Connor Peskin
RTOS Final
Weekly Report 1

Overview

I plan to do the inverted pendulum project, but will try to make my work as modular and adaptable as possible to be able to shift directions to one of the (seemingly) more complicated project options later on if I deem that I have enough time to do so.
I plan to use toggl track to track my time and each task that I'm working on at all times to get accurate numbers for my time estimates and actual time taken analysis. I also think this will be a great practice and very helpful.

Weekly Tasks & Time Estimates:

- Read/Digest project description
  - Estimated: 10 min
  - Time Taken: 10 min
- Create diagram showing all tasks and ITC concepts used
  - Estimated: 45 min
  - Time Taken: 47 min
- Create a unit testing plan, go back to lecture topic 2.5 and rewatch as well.
  - Estimated: 30 min
  - Time Taken: 49 min
- Project status summary (below)
  - Estimated: 5 min
  - Time Taken: 3 min
- In-Scope work items summary.
  - Estimated time: 30 min
  - Time Taken: 42 min
- Risk Register- I added one risk, nothing really has come up yet that I've recognized:
  - Estimated: 10 min
  - Time Taken: 7 min
- **TOTAL:**
  - Estimated: 2hr
  - Time Taken: 2hr 38 min

Diagram

My diagram can be found in my git repo here:
https://github.com/cope3319/RTOS-final-invertedPendulum/blob/master/InvertedPendulumDiagram.png

Unit Testing Plan

My initial function test outlines and comments can be found in this file in my repo:
https://github.com/cope3319/RTOS-final-invertedPendulum/blob/master/src/unittests.c

I plan to have a compile-time define that will signal for these tests to be run directly after every task has been initialized. The tests are as follows:

- Pushbutton Validation - Cutting off the GPIO IRQ (we know this works properly)
    - forceIncreaseTest()
        - Call the PB0 event and ensure that the PWM value is reflected on LED0 and that the force value is changed in the pendulumStateData struct
    - Force Decrease Check
        - Similar to above.
- Direction Change Test - Cutting off the touch sensor sampling function and LCD and LED
    - directionChangeTest()
        - Change the data in the AdjustmentData struct and ensure that the data in pendulum state is changed properly.
- LED1 Test
    - LED1Test()
        - Manually change the angle value in pendulumStateData struct and ensure that LED1 is lit as expected
- Force Limit Tests - Cutting off everything except pushbuttontask
    - upperLimitForceTest()
        - Call event to increase until it should go above limit force value and ensure that the force value stays at the top limit
    - lowerLimitForceTest()
        - Similar to above
- Display Frequency Test - Cutting off input tasks, LED task
    - displayUpdateFrequencyTest()
        - Ensure that the display updates on the proper interval ({tau}_display) by delaying that amount and ensuring the display has changed with some inputs
- Physics Task Test - cutting off all input and output tasks
    - physicsTaskTest()
        - Set the values to the adjustments, knowing what the adjustments to be made by the physics task should be. Ensure that the physics task makes the correct changes on the correct time interval.
- Garbage In, Garbage Out Test - cutting off all input and output tasks
    - GIGOtest()
        - Enter bad data into the adjustmentdata struct and ensure that the physics task recognizes this as bad data and doesn’t make changes to the pendulum state data.
- Mutex Tests
    - pendulumStateDataMutexTest()
        - Pend on the pendulumState mutex and ensure that no changes are made to it on the other periodic task’s frequencies.
    - adjusmentDataMutexTest()
        - Similar to above

Project Status Summary

Everything I've done so far has simply been planning and I'm pretty excited for this project. I think that it's entirely do-able in the timeframe and I'm optimistic that I'll be able to find the time to be able to do one of the more difficult project ideas. I'm going to put a lot of effort into designing with modularity in mind so that it should only be a change in physics (and some display) logic to get the other project ideas working.
Overall, this should be fun but I'm a bit worried about my design considerations. I'm not 100% sure that I've made the right decisions of Mutexes/Semaphores/Flags/Message Queues where I should have and I'm going to have to be adaptive and open to ideas in case I begin seeing issues with my current design/plan.

In-Scope Work Items Summary

- Planning - Estimated 2hr - **DONE**
    - (this includes the time to write this first report)
    - I will create the diagram, unit tests, and determine my baseline in-scope tasks to complete for this project. This is just the Weekly Report 1 :)
- Port over code from last project - Estimated: 1hr
    - Getting everything established, task definitions, etc always takes some time. I'll be using the code from Lab7 and will be changing task names and whatnot where necessary.
    - At this point, I should go through and implement modularity standards that I want to keep across all of my code, including pointers for my app.c helper functions and other things of that sort.
- Rewrite my display logic and get familiar with glib, etc. - Estimated: 3hr
    - I used the textdisplay.c module in my Lab7, so I don't have experience with drawing, updating, and the best methods of doing that yet.
    - I plan to firstly ensure that I have all the proper clocks and other variables set and get familiar with updating the display and how this will work. I know that the professor mentioned some tricks with updating the display by not clearing (took more time) but drawing white on white or something of that sort. These are the types of things I'd like to get ironed out prior to having much more complicated things in the mix.
    - At this point, I don't plan to write the code that the display task will ultimately be using. This is to gain familiarity while the environment is simple.
- Without using any inputs or outputs, write the physics logic and it's interaction with the two stated data structures. - Estimated 5hr
    - I'm giving this one a large chunk of time because I'm guessing that it's going to be much easier said than done. I still don't have a great grasp of the inverted pendulum and the physics behind it yet. I'll first handwrite and get a good understanding of the logic going into this. I'll then write the interactions with the two data structures
    - At this point, I'm going to utilize and develop my physicsTaskTest() pretty heavily to ensure that the physics task is making the correct adjustments as expected.

- Implement Pushbutton inputs - Estimated: 30 min
    - This should be pretty easy as we've done it for essentially every lab. Most of this work will be ensuring that the pushbuttons are correctly interacting with the physics task
- Implementing PWM - Estimated: 1.5hr
    - We haven't worked with PWM on this board yet (but I have with different boards), so I'll have to build out functionality for this. I'll build out functionality for this, then make sure it's working properly with the buttons and the physics task
- Logic for other LED - Estimated 15 mins
    - This should be easy- just need to have the logic built out for LED1 and I can use the LED1test() to test and ensure this is working properly.
- Touch Slider logic - Estimated 45 mins
    - This should be pretty straightforward implementing the already-made code from the previous labs and adapting to this project. I'll test by making sure that the LED1 task is adjusting properly.
- Display Integration with physics task- 4 hr
    - Similar to the physics engine/task, I think this is going to take a lot more time/work than I am anticipating so I've given it a little more time than I'm thinking.
    - At this point, I'm going to build out how the display task interacts with the state data structure and updates the screen. Thankfully, I'll have already put some time and thought into how I will be updating the screen in the best way possible, so hopefully this will be a little smoother than implementing this with everything else implemented and running.
- Iron out all functionality and ensure everything is working as expected- 3hr
    - With all tasks written, I'm positive that issues with interactions will be coming up along the way. I've allocated this time for making sure that everything is working properly and as expected.

I have completed 9.5% of my currently-scoped, estimated work (2/21hr) in 12% of the initially-estimated time. (2.6/21hr). My best guess of my say/do ratio is 79%, so to unbias my estimates after this class, I may want to multiply my estimates by 1.26 (12%/9%).
No scope changes have been made.

Risk Register

I don't really have any risks yet, as I haven't got to implementation of any aspect yet, just planning. My risk register file can be found in my git repo here:
https://github.com/cope3319/RTOS-final-invertedPendulum/blob/master/riskregister.xlsx