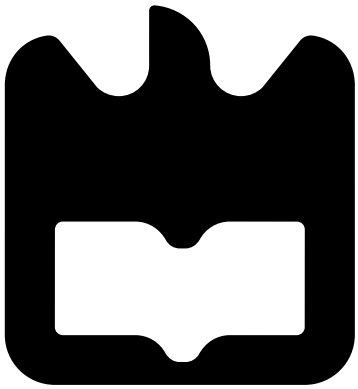

Zephyr RTOS Project Report

107474-Joseane Pereira
109050-Gabriel Costa
Universidade de Aveiro, DETI

December 23, 2024

Contents

1 Introduction

The aim of this project is to apply the Linux Real-Time Services and the Real-Time Model to the development of a real-life inspired real-time application. The project encompasses a set of cooperating tasks, involving synchronization, shared resources, access to a real-time database, etc.

2 Architecture

2.1 System Overview

In this project, the real-time monitoring system is structured around several primary tasks, each with distinct roles that simulate components in an industrial environment:

- **Task 0**: Responsible for updating the RTDB with the button states.
- **Task 1**: Updates the LED states based on the button states from the RTDB.
- **Task 2**: Validates RTDB entries and resets them if they are corrupted.
- **Task 3**: Placeholder for future tasks or additional functionality.

2.2 Data Structures and Access Methods

The primary data structure used in this project is the Real-Time Database (RTDB). The RTDB is used to store the states of LEDs and buttons, ensuring synchronized access and updates by different tasks.

2.3 Synchronization Mechanisms

The implementation relies on multiple threads that perform specific tasks concurrently. To ensure proper coordination, various synchronization methods are applied:

- **Thread Suspension and Resumption**: Tasks suspend themselves after execution and are resumed by the scheduler.
- **Mutexes**: Used to control access to shared resources and prevent data corruption.

3 Task Priorities and Activation Rates

Each task in the system is assigned a specific priority and activation rate to ensure timely execution and prevent task starvation. The following table outlines the task priorities and activation periods.

Task	Priority	Activation Period	Description
Task 0	1	50ms	Updates the RTDB with the button states.
Task 1	2	100ms	Updates the LED states based on the button states from the RTDB.
Task 2	1	100ms	Validates RTDB entries and resets them if they are corrupted.
Task 3	5	150ms	Placeholder for future tasks or additional functionality.

Table 1: Task Priorities and Activation Periods

4 System Schedulability

To confirm that all tasks meet their deadlines, we perform a schedulability analysis using the system's utilization factor.

Utilization Factor (U): Given that each task is independent and periodic, we can calculate the utilization factor U using:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

where C_i is the computation time and T_i is the period of each task. Assuming each task completes within its assigned period, this calculation helps ensure that the system remains schedulable.

5 Task Execution Sequence and Relevant Events

The system follows a well-defined sequence to maintain real-time performance. Below is the order of task execution and relevant interactions:

1. **Task 0**: Updates the RTDB with the button states.
2. **Task 1**: Updates the LED states based on the button states from the RTDB.
3. **Task 2**: Validates RTDB entries and resets them if they are corrupted.
4. **Task 3**: Placeholder for future tasks or additional functionality.

5.1 Relevant Events and Inter-Task Communication

RTDB Operations: The RTDB is critical for data synchronization:

- **Task 0** writes to the RTDB, while **Task 1** and **Task 2** read from it.
- Mutexes manage concurrent access to prevent data corruption.

6 Tests

The tests for the scheduler verify that tasks are added correctly and meet their deadlines. The test suite includes:

- **Scheduler Initialization Test**: Verifies that the scheduler initializes correctly.
- **Task Addition Test**: Verifies that tasks can be added and appear in the task table.
- **Task Deadline Test**: Verifies that tasks meet their deadlines.

7 Results

The tests confirm that the scheduler initializes correctly, tasks are added as expected, and all tasks meet their deadlines within the specified margins.

8 Conclusion

In this project, a multi-threaded system was successfully implemented to manage tasks in a real-time environment. The scheduler ensures that tasks are executed periodically and meet their deadlines. The RTDB provides synchronized access to shared data, preventing data corruption. Overall, the project demonstrates effective use of periodic task scheduling, synchronization, and real-time data management.