RTOS-Based Car Monitoring System

Project Report

Ain Shams University

Faculty of Engineering – Computer and Systems Engineering Dept.

Team Members

Name	ID
Ahmed Lotfy Abousalem	21P0332
Abdelruhman Reda Shehata	21P0059
Akram Abdel-Maksoud	21P0089

Supervisor: Dr. [SHRIEF HAMMAD]

Spring 2025

Table of Contents

1 Introduction

This project develops a real-time car monitoring system using the FreeRTOS operating system on a TM4C123 microcontroller. The system integrates multiple sensors and actuators to monitor and display critical vehicle parameters, ensuring timely responses to environmental and operational changes. The primary objective is to demonstrate the application of real-time operating system concepts in an embedded automotive context, focusing on task scheduling, interrupt handling, and inter-task synchronization using mutexes and queues.

2 System Overview

The car monitoring system comprises several hardware components interfaced with the TM4C123 microcontroller, managed by FreeRTOS tasks. The system monitors distance to obstacles, vehicle speed, door lock status, ignition state, and gear position, providing real-time feedback through an LCD display and a buzzer for alerts. The design emphasizes modularity, real-time responsiveness, and efficient resource utilization.

2.1 Hardware Components

- **Ultrasonic Sensor**: Measures distance to obstacles ahead of the vehicle.
- **Potentiometer (ADC)**: Simulates vehicle speed input.
- **LCD Display**: Displays system status, including distance, speed, door lock state, ignition state, and gear position.
- **Buzzer**: Provides audible alerts based on proximity to obstacles.
- **Limit Switch**: Detects door lock/unlock status.
- **Ignition Button**: Toggles the vehicle's ignition state.
- **Gear Selector Switches**: Indicates gear position (Park, Reverse, Neutral, Drive).

2.2 Software Architecture

The software is built around FreeRTOS, utilizing four tasks to handle sensor data acquisition, processing, and output. Each task operates at a defined priority to ensure critical operations, such as distance measurement and speed monitoring, are performed with minimal latency. A mutex (`xSharedDataMutex`) synchronizes access to shared variables (`distanceValue`, `speedValue`, `doorsLocked`, `ignitionOn`, and `gearState`), with tasks using `xSemaphoreTake` and `xSemaphoreGive`, and ISRs using `xSemaphoreTakeFromISR` and `xSemaphoreGiveFromISR`. A queue (`xEventQueue`) handles event communication, replacing global event flags. Interrupt handlers and the speed task send event messages (`EventMessage_t`) to the display task, which include door lock changes, ignition state changes, and gear shifts. The event message structure contains an event type (door, ignition, or gear) and relevant data (e.g., door lock state, gear state). The display task processes these messages using `xQueueReceive`, ensuring robust and modular event handling.

3 Wiring Diagram

The wiring diagram illustrates the connections between the TM4C123 microcontroller and the system components. The pin assignments are based on the GPIO configurations defined in the system.

- **Ultrasonic Sensor**:
  - Trigger: PA6
  - Echo: PA7
- **LCD Display**:
  - RS: PD0
  - E: PD1
  - D0: PB0
  - D1: PB1
  - D2: PB4
  - D3: PB5
  - D4: PE3
  - D5: PE4
  - D6: PE5
  - D7: PA5
- **Buzzer**: PB2
- **Potentiometer (ADC)**: PE2
- **Limit Switch**: PE0
- **Ignition Button**: PD6
- **Gear Selector Switches**:
  - Park (P): PC4
  - Reverse (R): PC5
  - Neutral (N): PC6
  - Drive (D): PC7

4 System Functionality

The system performs the following key functions:

- **Distance Monitoring**: When the ignition is on, the ultrasonic sensor measures the distance to obstacles every 100 ms. The buzzer activates with varying patterns based on

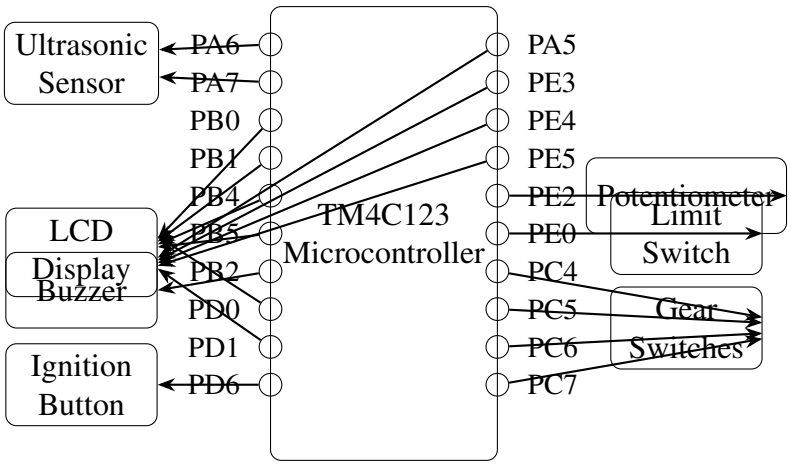

Figure 1: Wiring Diagram for the RTOS Car Monitoring System

proximity: rapid beeping for distances less than 30 cm and slower beeping for distances between 30 and 100 cm.

- **Speed Monitoring**: The ADC reads a potentiometer value every 200 ms to simulate vehicle speed, scaled to a range of 0 to 100 km/h. If the speed exceeds 20 km/h and the doors are unlocked, the system automatically locks them and sends a door event to the display task.
- **Door Lock Management**: A limit switch triggers an interrupt to toggle the door lock state, sending an event to the display task, which updates the LCD and activates a brief buzzer alert.
- **Ignition Control**: An ignition button toggles the vehicle's operational state, sending an event to the display task to update the LCD and trigger a buzzer alert.
- **Gear Selection**: Four switches represent gear positions. An interrupt ensures only one gear is active at a time, defaulting to Neutral if multiple gears are selected, and sends an event to the display task to update the LCD.
- **Display Management**: The LCD continuously updates to reflect the system state, processing event messages from the queue for door, ignition, and gear changes, and displaying real-time distance and speed data when no events are pending.

5 Design Details

5.1 Task Structure

The system employs four FreeRTOS tasks, each assigned a priority based on its criticality:

- **Ultrasonic Task (Priority 3)**: Periodically measures distance using the ultrasonic sensor when the ignition is on.
- **Speed Task (Priority 3)**: Reads ADC values to compute speed and triggers automatic door locking, sending door events to the queue if necessary.
- **Buzzer Task (Priority 2)**: Controls the buzzer based on distance thresholds, ensuring

timely alerts.

- **Display Task (Priority 1)**: Manages LCD output, processing event messages from the queue and displaying continuous data.

Higher-priority tasks (Ultrasonic and Speed) ensure timely sensor data processing, while the lower-priority Display task handles non-critical output.

5.2 Interrupt Handling

Interrupts are configured for the limit switch, ignition button, and gear selector switches. Each ISR updates shared variables within a mutex-protected critical section and sends an event message to the queue using `xQueueSendFromISR`. The gear selection ISR ensures robust handling by defaulting to Neutral in case of invalid inputs.

5.3 Peripheral Initialization

The system initializes the following peripherals:

- **GPIO**: Configured for sensor inputs, actuator outputs, and LCD control lines.
- **ADC**: Set up for single-ended sampling of the potentiometer.
- **LCD**: Initialized in 8-bit mode with commands for display control and cursor movement.

6 Challenges and Solutions

- **Task Synchronization**: Multiple tasks and ISRs access shared variables, risking data corruption. A mutex (`xSharedDataMutex`) was implemented to protect these variables, ensuring exclusive access.
- **Event Handling**: Coordinating events (door, ignition, gear changes) across tasks was complex with global flags. A queue (`xEventQueue`) was introduced to send structured event messages to the display task, improving modularity and reliability.
- **LCD Timing**: The LCD required precise timing for command and data operations. A custom delay function was implemented to ensure reliable communication.
- **Interrupt Debouncing**: Mechanical switches could generate multiple interrupts. Hardware pull-up/pull-down resistors and interrupt edge triggering minimized false events.
- **Real-Time Constraints**: Task priorities and periodic delays were tuned to meet real-time requirements, ensuring timely execution of critical tasks.

7 Testing and Validation

The system was tested in a controlled environment with the following scenarios:

- **Distance Measurement**: Objects were placed at varying distances to verify ultrasonic sensor accuracy and buzzer response.

- **Speed Simulation**: The potentiometer was adjusted to simulate speeds above and below 20 km/h, confirming automatic door locking and correct event queuing.
- **Input Events**: Limit switch, ignition button, and gear switches were toggled to validate interrupt handling, queue message delivery, and LCD updates.
- **System Integration**: The system was run continuously to ensure tasks operated without crashes, priority inversions, mutex-related deadlocks, or queue overflows.

All tests confirmed the system's reliability and responsiveness, with the LCD accurately reflecting system states and the buzzer providing appropriate alerts.

8 Conclusion

The RTOS-based car monitoring system successfully demonstrates the application of FreeRTOS in an embedded automotive system. By leveraging task scheduling, interrupt handling, mutex-based synchronization, and queue-based event communication, the system provides real-time monitoring and control of critical vehicle parameters. The modular design and robust synchronization mechanisms ensure scalability and reliability, making it a strong foundation for further enhancements.

9 Future Improvements

- **Additional Sensors**: Incorporate temperature or tire pressure sensors for comprehensive vehicle monitoring.
- **User Interface Enhancements**: Add a graphical LCD or touchscreen for improved user interaction.
- **Power Optimization**: Implement low-power modes to reduce energy consumption during idle states.
- **Fault Tolerance**: Add error handling for queue overflows or mutex acquisition failures to enhance system robustness.