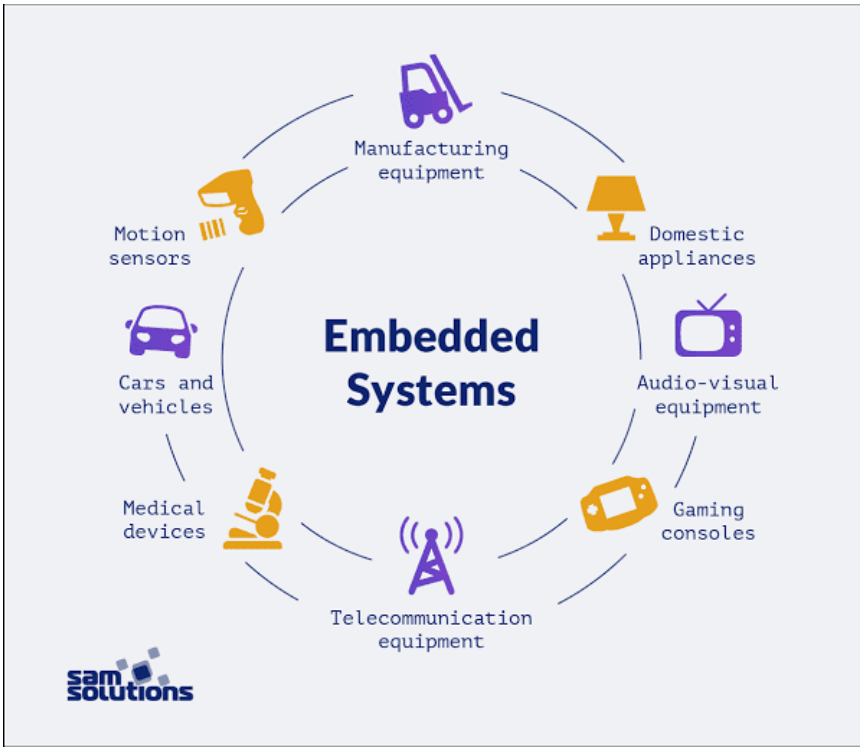
DALFIN Baptiste

LACOTTE Baptiste

Final assignement

Real-Time Systems



Examples of embbeded systems

Friday, 26th April

Mr. Singh JASDEEP

Summary

I. Abstract

The project aims to develop an embedded system using FreeRTOS to manage periodic tasks and perform temperature conversion and binary search operations. It consists of four periodic tasks: one indicating normal operation, another converting a fixed Fahrenheit temperature to Celsius, a third multiplying two large integers and displaying the result, and a fourth performing a binary search on a fixed list. The system utilizes a queue for inter-task communication and a software timer to trigger periodic data transmission. Lastly, the project provides an analysis of system output and suggests optimizations.

II. Introduction

Real-Time Operating Systems (RTOS) play a crucial role in embedded systems where tasks must be executed within strict timing constraints. In this project, we aim to design an RTOS capable of managing a set of periodic and optional aperiodic tasks efficiently. The system will be implemented using FreeRTOS, a popular open-source RTOS for embedded systems. Our task set comprises periodic tasks responsible for various operations such as printing status messages, temperature conversion, numerical operations, and binary searching. To ensure the system's reliability and predictability, it's essential to analyze each task's Worst Case Execution Time (WCET) accurately. This analysis guides the scheduling decisions, ensuring that tasks meet their deadlines consistently. In this report, we detail the design and implementation of the RTOS, including the analysis of task execution times, determination of WCETs, and scheduling using the Fixed-Priority (FP) scheduling algorithm. We provide insights into the schedulability analysis to demonstrate the system's ability to meet timing requirements under various scenarios. Finally, we share the repository containing the complete codebase along with a comprehensive report outlining the project's abstract, methodology, results, and conclusions. This project aims to showcase the practical application of RTOS concepts in real-world embedded systems development.

You can find the link of the git here: https://github.com/bapttit/Final_project.git

III. Methods

The first goal is to create our tasks in the code ipsa_sched.c. Let's start with the first task:

```
if( ulReceivedValue == mainVALUE_SENT_FROM_TASK1)
{
    console_print( "Working as normal\n" );
}
```

1st task: print "working"

```
else if( ulReceivedValue == mainVALUE_SENT_FROM_TASK2)
{
    double Fahrenheit = 15.0;
    double Celsius;
    Celsius = (Fahrenheit-32.0)*(5.0/9.0);
    console_print("%.2f °Fahrenheit = %.2f °Celsius\n", Fahrenheit, Celsius);
```

2nd task: convert Celsius to Fahrenheit

```
}else if( ulReceivedValue == mainVALUE_SENT_FROM_TASK3)
{
    long int a = 462134;
    long int b = 26581;
    long int multiply;
    multiply = a*b;
    console_print("The result is : %ld \n",multiply);
```

3rd task: product

```
}else if( ulReceivedValue == mainVALUE_SENT_FROM_TASK4)
{
    // Création de la liste de 50 éléments
    int array[SIZE];
    for (int i=0 ; i<SIZE; i++) {
        array[i] = i+1;
    }
    // bin est l'élément qu'on recherche
    int bin = 13;
    // Recherche binaire
    int low = 0, high = SIZE-1, middle;
    int found = 0;
    while (low <= high) {
        middle = low+(high-low)/2;
        if (array[middle] == bin) {
            found = 1;
            break;
        } else if (array[middle] < bin) {
            low = middle+1;
        } else {
            high = middle-1;
        }
    }
    console_print("L'élément %d se trouve à l'index : %d\n", bin, middle);
```

4th task: binary search a list of 50 elements

So now let's define priorities, WCETs and periods:

τ_i	c_i	C_i	T_i	p_i
Working	42000 ns	63000 ns	100000 ns	3
Convert	82000 ns	123000 ns	250000 ns	0
Multiply	69000 ns	104000 ns	450000 ns	1
Finding	54000 ns	81000 ns	600000 ns	2

We found execution time by running several times the different tasks and by making the mean. With these periods, we find an hyperperiod such as: $H = 0.009\,s = 9\,000\,000\,ns$. We have a CPU Utilization Time equals to $0.99 < 1$, which is perfect. Now, let's validate our schedule by calculating time response:

$$t_r = \frac{9\,000\,000}{100\,000} \times 42000 + \frac{9\,000\,000}{250\,000} \times 82000 + \frac{9\,000\,000}{450\,000} \times 69000 + \frac{9\,000\,000}{600\,000} \times 54000$$

$$\Leftrightarrow t_r = 7\,680\,000\,ns$$

We can see that $t_r < H$, so it is schedulable. We chose our WCETs by multiply our c_i by 1.4 after several research about order of magnitude concerning this work.

For the priorities we define like this:

```
/* Priorities at which the tasks are created. */
#define mainQUEUE_RECEIVE_TASK_PRIORITY     ( tskIDLE_PRIORITY + 5 )
#define mainQUEUE_SEND_TASK1_PRIORITY       ( tskIDLE_PRIORITY + 1 )
#define mainQUEUE_SEND_TASK2_PRIORITY       ( tskIDLE_PRIORITY + 4 )
#define mainQUEUE_SEND_TASK3_PRIORITY       ( tskIDLE_PRIORITY + 3 )
#define mainQUEUE_SEND_TASK4_PRIORITY       ( tskIDLE_PRIORITY + 2 )
```

Now we can define each period for each task:

```
/* The rate at which data is sent to the queue.  The times are converted from
 * milliseconds to ticks using the pdMS_TO_TICKS() macro. */
#define mainTASK1_SEND_FREQUENCY_MS         pdMS_TO_TICKS( 100UL )
#define mainTASK2_SEND_FREQUENCY_MS         pdMS_TO_TICKS( 250UL )
#define mainTASK3_SEND_FREQUENCY_MS         pdMS_TO_TICKS( 450UL )
#define mainTASK4_SEND_FREQUENCY_MS         pdMS_TO_TICKS( 600UL )
#define mainTIMER_SEND_FREQUENCY_MS         pdMS_TO_TICKS( 2000UL )
```

And finally, we define the values returned by each function:

IV. Results

```
baptistel@BaptisteL:/mnt/c/FreeRTOSv202107.00/FreeRTOS/Demo/Posix_GCC$ cd build
baptistel@BaptisteL:/mnt/c/FreeRTOSv202107.00/FreeRTOS/Demo/Posix_GCC/build$ ./posix_demo

Trace started.
The trace will be dumped to disk if a call to configASSERT() fails.
Starting ipsa sched
Working as normal
Working as normal
15.00 °Fahrenheit = -9.44 °Celsius
Working as normal
Working as normal
The result is : 12283983854
Working as normal
15.00 °Fahrenheit = -9.44 °Celsius
Working as normal
L'élément 13 se trouve à l'index : 12
Working as normal
15.00 °Fahrenheit = -9.44 °Celsius
Working as normal
Working as normal
The result is : 12283983854
Working as normal
15.00 °Fahrenheit = -9.44 °Celsius
Working as normal
Working as normal
L'élément 13 se trouve à l'index : 12
15.00 °Fahrenheit = -9.44 °Celsius
Working as normal
The result is : 12283983854
Working as normal
Working as normal
15.00 °Fahrenheit = -9.44 °Celsius
Working as normal
Working as normal
15.00 °Fahrenheit = -9.44 °Celsius
Working as normal
L'élément 13 se trouve à l'index : 12
The result is : 12283983854
Working as normal
```

Result of the execution

Here's a summary of the results:

1. "Working as normal": the periodic task 1 is functioning normally at each specified time interval.

2. "15°Fahrenheit=-9.44°Celsius": the periodic task 2 successfully converted a temperature to Fahrenheit.

3. "The result is: 12283983854": the multiplication of the two large numbers in periodic task 3 is correct.

4. "L'élément 13 se trouve à l'index: 12": The binary search in periodic task 4 found element 13 at index 12 in the array.

It seems that the priorities are respected. Each task executes according to its defined priority, and their respective messages are printed accordingly.

To optimize our project we can modify the priority. Even if the tasks are simple we can optimized the algorithm and make them more simple to take less time.