Where My Project Stands:

Functional:

- Platform bouncing off edge
- Platform crashing into side at too high of speed
- Railshot shooting
- Railshot collision with foundation (two black boxes)
- Railshot speed increased based on platform speed
- Satchel collision with platform
- Satchel collision with shield
- Satchel parabolic arc (Tracks with platform to make more difficult)
- Generator functionality (Made harder, charges only when R button pressed)
- Shield functionality
- Foundation damage (With visual indication)
- Win screen
- 2 separate loose screens, Crash, explode,

Partially Functional:

- Collision isn't perfect
- Screen doesn't update at a great speed
- Hard to play/ actually win game

Non-Functional:

- LED functionality
- Prisoners escape time, goes to Win screen instead
- Railshot harming platform.
- Extra Credit Options

Task	Estimated Effort	Actual Effort	Status
Button	5 min	1 hour	Ditched in the end due to bug
Capsense	15 min	5 min	Complete
Display	5 hours	10 hours	Complete
Physics	5 hours	10 hours	90% Complete
Timers	5 min	2 hours	Complete
Interrupts	1 hour	1 hour	Complete
Idle	Instant	Instant	Complete

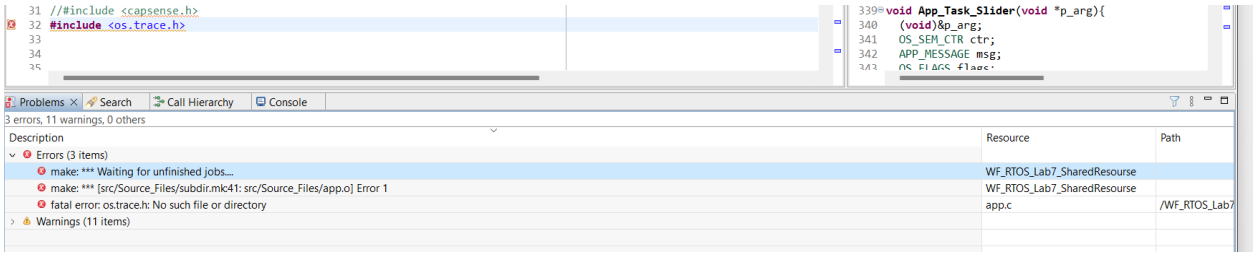
I listed what was complete/not complete above. However, I will talk about what I would have liked to include/ what I would want to change. I did not implement the LED functionality simply because I ran out of time. However, if I were to implement them I would continue to use OS timers for their functionality. I also didn't implement the rail shot harming the platform simply

because of how I implemented the generator functionality. It would seem odd to increase the total power held in the generator then fire and hit the platform. Lastly, I didn't implement the prisoner feature. This could have been done pretty easily by initiating some sort of "prisoner animation" when my game win screen is currently happening. However, the time to create all the new objects for the animation wasn't worth it.

Analysis:

RT Tasks:

I was unable to use Segger as my os.trace would not include:



Code Space:

My app.c is 1109 lines and honestly is a mess. I pride myself on code organization but because I had to inherit Will(TA)'s project there are a lot of artifacts from his code and my old working code that I copied over. Overall I have a lot of useful comments that helped me find/remember things but the code space is very crowded and chaotic.

Evaluation of your approaches to the physics update requirement:

I handled my physics by updating all of the appropriate values (ie velocity and speed) then I would check each edge case after the values were calculated. If the edge case requirements were met I would then adjust the calculated values accordingly. Firstly, I really underestimated the physics task. For some reason I thought it would be easy to send all of the updates of the physics task to the display task with a message queue, this is impractical and I used a Mutex system instead.

Scaling of Variable Spaces:

I found a lot of the information about the Canyon size and Castle size irrelevant. I felt like it was hard to translate those numbers into values that could be passed into the GLIB API and draw functional objects. The other variables for the physics task were extremely useful but they were much too large for a 128x128 screen. I choose to scale everything down. How I implemented everything was 1.Implement physics with given values, 2. Test those values 3. Adjust based on results on screen.

Next steps:

If I were to keep working on this project I would scale everything down more to make the game easier and more playable. This would require me to modify/change pretty much every constant in the Physics and Display task so it would take a while.