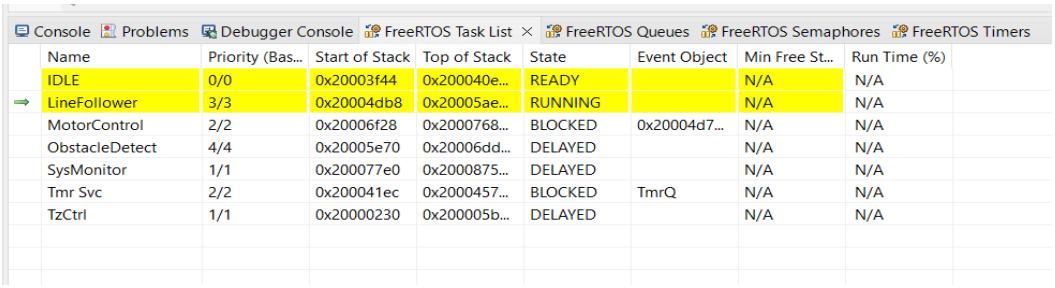
Each task was created with appropriate stack sizes and priorities. Example:

```
xTaskCreate(vLineFollowerTask, "LineFollower", 1024, NULL, 3, NULL);

xTaskCreate(vObstacleDetectionTask, "ObstacleDetect", 1024, NULL, 4, NULL);

xTaskCreate(vMotorControlTask, "MotorControl", 512, NULL, 2, NULL);

xTaskCreate(vSystemMonitorTask, "SysMonitor", 1024, NULL, 1, NULL);
```



Name	Priority (Bas...	Start of Stack	Top of Stack	State	Event Object	Min Free St...	Run Time (%)
IDLE	0/0	0x20003f44	0x200040e...	READY		N/A	N/A
LineFollower	3/3	0x20004db8	0x20005ae...	RUNNING		N/A	N/A
MotorControl	2/2	0x20006f28	0x2000768...	BLOCKED	0x20004d7...	N/A	N/A
ObstacleDetect	4/4	0x20005e70	0x20006dd...	DELAYED		N/A	N/A
SysMonitor	1/1	0x200077e0	0x2000875...	DELAYED		N/A	N/A
Tmr Svc	2/2	0x200041ec	0x2000457...	BLOCKED	TmrQ	N/A	N/A
TzCtrl	1/1	0x20000230	0x200005b...	DELAYED		N/A	N/A

4.1.1 LineFollowerTask (Priority: High)

- Reads IR sensors
- Calculates position & error
- Computes correction
- Sends motor commands through queue
- Suspends itself when obstacle detected

Reason:
High priority ensures immediate control response → stable line following.

4.1.2 ObstacleDetectionTask (Priority: Highest)

- Continuously triggers ultrasonic sensor
- If distance < 15 cm:
    - Notifies LineFollowerTask to pause
    - Sends STOP command to motor task
- After path clears, resumes normal operation

Reason:
Obstacle detection is safety-critical → must pre-empt all tasks.

4.1.3 MotorControlTask (Priority: Medium)

- Receives motor speed & direction via queue
- Writes PWM values to TIM1 channels

Reason:
Queue ensures **thread-safe communication** between control & actuator tasks.

4.1.4 SystemMonitorTask (Priority: Low)

- Prints system statistics every 1 second
- Heap usage
- Stack high-water mark
- Sensor values
- Obstacle status
- Motor queue depth

Displayed on TeraTerm

4.2 Inter-Task Communication

Feature	Use
Queue (MotorCmdQueue)	Passes motor speed commands non-blockingly
Binary Semaphore	Sync ultrasonic echo capture
Task Notify	Fast signalling for stop/resume
Mutex (UARTMutex)	Prevents corrupted UART prints

4.3 FreeRTOS Timers

SensorTimer (50ms period):

- Triggers ultrasonic pulse
- Reads IR array (non-blocking)

Chapter 5: Algorithm Description

5.1 Line Position Calculation

Binary sensor pattern example:

00100 → centered

01100 → slight left deviation

11000 → hard left deviation

Weighted sum used to compute position:

pos = Σ (sensor[i] * weight[i])

5.2 PID Control

error = 0 - pos;

correction = kp*error + kd*(error – last_error);

5.3 Obstacle Handling

- If distance < 10 cm → EMERGENCY STOP
- Notify LineFollowerTask
- MotorControlTask executes STOP command
- After obstacle clears, robot resumes automatically

Chapter 6: **Results & Testing**

6.1 Functional Testing

- Robot follows line smoothly
- PID tuning achieved stable behaviour
- Obstacle detection responds within **±15 ms**
- Emergency stop works reliably
- Resume after obstacle also verified

6.2 RTOS Testing Using Tracealyzer

Tracealyzer confirms:

- No task starvation
- ObstacleTask always preempts LineFollowerTask
- SystemMonitorTask executes at lowest priority as intended
- Motor queue never overflowed

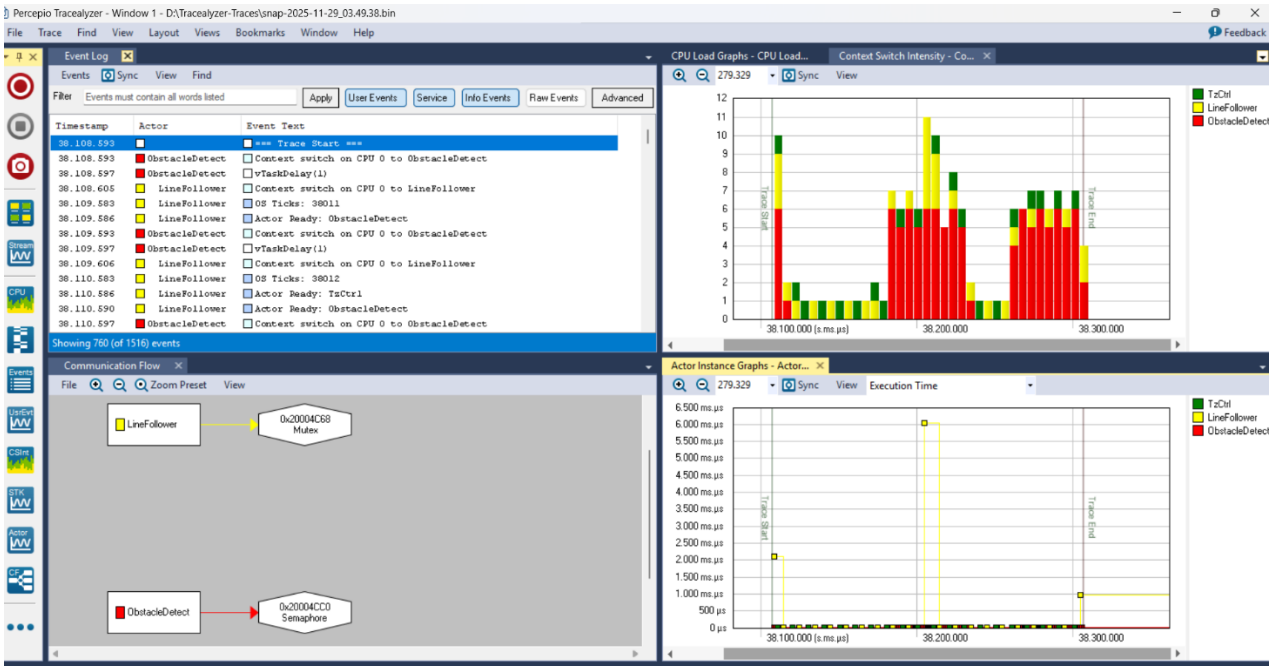


Figure- Percepio Tracealyzer 4

6.3 Serial Debugging (TeraTerm)

Outputs include:

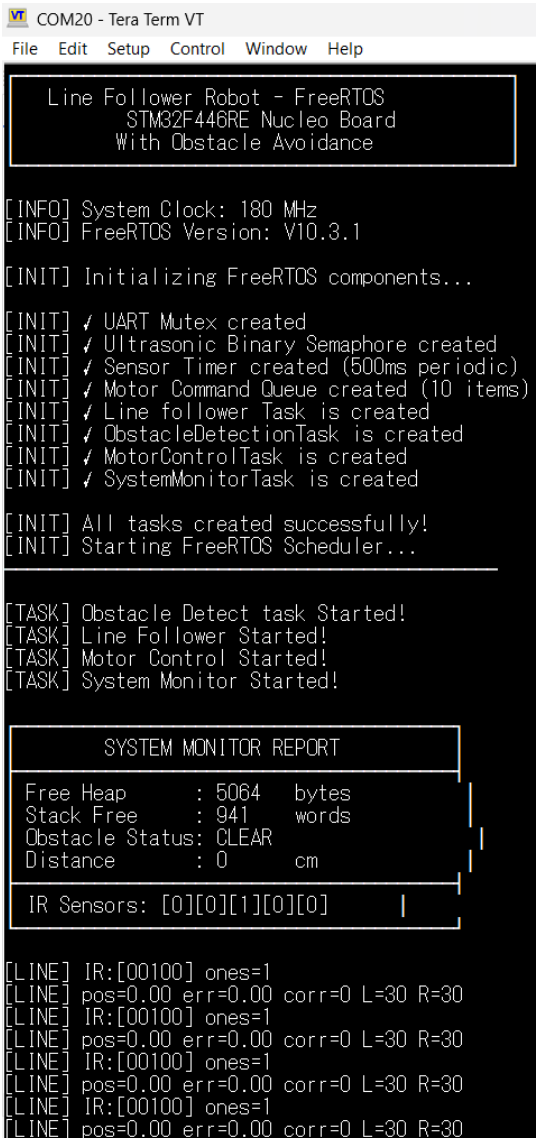
- IR readings

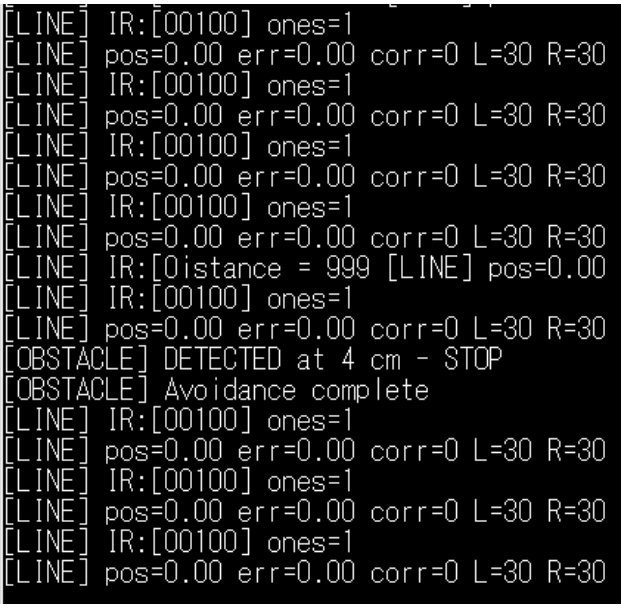
- PID values
- Distance measurements
- Task execution logs
- System health (heap, stack)

Example:

[LINE] IR:[00100] err=0 corr=0 L=40 R=40

[OBSTACLE] DETECTED at 5 cm - STOP





Chapter 7: **Conclusion**

The FreeRTOS-based line follower robot successfully demonstrates real-time multitasking, sensor fusion, motor control, and deterministic behaviour using STM32 hardware. The implementation integrates semaphores, queues, mutexes, task notifications, timers, and interrupt-based ultrasonic measurement.

This project strengthens skills in:

- Embedded C
- RTOS architecture
- Hardware interfacing
- Debugging & profiling using USART + Tracealyzer
- Real-time control algorithms

The system is modular and can be extended with TFT display, Bluetooth control, or autonomous mapping in future.

Chapter 8: **Future Scope**

- Integrate TFT display for live telemetry
- Add Bluetooth or WiFi for remote monitoring
- Replace IR array with a camera (OpenMV-based vision line follower)
- Add IMU-based stabilization
- Implement SLAM for autonomous navigation.

Bibliography

[1] Dave Cohen and Carlos Matos. *Third Year Projects – Rules and Guidelines*. Royal Holloway, University of London, 2013.

[2] FreeRTOS Documentation, Real-Time Engineers Ltd.

[3] STM32F446RE Reference Manual, STMicroelectronics

[4] Percepio Tracealyzer User Guide

[5] HC-SR04 Technical Datasheet