

# Real-Time Operating System (RTOS) Design and Implementation

## Abstract

This project proposes the design and implementation of a real-time operating system (RTOS) to efficiently manage a set of periodic tasks. The tasks encompass the periodic display of the message "Working," Fahrenheit to Celsius temperature conversion, multiplication of large integer numbers, and binary search in a pre-defined list.

## 1. Introduction

A real-time operating system is crucial for coordinating the execution of critical tasks in various domains such as embedded systems and industrial control. This project focuses on designing an RTOS capable of handling a diverse set of tasks, each with specific requirements.

## 2. Methodology

### 2.1 Task Analysis

Each task has undergone a detailed analysis to determine its execution time, with a specific emphasis on the worst-case execution time (WCET). The analysis involved the creation of `measuretime.py` (found in the git repository) which independently launches the `time` command in the command prompt, providing the WCET for each task.

After 10,000 iterations, the WCET results are as follows:

Task	Binary	Temperature	Multiplication	Working
WCET (in ms)	0.009	0.007ms	0.066	0.006

We need to find a period for each task. We can resolve this equation:

$$\frac{0.009}{0.009 * x} + \frac{0.007}{0.007x} + \frac{0.066}{0.066x} + \frac{0.006}{0.006x} \leq 1$$

$$\frac{1}{x} + \frac{1}{x} + \frac{1}{x} + \frac{1}{x} \leq 1$$

$$\frac{4}{x} \leq 1 \text{ So } x \geq 4$$

Multiplying by 4.5 for safety, the periods were calculated in seconds and milliseconds.

Task	Binary	Temperature	Multiplication	Working
WCET (in s)	0.009	0.007	0.066	0.006
Period (in s)	0.0405	0.0315	0.297	0.027
Period(in ms)	40.5	31.5	297	27

We can verify:  $\frac{0.009}{0.0405} + \frac{0.007}{0.0315} + \frac{0.066}{0.297} + \frac{0.006}{0.027} = 0.88 \leq 1$

So, tasks should be schedulable and can be verified in FreeRTOS.

## 2.2 System Architecture

The RTOS is based on FreeRTOS and uses a First-Come First-Serve (FCFS) scheduler. Task creation, scheduling, and synchronization are handled in accordance with each task's requirements.

Tasks have been implemented using FreeRTOS, adhering to the specifications defined in the task analysis.

## 3. Results

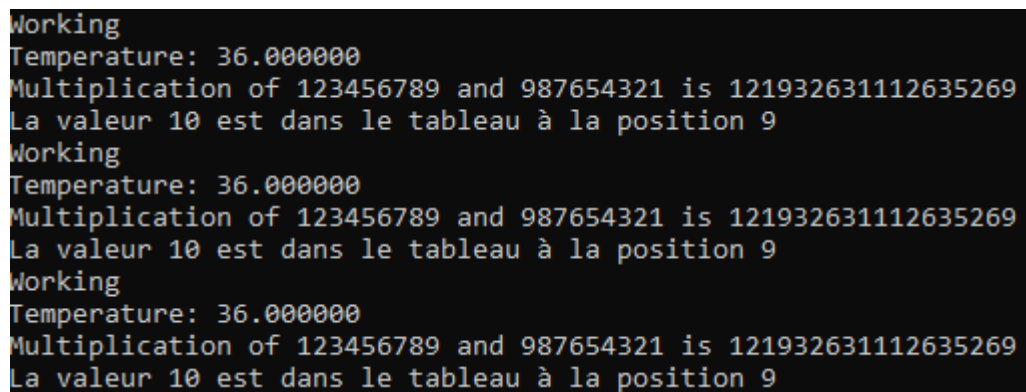
### 3.1 Task Schedulability Analysis

Calculations show that the entire task set can be successfully scheduled under the defined time constraints.

### 3.2 Execution Observations

We can observe the scheduling seems to perfectly process in the command prompt.

He is a screenshot of what we obtain.



```
Working
Temperature: 36.000000
Multiplication of 123456789 and 987654321 is 121932631112635269
La valeur 10 est dans le tableau à la position 9
Working
Temperature: 36.000000
Multiplication of 123456789 and 987654321 is 121932631112635269
La valeur 10 est dans le tableau à la position 9
Working
Temperature: 36.000000
Multiplication of 123456789 and 987654321 is 121932631112635269
La valeur 10 est dans le tableau à la position 9
```

The tasks appear to repeat as expected without out speed from task to another.

## 4. Conclusion

This project has demonstrated the feasibility of implementing an RTOS to manage a diversified set of tasks. The obtained results seem to be correct but are hard to measure without impacting the scheduling.

I had some difficulties to make my iterations on each task to measure the WCET, so I did a bash script and a Python script, but the Python script appear to give me more precise measures (in bash all is at 0 as if it round).

Finally, I calculated a period for each task according to the condition for scheduling. Then I implemented the founded period in the c script, and everything is ok and execute respecting the scheduling.