ECEE 5623, Real-Time Systems:

Exercise #6 – Real-Time Software Systems

Final Extended Lab (Individual or Group Work)

DUE: As Indicated on Canvas

Please thoroughly read Chapters 12, 13 & 14 in RTECS with Linux and RTOS

Please see example code provided - Linux, FreeRTOS, VxWorks, Zephyr

This lab is written **to be completed with embedded Linux running on the R-Pi3b (order or borrow and use NOOB image)**. **It is possible to also use a Jetson Nano (Getting started)**, Jetson TK1/TX1/TX2 (JetPack install) or Beagle board(s). **The standard final project requires use of a camera, which is simple with embedded Linux using the UVC driver**. Linux examples can be found here - http://ecee.colorado.edu/~ecen5623/ecen/ex/Linux/code/

This group exercise (teams of 2, 3 or 5) asks you to explore real-time software system analysis and design concepts by applying RM theory we have learned as well as software engineering skills you have learned as a computer and/or software engineering student. The goal is for your team to design an aspect of a real-time system that includes 2+ real-time services and requires RM analysis along with priority-preemptive (POSIX SCHED_FIFO) or RTOS priority scheduling to meet required periodic deadlines.

Exercise #6 Requirements and High Level Design:

1) [10 points] Provide all major functional capability requirements for your real-time software system [not just for the proof-of-concept aspect to be demonstrated and analyzed, but the whole design concept envisioned]. You should have at least 5 major requirements or more.

2) [10 points] Complete a high-level real-time software system functional description and proposal along with a single page block diagram showing major elements (hardware and software) in your system (example). You must include a single page block diagram, but also back this up with more details in corresponding CFD/DFD, state-machine, ERD and flow-chart diagrams as you see fit (2 more minimum). Please provide this in your report for this assignment.

3) [10 points] Provide all major real-time service requirements with a description of each S_i including C_i, T_i, D_i with a description of how the request frequency and deadline was determined for each as well as how C_i was determined, estimated or measured as WCET for the service.

4) [10 points] For your 2+ services S_i with requirements C_i, T_i, D_i provide Cheddar Simulation and Worst-Case analysis, but double check this with your own Scheduling Point and Completion Test analysis as well as hand-drawn timing (if feasible – i.e. if LCM is not ridiculously long). Provide an estimate of the safety margin your system should have compared to what it must have for feasibility and margin of error [variations seen in C_i to account for expected C_i and WCET].

5) [10 pts] Analyze your system in terms of how well it meets real-time requirements as outlined and used in Lab Exercise #5, found here. Present the results along with a description of how predictable the responses are relative to the request times as well as how constant the request frequency is for your system design.

6) [50 pts] Write a report that includes [16 total pages including cover page, but not appendices or code]. The report should be composed of results from your work in Lab Exercise #5 and work above, but documented in complete form as follows:

   [1 page] **Cover Page (list all group members clearly)**

   [1 paragraph] **Introduction**

   [1 page] **Functional (capability) Requirements**

   [1 page] **Real-Time Requirements**

   [N pages] **Functional Design Overview and Diagrams**

   [N pages] **Real-Time Analysis and Design with Timing Diagrams, both measured and expected based upon theory**

   [N pages] **Proof-of-Concept with Example Output and Tests Completed** - **(Must contain final time-lapse MPEGs and verification RAW frames with time-stamps and uname stamp)**

   [1 paragraph] **Conclusion**

   [1 page] **Formal References (and Attributions to Anyone who helped not on the team)**

   [N pages] **Appendices with results, code and supporting material** (to stay in page bounds)

Overall, provide a well-documented professional report of your findings, output, and tests so that it is easy for a colleague (or instructor) to understand what you've done. Include any C/C++ source code you write (or modify) and Makefiles needed to build your code and make sure your code is well commented, documented and follows coding style guidelines. I will look at your report first, so it must be well written and clearly address each problem providing clear and concise responses to receive credit.

Note: Linux manual pages can be found for all system calls (e.g. fork()) on the web at http://linux.die.net/man/ - e.g. http://linux.die.net/man/2/fork

In this class, you'll be expected to consult the Linux manual pages and to do some reading and research on your own, so practice this in this first lab and try to answer as many of your own questions as possible, but do come to office hours and ask for help if you get stuck.

Upload all code and your report completed using MS Word or as a PDF to Canvas and include all source code (ideally example output should be integrated into the report directly, but if not, clearly label in the report and by filename if test and example output is not pasted directly into the report). ***Your code must include a Makefile so I can build your solution on an embedded Linux system (R-Pi 3b+ or Jetson). Please zip or tar.gz your solution with your first and last name embedded in the directory name and/or provide a GitHub public or private repository link. Note that I may ask you or SA graders may ask you to walk-through and explain your code. Any code that you present as your own that is "re-used" and not cited with the original source is plagiarism. So, be sure to cite code you did not author and be sure you can explain it in good detail if you do re-use, you must provide a proper citation and prove that you understand the code you are using.***

Grading Rubric

[10 points] Functional (capability) requirements for system design:

[2 points] R#1 ____

[2 points] R#2 ____

[2 points] R#3 ____

[2 points] R#4 ____

[2 points] R#5 to N____

[10 points] High level system and software design:

[5 points] Block diagram ____

[5 points] Detail diagram #1 #N_______________________________________

[10 points] Real-Time Services and Requirements:

[5 points] ServicesSi, C_i and WCET____________________________________

[5 points] T_i and D_i specification_______________________________________\

[10 points] Real-Time Services Feasibility, Safety and Margin:

[5 points] Cheddar analysis of scheduling by hand ____________________________

[5 points] Time-stamp tracing using syslog of other non-intrusive mechanism

____

[10 pts] Analyze your system

[5 points] 1 Hz analysis (or lowest rate for creative project)

____

[5 points] 10 Hz (or highest rate for creative project)

____

[50 points] Report required elements:

[5 points] - **Cover Page (list all group members clearly)**

[5 points] - **Introduction**

[5 points] - **Functional (capability) Requirements**

[5 points] - **Real-Time Requirements**

[5 points] - **Functional Design Overview and Diagrams**

[5 points] - **Real-Time Analysis and Design with Timing Diagrams, both measured and expected based upon theory**

[5 points] - **Proof-of-Concept with Example Output and Tests Completed** - **(Must contain final time-lapse MPEGs and verification RAW frames with time-stamps and uname stamp)**

[5 points] - **Conclusion**

[5 points] - **Formal References (and Attributions to Anyone who helped not on the team)**

[5 points] - **Appendices with results, code and supporting material** (to stay in page bounds)