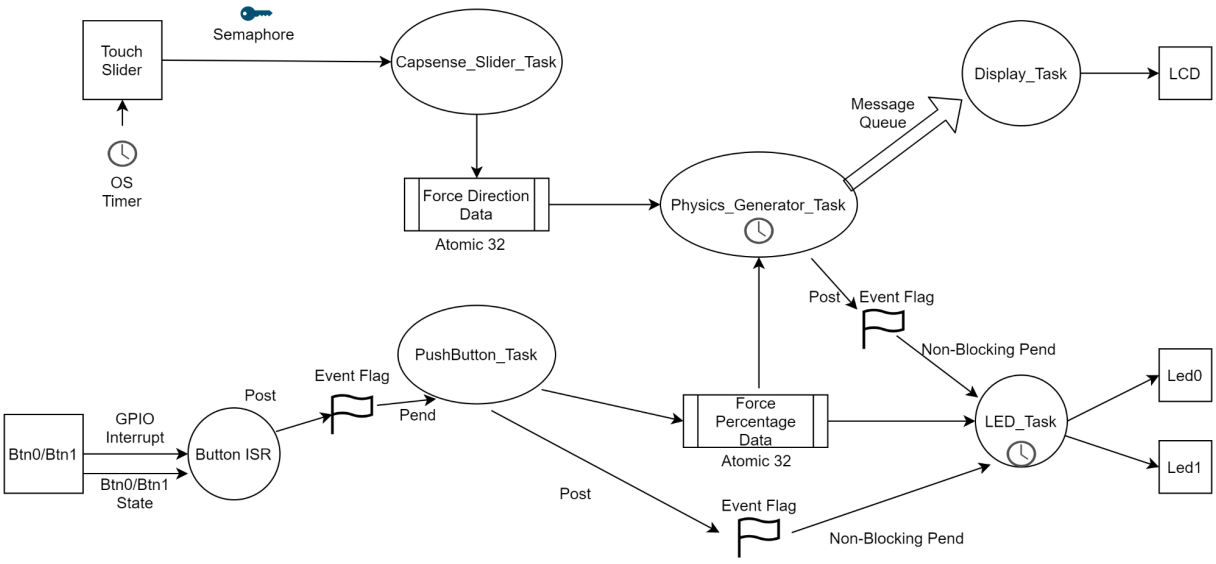
Justin Thwaites
ECEN 3753
Final Project

Week 4 Report

I am currently 97% 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 4 more times and ensure the LED0 is much brighter at approximately 50% brightness.
    c. Press pushbutton 1 5 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 4 more times and ensure the LED0 is much brighter at approximately 50% brightness.
    c. Press pushbutton 0 5 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 and begins blinking.
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 and begins blinking.
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 and begins blinking.
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 and begins blinking.
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.

Project Summary Statement

This week I implemented the integration of all tasks of the project. This has allowed me to pass all 10 functional tests this week through my implementation of the project.

I have completed 64% (16.5/17hrs) of the project scope by this third week. This has taken me 112% (19/17hrs) of the original time. This will mean I will be very close to my estimated adjustment of 1.25x the amount of time at approximately 21.25 hours. This week I was able to complete each task in approximately the same amount of time as was allotted for it, except for the physics task. However, since I am almost complete with my project I will not need to increase the time allotted.

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

Because of the logic already being implemented the pushbutton task was able to be completed very quickly. Therefore I reduced the amount of time I actually spent working on this.

- Slider Task implementation
    - Complete
    - Takes 0.5 hours
    - Spent 0.5 hours

Similar to the pushbutton task the slider task was completed without much difficulty allowing me to complete it in a half hour.

- Physics Engine Task Implementation
    - Complete
    - Takes 1.75 hours
    - Spent 2.5 hours

While the implementation of the task did not in itself take too long, adding to it extra custom functions and modifications such as tables for trig lookup to increase speed made it take a bit longer than expected.

- Display Task Implementation
    - Complete
    - Takes 2 hours
    - Spent 2 hours

While I did enjoy making the graphics for the game it did take quite a bit of time and I am glad I allotted 2 hours for implementation of the task logic.

- LED Task Implementation
    - Complete
    - Takes 0.5 hours
    - Spent 0.5 hours

The creation of the PWM which seemed tricky at first, ended up being quite easy to implement, allowing me to finish in time for this task.

- Functional Tests
    - Complete
    - Takes 1 hour
    - Spent 1 hour
- Integrating project subsystems together for complete functionality
    - Complete
    - Takes 1 hour

I am quite glad that I included this time, I had to debug quite a few glitches when playing the game such as the pendulum never crossing over to the other side.

- Final quality inspection

- Yet to complete
- Takes 0.5 hours

Total Complete/ Total Required = 16.5/17 hours 97%

Risks

See the attached spreadsheet for my filled out Risks register.