1. RT Tasks:

a. Priority

Shield Task is 6

Slider Task is 3

Physics Task is 7

Display Task is 4

Idle Task is 30

Contexts

Name	Type	Stack Information	ctivations	Total Blocked Time	Total Run Time	Time Interrupte
SysTick IRQ	#15		0		0.000 000 000 s	0.000 000 r
Scheduler			401		0.009 347 025 s	0.000 000 r
Task 0x3453	@0	0 @ 0x00000000	28	0.113 988 250 s	0.006 653 675 s	0.000 000 r
slider task	@3	512 @ 0x2000E4...	110	0.004 276 675 s	0.007 641 000 s	0.000 000 r
display task	@4	256 @ 0x2000F0...	22	0.000 633 600 s	0.639 301 050 s	0.000 000 r
Kernel's Timer ...	@5	256 @ 0x2000B6...	264	0.009 131 875 s	0.039 455 900 s	0.000 000 r
shield task	@6	256 @ 0x2000E0...	1	0.000 024 825 s	0.000 061 400 s	0.000 000 r
physics task	@7	256 @ 0x2000E...	23	0.611 718 675 s	0.002 035 700 s	0.000 000 r
idle task	@30	256 @ 0x2000F4...	331	0.000 000 000 s	2.189 313 450 s	0.000 000 r
Idle			0		0.000 000 000 s	0.000 000 r

b. Execution times vs. deadlines

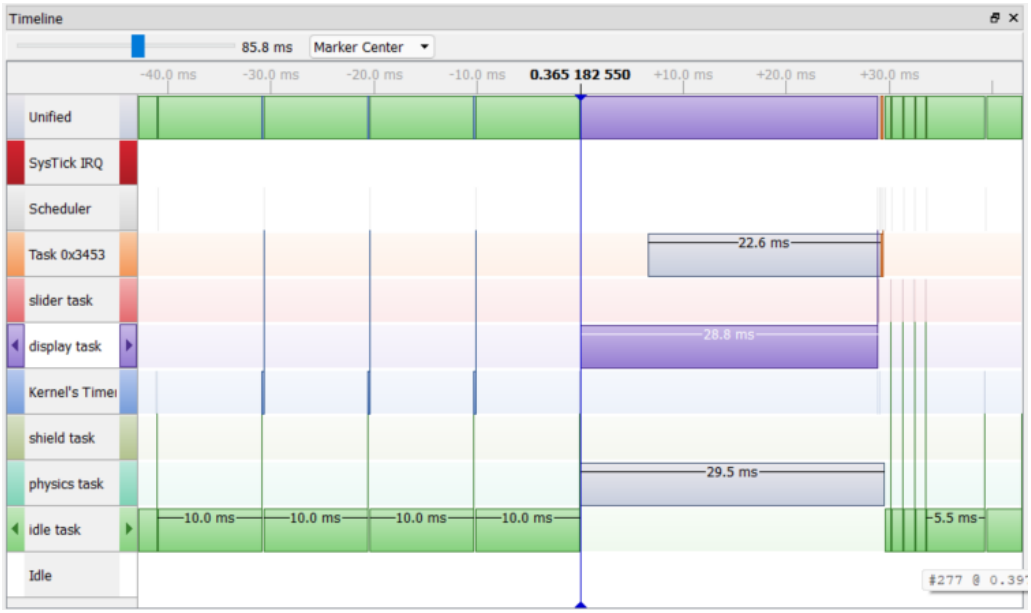


Figure2. LCD Task's execution time and Physics Task's execution time

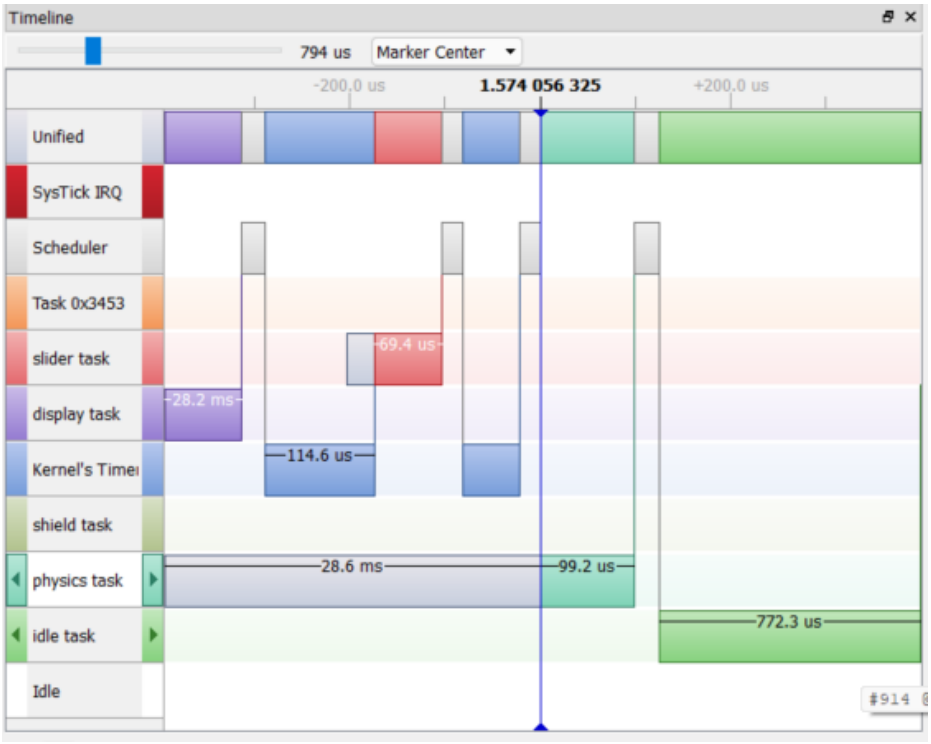
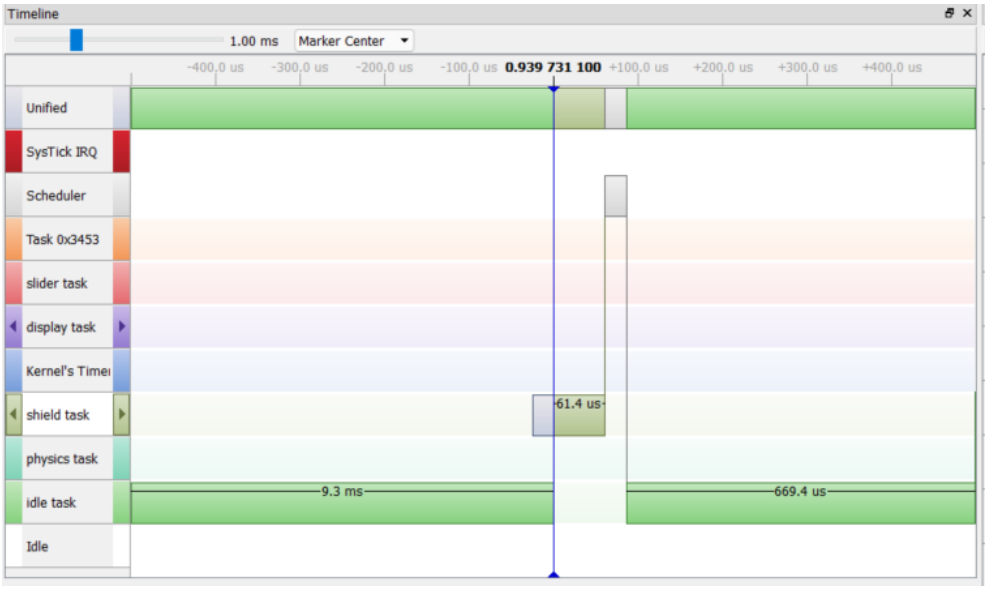


Figure3. Physics Task

It looks like LCD task and physics task occur at the same time, which seems odd because the priority of display tasks is higher than physics. Additionally, they both have about the same activations during the recording. Their behavior is more like in parallel instead of in sequence. However, the fact is the physics task only executes 0.1ms, but it has been blocked for a very long time. My best guess is that the gray period is used to calling other movement calculation functions which include several float calculations.

And one interesting thing I noticed is that the execution time of physics is equal to the execution time of LCD task, Timer Task and Slider Task.



Shield task looks normal as I planned.

Figure3. Shield Task's execution time

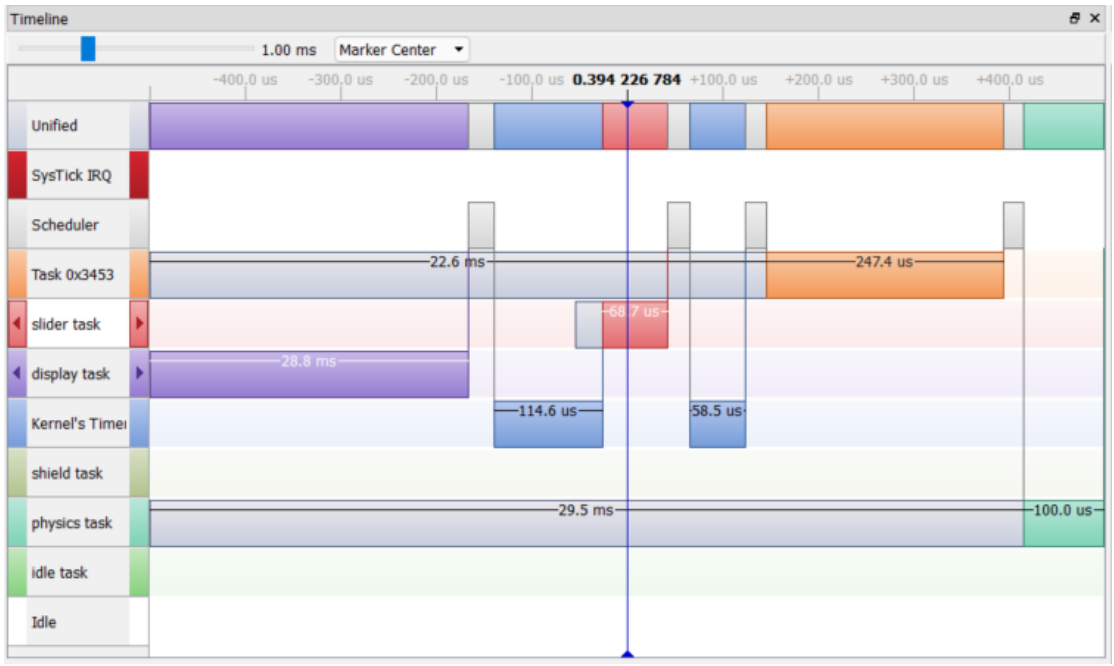


Figure4. Slider Task's execution time

Slider task looks normal because the timer task sends semaphore to it to start the sensing and calculation of the movement of the platform.

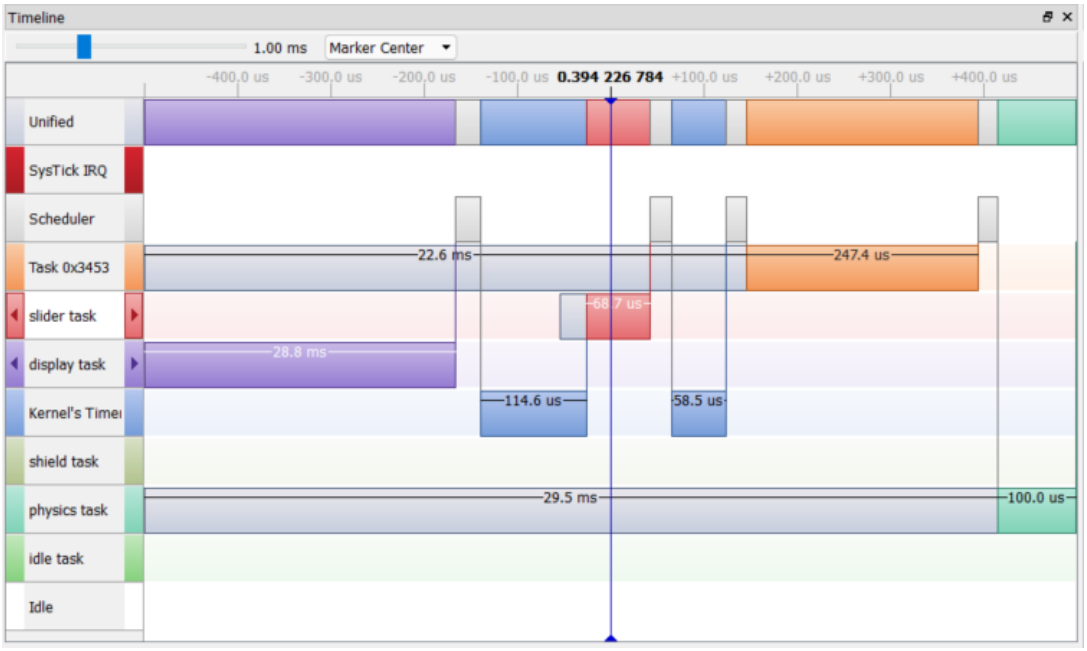


Figure5. Timer Task's execution time

I believe that the semaphore works in the way I planned, because it sends a semaphore to the slider task, and then the slider task starts to run.

2. **Code Space**:

```
C:\Users\LiChe\SimplicityStudio\v5_workspace\Lab7_All_Includes\GNU ARM v10.2.1 - Default>bash get_size.sh Lab7_All_Includes.axf 1024000 256000
Flash used: 71788 / 1024000 (7%)
RAM used: 64792 / 256000 (25%)
```

Flash used: 71788/1024000

RAM used: 64792/256000

I think the reason why RAM uses about 25% of space is because the LCD task spends lots of space to draw the circle, string and line. I can really feel the building project time is getting slower when I add 4 strings on the LCD. And I guess that is where the code space is being used.

Another reason I can think of is because I have done lots of float calculations in my physics task. I believe float calculations are much slower than integer calculations and they use more code space than integer calculations.

3. Evaluation of my approaches to the physics

In the physics task , I didn't use ODE. Instead, I use a linear approximation(1) to approximate the position of the ball in the game.

$$V_f = V_0 + a * t \qquad (1)$$

The limits I met before are that the initial condition can not be too large, otherwise the LCD can't show the animation properly. Additionally, for the calculation of KE boost function, I change the velocity directly by multiplying the original number with some constants. Because the real-word physics formulas make calculation more complex than I expect.

I have also tried to use all real-word physics formulas, but it didn't work very well because the numbers I have are too small. Then, the real-word physics formulas usually give me some number close to 0, which makes the game not playable.

I have also met the hardware issue related to Capsense which is not obvious at first. My best guess for this issue is related to the timing. In other words, the Capsense didn't get enough time to fully sense the position of human fingers before it started the next sensing task. Hence, the Capsense messed up itself due to lack of sensing time. Moreover, this issue is not reproducible because it doesn't happen periodically.

4. Scaling of variable spaces:

I find the game is playable when the magnitude of range is between 0-40 pixels. Otherwise, everything is going too fast so that the player may not have enough reaction time. Moreover, given the length of the platform as 40 makes the game easier. i.e. easier to catch the ball.

The range of m and Fmax given in the data set are not playable because they will make the acceleration of the ball too fast.

Beyond that, my data set is not scalable for some realistic units, like meter, kg, etc. However, the gravity g, percentage of KE increasing and reduction are acceptable values in my game design.

5. Functional Tests:

a. Direction of slider by controlling the Capsense. Success when the player can control the direction of the platform in either right or left.
b. HM will pass through the platform when its velocity is less than some threshold.
c. Laser function. Success when the current HM on LCD is gone.
d. Boost function. Success when the current HM rebounce with faster speed.
e. Wall rebounce. Success when the platform will rebounce from the canyon wall.
f. KE reduction. Success when the HM's speed is getting slower after impact with the platform.
g. LCD for Game over and Won the game.

I am able to pass all the functional tests shown in above.

6. Next Steps:

My next step would be to find scalable range, like converting 128*128 pixels to meters so that it's more like a real-world physics engine.

Beyond that, I hope I can find the problems with the LED prediction function and solve it with some playable values. Additionally, I want to make my animation more realistic, especially in the boost and laser function. And I hope I can get more time to dig deeper into the story of LCD drawing.