Demo

Going to demo in class and show a mostly working game. Only aspect of the game that didn't function was the changing rate of one of the LEDs dependent on the percentage of the acceleration to the blackout acceleration. The three functional tests I am going to demo are tests 3, 4, and 8 shown below.

Work Items

Task	Estimated Time (Hrs)	Actual Time (Hrs)	In Progress/Done
Project Planning	2	3.25	
Create Unit Tests	3	3	
Create Project Files	0.5	0.5	
Implement Button IRQs	1.5	1	
Instantiate all OS Resources	2	2	
Add Functional Tests	2	1.5	
Import LCD Sources	2	3.5	
Implement Config File	1.5	1	
Implement Thrust Task	3	4	
Implement Angle Task	3	1.5	
Implement Physics Task	8	9	
Implement LCD Task	4	4	
Implement LED Tasks	3	3.5	
Total	35.5	37.75	
Percent Done	106.34%	Tasks Completed	92.31%

Testing Plan

Unit Tests

1. Test Case: Valid AngleValues
    a. Test that the Angle task is getting proper values into the angle data structure when no input
    b. Value with no input should be -1.
2. Test Case: Valid Angle Values
    a. Test Angle task is setting proper value to angle data structure with user input
    b. Value with input to far left of slider should be 40.
3. Test Case: Valid Angle Values
    a. Test Angle task is setting proper value to angle data structure with user input
    b. Value with input to left of slider should be 40
4. Test Case: Valid Angle Values
    a. Test Angle task is setting proper value to angle data structure with user input
    b. Value with input to right of slider should be 40.
5. Test Case: Valid Angle Values
    a. Test Angle task is setting proper value to angle data structure with user input
    b. Value with input to far right of slider should be 40.
6. Test Case: Valid Thrust Values
    a. Test thrust task is correctly setting thrust data on button 0 interrupts.
    b. Should set button 0 field in struct to true
    c. Should decrement the thrust counter by 40.
7. Test Case: Valid Thrust Values
    a. Test thrust task is correctly setting thrust data on button 1 interrupts.
    b. Should set button 1 field in struct to true.
    c. Should increment the thrust counter by 40.
8. Test Case: Valid Thrust Values
    a. Test thrust task with zero thrust percentage.
    b. X component for acceleration should have a zero value
    c. Y component for acceleration should only be the gravitational constant
9. Test Case: Valid Thrust Values
    a. Test thrust task with increment button held for longer period of time (3 presses)
    b. Decrement button once to test in unison.
    c. Verify by checking value of thrust.
10. Test Case: Valid Thrust Values
    a. Test thrust task with increment button held for longer period of time (11 presses)
    b. Verify by checking thrust caps out at 400.

Functional Tests

1. Test Case: Test LCD Functionality.

    a. Check that LCD is correctly initialized by displaying a Message
2. Test Case: TEST LCD Functionality
    a. Ensure that LCD is updating at a reasonable speed to ensure playability.
3. Test Case: LCD Functionality
    a. Verify that exiting the bounds of the screen ends the game.
4. Test Case: Test LED0
    a. Test LED brightness at different duty cycles to verify correct PWM implementation. Verify by seeing rate of LED with different values of thrust (0-10 button pushes).
5. Test Case: Test LED
    a. Test with healthy flight conditions
6. Test Case: Test LED
    a. Test with blackout flight conditions
7. Test Case: Test LED
    a. Test with game over conditions
8. Test Case: Test Physics Model
    a. Test whether thrust works correctly in changing velocity
9. Test Case: Test Physics Model
    a. Test whether changing the angle with thrust will change the horizontal velocity and acceleration.
10. Test Case: Win Game Conditions
    a. Test winning conditions by attempting to win the game with soft landing, also have to land straight up.

Test Summary

All the above unit tests have been implemented. Only the tests pertaining to the angle and thrust tasks have been completed completely and are passing. I didn't add any other functions that are unit testable so I only included tests for the two helper functions dealing with setting thrust and angle based on the user inputs. I did make one change to testing as I have edited the rate of change of thrust for each button push. Since the max thrust is 400 N, I made the change rate 40 so that 10 pushes would max it out.

In terms of the functional tests, I did remove any of the tests that dealt with checking if the configuration data was valid. Since I was only going to use the data setup in my config.h file, I figured that the user would just have to manually edit the header file for any changes to the config data so it wouldn't really need any functional testing for now. As of now, all of the functions tests pass except for the one dealing with the LED and displaying both of the blackout conditions.

Project Standing

This week, I worked on wrapping up the physics task, angle task, LCD task, and most of the LED task. The angle task was pretty simple to implement since I basically reused a lot of my code from the 2 previous labs. The physics task was indeed the task I spent most of the time implementing as I had originally anticipated. I had originally planned to use actual realistic values for my configuration data, but quickly found that the model I had made didn't work well with those values. Instead, I just debugged with different values to find some that made the game more playable as that aspect was more important to me. Overall, I was able to complete the game and make it playable for the most part. As previously mentioned, the only non-working part of the game is that one of the LEDs doesn't correctly PWM at the rate of the blackout acceleration, but it does function properly to show the end game pattern. The other LED does properly modulate at the rate between the max thrust and the current thrust.

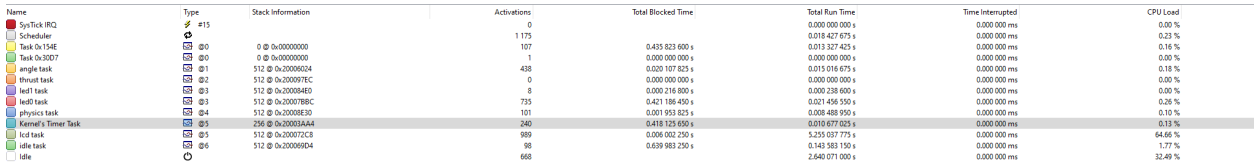
At the beginning, I had anticipated around 35.5 hours of work for the project. In the end, I spent a couple more hours than anticipated bringing the total amount of work to around 37.75 hours. I spent a bit more trying to implement the physics model and LED tasks.

Solution Analysis

My Priorities for my tasks are as follows:

```
#define LCD_TASK_STACK_SIZE         512
#define LCD_TASK_PRIO               5

#define ANGLE_TASK_STACK_SIZE       512
#define ANGLE_TASK_PRIO             1

#define THRUST_TASK_STACK_SIZE      512
#define THRUST_TASK_PRIO            2

#define PHYSICS_TASK_STACK_SIZE     512
#define PHYSICS_TASK_PRIO           4

#define LED0_TASK_STACK_SIZE        512
#define LED0_TASK_PRIO              3

#define LED1_TASK_STACK_SIZE       512
#define LED1_TASK_PRIO              3

#define IDLE_TASK_STACK_SIZE        512
#define IDLE_TASK_PRIO              6
```

As seen, I made the LCD task the most important task since I wanted it to have priority so that movement across the screen was relatively smooth in terms of refresh rate. For the LCD, I had it run every 100ms resulting in ~10 Hz refresh rate. The movement seemed smooth enough for a simple pixel game.



Name	Type	Stack Information	Activations	Total Blocked Time	Total Run Time	Time Interrupted	CPU Load
SysTick IRQ	#15		0		0.000 000 000 s	0.000 000 ms	0.00 %
Scheduler			1 175		0.018 427 675 s	0.000 000 ms	0.23 %
Task 0x154E	@0	0 @ 0x00000000	107	0.435 823 600 s	0.013 327 425 s	0.000 000 ms	0.16 %
Task 0x3DD7	@0	0 @ 0x00000000	1	0.000 000 000 s	0.000 000 000 s	0.000 000 ms	0.00 %
angle task	@1	512 @ 0x200060D4	438	0.020 107 825 s	0.015 016 675 s	0.000 000 ms	0.18 %
thrust task	@2	512 @ 0x200097EC	0	0.000 000 000 s	0.000 000 000 s	0.000 000 ms	0.00 %
led1 task	@3	512 @ 0x200084E0	8	0.000 216 800 s	0.000 238 600 s	0.000 000 ms	0.00 %
led0 task	@3	512 @ 0x20007B8C	735	0.421 186 450 s	0.021 456 550 s	0.000 000 ms	0.26 %
physics task	@4	512 @ 0x20008E30	101	0.001 953 825 s	0.008 488 950 s	0.000 000 ms	0.10 %
Kernel's Timer Task	@5	256 @ 0x20003AA4	240	0.418 125 650 s	0.010 677 025 s	0.000 000 ms	0.13 %
lcd task	@5	512 @ 0x200072C8	989	0.006 002 250 s	5.255 037 775 s	0.000 000 ms	64.66 %
idle task	@6	512 @ 0x200069D4	98	0.639 983 250 s	0.143 583 150 s	0.000 000 ms	1.77 %
Idle			668		2.640 071 000 s	0.000 000 ms	32.49 %

I was unfortunately unable to get SystemView to show the times in the CPU Load window, but above we can see that the LCD task was taking up most the CPU load as expected since I set it to run every 100ms.

Below is a screenshot of the code space that my project utilizes

```
Flash used: 41732 / 1024000 (4%)
RAM used: 41188 / 256000 (16%)
```

I am pretty surprised by how small that number seems since I used a lot of OS resources in my projects, especially timers in order to change the PWM rate of the LEDs (for the case that did work).

My physics was pretty simple. For the cases when there was no thrust, I only had to worry about gravity being the only acceleration acting on the system and only in the y axis. To account for thrust I used the percentage of current thrust vs max thrust and its efficiency to be an acceleration value and then subtracted the gravitational acceleration to find the resultant velocity. To find the resulting velocities for the +/- 45 degree angles, I just approximated the trig values for that angle and multiplied by the thrust/gravitational accelerations to get both x and y velocities. One part of the physics that I didn't really implement was with the mass of the ship. I did apply physics to deal with depreciating fuel reserves though. I once again used the percentage of current thrust vs max thrust to decrement the total fuel supply on each time cycle the physics task was handled.

For the scaling of variable spaces, I found that it was much more difficult than I originally thought to try to use realistic physics values for the configuration data. I had to really experiment with the values and in the end, I ended up just using values that made the game playable regardless of if they were realistic or not.

If I had a couple more weeks to work on the project, there would be a couple of more things that I would also implement. First of all, I would fix the LED to properly change the pwm rate based on the blackout condition. This one is a given since it was one of the requirements of the project. Also, I wanted to create a fuel gauge on screen to show the player how much fuel is left for thrusting. I found it hard to know sometimes whether or not I had run out of fuel on some longer playthroughs. Lastly, I would probably make my ship model larger because right now it is insanely small (4 pixels).