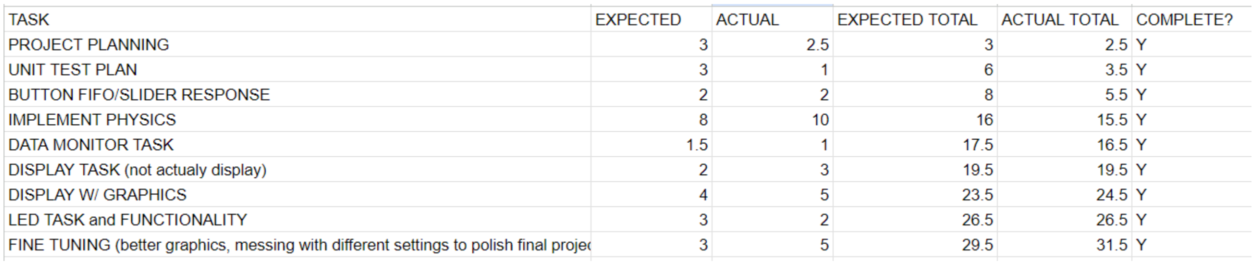
FINAL SUMMARY

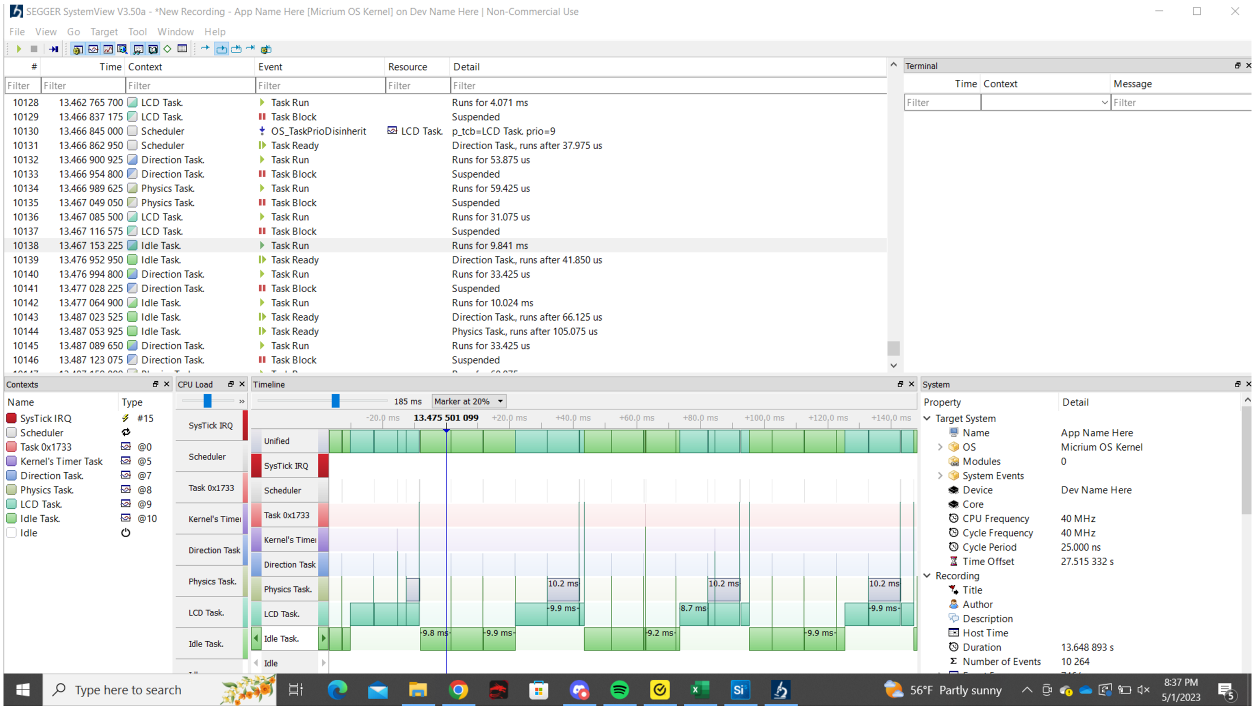
I was able to get my project fully functioning based on how I anticipated it to work. The slider applies a force to the sled, moving it around. The left button activates a shield which destroys any nearby satchels. The right button charges a shot that can be fired at the foundation to damage it. The satchels are correctly implemented to have one on screen at all times. The sled will be destroyed if you hit the wall going too fast. The shield and shot are dependent on an energy system, with a bar displayed at the top of the screen. Finally, when the foundation is destroyed, an animation plays showing the prisoners escaping. I accomplished this using three tasks - a direction task, physics task, and display task. The direction task simply indicates where on the slider your finger is. The display task mainly just updates what is on the display. The physics task does most of the heavy lifting, calculating things such as position of objects, whether objects collide, and if the slider is destroyed.

SUMMARY EFFORT and ESTIMATE NUMBERS

I did a little bit extra this weekend to prepare for the demo, such as adjusting some numbers and trying to get my graphics a little smoother. This brought my total to 31.5 hours, meaning I took about 107% of the expected time. I put this under the “Fine Tuning” task.



TASK	EXPECTED	ACTUAL	EXPECTED TOTAL	ACTUAL TOTAL	COMPLETE?
PROJECT PLANNING	3	2.5	3	2.5	Y
UNIT TEST PLAN	3	1	6	3.5	Y
BUTTON FIFO/SLIDER RESPONSE	2	2	8	5.5	Y
IMPLEMENT PHYSICS	8	10	16	15.5	Y
DATA MONITOR TASK	1.5	1	17.5	16.5	Y
DISPLAY TASK (not actualy display)	2	3	19.5	19.5	Y
DISPLAY W/ GRAPHICS	4	5	23.5	24.5	Y
LED TASK and FUNCTIONALITY	3	2	26.5	26.5	Y
FINE TUNING (better graphics, messing with different settings to polish final projec	3	5	29.5	31.5	Y



#	Time	Context	Event	Resource	Detail
10128	13.462 765 700	LCD Task.	Task Run		Runs for 4.071 ms
10129	13.466 837 175	LCD Task.	Task Block		Suspended
10130	13.466 845 000	Scheduler	OS_TaskPrioDisinherit	LCD Task.	p_tcb=LCD Task. prio=9
10131	13.466 862 950	Scheduler	Task Ready		Direction Task., runs after 37.975 us
10132	13.466 900 925	Direction Task.	Task Run		Runs for 53.875 us
10133	13.466 954 800	Direction Task.	Task Block		Suspended
10134	13.466 989 625	Physics Task.	Task Run		Runs for 59.425 us
10135	13.467 049 050	Physics Task.	Task Block		Suspended
10136	13.467 085 500	LCD Task.	Task Run		Runs for 31.075 us
10137	13.467 116 575	LCD Task.	Task Block		Suspended
10138	13.467 153 225	Idle Task.	Task Run		Runs for 9.841 ms
10139	13.476 952 950	Idle Task.	Task Ready		Direction Task., runs after 41.850 us
10140	13.476 994 800	Direction Task.	Task Run		Runs for 33.425 us
10141	13.477 028 225	Direction Task.	Task Block		Suspended
10142	13.477 064 900	Idle Task.	Task Run		Runs for 10.024 ms
10143	13.487 023 525	Idle Task.	Task Ready		Direction Task., runs after 66.125 us
10144	13.487 053 925	Idle Task.	Task Ready		Physics Task., runs after 105.075 us
10145	13.487 089 650	Direction Task.	Task Run		Runs for 33.425 us
10146	13.487 123 075	Direction Task.	Task Block		Suspended

Contexts

Name	Type
SysTick IRQ	#15
Scheduler	
Task 0x1733	@0
Kernel's Timer Task	@5
Direction Task.	@7
Physics Task.	@8
LCD Task.	@9
Idle Task.	@10
Idle	

System

Property	Detail
Target System	
Name	App Name Here
OS	Micrium OS Kernel
Modules	0
System Events	
Device	Dev Name Here
Core	
CPU Frequency	40 MHz
Cycle Frequency	40 MHz
Cycle Period	25.000 ns
Time Offset	27.515 332 s
Recording	
Title	
Author	
Description	
Host Time	
Duration	13.648 893 s
Number of Events	10 264

From the SystemView screenshot, we can see the three tasks listed in order of priority, with the Direction Task being first with a priority of 7, Physics with 8 going second, LCD with 9, and lastly the Idle Task with 10. I thought it was interesting that the direction task ran for such a short time compared to the other tasks, but it makes sense as my direction task basically just had to check where on the slider I was touching, which does not take long.

I did not realize at first that the order of how I did my code in the physics task would have a significant impact. I ended up deciding to handle collisions after I updated the positions of all of my moving parts. That is, I did all the math to find out where stuff was, then checked for collisions. One thing I did not foresee was missed collisions if something was moving fast enough, but I never noticed this issue occurring so I decided it was good. Perhaps if this was an issue, I could check both before and after things moved. A lot of my edge cases were dealt with by basically checking if the object is anywhere beyond a certain limit, which made it easy to deal with for some of the edge cases.

For scaling my display, I mostly just eyeballed what I thought was a reasonable playing space. I started with the given numbers, then felt that some of the conversions were annoying so I decided to fudge some numbers.

If I had another two weeks, I would put a lot of effort into improving my graphics. Now that I have the base game done, I feel like the next logical step would be to start making the game look better. The display feels more time consuming that it is difficult to implement, so I feel like two weeks dedicated to it would see good progress made. I also had a bug where my program takes a noticeable amount of time to register where on the slider I'm pressing. It was not game breaking, so I did not bother to fix it before the final submission, but if I had more time I would definitely fix this as it becomes annoying once you notice it.