PROJECT REPORT ON RTOS - BASED RAILWAY CROSSING AUTOMATION SYSTEM

Carried Out at

CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING
ELECTRONIC CITY, BANGALORE.

UNDER THE SUPERVISION OF

Pranav Sangar
Embedded Software Engineer
C-DAC Bangalore

Submitted By

A. Rohith Naga Sai	250850130004
Dhanush Gowda K	250850130012
Kapil Rajgopal	250850130017
Konakalla Vivek Sri Krishna Chaitanya	250850130021

Post Graduate Diploma in Embedded System Design
(PG - DESD)
C-DAC, BANGALORE

Candidate's Declaration

We hereby certify that the work being presented in the report entitled "RTOS - BASED RAILWAY CROSSING AUTOMATION SYSTEM", in the fulfillment of the requirements for the award of Post Graduate Diploma in Embedded Systems Design and submitted in the department of Embedded System Design of the C-DAC Bangalore, is an authentic record of our work carried out during the period 21st August 2025 – 4th February 2026 under the supervision of Pranav Sangar, Embedded Software Engineer, C-DAC Bangalore.

The matter presented in the report has been submitted by me for the award of any degree of this Institute.

A. Rohith Naga Sai

A. Rohith Naga Sai

Dhanush

Dhanush Gowda K

Kapil R

Kapil Rajgopal

K.V.S.K. Chaitanya

Konakalla Vivek Sri Krishna Chaitanya

[illegible]

Counter Signed by

Pranav Sangar
Embedded Software Engineer, C-DAC Bangalore

ACKNOWLEDGMENT

I take this opportunity to express my gratitude to all those people who have been directly and indirectly with me during the competition of this project

I pay thank to “Pranav Sangar” who has given guidance and a light to me during this major project. His versatile knowledge about “RTOS - BASED RAILWAY CROSSING AUTOMATION SYSTEM” has eased me in the critical times during the span of this Final Project.

I acknowledge here out debt to those who contributed significantly to one or more steps. I take full responsibility for any remaining sins of omission and commission.

A. Rohith Naga Sai

Dhanush Gowda K

Kapil Rajgopal

Konakalla Vivek Sri Krishna Chaitanya

ABSTRACT

Railway level crossings continue to pose significant safety risks due to human error, delayed gate operation, and inadequate real-time monitoring, particularly at unmanned or semi-manned locations. Traditional manual systems often result in accidents involving road users, pedestrians, and trains, especially when gates remain open or close prematurely. This project addresses these challenges by designing and implementing an automated railway level crossing gate control system using the high-performance microcontroller on the Nucleo-STM32H753ZI development board. The system integrates two independent servo-controlled gates, infrared (IR) sensors for train arrival and exit detection, ultrasonic sensors for obstacle detection during gate closure, LED traffic light modules for visual signaling (red during closure and passing, yellow warning, green when open), and active buzzers for audible alerts. Safety is ensured by allowing each gate to pause independently at 90 degrees if an obstacle is detected within 2-15 cm, while the other gate continues its motion. The design supports smooth 10 degrees incremental movement with 100 ms steps for fast response suitable for scaled demonstrations.

Two versions were developed and tested, a bare-metal (non-RTOS) implementation using non-blocking timing, and a FreeRTOS-based with dedicated tasks for sensor monitoring, independent servo control, LED and buzzer feedback, and thread-safe UART logging. The RTOS version leverages event flags, mutexes, and a message queue to achieve true concurrency, deterministic timing, and scalable real-time performance. Hardware testing on the Nucleo board demonstrated reliable operation, gates close within approximately 1.8-2 seconds, pause correctly on obstacle detection, maintain a 3-second passing delay, and reopen after train exit detection with a 1-second safety buffer. UART logs confirmed independent gate behaviour and accurate sensor fusion. The system significantly reduces human dependency, enhances safety through proactive obstacle handling, and provides clear visual and audible feedback.

This work highlights the advantages of RTOS in embedded safety-critical applications and lays the foundation for future enhancements, including predictive component monitoring using machine learning models integrated within STM32CubeIDE.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

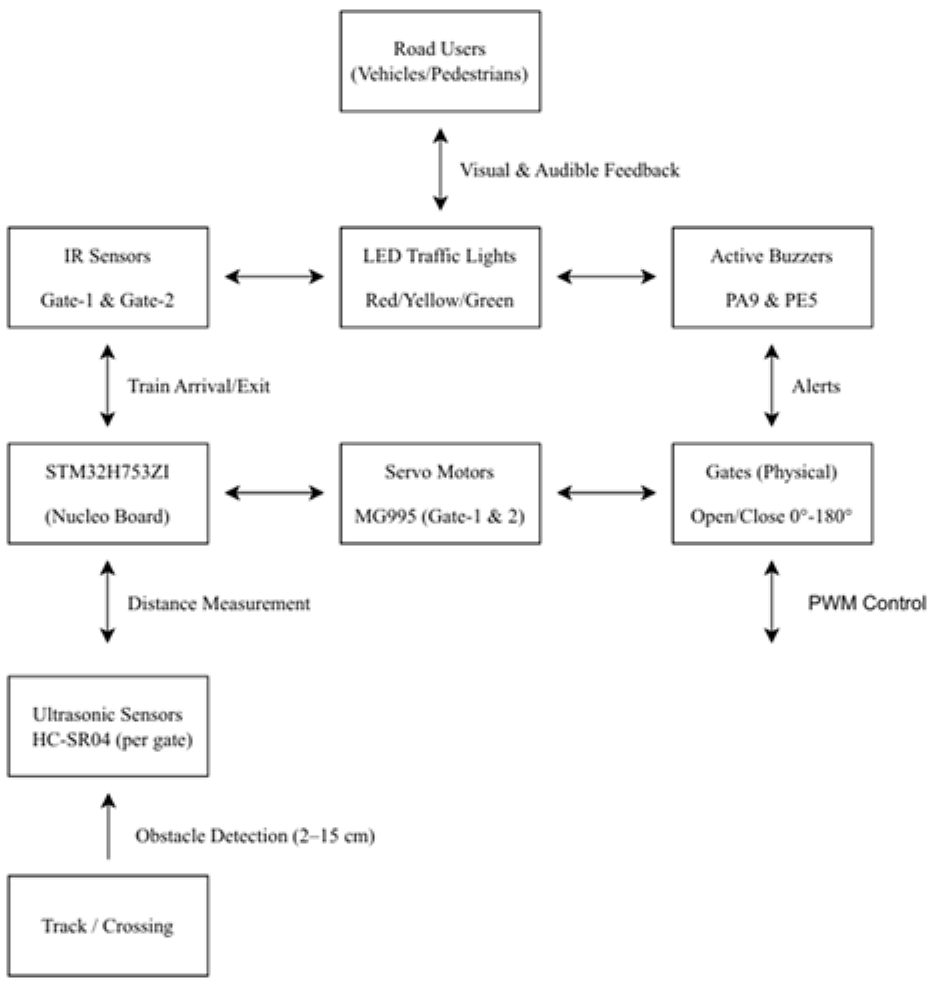
ABBREVIATIONS & ACRONYMS

ADC	Analog-to-Digital Converter
AF	Alternate Function
ARR	Auto-Reload Register
CMSIS	Cortex Microcontroller Software Interface Standard
GPIO	General Purpose Input/Output
HAL	Hardware Abstraction Layer
IC	Input Capture
IR	Infrared
MCU	Microcontroller Unit
PWM	Pulse Width Modulation
RTOS	Real-Time Operating System
SRS	Software Requirement Specification
TIM	Timer
USART	Universal Synchronous and Asynchronous Receiver Transmitter

Chapter 1

INTRODUCTION

1.1 Background



Road Users (Vehicles/Pedestrians)
Visual & Audible Feedback
IR Sensors Gate-1 & Gate-2
LED Traffic Lights Red/Yellow/Green
Active Buzzers PA9 & PE5
Train Arrival/Exit
Alerts
STM32H753ZI (Nucleo Board)
Servo Motors MG995 (Gate-1 & 2)
Gates (Physical) Open/Close 0°-180°
Distance Measurement
PWM Control
Ultrasonic Sensors HC-SR04 (per gate)
Obstacle Detection (2–15 cm)
Track / Crossing

Fig. 1.1 Block diagram of the automated railway level crossing system

Railway transportation remains one of the most efficient and widely used modes of mass transit across the globe, particularly in densely populated countries like India. However, railway level crossings-points where rail tracks intersect with roadways-continue to be among the most hazardous locations in the rail network. According to various safety reports, a substantial proportion of railway accidents occur at these crossings, often due to human error, delayed gate closure, or failure to detect obstacles such as vehicles and pedestrians trapped beneath descending gates. Manual or semi-manual gate operations exacerbate these risks, as gatekeepers may respond late or become distracted, leading to catastrophic collisions between trains and road users.

1.2 Problem Statement

The problem is particularly acute at unmanned or semi-manned crossings, where there is no dedicated personnel to monitor approaching trains or control gate mechanisms. Conventional systems relying solely on manual signaling or basic sensor triggers often lack real-time responsiveness, independent safety checks, and clear feedback to road users. In addition, the absence of obstacle detection during gate closure can result in entrapment, while inadequate visual and audible warnings increase the likelihood of violations by impatient drivers or pedestrians.

1.3 Objectives

This project aims to develop a reliable, automated railway level crossing gate control system that eliminates human dependency and enhances safety through intelligent sensor integration and real-time control. The primary objectives are:

1. To detect train arrival and exit automatically using infrared (IR) sensors placed at entrance and exit sides of the crossing.
2. To control two independent servo-operated gates with smooth incremental movement and individual obstacle detection using ultrasonic sensors during closure.
3. To implement safety pauses at 90 degrees for any gate detecting an obstacle, while allowing the unaffected gate to continue its motion.
4. To provide clear visual (traffic light signals) and audible (buzzer alerts) feedback to road users throughout the operation cycle.
5. To develop and compare two versions, a bare-metal (non-RTOS) implementation using timestamp-based non-blocking timing, and a FreeRTOS-based version

employing task concurrency, event flags, mutexes, and thread-safe logging for superior real-time performance.

6. To validate the system on hardware using the Nucleo-STM32H753ZI board, ensuring fast response suitable for scaled or toy-train demonstrations.

1.4 Scope of the Work

The scope of the work includes hardware integration of IR sensors, HC-SR04 ultrasonic modules, MG995 servo motors, LED traffic light modules, and active buzzers on the STM32H753ZI platform. Both software versions are implemented in STM32CubeIDE using HAL libraries, with the RTOS version configured via STM32CubeMX middleware. The system supports UART-based event logging for debugging and performance analysis. Testing focuses on functional correctness, independent gate behaviour, obstacle handling, and timing accuracy. Limitations include the use of a scaled prototype (toy-train setup) rather than full-scale railway hardware, and exclusion of advanced wireless communication or cloud integration at this stage.

1.5 Methodology Overview

The methodology follows a structured approach, requirements analysis, hardware selection and interfacing, bare-metal prototyping with non-blocking timing, RTOS migration using FreeRTOS CMSIS-V2, code optimization, and iterative hardware testing. The project contributes to improving safety at level crossings by demonstrating practical embedded automation with real-time capabilities, while providing a foundation for future extensions such as predictive maintenance using machine learning.

Chapter 2

LITERATURE SURVEY

2.1 Traditional Microcontroller-Based Gate Automation

Railway level crossings remain a persistent safety challenge in transportation networks, where delays in gate operation, human error, and inadequate monitoring contribute to accidents. This chapter reviews 15 key papers provided, spanning 2015 to 2025, with a focus on their alignment to the project-an automated gate system using STM32H7 with IR and ultrasonic sensors, servo motors, LEDs and buzzers, and RTOS for real-time concurrency. The papers are analysed for traditional microcontroller automation, sensor-based detection, backup and safety mechanisms, and AI and predictive enhancements. Sources from IEEE, Springer, Elsevier, and MDPI dominate, offering insights into embedded control, vibration and IR sensing, and intelligent surveillance. Each study is evaluated for contributions to gate timing, obstacle detection, and future ML and DL predictive analysis (e.g: servo monitoring), identifying gaps in independent multi-gate concurrency and STM32 and RTOS integration that this project addresses.

Early works emphasized basic automation to replace manual gates. Mansingh et al. [1] proposed an infrared detector-based system for unmanned crossings, using L-shaped sensors and transmitters to trigger gate closure. While simple and cost-effective, it lacks obstacle checks or real-time multitasking, aligning with the project's IR detection but highlighting the need for ultrasonic integration as implemented in this work. Similarly, Jesuraj et al. [2] introduced a piezoelectric sensor prototype for vibration-based train detection, wirelessly signaling gate closure in 30 seconds. This resonates with the project's focus on unmanned automation but omits independent gate pausing, which our design achieves via per-gate ultrasonic pauses.

2.2 Sensor-Integrated Safety Enhancements

Sensor-driven systems for improved reliability followed. Zawodny et al. [3] explored advanced train detection to optimize crossing closure timing, using position/speed data to reduce unnecessary delays. This supports the project's 100 ms step timing for fast closure but could benefit from our IR edge detection with debounce for accuracy. Maran et al. [4] utilized ATmega328P and PIR sensors for motion-based gate control, demonstrating low-cost installation suitable for rural areas. It mirrors our IR polling but lacks the ultrasonic safety layer, emphasizing the project's enhancement for obstacle avoidance.

2.3 RTOS-Enabled Real-Time Control

Backup and multi-mode designs addressed system failures. Kulkarni et al. [5] developed an automated gate with redundant sensors (TDIR and ODIR) and camera backup, using Raspberry Pi and SSD algorithms for robust operation. This aligns with our dual-sensor approach (IR and ultrasonic) and suggests potential for camera integration in future iterations, though our STM32 focus prioritizes embedded efficiency over Pi's processing. Gangwar et al. [6] presented a complete automation solution with track-side sensors and safety checks, reducing human supervision needs. It complements our independent gate logic but could incorporate the RTOS for better concurrency in multi-task environments.

2.4 AI and Predictive Approaches

Integrated security and risk mitigation systems emerged next. Kumar et al. [7] proposed an anti-collision framework with RFID, metal and IR sensors, servo gates, and GSM and GPS modules for alerts. This extends our project's scope by adding communication, aligning with future ML-based predictive alerts for components like servos, though our design emphasizes real-time embedded control without external modules. Kumar et al. [8] described a sensor-linked automated security system for anomaly detection and quick responses. It supports our event-driven logging but lacks the per-gate independence central to our ultrasonic pauses.

Intelligent surveillance using AI advanced monitoring capabilities. Sikora et al. [9] introduced deep learning for object and barrier detection at crossings via image processing, achieving high accuracy in unsafe scenario identification. This could enhance our ultrasonic and IR fusion with vision-based predictive analysis, particularly for servo wear monitoring via ML in STM32CubeIDE. Staino et al. [10] utilized deep learning for real-time signal recognition, enabling faster automated responses. It parallels our fast 1.8-second closure but suggests DL extensions for our project's future predictive fault detection in sensors and actuators.

Kapoor et al. [11] proposed a deep learning framework for object and track recognition in railway surveillance, reliable under varying conditions. This directly aligns with our obstacle detection needs, offering a path to embed DL models for servo prediction in STM32 environments.

Raspberry Pi-based automation provided scalable alternatives. Sabarish et al. [12] designed a Pi-controlled system with IR sensors and servos for gate operation, emphasizing low-cost scalability. While effective, it lacks our STM32's efficiency and RTOS concurrency for independent gates, but supports similar IR-triggered closure. Crack detection integration added structural safety. Amusan et al. [13] combined gate automation with track crack alerts using sensors, preventing derailments. This extends our ultrasonic focus to broader predictive maintenance, relevant for future ML analysis of servo/components.

Zhang et al. [14] optimized STM32 and FreeRTOS scheduling for real-time embedded systems, improving latency in multi-task setups. This directly informs our RTOS version's task priorities and event flags, addressing concurrency gaps in earlier works. Finally, Kumar et al. [15] integrated IR and roadside sensors with camera monitoring and manual override for traffic management. It complements our feedback system but emphasizes remote aspects, aligning with future IoT extensions.

2.5 Research Gaps and Project Contribution

The surveyed papers demonstrate progression from basic detection to AI-enhanced systems, yet gaps in RTOS-driven independent gate safety and embedded ML for predictive component analysis (e.g: servo motors) persist. This project fills these by leveraging STM32H7 with bare-metal and RTOS designs, paving the way for DL-based monitoring in STM32CubeIDE.

Chapter 3

SOFTWARE REQUIREMENT SPECIFICATION

The software requirement specification (SRS) defines the functional and non-functional requirements of the automated railway level crossing gate control system. It serves as the foundation for design, implementation, and testing phases, ensuring the system meets safety, reliability, and performance expectations for both bare-metal and FreeRTOS-based versions.

3.1 Functional Requirements

The system shall provide the following core functionalities:

1. **Train Arrival and Exit Detection**

   - The system shall detect an approaching train using an active-low IR sensor at the entrance side (Gate-1, connected to PA0).
   - The system shall detect train departure using an active-low IR sensor at the exit side (Gate-2, connected to PF0).
   - Detection shall incorporate edge-triggered logic with 200 ms software debounce to prevent false triggers due to noise or vibration.

2. **Gate Control and Movement**

   - Upon train arrival detection, both gates shall move from 180 degrees (open) to 0 degrees (closed) in 10 degrees increments with 100 ms step intervals.
   - Gate movement shall be controlled independently using MG995 servo motors via PWM signals (Gate-1: TIM1_CH1 on PA8, Gate-2: TIM2_CH1 on PA15).
   - Each gate shall support individual pause at 90 degrees for 1 second if an obstacle is detected, without interrupting the motion of the other gate.

3. **Obstacle Detection During Closure**

   - Each gate shall use an HC-SR04 ultrasonic sensor to measure distance to obstacles in the gate path (Gate-1: PB2 Trig and PB1 Echo on TIM3_CH4, Gate-2: PB9 Trig and PB8 Echo on TIM4_CH3).

- Obstacles within 2-15 cm shall trigger a safety pause at 90 degrees for the affected gate only.
- Ultrasonic measurement shall occur at each step during closing, with a 60 ms wait for echo response.

4. **Visual and Audible Feedback**

- The system shall control two LED traffic light modules (Gate-1: PA5 Red, PA6 Yellow, PA7 Green, Gate-2: PE2 Red, PE3 Yellow, PE4 Green).
- Red LEDs and buzzers (PA9 and PE5) shall blink during gate movement and remain steady during the 3-second train passing delay.
- Yellow LEDs shall indicate a 2-second warning period before gate closure.
- Green LEDs shall remain ON during idle (gates open) state.

5. **Timing and State Management**

- After both gates reach 0 degrees, the system shall enforce a 3-second train passing delay.
- After train exit detection, the system shall delay 1 second before initiating gate opening (0 degrees to 180 degrees).
- The system shall return to idle state (gates open, green LEDs ON) upon completion of the opening sequence.

6. **Logging and Debugging**

- All major events (IR detection, angle changes, ultrasonic pauses, state transitions) shall be logged via USART2 at 115200 baud.
- In the RTOS version, logging shall be thread-safe using a message queue and dedicated UART task.

Chapter 3

TABLE 3.1: FUNCTIONAL REQUIREMENTS SUMMARY

Req. No.	**Requirement Description**	**Priority**	**Implemented In (Bare-metal and RTOS)**
FR-1	Detect train arrival using IR sensor (Gate-1, PA0 – active low) with edge detection	High	Both
FR-2	Detect train exit using IR sensor (Gate-2, PF0 – active low) with edge detection	High	Both
FR-3	Move both gates from 180 degrees to 0 degrees in 10 degrees steps (100 ms interval) on arrival detection	High	Both
FR-4	Independent ultrasonic obstacle check (2–15 cm) during closing for each gate	High	Both
FR-5	Pause affected gate at 90 degrees for 1 second on obstacle detection (other gate continues)	Critical	Both
FR-6	3-second train passing delay with steady red LEDs + buzzers after both gates closed	High	Both
FR-7	1-second delay then open gates (0 degrees to 180 degrees) on train exit detection	High	Both
FR-8	Yellow LED warning blink for 2 seconds before gate closure	Medium	Both
FR-9	Green LEDs ON in idle state (gates open)	Medium	Both
FR-10	Thread-safe UART logging of all events (angles, detections, pauses, states)	Medium	RTOS only (queue-based)

3.2 Non-Functional Requirements

1. **Performance**

   - Full gate closure (180 degrees to 0 degrees) shall complete in ≤ 2 seconds under normal conditions.
   - Response time to IR detection shall be ≤ 50 ms.
   - Ultrasonic measurement and decision shall occur within each 100 ms step.

2. **Reliability and Safety**

   - The system shall ensure independent gate operation, obstacle detection on one gate shall not block the other.
   - Software shall include debounce mechanisms to prevent false IR triggers.
   - In case of sensor failure (e.g: no ultrasonic echo), the system shall continue normal operation using the last valid distance or assume no obstacle.

3. **Usability and Maintainability**

   - UART logging shall provide detailed, timestamp-free event traces for debugging.
   - Code shall be modular, well-commented, and structured to allow easy extension (e.g: future ML integration).

4. **Hardware and Software Constraints**

   - Target platform: STM32H753ZI on Nucleo-H753ZI board.
   - Development environment: STM32CubeIDE with HAL libraries and FreeRTOS CMSIS-V2 middleware.

TABLE 3.2: NON-FUNCTIONAL REQUIREMENTS SUMMARY

Req. No.	Requirement Description	Target Value and Constraint	Verification Method
NFR-1	Gate closure time (180 degrees to 0 degrees) under no-obstacle condition	$\leq$ 2.0 seconds	Hardware timing measurement
NFR-2	Response time to IR detection	$\leq$ 50 ms	Oscilloscope and log timestamp
NFR-3	Ultrasonic measurement + decision time per step	Within 100 ms step interval	Code profiling and log
NFR-4	Independent gate operation (one gate pause does not block the other)	Guaranteed in RTOS version	Log analysis during test
NFR-5	Debounce mechanism for IR sensors to prevent false triggers	200 ms software debounce	Repeated trigger test
NFR-6	System reliability under normal lab conditions	No missed detections or false pauses in 50 cycles	Long-duration test runs
NFR-7	Memory usage (RTOS version)	< 50% of STM32H753ZI SRAM	FreeRTOS task list and heap stats
NFR-8	Code maintainability and modularity	Well-commented, modular functions	Code review
NFR-9	Logging without blocking critical tasks (RTOS version)	Thread-safe via queue	Log consistency check
NFR-10	No dependency on external internet or cloud services	Fully standalone embedded system	Design review

Chapter 4

ARCHITECTURE

This chapter presents the overall architecture of the automated railway level crossing gate control system, describing both the bare-metal and FreeRTOS-based implementations. The design emphasizes modularity, real-time responsiveness, and safety through independent gate control and sensor fusion.

4.1 System Block Diagram

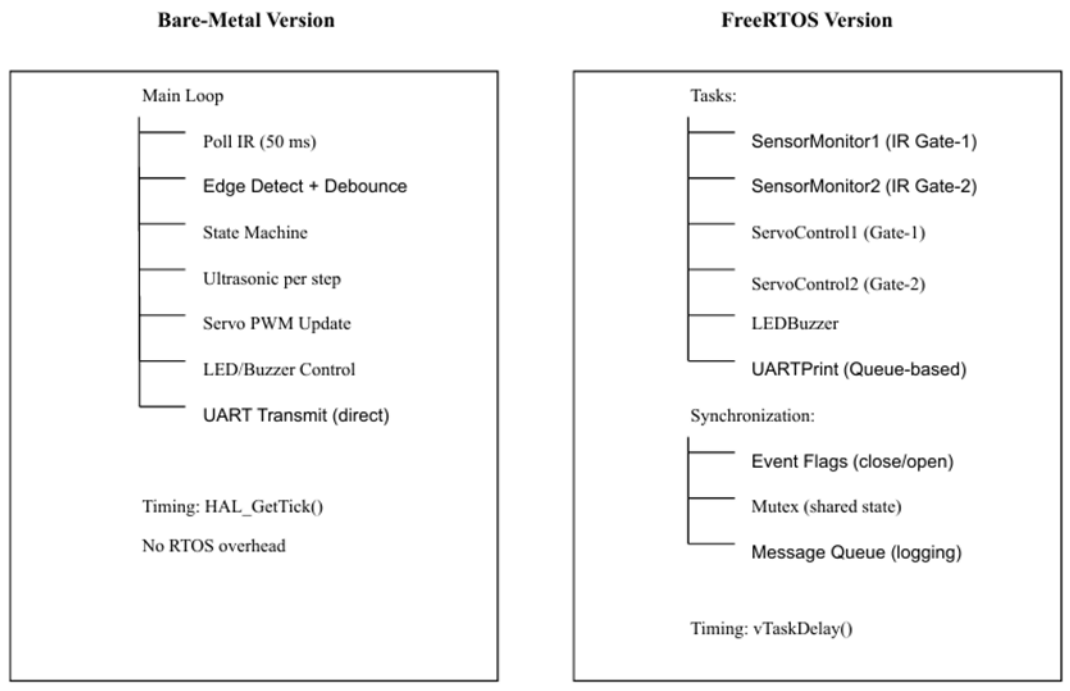


Fig. 4.1 High-level system architecture (bare-metal vs RTOS)

The high-level block diagram consists of the following major components:

- **Central Controller**: STM32H753ZI microcontroller (Nucleo-H753ZI board)
- **Input Sensors**
  - 2 × IR sensors (train arrival and exit detection)
  - 2 × HC-SR04 ultrasonic sensors (obstacle detection)

- **Actuators**
    - 2 × MG995 servo motors (gate movement)
- **Feedback Devices**
    - 2 × LED traffic light modules (Red, Yellow, Green signaling)
    - 2 × 5V active buzzers (audible alerts)
- **Debug Interface**
    - USART2 for event logging to PC terminal

Data flow: Sensors → MCU (GPIO and TIM) → Processing logic → PWM output to servos → GPIO control for LEDs and buzzers → UART logging.

4.2 Bare-Metal Architecture (Non-RTOS Version)

The bare-metal implementation uses a single-threaded, super-loop structure with non-blocking timing:

- **Main Loop** polls IR sensors every 50 ms with edge detection and debounce.
- **State Variables** (sweeping_down, gates_closed, system_state, pause_end timestamps) control gate motion and feedback.
- **Ultrasonic Polling** occurs within the gate movement loop; each gate checks its own sensor independently using separate pause_end timers.
- **Timing Management** uses HAL_GetTick() for 100 ms steps, 1 s pauses, 3 s passing delay, and 1 s open delay.
- **Feedback** handled by dedicated functions (BlinkRedAndBuzzBoth, TurnOffAllFeedback) called conditionally.

Advantages: Simplicity, deterministic execution, no RTOS overhead. Limitation: Single-threaded nature limits true parallelism; long delays (e.g: ultrasonic wait) can temporarily stall the loop.

4.3 FreeRTOS Architecture (RTOS Version)

The RTOS version leverages FreeRTOS CMSIS-V2 for multitasking and deterministic real-time behavior.

- **Tasks** (7 total):
    - SensorMonitor1: Polls IR Gate-1 (PA0), sets closeEvent on falling edge.
    - SensorMonitor2: Polls IR Gate-2 (PF0), sets openEvent on falling edge.
    - ServoControl1: Handles Gate-1 motion (close and open with pauses), waits on closeEvent and openEvent.
    - ServoControl2: Handles Gate-2 motion independently.
    - LEDBuzzer: Monitors system_state, controls blinking and steady LEDs and buzzers.
    - UARTPrint: Dequeues and transmits log messages.
    - defaultTask: Idle loop.
- **Synchronization Primitives**
    - Event Flags: closeEvent and openEvent for signaling gate commands.
    - Mutex: stateMutex protects shared variables (angles, counts, system_state).
    - Message Queue: printQueue for thread-safe UART logging.
- **Data Flow**
    - IR detection in sensor tasks → event flag set → servo tasks awaken → ultrasonic check per step → angle update via PWM → feedback task reacts to state changes → logs queued to UART task.

Advantages: Independent gate pausing without blocking other tasks, deterministic timing via vTaskDelay, scalable for future additions (e.g: ML tasks).

Limitation: Slightly higher memory and CPU overhead compared to bare-metal.

4.4 Comparison of Architectures

The bare-metal version suits simple, deterministic applications with minimal concurrency needs. The FreeRTOS version excels in scenarios requiring parallel execution, such as independent gate safety logic and real-time sensor-actuator coordination. Both versions share identical hardware interfacing (HAL-based GPIO, TIM PWM, TIM Input Capture) and achieve the same functional behaviour, with RTOS providing superior responsiveness and robustness.

Chapter 5

SYSTEM DESIGN

This chapter presents the design of the automated railway level crossing gate control system, covering hardware interfacing, timer configurations, state machine logic, and task interactions for both bare-metal and FreeRTOS implementations. The design ensures safety, real-time responsiveness, and independent gate behaviour.

5.1 Pin Configuration

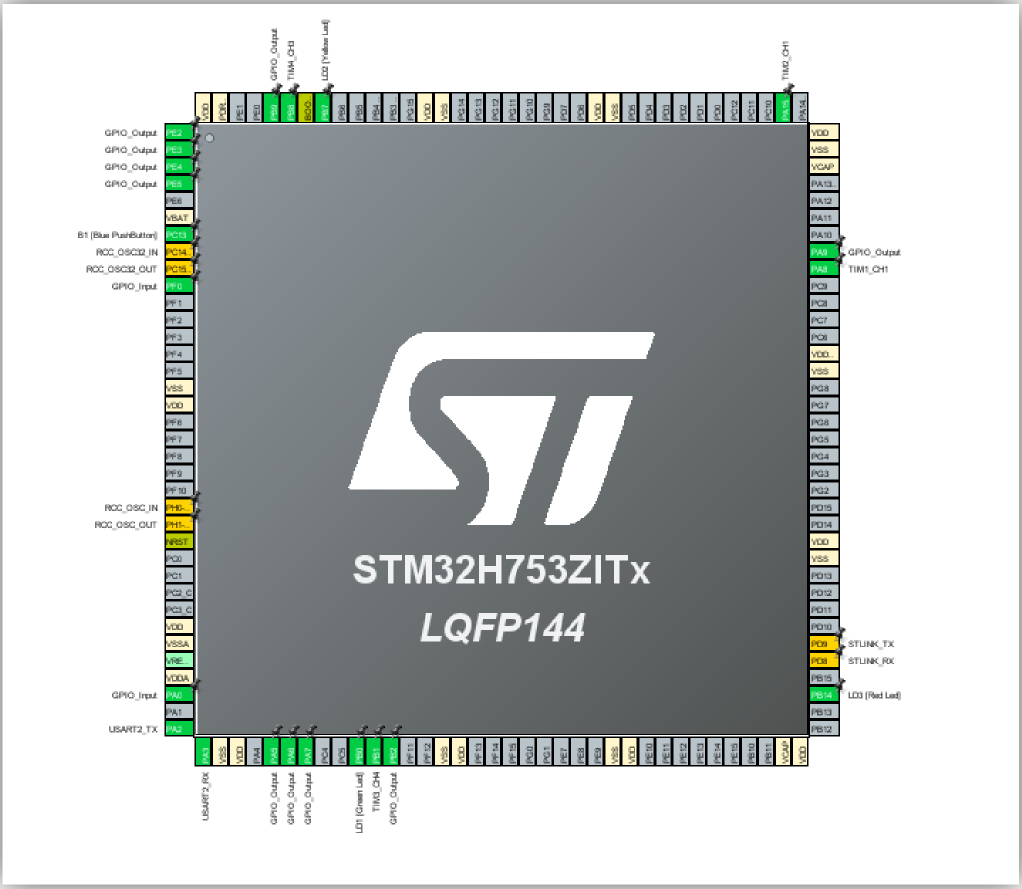


Fig. 5.1 MCU pin configuration in STM32CubeIDE

Chapter 5

The STM32H753ZI microcontroller interfaces with all peripherals using the following pin assignments:

Gate-1 (Entrance Side)

- IR Sensor Output (train arrival): PA0 → GPIO Input (active low)
- Ultrasonic Trigger: PB2 → GPIO Output
- Ultrasonic Echo: PB1 → TIM3_CH4 (Input Capture)
- Servo Signal: PA8 → TIM1_CH1 (PWM Generation)
- Traffic Light: PA5 (Red), PA6 (Yellow), PA7 (Green) → GPIO Output
- Buzzer: PA9 → GPIO Output

Gate-2 (Exit Side)

- IR Sensor Output (train exit): PF0 → GPIO Input (active low)
- Ultrasonic Trigger: PB9 → GPIO Output
- Ultrasonic Echo: PB8 → TIM4_CH3 (Input Capture)
- Servo Signal: PA15 → TIM2_CH1 (PWM Generation)
- Traffic Light: PE2 (Red), PE3 (Yellow), PE4 (Green) → GPIO Output
- Buzzer: PE5 → GPIO Output

Debug Interface

- USART2: PA2 (TX), PA3 (RX) → 115200 baud

All GPIO pins are configured with no pull-up and pull-down and low-speed mode unless timing-critical.

Chapter 5

TABLE 5.1: PIN CONFIGURATION SUMMARY FOR GATE-1 AND GATE-2

Function	Gate-1 Pin	Gate-2 Pin	Mode and Alternate Function	Description
IR Sensor Output	PA0	PF0	GPIO Input	Active low – train arrival and exit
Ultrasonic Trigger	PB2	PB9	GPIO Output	10 µs pulse to HC-SR04
Ultrasonic Echo	PB1	PB8	TIM3_CH4 and TIM4_CH3	Input Capture for distance measurement
Servo Signal (PWM)	PA8	PA15	TIM1_CH1 and TIM2_CH1	50 Hz PWM for MG995 servo
Red LED (Traffic Light)	PA5	PE2	GPIO Output	ON during closing/passing
Yellow LED (Traffic Light)	PA6	PE3	GPIO Output	Warning blink before closing
Green LED (Traffic Light)	PA7	PE4	GPIO Output	ON in idle state
Buzzer	PA9	PE5	GPIO Output	Blink during movement, steady in passing

5.2 Timer Configuration

Timers are configured for precise PWM and input capture operations:

- **TIM1** (Gate-1 Servo): Prescaler = 639, Period (ARR) = 1999, CH1 PWM Generation → 50 Hz (20 ms period)

- **TIM2** (Gate-2 Servo): Identical to TIM1 (Prescaler 639, ARR 1999, CH1 PWM)
- **TIM3** (Gate-1 Ultrasonic Echo): Prescaler = 199, Period = 65535, CH4 Input Capture (rising/falling edge interrupt)
- **TIM4** (Gate-2 Ultrasonic Echo): Identical to TIM3 (Prescaler 199, Period 65535, CH3 Input Capture)
- **TIM6**: FreeRTOS system timebase (1 ms tick)

PWM pulse width mapping:

- 0 degrees (closed) ≈ 1 ms pulse → CCR ≈ 1000 µs equivalent
- 90 degrees ≈ 1.5 ms → CCR midpoint
- 180 degrees (open) ≈ 2 ms → CCR ≈ 2000 µs equivalent

Distance calculation: distance_cm = (echo_ticks × 0.0343 × prescaler_factor) / 2, with prescaler_factor = 3.125 (for 199 prescaler at 64 MHz APB clock).

TABLE 5.2: TIMER CONFIGURATION DETAILS

Timer	Purpose	Prescaler	Period (ARR)	Channel	Clock Source	Notes
TIM1	Gate-1 Servo PWM	639	1999	CH1	Internal (64 MHz)	50 Hz (20 ms period), PWM Generation
TIM2	Gate-2 Servo PWM	639	1999	CH1	Internal (64 MHz)	Identical to TIM1
TIM3	Gate-1 Ultrasonic Echo Capture	199	65535	CH4	Internal (64 MHz)	Input Capture rising/falling edge interrupt
TIM4	Gate-2 Ultrasonic Echo Capture	199	65535	CH3	Internal (64 MHz)	Identical to TIM3
TIM6	FreeRTOS System Tick				Internal	1 ms tick for vTaskDelay

5.3 State Machine Design

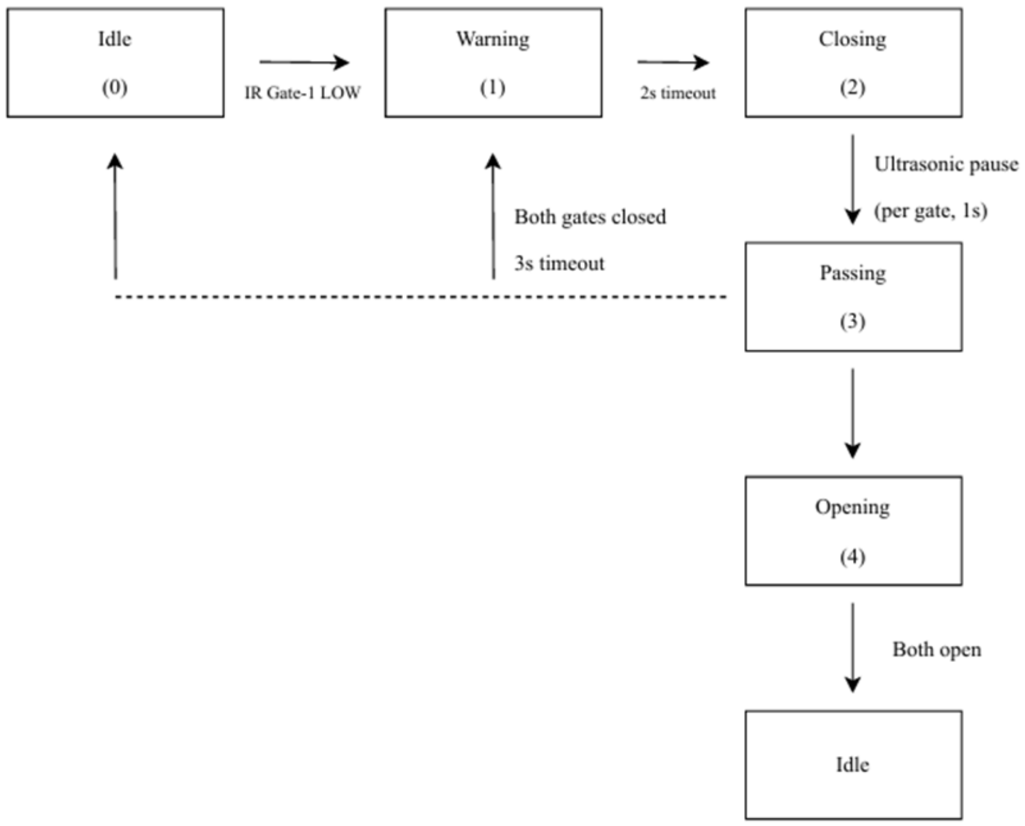


Fig. 5.3 State transition diagram of the system

The system operates in five discrete states (tracked via system_state variable):

- **State 0: Idle** – Gates open (180 degrees), green LEDs ON
- **State 1: Warning** – Yellow LEDs blink for 2 s after IR Gate-1 detection
- **State 2: Closing** – Gates move 180 degrees → 0 degrees in 10 degrees steps, ultrasonic checks active
- **State 3: Passing** – Both gates closed, steady red LEDs + buzzers for 3 s
- **State 4: Opening** – Gates move 0 degrees → 180 degrees after IR Gate-2 detection + 1 s delay

Transitions are event-driven (IR detection) or condition-based (both gates closed/opened).

5.4 Task Interaction (FreeRTOS Version)

- SensorMonitor1 and SensorMonitor 2 tasks poll IR pins (50 ms interval), detect falling edges, and set closeEvent and openEvent flags.
- ServoControl1 and ServoControl 2 tasks wait on respective events, execute closing and opening loops with ultrasonic polling, update shared angle variables under stateMutex.
- LEDBuzzer task continuously monitors system_state (50 ms interval), applies blinking and steady patterns.
- UARTPrint task blocks on printQueue, transmits dequeued messages.
- Mutual exclusion via stateMutex prevents race conditions on angles, counts, and state.

Chapter 6

IMPLEMENTATION

This chapter explains the implementation of both bare-metal and FreeRTOS versions, including key code structures, hardware integration, testing methodology, and results.

6.1 Bare-Metal Implementation

The bare-metal version uses a single main loop with HAL_GetTick() for non-blocking timing.

Key Functions

- HCSR04_Trigger_GateX(): Generates 10 µs pulse on trigger pin.
- Get_Ultrasonic_Distance_Gate(): Triggers sensor, waits 60 ms, returns distance from callback.
- SetServoGate(angle): Maps angle to PWM CCR value using Angle_to_CCR().
- BlinkRedAndBuzzBoth() and TurnOffAllFeedback(): Control LEDs and buzzers.
- Print(): UART transmit for logging.

Main Logic

- IR polling with edge detection and 200 ms debounce.
- On arrival: 2 s yellow warning → closing loop (100 ms steps, ultrasonic check per step).
- Pause at 90 degrees if obstacle (2–15 cm) → resume independently via per-gate pause_end timestamp.
- Both closed → 3 s passing delay with steady red and buzz.
- On exit: 1 s delay → opening loop → green LEDs ON.

Ultrasonic Callback

Code:

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim) {
   if (htim->Instance == TIM3)
   { /* Gate-1 logic */ }
```

```
 if (htim->Instance == TIM4)
    { /* Gate-2 logic */ }
    // Calculate distance using echo_ticks and factor 3.125 × 0.0343 / 2
}
```

TABLE 6.1: COMPARISON OF BARE-METAL AND FREERTOS IMPLEMENTATIONS

Feature	Bare-Metal Version	FreeRTOS Version	Advantage of RTOS Version
Architecture	Single-threaded main loop	Multi-tasking (7 tasks)	True concurrency
Gate Independence	Achieved via separate pause_end timestamps	Native – each gate has dedicated task	No blocking during one gate's pause
Timing Mechanism	HAL_GetTick() non-blocking delays	vTaskDelay() + event flags	Deterministic, scheduler-managed
IR Detection	Polling in main loop (50 ms)	Dedicated SensorMonitor tasks (50 ms)	Higher priority, non-blocking
Ultrasonic Handling	Called inside movement loop	Polled inside each servo task	Independent, parallel execution
Feedback (LED/Buzzer)	Function calls in main loop	Dedicated LEDBuzzer task	Continuous monitoring without interference
Logging	Direct HAL_UART_Transmit	Thread-safe queue + UARTPrint task	No blocking of critical tasks
Memory Overhead	Minimal (no RTOS)	Moderate (tasks, queue, mutexes)	Acceptable for STM32H7 resources
Scalability for Future ML	Difficult (single thread)	Easier (add ML task)	Better foundation for predictive analysis
Response Time Consistency	Good under light load	Excellent under multi-task load	Scheduler ensures priority

6.2 FreeRTOS Implementation

The RTOS version uses seven tasks configured via STM32CubeMX (CMSIS-V2).

Task Breakdown

- SensorMonitor1 and SensorMonitor2: 50 ms polling, edge detection, set closeEvent/openEvent.
- ServoControl1 and ServoControl2: Wait on events → 2 s warning delay → 100 ms step loop with ultrasonic polling → pause at 90 degrees if needed → count closed and opened gates → signal next phase.
- LEDBuzzer: Monitors system_state → blink during movement and passing → steady green when idle.
- UARTPrint: Blocks on printQueue → transmits → frees allocated string.

Synchronization

- Event flags for gate commands.
- Mutex for shared state (angles, counts, system_state).
- Queue for thread-safe logging (pvPortMalloc + vPortFree).

Logging Example

Code:

```
void Print(const char *format, ...)
{
   char buffer[128];
   va_list args;
   va_start(args, format);
   vsnprintf(buffer, sizeof(buffer), format, args);
   va_end(args);
   osMutexAcquire(printMutexHandle, osWaitForever);

   char *msg = pvPortMalloc(strlen(buffer) + 1);
   strcpy(msg, buffer);
   osMessageQueuePut(printQueueHandle, &msg, 0, osWaitForever);
   osMutexRelease(printMutexHandle);
}
```

6.3 Testing and Results

Hardware testing used a toy-train setup on a scaled track. Key observations:

- Full close and open cycle: ≈1.8-2.0 s without obstacles.
- Independent pausing: Gate 2 paused multiple times at 5-7 cm while Gate 1 continued (verified in UART logs).
- Feedback: Yellow blink (2 s), red and buzz blink during movement, steady red and buzz during passing, green idle.
- No missed IR triggers (debounce effective).
- RTOS version showed no blocking delays during pauses.

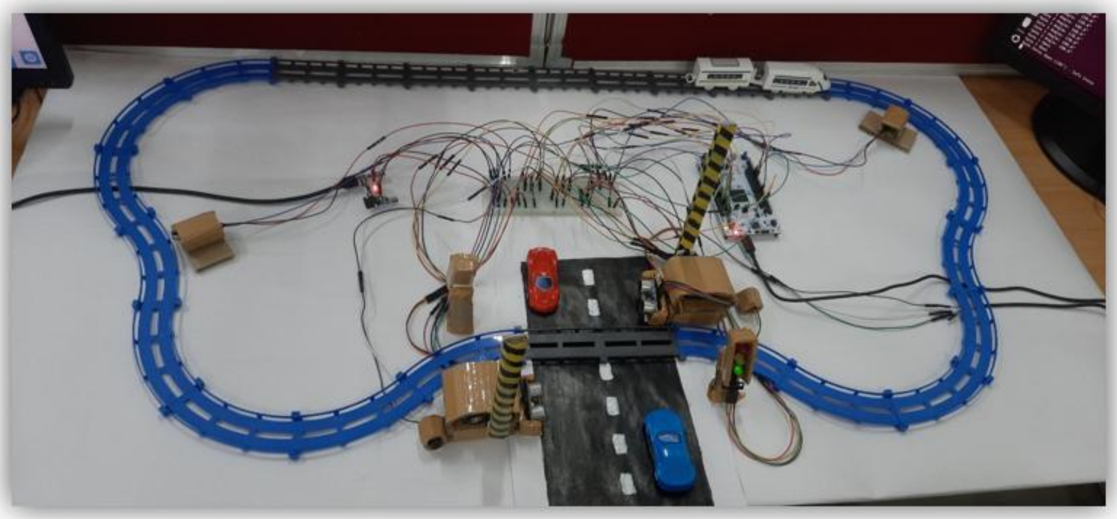
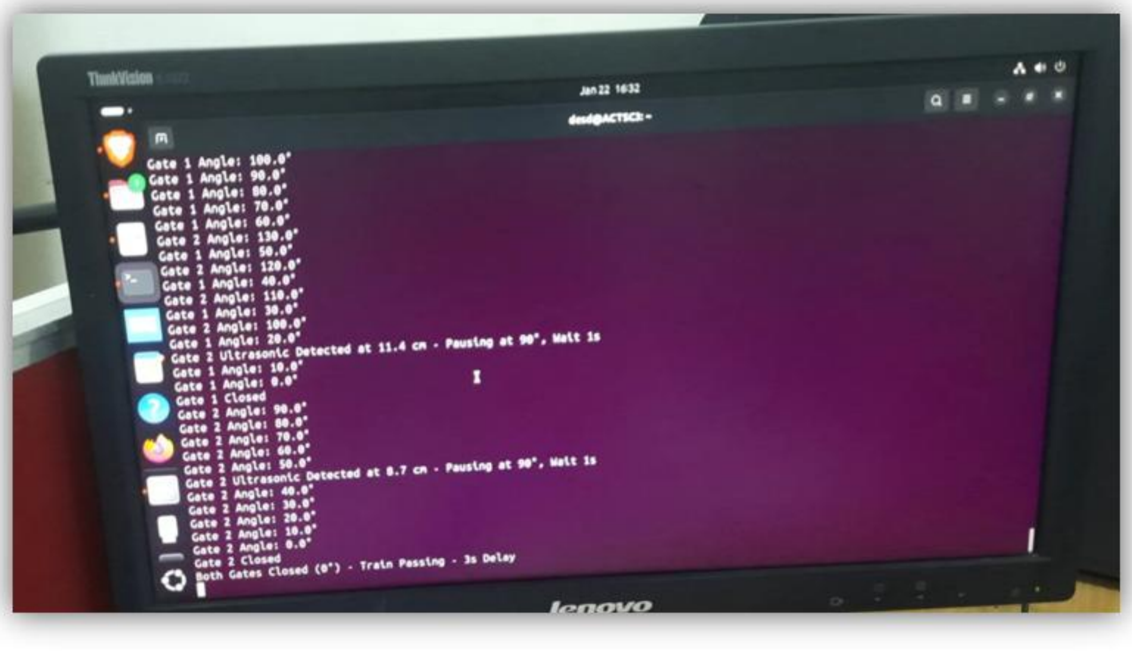


Fig. 6.3.1 Photograph of the demonstrated project prototype



```
Gate 1 Angle: 100.0°
Gate 1 Angle: 90.0°
Gate 1 Angle: 80.0°
Gate 1 Angle: 70.0°
Gate 1 Angle: 60.0°
Gate 2 Angle: 130.0°
Gate 1 Angle: 50.0°
Gate 2 Angle: 120.0°
Gate 1 Angle: 40.0°
Gate 2 Angle: 110.0°
Gate 1 Angle: 30.0°
Gate 2 Angle: 100.0°
Gate 1 Angle: 20.0°
Gate 2 Ultrasonic Detected at 11.4 cm - Pausing at 90°, Wait 1s
Gate 1 Angle: 10.0°
Gate 1 Angle: 0.0°
Gate 1 Closed
Gate 2 Angle: 90.0°
Gate 2 Angle: 80.0°
Gate 2 Angle: 70.0°
Gate 2 Angle: 60.0°
Gate 2 Angle: 50.0°
Gate 2 Ultrasonic Detected at 8.7 cm - Pausing at 90°, Wait 1s
Gate 2 Angle: 40.0°
Gate 2 Angle: 30.0°
Gate 2 Angle: 20.0°
Gate 2 Angle: 10.0°
Gate 2 Angle: 0.0°
Gate 2 Closed
Both Gates Closed (0°) - Train Passing - 3s Delay
```

Fig. 6.3.2 Screenshot of UART log showing independent gate pause (RTOS)

Chapter 7

CONCLUSION

7.1 Summary of Achievements

This project successfully designed and implemented an automated railway level crossing gate control system using the STM32H753ZI microcontroller on the Nucleo-H753ZI board. The system integrates IR sensors for train detection, ultrasonic sensors for obstacle safety, servo motors for gate actuation, and LED and buzzer modules for clear feedback, effectively eliminating human dependency and enhancing safety at level crossings.

Two versions were developed, a bare-metal implementation utilizing non-blocking timing and a FreeRTOS-based version employing task concurrency, event flags, mutexes, and thread-safe logging. Both versions demonstrated reliable operation on hardware, with full gate closure in approximately 1.8-2 seconds, independent obstacle-based pauses at 90 degrees, a 3-second passing delay, and smooth reopening after train exit. UART logs confirmed correct sequencing, independent gate behaviour, and accurate sensor fusion.

Key achievements include real-time responsiveness suitable for fast toy-train demonstrations, independent safety logic for each gate, and effective visual and audible signaling. The RTOS version provided superior concurrency, ensuring one gate's pause does not block the other, highlighting the advantages of multitasking in safety-critical embedded applications.

7.2 Limitations

Limitations include the scaled prototype nature (toy-train setup), absence of full-scale railway hardware testing, and current exclusion of wireless communication or cloud integration. Sensor reliability in extreme conditions (e.g: dust, temperature) and servo current draw under load were not exhaustively evaluated.

7.3 Future Scope

Future scope includes predictive analysis of critical components, particularly servo motors, using machine learning or deep learning models integrated directly within STM32CubeIDE. This would enable real-time health monitoring, fault prediction, and proactive maintenance based on embedded sensor data, further enhancing system longevity and reliability for real-world deployment.

REFERENCES

[1] B. B. Mansingh, K. S. Selvakumar and S. R. V. Kumar, "Automation in unmanned railway level crossing," *2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO)*, Coimbatore, India, 2015, pp. 1-4, doi: 10.1109/ISCO.2015.7282344.

[2] Jesuraj, Y. Arockia, and K. Hemalatha. "A prototype model of unmanned automatic level crossing system using piezoelectric sensor." *Microprocessors and Microsystems* 79 (2020): 103265.

[3] Zawodny, M.; Kruszyna, M.; Szczepanek, W.K.; Korzeń, M. A New Form of Train Detection as a Solution to Improve Level Crossing Closing Time. Sensors 2023, 23, 6619. https://doi.org/10.3390/s23146619

[4] Maran, Madhan Kumar, Kannan Ramakrishnan, and Kamalesh Murugan. "Advanced Train Detection and Gate Control System using ATmega328P and Passive Infrared Motion Sensor." *2025 5th International Conference on Trends in Material Science and Inventive Materials (ICTMIM)*. IEEE, 2025.

[5] Kulkarni, Narasimha, et al. "Automated Gate control with backup systems at Level Crossing." *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. IEEE, 2020.

[6] A. K. Gangwar, A. Sarojwal and P. P. Singh, "A Study and Solution for Unmanned Railway Crossings with Complete Automation," *2024 13th International Conference on System Modeling & Advancement in Research Trends (SMART)*, Moradabad, India, 2024, pp. 820-824, doi: 10.1109/SMART63812.2024.10882235.

[7] P. Kumar *et al*., "Automatic Protection System and Risk Mitigation in Railways," *2023 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, Greater Noida, India, 2023, pp. 1647-1652, doi: 10.1109/ICAC3N60023.2023.10541376.

[8] Kumar, Prince, et al. "Automatic Protection System and Risk Mitigation in Railways." *2023 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*. IEEE, 2023.

[9] Sikora, Pavel, et al. "Artificial intelligence-based surveillance system for railway crossing traffic." *IEEE Sensors Journal* 21.14 (2020): 15515-15526.

[10] Staino, A., Suwalka, A., Mitra, P. *et al.* Real-Time Detection and Recognition of Railway Traffic Signals Using Deep Learning. *J. Big Data Anal. Transp.* **4**, 57–71 (2022). https://doi.org/10.1007/s42421-022-00054-7

[11] Kapoor, Rajiv, Rohini Goel, and Avinash Sharma. "An intelligent railway surveillance framework based on recognition of object and railway track using deep learning." *Multimedia tools and applications* 81.15 (2022): 21083-21109.

[12] Sabarish, K., et al. "Automation of Railway Gate Using Raspberry Pi." *International Conference on Soft Computing and Signal Processing*. Singapore: Springer Nature Singapore, 2022.

[13] Amusan, Akinwumi Abimbola, and Yusuf Kolawole Adebakin. "An automatic railway level crossing system with crack detection." *2022 5th Information Technology for Education and Development (ITED)*. IEEE, 2022.

[14] Zhang, Yuanyuan, and Junnan Li. "Design of real-time embedded system based on STM32 and free RTOS with optimization of advanced task scheduling algorithm." *2025 3rd Cognitive Models and Artificial Intelligence Conference (AICCONF)*. IEEE, 2025.

[15] Kumar, Abhishek, and Raghav Khajuria. "Enhancing railway operations: machine learning approaches for stopping distance prediction." *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, 2024.