Real-Time Systems

Lab 5: Scheduling

Due: March 9, 2017 (Before Class)

Introduction

At this point you have written all the essential tasks required for successful navigation and control of Ringo through an environment. We have, to this point, relied on FreeRTOS's scheduling to ensure timing deadlines are met. But that's no fun!

In this lab you will design and implement an offline cyclic executive schedule that schedules your tasks manually rather than using FreeRTOS. For bonus points you can also implement a Rate Monotonic Scheduler using FreeRTOS.

Cyclic Executive

From class you know how to design a cyclic executive scheduler. For this part of the lab you will need to throw out FreeRTOS and go back to the traditional Arduino structure of just having `setup()` and `loop()` function. The cyclic executive should reside in the `loop()` function. This means you will need to port your tasks to functions call (easy peasy)! Design and implement a cyclic executive schedule for Ringo that accomplishes the same mission objectives that it did in Lab 4. Then compare the performance of the system with this scheduler to the implementation you did in Lab 4.

Here are some (maybe) helpful steps:

1. Demonstrate that your task set is schedulable (consider only periodic tasks for this part) using the exact schedulability test we examined for EDF. This shows us whether or not we should even bother with designing a cyclic executive scheduler. If your task set cannot be scheduled, modify it so that it can be and discuss in the writeup how you modified it and why.
   (a) Remember that to do this you will need to have a period, P_i, and execution time, e_i, for each task. e_i comes from your WCET analysis from lab 4 which you may need to adjust for this lab depending on whether or not your task set is schedulable.
2. Develop a feasible schedule for **all tasks** (including aperiodic tasks) using a cyclic executive. Show your cyclic executive schedule in your lab writeup.
3. Port all your tasks to functions which can be called in the `loop()` function.
4. Implement the cyclic executive using the `loop()` function in the Arduino.
5. Assess performance by comparing Ringo's ability to navigate the same mission you executed in Lab 4. This can be done visually, noting any significant differences in the video.

Bonus 20%: Rate Monotonic Scheduler

For an additional 20% bonus, design and implement a RMS using FreeRTOS. First prove that the task set is RMS schedulable. If your task set doesn't meet the sufficient condition you can use the necessary condition using the "Criticality of First Instances" theorem. Now implement RMS for your system using FreeRTOS.

Here are a few tips:

- FreeRTOS is a fixed priority preemptive system. You can change any task's priority, but since RMS is static priority, avoid using vTaskPrioritySet() in FreeRTOS. In RMS, priority is assigned based on periodicity of the task, and since that's already determined (and is fixed) using the WCET values from the previous lab, you should be able to assign priority easily to obtain the RMS schedule.
- Make sure each task has a unique priority. Although this isn't a strict requirement, it's **significantly easier** to assess deterministic behavior with unique priorities.

If you do this part of the lab be sure to adhere to the additional submission requirements so you can get all your points.

Reminder from last lab: Worst Case Execution Time (WCET)

You can use a static analysis tool or a measurement based technique to estimate WCET of each task. A reasonable list of WCET tools can be found at `https://www.rapitasystems.com/WCET-Tools`. Alternatively, you can do a statistical analysis by running your task enough times and estimating WCET. Here "enough times" is ill-defined as discussed in class. Whichever method you use, remember to add a ~20% cushion to the time for a margin of "safety."

What to submit

1. **(35 points)** Zip the entire Arduino project so that we can just unzip and execute the source code.
   - Please include a diff of the source code between this lab and previous one. Please use a "diff" tool of some kind that highlights the changes from previous files (e.g. `https://www.diffchecker.com/`, meld, github, etc.).
   - **(10 points) BONUS**: if you did the bonus, zip up the code from that part of the lab and give me the "diff" from the Lab 4. Note that this should only be a few small changes to some period and priority values in the tasks.
2. **(30 points)** Documentation of the task(s) you have implemented for this lab. **(1-2 pages long. NOTE: this is the only lab with different page length requirements!)**
   - Show the table of your cyclic executive. Show the hyperperiod and all tasks for one complete cycle of the schedule. This is best done in a timing table (Gantt chart style).
   - Answer the following:
     - For each task why did you choose the period, P_i, that you chose?
     - How did you determine WCET for each task?
     - How does your custom cyclic executive system compare with FreeRTOS? Is the performance different? Which is easier to implement, maintain? Which is easier to adjust given a change in requirements to the system? What other differences do you notice?
   - **(10 points) BONUS**: if you did the bonus part of the lab include a single paragraph on how you implemented RMS using FreeRTOS. Include your proof of schedulability.
3. **(35 points)** Record a short video of Ringo doing the same thing you did in the video for Lab 4. To help me know you aren't just submitting the same video include some audio or video with the date you recorded the video.
   - Be sure to describe or narrate the test that Ringo is doing that demonstrates that you have accomplished the objectives of the lab.

- Please upload the video to YouTube, Vimeo, or something similar and then provide me with a link. Just put the link somewhere in the writeup you did in #2 above.

4. Upload all of this into "handin" at `https://cse-apps.unl.edu/handin`