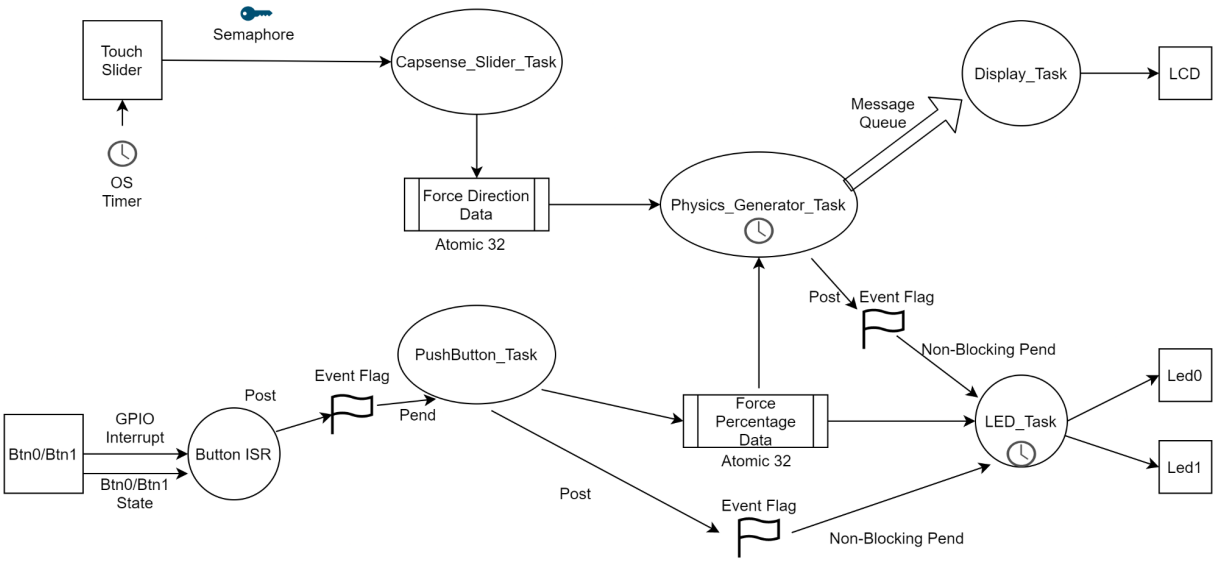
Justin Thwaites
ECEN 3753
Final Project

Final Report

I am currently 100% complete with the project.

Updated Task Diagram



Unit Tests and Functional tests:

As of now, of the 12 units, 12 of them are passing. These unit tests are currently passing because I have now implemented the logic behind these functions. The unit tests are as follows:

1. Button 0 reads to ensure a button0 pressed event and then a button0 released event is sent into the pushbutton task the button recording function of PushButton_task will return that pushbutton 0 is pressed. This is cut by the Button ISR and before the writing of force percentage to the shared Force Percentage Data.
2. Button 1 reads to ensure a button1 pressed event and then a button1 released event is sent into the pushbutton task the button recording function of PushButton_task will return that pushbutton 1 is pressed. This is cut by the Button ISR and before the writing of force percentage to the shared Force Percentage Data.
3. Button error read to insure a button1 pressed event and then a button0 pressed event and then a button1 released event are sent into the pushbutton task the button recording function of PushButton_task will return that no push buttons are pressed. This is cut by the Button ISR and before the writing of force percentage to the shared Force Percentage Data.

4. Button error read to ensure a button0 pressed event and then a button1 pressed event and then a button0 released event are sent into the pushbutton task the button recording function of PushButton_task will return that no push buttons are pressed. This is cut by the Button ISR and before the writing of force percentage to the shared Force Percentage Data.
5. Slider Reading pressed test, ensures that given values of the normalized slider positions function with all below the pressed threshold, it will return the lowest slider position. This unit test is cut by the Touch slider semaphore and the input to the force direction data.
6. Slider Reading not pressed, ensures that given values of the normalized slider positions function with all values above the pressed threshold, it will return that no slider is pressed. This unit test is cut by the Touch slider semaphore and the input to the force direction data.
7. Slider Reading out of bounds error, ensures that given values of the normalized slider positions are all 0 or all above the given threshold, it will return an error code that can be caught to prevent buggy performance. This unit test is cut by the Touch slider semaphore and the input to the force direction data.
8. LED1 Control, this unit test looks at the handling logic and makes sure if a stub provides the Event Flag for LED1 to be turned on, the handling logic will turn on LED1. This Unit test is cut Around the LED Task by the Event Flag data structure and the BSP LED functions.
9. LED0 Control, this unit test looks at the handling logic and makes sure if a stub provides the Force percentage at 100, the logic will return that LED0 should be high. This Unit test is cut Around the LED Task by the force percentage data structure and the BSP LED functions.
10. LED0 Control, this unit test looks at the handling logic and makes sure if a stub provides the Force percentage at 50%, the logic will return that LED0 should be low. This Unit test is cut Around the LED Task by the force percentage data structure and the BSP LED functions.
11. LED0 Control, this unit test looks at the handling logic and makes sure if a stub provides the Force percentage at 0, the logic will return that LED0 should be low. This Unit test is cut Around the LED Task by the force percentage data structure and the BSP LED functions.
12. Physics_generator task, this unit test tests the physics engine with given global inputs as well as the force direction and force percentage and makes sure that the correct force is computed as a result of the global constants as well as the force direction and force percentage.

Functional Tests

1. Upon startup of the project, verify that without pressing any of the inputs the inverted pendulum will remain completely upright.
    a. Now press pushbutton 1 10 times without pressing any other inputs, verify that the slider is still straight up.

    b. Now press pushbutton 0 10 times without pressing any other inputs, verify the slider is still straight up.
2. Upon restart of the project, verify that neither of the LEDs are on.
    a. Press pushbutton 1 once and ensure the LED0 is slightly on
    b. Press pushbutton 1 3 more times and ensure the LED0 is much brighter at approximately 50% brightness.
    c. Press pushbutton 1 4 more times and LED0 should be fully on.
3. With LED0 fully on ensure it remains on and does not change.
    a. Press pushbutton 0 once and ensure LED0 dims slightly
    b. Press pushbutton 0 3 more times and ensure the LED0 is much brighter at approximately 50% brightness.
    c. Press pushbutton 0 4 more times and LED0 should be fully off.
4. Upon restart of the project, verify that neither of the LEDs are on.
    a. Press the far left slider, ensure that the inverted pendulum has not moved on the screen.
    b. Press the left slider, ensure that the inverted pendulum has not moved on the screen.
    c. Press the far right slider, ensure that the inverted pendulum has not moved on the screen.
    d. Press the right slider, ensure that the inverted pendulum has not moved on the screen.
5. Upon restart of the project, verify that neither of the LEDs are on.
    a. Press pushbutton 1 once and ensure the LED0 is slightly on
    b. Press the **far left** slider, ensure that the inverted pendulum has tilted right on the screen.
    c. Stop pressing and observe the pendulum continue falling.
    d. Let the pendulum fully fall.
    e. Ensure that the LED 1 turns on.
6. Upon restart of the project, verify that neither of the LEDs are on.
    a. Press pushbutton 1 once and ensure the LED0 is slightly on
    b. Press the **left** slider, ensure that the inverted pendulum has tilted right on the screen.
    c. Stop pressing and observe the pendulum continue falling.
    d. Let the pendulum fully fall.
    e. Ensure that the LED 1 turns on.
7. Upon restart of the project, verify that neither of the LEDs are on.
    a. Press pushbutton 1 once and ensure the LED0 is slightly on
    b. Press the **far right** slider, ensure that the inverted pendulum has tilted left on the screen.
    c. Stop pressing and observe the pendulum continue falling.
    d. Let the pendulum fully fall.
    e. Ensure that the LED 1 turns on.
8. Upon restart of the project, verify that neither of the LEDs are on.
    a. Press pushbutton 1 once and ensure the LED0 is slightly on

    b. Press the **right** slider, ensure that the inverted pendulum has tilted left on the screen.
    c. Stop pressing and observe the pendulum continue falling.
    d. Let the pendulum fully fall.
    e. Ensure that the LED 1 turns on.
9. With the Pendulum on its side and LED 1 no, ensure LED 1 remains lit.
    a. Press pushbutton 1 once and ensure the LED0 is slightly on
    b. Press the far right slider, ensure that the inverted pendulum does not move and LED1 stays lit.
    c. Press the right slider, ensure that the inverted pendulum does not move and LED1 stays lit.
    d. Press the left slider, ensure that the inverted pendulum does not move and LED1 stays lit.
    e. Press the far left slider, ensure that the inverted pendulum does not move and LED1 stays lit.
10. Upon restart of the project, press pushbutton 1 once and ensure the LED0 is slightly on.
    a. Press the left slider and ensure the pendulum begins moving left falling right.
    b. While the pendulum is falling move your finger quickly to the far right slider.
    c. The pendulum should begin changing direction and go back up to the top and fall the other way.
    d. Move your finger back to the far left slider once more and the pendulum should change direction one more time.

Testing Status Summary

Of all the unit tests 1-12, all 12 are currently at status “Run”. These tests are running and passing because the logic has been implemented.

Of all the functional tests 1-10, all 10 are currently at status “Run”. This is because the logic is implemented and the integration of the tests are implemented. Of these functional tests all are passing. This is because my rtos solution is complete.

Project Summary Statement

This week I ensured the entire project was functional and made any slight modifications to make sure that the project fit the requirements. It is now a complete and functional game to be played.

I have completed 100% (17/17hrs) of the project scope by this third week. This has taken me 118% (20/17hrs) of the original estimated time. My best guess for a say do ratio is 114% (multiplying my estimates by 8/7) for future planned projects. Given my scope estimate change to 21.25, I was very close to the total time for the adjusted scope (20/21.25) 94%.

Project Scope

- Task Diagram
    - Complete
    - Takes 1 hour
    - Spent 1 hour
- Unit Test generating
    - Complete
    - Takes 0.5 hours
    - Spent 1 hour
- Project Scoping
    - Complete
    - Takes 1 hours
    - Spent 0.5 hours
- Risks
    - Complete
    - Takes 0.5 hours
    - Spent 0.5 hours
- Setting up the Project In simplicity with skeleton tasks and stub functions
    - Complete
    - Takes 1 hour
    - Spent 1.5 hours
- Data Structure Definitions
    - Complete
    - Takes 1 hour
    - Spent 1 hour
- Unit testing implementation of all unit tests
    - Complete
    - Takes 2 hours
    - Spent 2.5 hours
- Pushbutton Task Logic
    - Complete
    - Takes 0.75 hours
    - Spent 0.75 hours
- Slider Task Logic
    - Complete
    - Takes 0.5 hour
    - Spent 0.5 hour
- Physics Engine Task Logic
    - Complete
    - Takes 1.75 hours
    - Spent 1.75 hours
- LED Task Logic
    - Complete

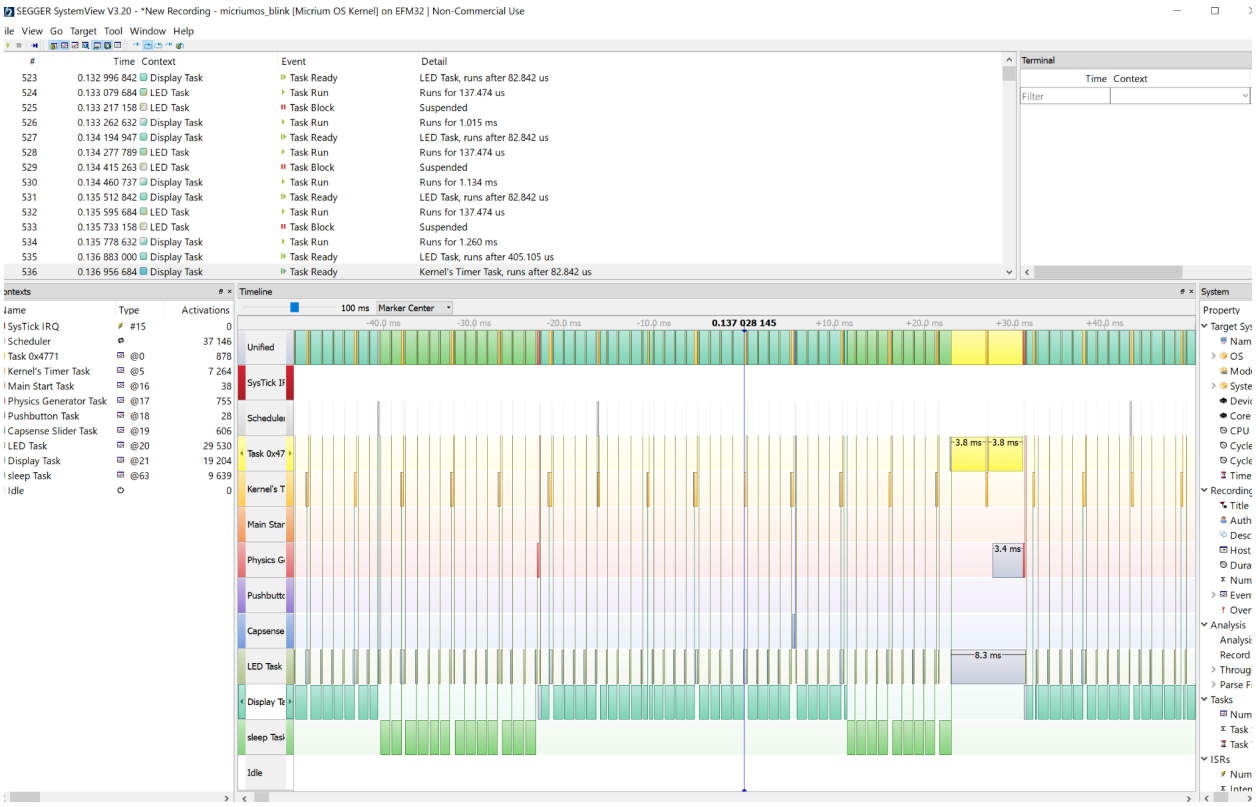
    - Takes 0.5 hours
    - Spent 0.5 hours
- Pushbutton Task Implementation
    - Complete
    - Takes 0.75 hours
    - Spent 0.5 hours
- Slider Task implementation
    - Complete
    - Takes 0.5 hours
    - Spent 0.5 hours
- Physics Engine Task Implementation
    - Complete
    - Takes 1.75 hours
    - Spent 2.5 hours
- Display Task Implementation
    - Complete
    - Takes 2 hours
    - Spent 2 hours
- LED Task Implementation
    - Complete
    - Takes 0.5 hours
    - Spent 0.5 hours
- Functional Tests
    - Complete
    - Takes 1 hour
    - Spent 1 hour
- Integrating project subsystems together for complete functionality
    - Complete
    - Takes 1 hour
    - Spent 1 hour
- Final quality inspection
    - Complete
    - Takes 0.5 hours
    - Spent 1 hour

While this part of the project was quite easy, the part took twice as long because it was easy to be distracted playing the game as well as making slight modifications to the game to ensure that it is working optimally.

Total Complete/ Total Required = 17/17 hours 100%

Analysis of Solution

RT tasks



My task priorities are:
Physics Generator task 17
Pushbutton task 18
Capsense Slider task at 19
LED task 20
Display task 21

My task runtimes are:
Physics Generator task 0.39ms
Pushbutton task 0.08ms
Capsense Slider task at 0.23ms
LED task 0.26ms
Display task 1.32ms

My hard real time requirement of my Physics task occurring every 50 ms is met, however there is sometimes a 5 ms delay on this task. However this does not prevent this task from meeting its deadline of occuring every 50 ms, this is partly because the physics task only takes 0.5 ms due to only using the integer math.

Code Space

Flash used: 84124 / 1024000 (8%)

RAM used: 73680 / 256000 (28%)
The amount of code used on this processor is very minimal when compared to the other projects and their relative scope compared to this project. This shows that using an RTOS has a flat upfront fee in RAM and Flash usage, but then decreases additional code space increases.

Evaluation of Approaches

For my game I used an euler integration physics solver to solve the movement of the top of the mass based on the calculated forces based on time step. When I built this I wanted to make a simplified sin and cosine calculation so I defined my own trig functions that took an input of radians multiplied by 128 and returned a value multiplied by 1024. This allowed me to make most of my computations work without the need for floating point arithmetic. While possibly not necessary it was a challenge that i feel was worth it in reducing my time spent computing. Aside from this crossing over proved challenging for me, I ended up implementing this with some if statements to cross over if the approximator functions outputted a result greater than the max possible height. This challenge did not seem obvious to me at first.

Scaling of Variable Spaces

The game is dependent on g/l and m/Fmax, not a variation of this, the reason for this is that if gravity is increased and l is increased its still going to have the same playability speed because it may fall faster but it has longer to fall. Likewise with a greater mass it may take more force to change direction, but if the max force is increased it is impossible to notice. Due to the scaling on the screen it is better to have an l that is less than 1/10th of the playing field otherwise, the magnification makes it so it is hard to tell when it is close to going off the screen.

The corner ranges are around:
g/l max: 2.5 1/s^2
g/l min: 1/64 1/s^2

m/Fmax max: 1/3 g/mN
m/Fmax min: 1/100 g/mN

What would be Next Steps

If I had 2 extra weeks on the project I would focus on trying to get the project to be operating at a much more efficient rate. I would look for any code bloat in my solution and I would cut it down, and optimize to allow for an even smaller step size. Additionally, I think I would increase the functionality of the LCD display, I would make it so that the screen would appear more like a mario type game where you can tell that the cart is moving but not see the the whole world so that the walls can be seen only when you are close to them and the pendulum does not need to be scaled.