

**EDF Scheduler**

**Project**

***Team Members***

**Arafa Arafa**

**Anas Mahmoud**

**Bassel Yasser**

**Mahmoud Adel**

**Youssef Abbas**

Table of Contents

[Project Description 2](#_Toc140394732)

[Requirements 2](#_Toc140394733)

[Tasks 3](#_Toc140394734)

[Hyper period 3](#_Toc140394735)

[CPU Load 4](#_Toc140394736)

[URM Calculation 4](#_Toc140394737)

[Time demand Calculation 5](#_Toc140394738)

[Simulation on SimSo 8](#_Toc140394739)

# Project Description

The EDF Scheduler project aims to implement an Earliest Deadline First (EDF) scheduling algorithm using FreeRTOS. The EDF scheduler is a dynamic priority-based preemptive scheduling policy commonly used in real-time systems. This project involves implementing the necessary changes outlined in a provided thesis, creating tasks with specific criteria, verifying the system implementation, and optionally enhancing the functionality of the scheduler.

# Requirements

The following requirements must be fulfilled to successfully complete the EDF scheduler implementation project:

* Read the provided thesis, specifically Chapter 2: "FreeRTOS Task Scheduling" and Chapter 3: "EDF Scheduler."
* Implement the changes mentioned in Chapter 3.2.2: "Implementation in FreeRTOS." The changes should be made in the tasks.c source file.
* Implement any missing changes not mentioned in the thesis:
* Modify the prvIdleTask function to ensure the idle task always has the farthest deadline.
* In the xTaskIncrementTick function, calculate the new task deadline and insert it in the correct position in the EDF ready list for every tick increment.
* Modify the preemption mechanism in the xTaskIncrementTick function, so that a task with a sooner deadline preempts a task with a larger deadline.
* Implement four tasks using the EDF scheduler. The tasks and their criteria are as follows:
  + Task 1: "Button\_1\_Monitor," Periodicity: 50ms, Deadline: 50ms.
  + Task 2: "Button\_2\_Monitor," Periodicity: 50ms, Deadline: 50ms.
  + Task 3: "Periodic\_Transmitter," Periodicity: 100ms, Deadline: 100ms.
  + Task 4: "Uart\_Receiver," Periodicity: 20ms, Deadline: 20ms.
  + Add two additional tasks to simulate a heavier load:
  + Task 5: "Load\_1\_Simulation," Periodicity: 10ms, Deadline: 10ms, Execution time: 5ms.
  + Task 6: "Load\_2\_Simulation," Periodicity: 100ms, Deadline: 100ms, Execution time: 12ms.
* Implement all the tasks mentioned above in the same main.c source file.

# Tasks

|  |  |  |
| --- | --- | --- |
| Task id | Periodicity | Execution time (ms) |
| T1\_Button\_1\_Monitor | 50 | 0.0144 |
| T2\_Button\_2\_Monitor | 50 | 0.0144 |
| T3\_Periodic\_Transmitter | 100 | 0.013 |
| T4\_Uart\_Receiver | 20 | 0.0246 |
| T5\_ Load\_1\_Simulation | 10 | 5 |
| T6\_Load\_2\_Simulation | 100 | 12 |

The following table is shown the tasks with each (Periodicity, Deadline, Busy time)

|  |  |  |  |
| --- | --- | --- | --- |
| Task id | Periodicity | Deadline | Busy time  (E\*(P/H)) (ms) |
| T1\_Button\_1\_Monitor | 50 | 50 | 0.0288 |
| T2\_Button\_2\_Monitor | 50 | 50 | 0.0288 |
| T3\_Periodic\_Transmitter | 100 | 100 | 0.01308 |
| T4\_Uart\_Receiver | 20 | 20 | 0.0984 |
| T5\_ Load\_1\_Simulation | 10 | 10 | 25 |
| T6\_Load\_2\_Simulation | 100 | 100 | 12 |
| **Total** | | | 37.169 |

# Hyper period

Hyper period = LCM (tasks periodicity) =LCM (100,20,10,50) = 100

# 

# CPU Load

CPU load = (Total busy time / Hyper period)

CPU load = [(13\*2) (0.014\*2) +0.013+(5\*0.0132) +(5\*10)+12 ]/100

= 62.831%

# URM Calculation

U=

|  |  |  |  |
| --- | --- | --- | --- |
| Task id | Periodicity | Execution time | U |
| T1\_Button\_1\_Monitor | 50 | 0.0144 | 0.0144/50 |
| T2\_Button\_2\_Monitor | 50 | 0.0144 | 0.0144/50 |
| T3\_Periodic\_Transmitter | 100 | 0.013 | 0.013/100 |
| T4\_Uart\_Receiver | 20 | 0.0246 | 0.0246/20 |
| T5\_ Load\_1\_Simulation | 10 | 5 | 5/10 |
| T6\_Load\_2\_Simulation | 100 | 12 | 12/100 |
| Total | | | 0.6219 |

U = 0.6219 (62.19%)

URM= = 0.7347 (73.47%)

U ? URM🡪 U < URM

# Time demand Calculation

w(t)=

**Task1**

w(20) =

w(25) =

w(40) =

w(50) =

*: - w(50) < 50 , T1 is schedulable*

**Task2**

w(20) =

w(25) =

w(40) =

w(50) =

*: - w(50) < 50 , T2 is schedulable*

**Task3**

w(50) =

w(70) =

w(90) =

w(100) =

*: - w(100) < 100 , T3 is schedulable*

**Task4**

w(10) =

w(20) =

: - *w(20) < 20 , T4 is schedulable*

**Task5**

w(1) = 5 + 0 = 5

w(2) = 5 + 0 = 5

w(5) = 5 + 0 = 5

: *- w(5) < 5 , T4 is schedulable*

**Task6**

w(50) =

w(70) =

w(90) =

w(100) =

: - *w(100) < 100 , T6 is schedulable*

**NB: After we made URM calculation and Time demand calculation on this task , we noticed that the two calculations leading to the same result that the system is schedulable**

# Simulation on SimSo

A screenshot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated