

STR (online)

Real Time Systems

Exam2:

FreeRTOS Windows port for the
implementation of an EDF scheduler

FIB Barcelona School of Informatics
UPC Technical University of Catalonia

Antonio Camacho Santiago

May 2020

The 2nd exam of the Real Time Systems course is intended to implement an Earliest Deadline First Scheduler based on the FreeRTOS kernel.

Detailed explanations of the deliverables are shown in 2-exam_edf_freertos_r2.pdf

<https://raco.fib.upc.edu/practiques/practica.jsp?espai=270071&action=view&id=74485>

In order to develop the work proposed for this exam, a pseudo-real time Simulator is delivered with this file.

The pseudo-real time Simulator consists of the FreeRTOS kernel ported to Windows platform. Note that this is not useful to implement hard real-time Systems but to “simulate” them. However, it can be used to implement the EDF scheduler on top of FreeRTOS and test it as in any other microcontroller. The time accuracy of the environment is not very good but it is good enough for testing the new proposed scheduler.

In order to install the FreeRTOS Windows port for simulation, download the str_exam2.zip file from google drive. The link is

https://drive.google.com/open?id=1ElUd3iCjx4g2rmdD6jD_qEz-YSSis3Sz

Note that you need to be logged into your upc account in order to download it.

The zip file contains all the programs needed to start coding the new solution. In particular, the eclipse IDE for C/C++ Developers version 2020-03 M1 (4.15.0), the minGW minimalist development environment for native Microsoft Windows Applications, the complete FreeRTOS kernel for all the supported microcontrollers, and an eclipse workspace with the template to start developing the project.

All the software used along this work is free of charge, and can be download from external sources although it is not recommended because it should work flawlessly without installing anything.

Note that the FreeRTOS simulation environment can be also deployed in Linus or Mac operating systems although it is not covered in this document.

The eclipse application has been download from:

<https://www.eclipse.org/downloads/packages/release/2020-03/r/eclipse-ide-cc-developers-includes-incubating-components>

The minGW provided within the zip file has been downloaded from Mathworks as an external compiler for Matlab. It can be download from any other external source.

The FreeRTOS source code has been downloaded from

<https://www.freertos.org/a00104.html>

Additionally, you can install Matlab by creating a new account in mathworks with the upc e-mail. Or even better, you can use the Online Matlab application with a new mathworks's account based on the upc-email.

Also, it is important to highlight that you can work in remote desktops by connecting to the fib laboratories. Details can be found in:

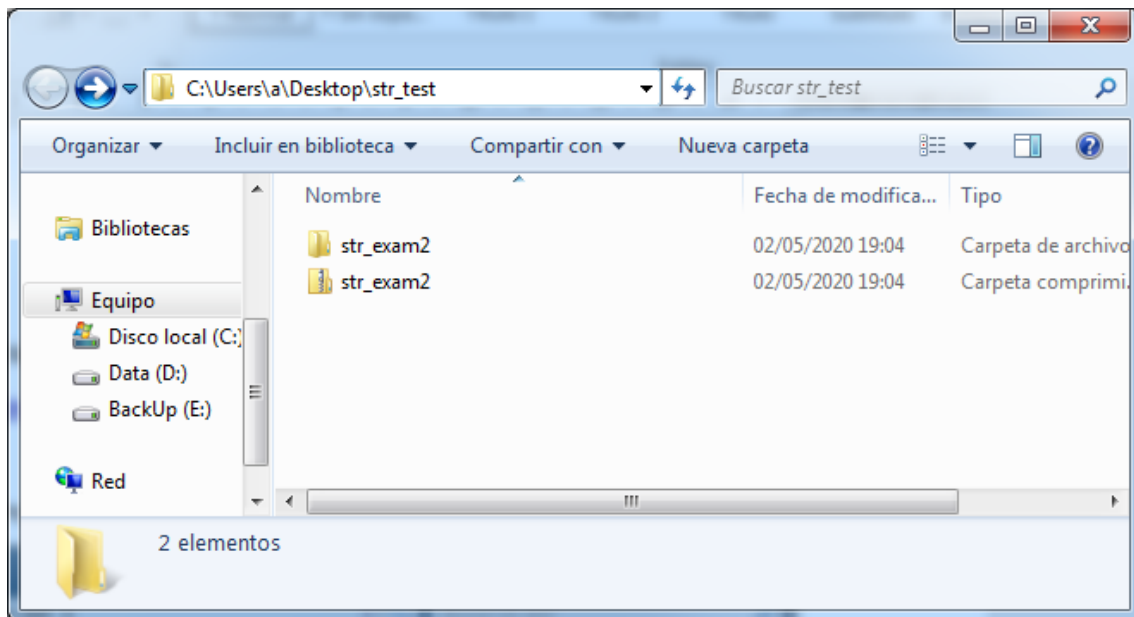
<https://www.fib.upc.edu/es/la-fib/servicios-tic/conexion-remota-los-pc-de-los-laboratorios>

Step by step procedure to start with the FreeRTOS emulation.

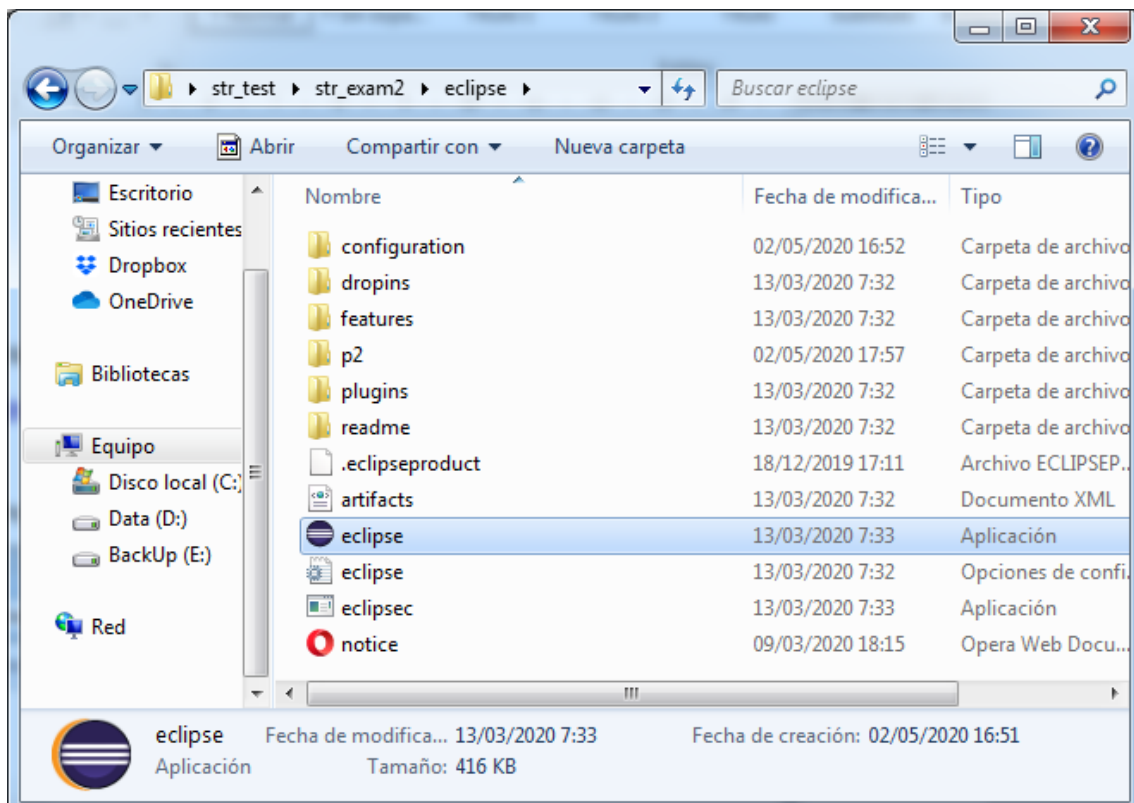
Download the zip file from

https://drive.google.com/open?id=1ElUd3iCjx4g2rmdD6jD_qEz-YSsis3Sz

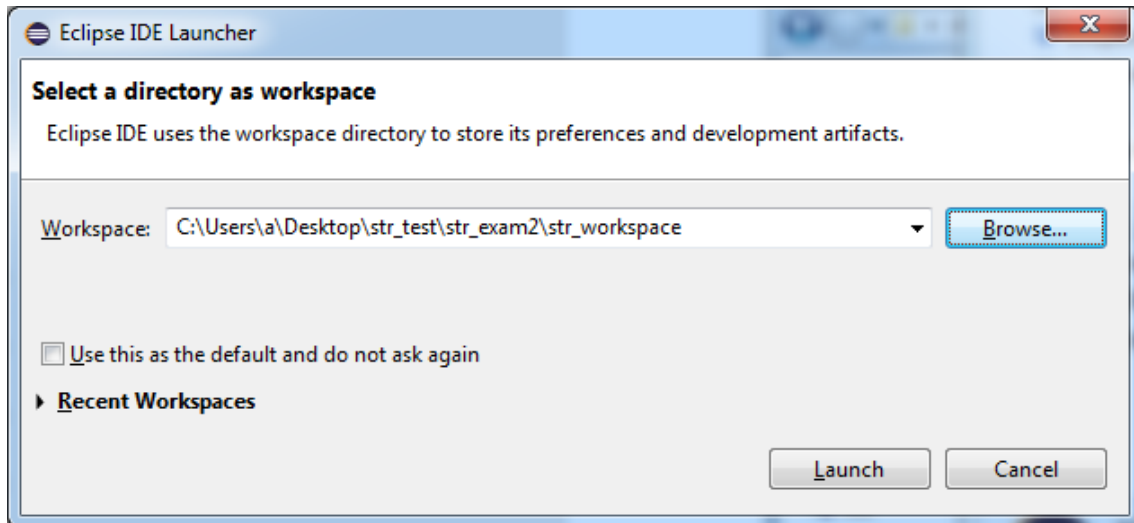
Unzip the file in any folder, for example C:\Users\A\Desktop\str_test



Open the str_exam2 folder, go to the eclipse folder and double click into the eclipse.exe icon.



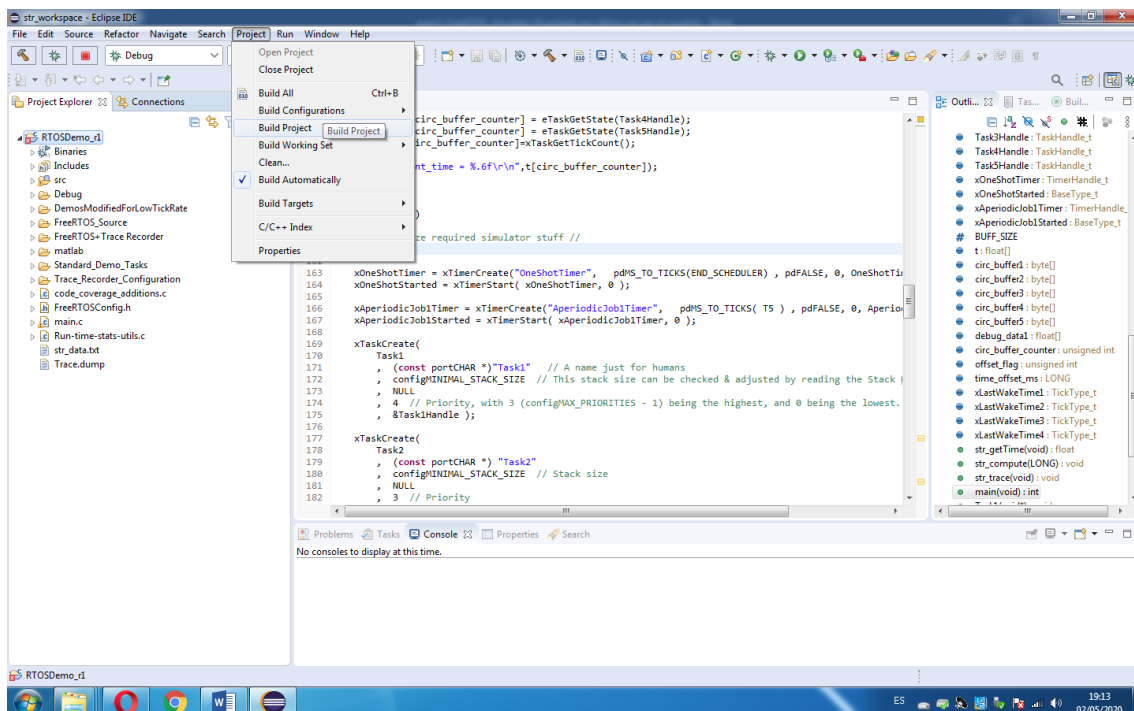
Eclipse will launch, after few seconds the workspace folder will be required. In this case the folder is str_workspace that has also been released with the zip file



For a different folder, click Browse... and select the str_workspace provided in the zip file.

Once eclipse has started, a project will be shown in the Project Explorer. The name of the template is RTOSDemo_r1.

To compile the project, go to the Project menu and click Build project

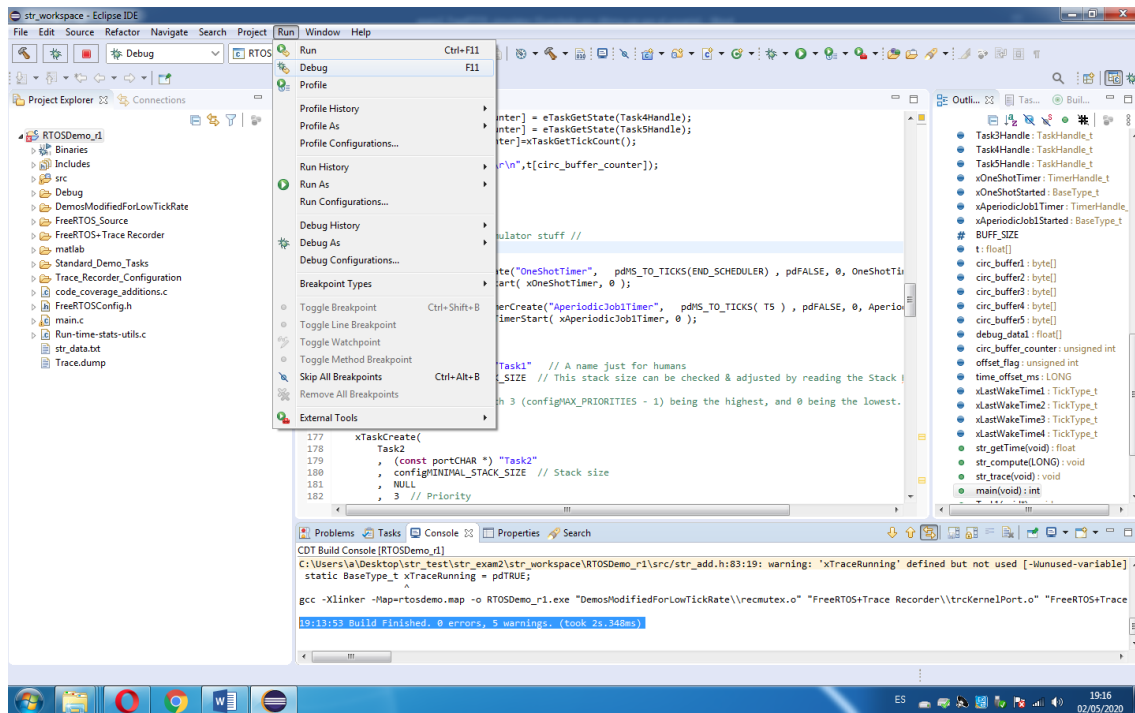


If the compilation is successful, the following message should appear in the console view

19:13:53 Build Finished. 0 errors, 5 warnings. (took 2s.348ms)

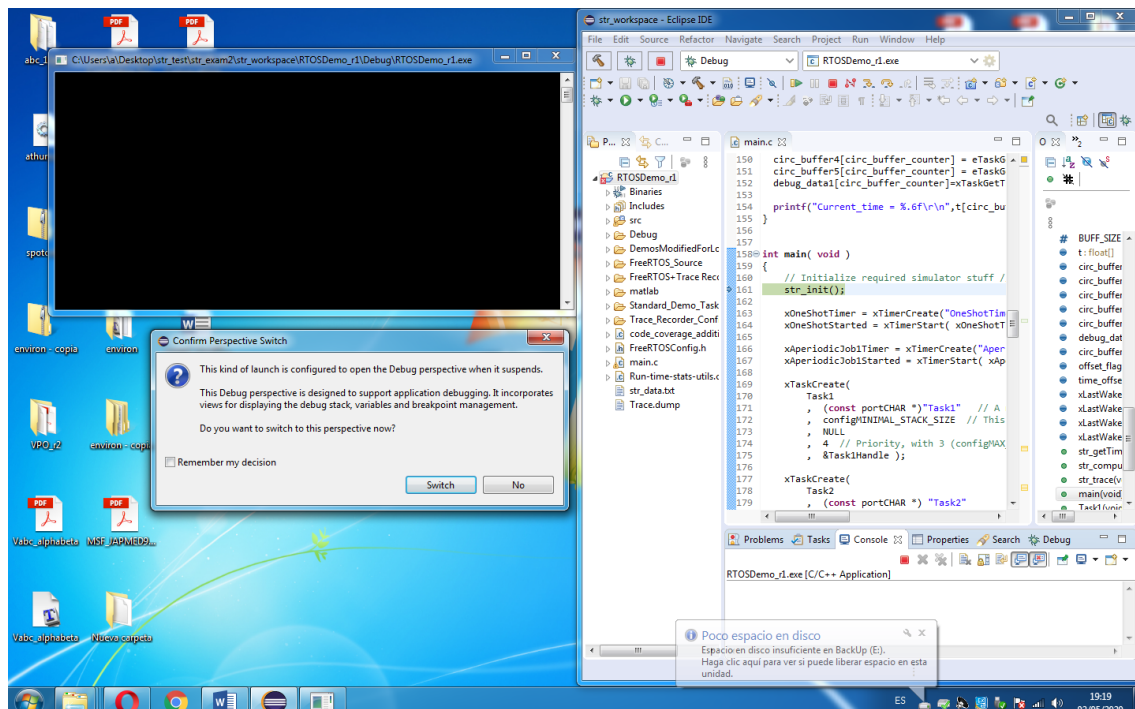
If the compiler is not started, probably it cannot find the mingw toolchain. In such a case, copy the mingw file into c:\ and try again.

Now it is time to debug the program. Open the Run menu and click Debug F11



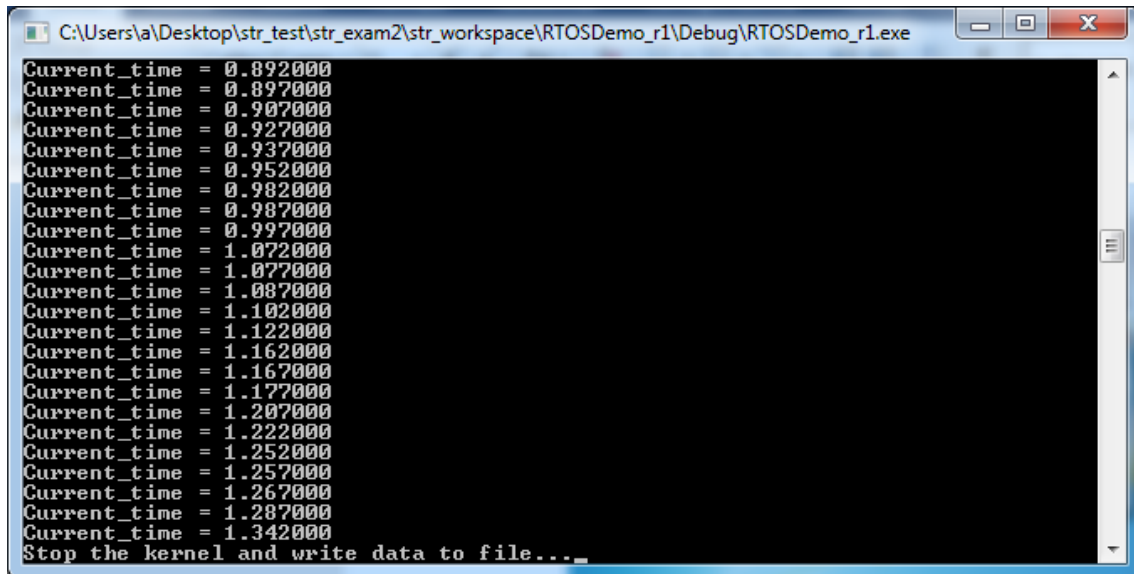
A new output console is opened for printf messages.

Also, eclipse ask to switch the perspective to start the debugger. Click Switch to enter the debug mode.



Start running the code by clicking on the Run menu, and then click Resume.

The console should present the following information:



```
C:\Users\al\Desktop\str_test\str_exam2\str_workspace\RTOSDemo_r1\Debug\RTOSDemo_r1.exe
Current_time = 0.892000
Current_time = 0.897000
Current_time = 0.907000
Current_time = 0.927000
Current_time = 0.937000
Current_time = 0.952000
Current_time = 0.982000
Current_time = 0.987000
Current_time = 0.997000
Current_time = 1.072000
Current_time = 1.077000
Current_time = 1.087000
Current_time = 1.102000
Current_time = 1.122000
Current_time = 1.162000
Current_time = 1.167000
Current_time = 1.177000
Current_time = 1.207000
Current_time = 1.222000
Current_time = 1.252000
Current_time = 1.257000
Current_time = 1.267000
Current_time = 1.287000
Current_time = 1.342000
Stop the kernel and write data to file..._
```

Stop the debugging session by clicking terminate button.

Note that the debugger can be executed step by step by using the step over and step in buttons to help on the coding process.

Additionally, it is possible to debug var values in the Expressions tab on the top right part of the eclipse by clicking on the add new expression.

The code is made of 5 tasks running at different priorities, with different periods and execution times. The code is self-explained although you can ask for further information at antonio.camacho.santiago@upc.edu.

At the end of the execution, a timer stops the scheduler and starts writing the str_trace information inside the circular buffers to a str_data.txt file. The state of each tasks and the time at each traceTASK_SWITCHED_IN() are written to the text file. It can be modified to write anything you want.

The kernel modifications are

trcKernelPort.h call to custom str_trace() in traceTASK_SWITCHED_IN()

#define configTICK_RATE_HZ in FreeRTOSConfig.h as 100

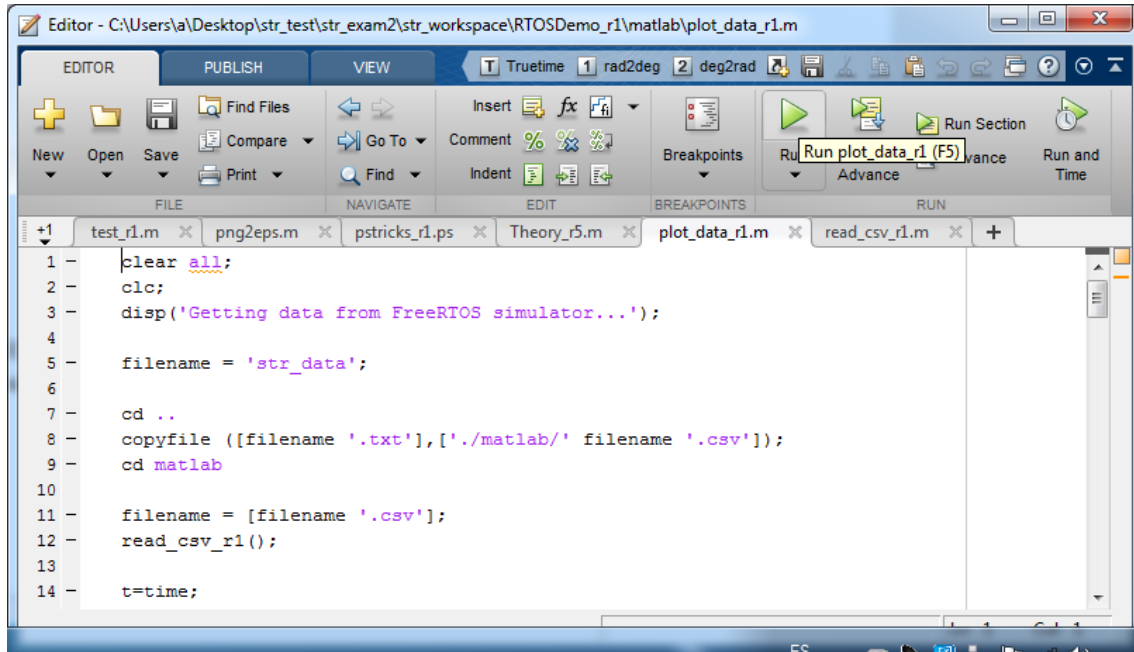
#define portTICK_PERIOD_MS as 9 in portmacro.h

The first modification allows to include the str_trace() functionality.

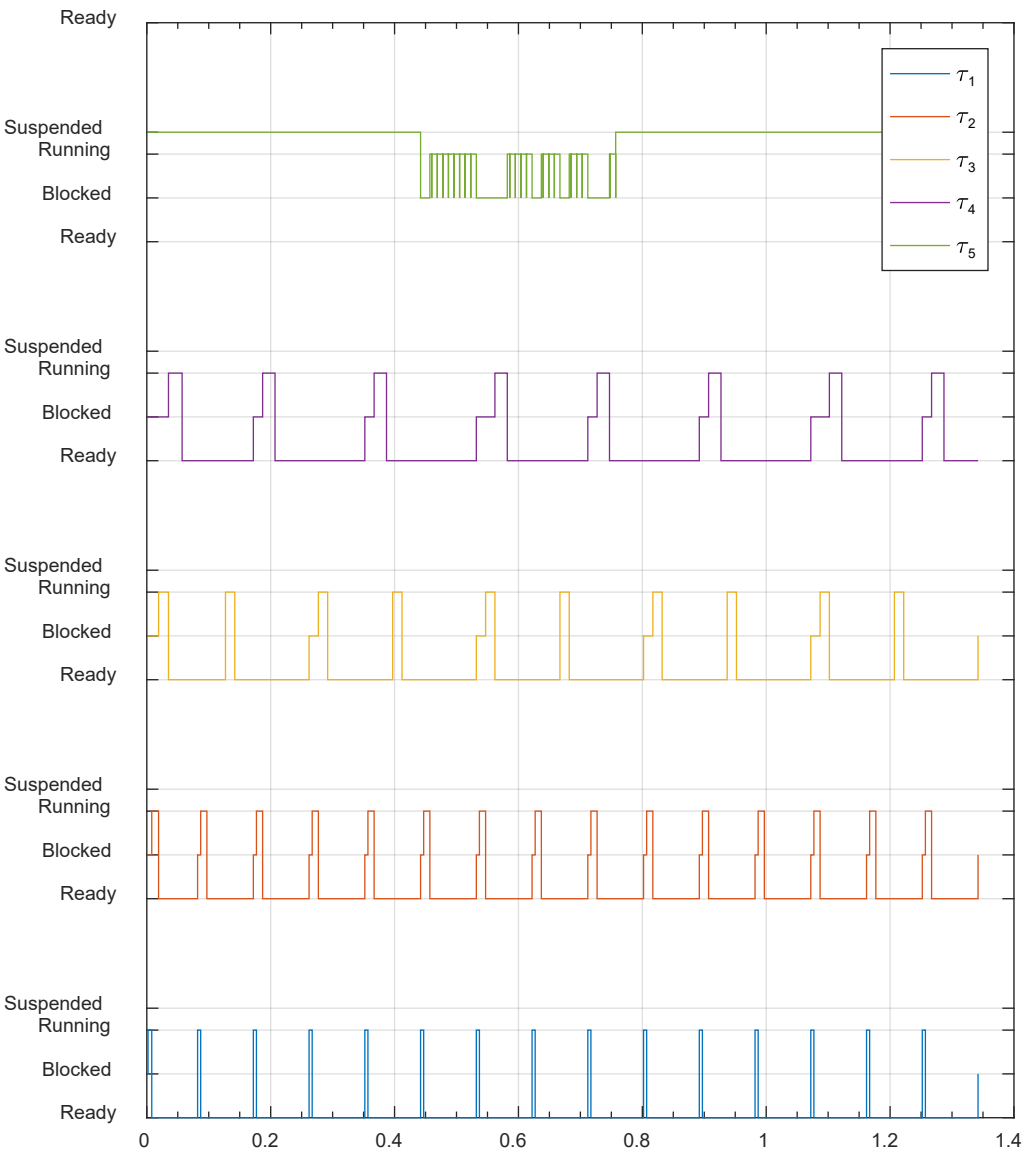
The second modification allows to set the tick rate at 100 Hz.

This macro works in conjunction with the third modification that configures the timer interrupt in milliseconds. Faster processors will require a higher value for this value.

Finally, an additional matlab script `..\str_workspace\RTOSDemo_r1\matlab\process_data_r1.m` is provided to plot the data that has been written to the text file. This custom application is optional, and it is provided to reduce the development time of the EDF scheduler by using a graphical interface. To use the script, you need to install matlab. Double click the matlab script and click the Run button



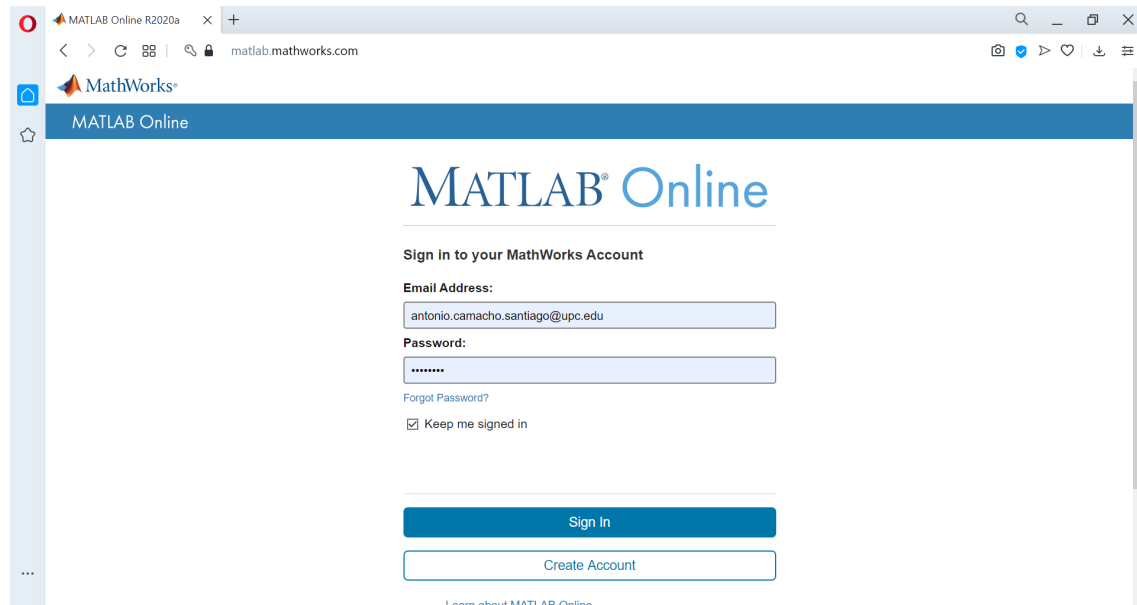
The results are shown in the next figure



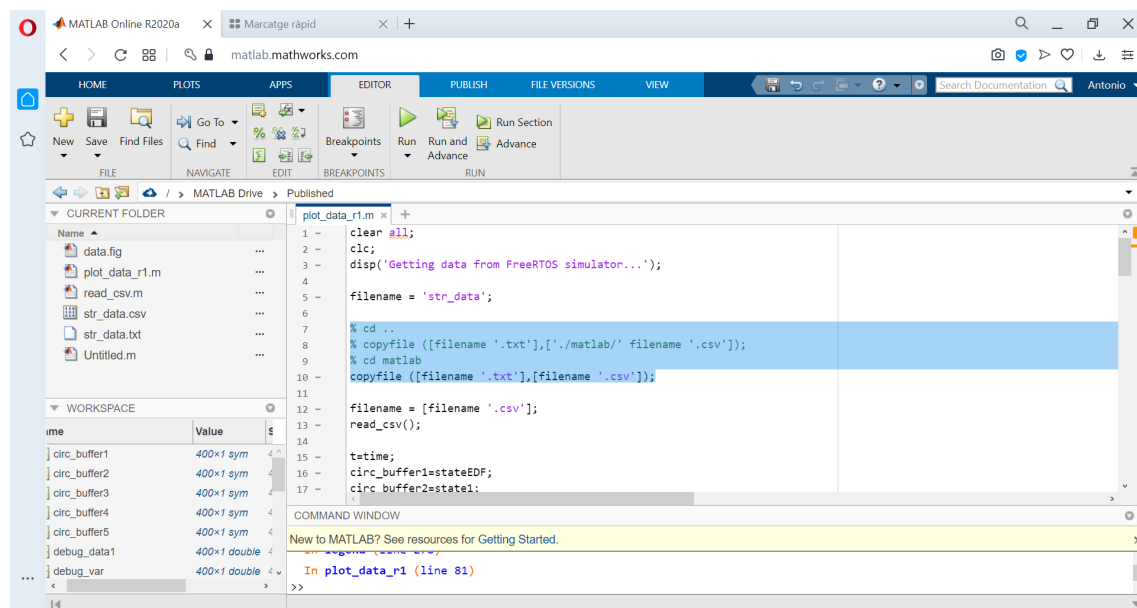
If you prefer to use the Online Matlab, open a web browser and go to

<https://es.mathworks.com/products/matlab-online.html>

You have to create an account with the upc e-mail, then sign in



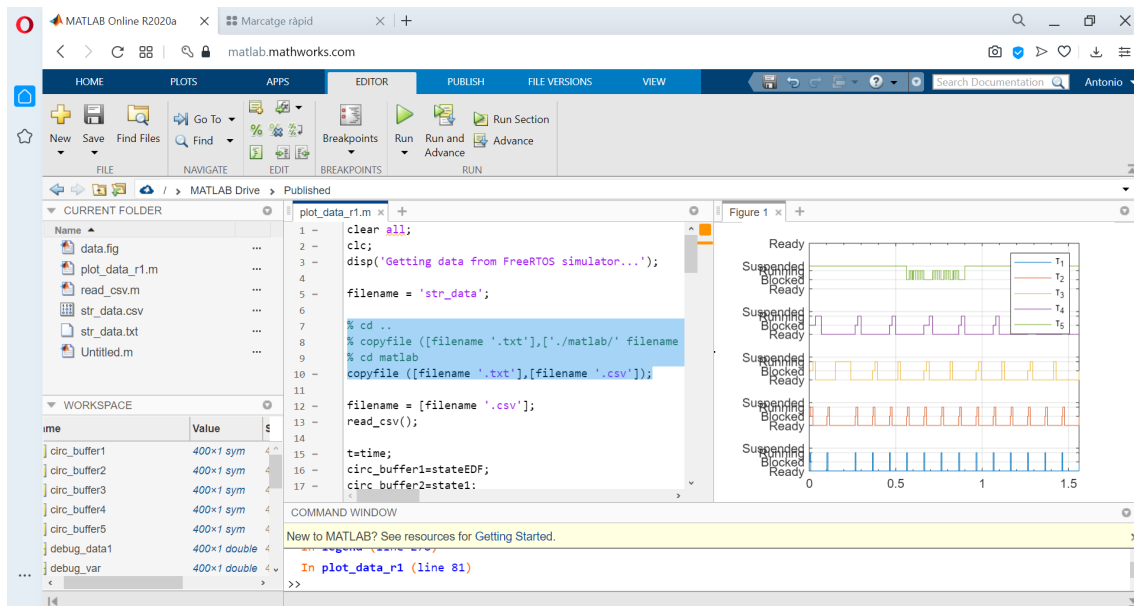
Drag and drop the str_data.txt into the working folder. Also drag and drop the plot_data_r1.m and read_csv.m files. Open the plot_data_r1.m script by double clicking it.



In order to open the text file and convert to csv from the same folder, modify lines 7 to 9 in the script as follows:

```
% cd ..
% copyfile ([filename '.txt'], ['./matlab/' filename '.csv']);
% cd matlab
copyfile ([filename '.txt'], [filename '.csv']);
```

Run the script and the result should be something similar to



Each time a new data has to be plotted, it is mandatory to upload the str_data.txt file and run the plot_data_r1.m script.

This is the template to start implementing the EDF scheduler on top of the FreeRTOS kernel.

Any doubt, don't hesitate to contact me.