Peer-graded Assignment: Assignment 4

Submit by October 29, 11:59 PM PDT

Important Information
It is especially important to submit this assignment before the deadline, October 29, 11:59 PM PDT, because it must be graded by others. If you submit late, there may not be enough classmates around to review your work. This makes it difficult - and in some cases, impossible - to produce a grade. Submit on time to avoid these risks.

Development of Real-Time Systems – Assignment 4

Overview

In this assignment we are going to use our previously learned skills in FreeRTOS and SimSo to **schedule non-periodic jobs**. First we will start off by setting up a set of periodic tasks in SimSo and then extend the schedule with a non-periodic job. We will compare difference schedulers here and argue for which one is better for different types of tasks. Then we will use FreeRTOS to implement non-periodic jobs in practice. With the previously learned skills in measuring time, we will measure the response time of non-periodic jobs and argue for or against a given schedule.

Pre-requisite

It is recommended to use the free software Visual Studio Express for this assignment. The FreeRTOS system you are about to setup will execute in Visual Studio Express, and output from the system is available from its console. More help Here about setting up Visual Studio Express with the FreeRTOS project.

It is also possible to use the free software Eclipse for this assignment. Notes on setting up Eclipse with the FreeRTOS project Here.

After completing this assignment, you will be able to

- Schedule non-periodic jobs in SimSo
- Implement non-periodic jobs in FreeRTOS
- Measure the response time of non-periodic jobs in FreeRTOS

Simulation assignment

-Download the SimSo scheduler from Here!

-Install SimSo and familiarize yourself with the tool. Documentation available Here.

Consider the tasks T1(3, 0.5), T2(4, 1.5, 3), T3(7, 1.0, 5) and the EDF scheduler. A sporadic job arrives at t=50 having the execution time of 10 and a relative deadline of 30. Create the sporadic task in SimSo by selecting: "generate task set" and then list of act. Dates to the release time

-Use SimSo to schedule the task set and provide a report answering the following questions:

- What is the minimum/maximum/average response time of all tasks?
- Is any task missing the deadline? Which task? Where?

- Is the sporadic job meeting its deadline?
- What is the response time for the sporadic job?

Consider the tasks T1(3, 0.5), T2(4, 1.5, 3), T3(7, 1.0, 5) and the RM scheduler. A sporadic job arrives at t=50 having the execution time of 10 and a relative deadline of 30. Create the sporadic task in SimSo by selecting: ”generate task set” and then list of act. Dates to the release time

-Use SimSo to schedule the task set and provide a report answering the following questions:

- What is the minimum/maximum/average response time of all tasks?
- Is any task missing the deadline? Which task? Where?
- Is the sporadic job meeting its deadline?
- What is the response time for the sporadic job?
- Which scheduler is better is better in this example; EDF or RM?

Programming assignment

In this programming assignment, you will handle aperiodic jobs.

-Download the FreeRTOS project Here

-Import and build the project into Visual Studio Express 2015. More information from our Documentation.

-Now run the project. More information from our Documentation

-Familiarize yourself with the FreeRTOS API and locate to the main() function in the FreeRTOS project in Visual Studio Express

-Here create a task "matrixtask" containing the functionality given in Assignment 2.

(Copy the C-code from matrixtask in Assignment 2)

-Add a software timer in main() to trigger a software interrupt every 5 seconds. (Documentation found Here.)

-Define a Timer callback function outside main() with the following functionality:

```
/* A variable to hold a count of the number of times the timer expires. */
long lExpireCounters = 0;
void vTimerCallback(TimerHandle_t pxTimer)
{
  printf("Timer callback!\n");
  xTaskCreate((pdTASK_CODE)aperiodic_task, (signed char *)"Aperiodic",
    configMINIMAL_STACK_SIZE, NULL, 2, &aperiodic_handle);
  long lArrayIndex;
  const long xMaxExpiryCountBeforeStopping = 10;
  /* Optionally do something if the pxTimer parameter is NULL. */
  configASSERT(pxTimer);
  /* Increment the number of times that pxTimer has expired. */
  lExpireCounters += 1;
  /* If the timer has expired 10 times then stop it from running. */
  if (lExpireCounters == xMaxExpiryCountBeforeStopping) {
    /* Do not use a block time if calling a timer API function from a
    timer callback function, as doing so could cause a deadlock! */
    xTimerStop(pxTimer, 0);
  }
}

```

-Create an aperiodic task using the following functionality:

```
static void aperiodic_task()
{
  printf("Aperiodic task started!\n");
  fflush(stdout);
  long i;
  for (i = 0; i<1000000000; i++); //Dummy workload
  printf("Aperiodic task done!\n");
  fflush(stdout);
  vTaskDelete(aperiodic_handle);
}

```

Help Center

The following questions should be solved with programming and the questions should be answered in a report:

- Is the system fast enough to handle all aperiodic tasks? Why?
- If not, solve this problem without alter the functionality of any task
- What is the response time of the aperiodic task?
- Provide a screenshot of the running system

Review criteria

less

Everyone enrolled in this course must review at least three other submissions to ensure everyone receives a grade.