Design, Development, and Testing of a Real-Time Intelligent Traffic Control System using FreeRTOS

Andukuri M N Sri Harsha(N200484)
Dept. of Electronics and Communication Engineering
Rajiv Gandhi University of Knowledge Technologies (RGUKT)
Email: N200484@rguktn.ac.in

***Abstract*—Urban intersections are traditionally controlled using fixed-time or manually configured traffic signals, which are often unable to adapt to rapidly changing traffic patterns. This leads to increased waiting time, fuel consumption, and congestion. This paper presents the design, development, and testing of a real-time intelligent traffic control system implemented on an STM32F446RE microcontroller using the FreeRTOS real-time operating system. The proposed system employs infrared (IR) sensors to measure lane-wise vehicle density and dynamically allocates green light durations based on three discrete density levels: no vehicles, low density, and high density. A master task and three lane tasks are created in FreeRTOS to realize a clean separation of responsibilities, ensure time determinism, and improve modularity. The master task executes periodically at 1 Hz using `vTaskDelayUntil()`, ensuring predictable timing, while lane tasks periodically sample IR sensors and update shared density variables. A multi-color LED scheme (red, blue, green) is used to represent stop, waiting/next, and go states, respectively. Experimental evaluation of the prototype demonstrates deterministic scheduling, fair access for all lanes, bounded maximum waiting time, and correct handling of both empty and heavily loaded traffic conditions. The proposed architecture can be extended to real intersections with higher sensor resolution and communication to a central controller.**

***Index Terms*—FreeRTOS, real-time systems, STM32, intelligent traffic light control, embedded systems, lane density.**

I. Introduction

Traffic congestion is a critical problem in modern cities, leading to increased travel time, fuel wastage, and pollution. Conventional traffic lights are typically configured based on fixed timing plans or simple sensor-based actuation, which may not adapt sufficiently to real-time traffic variations. As a result, vehicles may be forced to wait at a red signal even when no cross traffic is present, while congested lanes may receive insufficient green time. To address these issues, intersection control must become more intelligent, adaptive, and deterministic.

Real-time operating systems (RTOS) are widely used in embedded applications that require predictable timing and concurrent task management. FreeRTOS is a lightweight, open-source RTOS designed for microcontrollers, supporting preemptive multitasking, priorities, and deterministic scheduling. Using FreeRTOS for traffic control not only improves modularity and code structure but also makes timing behavior explicit and analyzable.

In this work, an intelligent traffic signal controller is implemented using FreeRTOS on an STM32F446RE Nucleo board. Three traffic lanes are modeled, each equipped with two IR sensors to detect vehicle presence at near and far positions. Based on the combined sensor state, the lane is classified as having no traffic, low traffic, or high traffic. A master task uses this density information to dynamically choose the green and blue (next) lanes and to compute the green phase duration subject to fairness and maximum green time constraints. The final system offers:

- Density-based adaptive green timings (1 s, 3 s, or 5 s).
- Fairness enforced by a maximum cumulative green time of 10 s per lane when other lanes have vehicles.
- Deterministic 1 Hz decision loop implemented via `vTaskDelayUntil()`.
- Separation into one master task and three lane tasks to reflect a typical RTOS design pattern.

II. Related Work

Several research efforts have explored the application of intelligent control to traffic signals. Adaptive control systems based on loop detectors, image processing, or wireless sensor networks attempt to dynamically tune signal timings in response to real-time traffic information. Fuzzy logic controllers and reinforcement learning-based controllers have also been proposed for more complex intersections.

While many works focus on algorithmic optimization, relatively fewer concentrate on the embedded implementation details, especially in resource-constrained microcontrollers. Some solutions rely on high-end processors, general-purpose operating systems, or centralized servers, which may increase cost and complexity. In contrast, the present work demonstrates how a practical intersection controller can be realized on a low-cost STM32 microcontroller using FreeRTOS, with a clear mapping from traffic logic to RTOS tasks.

III. System Overview

A. Hardware Platform

The proposed system is implemented on an STM32F446RE Nucleo development board. It features an ARM Cortex-M4 core, integrated memory, and rich peripheral support suitable for real-time control. The following peripheral connections are used:

- Six IR sensors (two per lane) connected to GPIO pins.
- Nine LEDs (three per lane) for red, blue, and green indications.

TABLE I
GPIO PIN ASSIGNMENT FOR IR SENSORS AND LEDS

Signal	Description	STM32 Pin
L1_HIGH	Lane 1 far IR sensor	PA0
L1_LOW	Lane 1 near IR sensor	PA1
L2_HIGH	Lane 2 far IR sensor	PA4
L2_LOW	Lane 2 near IR sensor	PB0
L3_HIGH	Lane 3 far IR sensor	PC1
L3_LOW	Lane 3 near IR sensor	PC0
L1_RED	Lane 1 red LED	PA10
L1_BLUE	Lane 1 blue LED	PB3
L1_GREEN	Lane 1 green LED	PB5
L2_RED	Lane 2 red LED	PB4
L2_BLUE	Lane 2 blue LED	PB10
L2_GREEN	Lane 2 green LED	PA8
L3_RED	Lane 3 red LED	PA9
L3_BLUE	Lane 3 blue LED	PC7
L3_GREEN	Lane 3 green LED	PB6

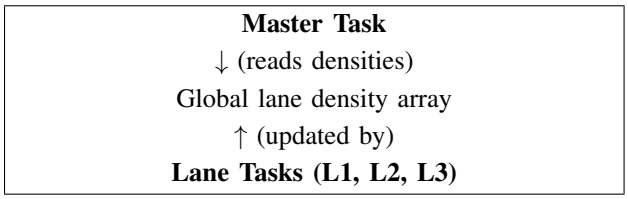

Fig. 1. High-level software architecture showing one master task and three lane tasks.

- UART2 configured at 115200 baud for logging system behavior to a PC terminal.

Each lane has a near and far IR sensor. The far sensor represents vehicles queued deeper in the lane, while the near sensor indicates vehicles close to the intersection. Both sensors are modeled as active-low digital inputs (i.e., logic 0 when blocked). LED states are defined as follows:

- Red: Lane must stop (no crossing).
- Blue: Lane is next; drivers are informed to be ready.
- Green: Lane has right-of-way and can cross.

B. Software Architecture

The software is divided into two major parts: hardware abstraction using the STM32 HAL and task-level scheduling using FreeRTOS. The overall architecture is shown conceptually in Fig. 1.

- **Master Task:** Runs at 1 Hz, reads the latest lane densities, chooses the active green and blue lanes, computes green phase duration, enforces maximum green time of 10 s when other lanes have traffic, and updates LED states.
- **Lane Tasks (3 tasks):** Each lane task runs every 100 ms, samples its two IR sensors, and updates the corresponding entry in the global array `g_laneDensity[3]`.
- **UART Logging:** The master task logs the current densities, green and blue lanes, and remaining time to a UART terminal once per second.

TABLE II
FREERTOS TASK CONFIGURATION

Task Name	Role	Priority	Period
master_task	Traffic decision & LEDs	2	1 s
lane1_task	Density for Lane 1	1	100 ms
lane2_task	Density for Lane 2	1	100 ms
lane3_task	Density for Lane 3	1	100 ms

IV. TRAFFIC DENSITY EVALUATION AND CONTROL LOGIC

A. Density Computation

Each lane is associated with two IR sensors, denoted as high (far) and low (near). The raw digital values are converted into logical activity using

$$\texttt{IR\_ACTIVE(raw)} = (raw == 0),$$

assuming active-low sensors. The density for lane i is then defined as:

- `DENSITY_HIGH`: both sensors active.
- `DENSITY_LOW`: exactly one sensor active.
- `DENSITY_NONE`: no sensor active.

This mapping provides a simple but effective representation where the number of observed vehicles influences the phase duration. The C function `read_lane_density()` encapsulates this logic and returns the corresponding `density_t` value.

B. FreeRTOS Task Configuration

Table II summarizes the FreeRTOS tasks created in the system, along with their priorities and periods.

The master task is given a higher priority than lane tasks to ensure that the 1 Hz control loop executes in a timely manner. The lane tasks are simple, sensor-sampling tasks that tolerate small jitter in their execution.

C. Phase Selection and Fairness Constraints

The master task enforces the following rules:

1) If all lanes have `DENSITY_NONE`, the system cycles through the lanes in a round-robin fashion with a fixed 3 s green time per lane. This models a default cycle when no vehicles are present.
2) If at least one lane has non-zero density, the lane with the highest density is selected as the green lane. In case of a tie, the lane chosen next is the one following the current green lane in cyclic order, which ensures fairness.
3) The blue lane is chosen as another lane (different from green) that has non-zero density, whenever possible. If no other lane has density, the next lane in order is chosen to be blue purely as an indicator that it is next, even if currently empty.
4) The green phase duration is set to:
   - 5 s for `DENSITY_HIGH`
   - 3 s for `DENSITY_LOW`
   - 1 s when the chosen green lane itself has no vehicles but other lanes have density (a rare transient case)

5) If any other lane has density and the current green lane has already accumulated 10 s of continuous green time, the controller forces a switch to another lane with the highest density to avoid starvation.

D. Time Determinism

The master task uses `vTaskDelayUntil()` with a period of 1 second. This API ensures that the task wakes up at fixed intervals relative to an initial reference tick, making the main control loop periodic and time-deterministic. Lane tasks use `vTaskDelay()` with 100 ms delays, which is sufficient for sensor sampling and does not affect the determinism of the higher-priority master task.

V. Implementation Details

A. UART Logging

UART2 is configured at 115200 baud. The standard C library `printf` is redirected to UART by overriding the `_write()` function. This allows the system to print formatted status lines such as `dens[L1,L2,L3]=[1,0,2], GREEN=L3, BLUE=L1, remain=5s`. This line length fits comfortably in a single column and does not overflow into the adjacent column in the IEEE two-column layout.

B. LED Control

Each lane has three LEDs driven by GPIO pins configured as push-pull outputs. The helper function `set_lane_led()` abstracts away the specific GPIO ports and pins. The master task calls this function after deciding the green and blue lanes to ensure that exactly one color is active per lane.

C. Illustrative Code Snippet

To avoid tables floating into the middle of the code example, the main scheduling logic is placed inside a figure float with a listing environment, as shown in Fig. 2.

```
/* Snapshot densities */
density_t d0 = g_laneDensity[0];
density_t d1 = g_laneDensity[1];
density_t d2 = g_laneDensity[2];

/* Check if all empty */
uint8_t all_empty = (d0 == DENSITY_NONE &&
                     d1 == DENSITY_NONE &&
                     d2 == DENSITY_NONE);

if (phase_remaining_s == 0) {
   if (all_empty) {
      current_green = (current_green + 1) % 3;
      current_blue = (current_green + 1) % 3;
      phase_remaining_s = 3;
      green_continuous_s = 0;
   } else {
      /* Decide new_green with 10 s fairness constraint */
      uint8_t new_green =
         select_next_nonempty_lane(current_green);
      /* ... fairness and max 10 s logic ... */

      if (new_green != current_green) {
         current_green = new_green;
         green_continuous_s = 0;
      }

      /* Choose blue lane and green duration */
      /* ... */
   }

   /* Apply LEDs */
   set_lane_led(1, LANE_LED_RED);
   set_lane_led(2, LANE_LED_RED);
   set_lane_led(3, LANE_LED_RED);
   set_lane_led(current_green + 1, LANE_LED_GREEN);
   set_lane_led(current_blue + 1, LANE_LED_BLUE);
}
```

Fig. 2. Excerpt of master task logic implementing phase timing and fairness.

VI. Experimental Results

A. Test Scenarios

The system was tested under several representative scenarios by manually covering/uncovering IR sensors to emulate vehicle presence. Table III summarizes three key scenarios and the corresponding behavior observed at the UART terminal and LEDs.

TABLE III
Selected Test Scenarios and Observed Behavior

Scenario	Sensor Pattern	Observed Behavior
S1: All empty	All IR sensors inactive (logic 1)	Controller cycles L1, L2, L3 with 3 s green each; blue indicates next lane; no lane exceeds 3 s since all are empty.
S2: High density on L3 only	L3_HIGH = 0, L3_LOW = 0; others inactive	L3 repeatedly selected as green with 5 s phases; L1 or L2 becomes blue; if occasional vehicles appear on L1 or L2, they receive green after L3 accumulates 10 s.
S3: Low density on L1, high density on L2	L1 has one sensor active, L2 both, L3 empty	L2 is preferred as green with 5 s, L1 is often blue and later green with 3 s; system prevents L2 from monopolizing more than 10 s when L1 has pending vehicles.

B. Timing and Determinism

The periodicity of the master task was verified by observing the timestamped UART messages. The interval between consecutive log lines remained approximately 1 s, confirming correct operation of `vTaskDelayUntil()`. Lane tasks, running at 100 ms, updated density values quickly enough so that the master task always had up-to-date information.

The fairness constraint was also validated. For example, when L3 had continuous high density and L1 had low density, L3 initially received a series of 5 s green phases. Once the accumulated green time for L3 reached approximately 10 s, the controller forced a switch to L1, granting it a 3 s green phase before returning to L3, thus avoiding starvation.

C. Discussion

The prototype demonstrates that even with simple binary sensors and a small set of density states, the system can adapt green times based on actual demand and guarantee fairness. While the discrete timing (1 s resolution) may be coarse for real intersections, it is adequate for a laboratory-

scale demonstrator. In real deployments, the same architecture can be retained while scaling up to more sophisticated sensors (e.g., cameras, radar) and larger timing ranges.

VII. Conclusion and Future Work

This paper presented the design, development, and testing of an intelligent traffic control system implemented on an STM32F446RE microcontroller using FreeRTOS. The system uses six IR sensors to estimate lane-wise density and dynamically assigns green times based on three density levels. By structuring the application into one master task and three lane tasks, the design achieves clear modularity, deterministic timing, and fairness across lanes. Experimental validation on a physical prototype verified the intended behavior, including:

- Adaptive green times: 5 s for high, 3 s for low, and 1 s for rare empty-green cases.
- A maximum continuous green time of 10 s for any lane when others also have traffic.
- Deterministic 1 Hz control loop using `vTaskDelayUntil()`.

Future work will focus on:

- Integrating more advanced sensing (e.g., computer vision or wireless communication with vehicles).
- Extending the system to four-way intersections and pedestrian crossings.
- Logging traffic statistics to non-volatile memory or a cloud server for offline optimization.
- Exploring more advanced scheduling algorithms (e.g., fuzzy logic or reinforcement learning) while retaining real-time guarantees.

Acknowledgment

The author would like to thank the faculty and laboratory staff of the Department of ECE, RGUKT, for providing guidance and access to development tools and equipment.

References

[1] R. Barry, *Using the FreeRTOS Real Time Kernel – A Practical Guide*, FreeRTOS, 2013.

[2] STMicroelectronics, "STM32F446xx advanced Arm-based 32-bit MCUs," Data Sheet, 2019.

[3] P. Varaiya, "The max-pressure controller for arbitrary networks of signalized intersections," in *Advances in Dynamic Network Modeling in Complex Transportation Systems*, Springer, 2013.

[4] R. Panda and R. Agrawalla, "Intelligent traffic control system using adaptive traffic signal control," in *Proc. Int. Conf. on Smart Cities*, 2015.

[5] J. J. Labrosse, *MicroC/OS-II: The Real-Time Kernel*, 2nd ed., CMP Books, 2002.