Tommy Ramirez
4/27/22
ECEN 3753

Project HarkonnenPong Final Week

Unit Testing

Platform Motion

- Test if platform doesn't move when no force is applied and velocity is 0 (Pass)
- Test if platform maintains nonzero velocity when no force is applied (Pass)
- Test if platform increases velocity when force is applied in the same direction of the velocity (Pass)
- Test if platform decreases velocity when force is applied in the opposite direction of the velocity (Pass)
- Test if low and high force has difference on accelerations (Pass)

Holtzman Mass Motion

- Test maintained x velocity of Holtzman Mass (Pass)
- Test constant downward acceleration of Holtzman Mass (Pass)

Holtzman Mass Out of Bounds Check

- Test LED flag set when Holtzman Mass crashes (Pass)
- Test if Holtzman Mass is replaced after exiting the range (Pass)
- Test if Holtzman Mass is replaced for a number of times based on a variable (Pass)

Functional Testing

1. At reset observe that the left LED is blinking at the slowest rate. (Pass)
2. Place your finger on the touch slider and slide your finger to the middle right and observe the platform accelerating rightward. (Pass)
    a. Release your finger and observe the platform maintaining velocity. (Pass)
3. Slide your finger to the far right before it hits the border and observe the platform accelerating faster rightward. (Pass)
    a. Release your finger and observe the platform maintaining velocity. (Pass)
4. Slide your finger to the far left before it hits the border and observe the platform decelerating before accelerating leftward. (Pass)
    a. Release your finger and observe the platform maintaining velocity. (Pass)
5. Press Btn1 and observe the Holtzman Mass count and the y-position of the holtzman mass reset to the top of the screen. (Pass)
6. After pressing Btn1 several times, observe that the button no longer has any effect. (Pass)
7. Move the platform away from the falling Holtzman Mass and observe it activates the game over screen and the right LED blinks at 1 Hz. (Pass)
8. Move the platform in the Holtzman Mass' path and observe it bounce off the platform. (Pass)

    a. Observe after repeated bounces that the peak of the parabola is getting lower. (Pass)
    b. After several bounces, observe that when the peak is vertically close to the platform that the mass phases through the platform. (Pass)

9. Press Btn0 when the Holtzman Mass is near the platform but hasn't hit it yet and observe that upon bouncing, the peak of the parabola is higher than before. (Pass)
10. Use the slider to match the right and left LED's pwm while the right LED is blinking and observe that the platform will collide with the holtzman mass. (Pass)
11. Bounce the holtzman mass outside of the canyon and observe the same thing happen as what happens when btn1 is pressed. (Pass)

Functionality Deliverables and Usability Summary

This week I fully implemented all the remaining required features including the right LED required force prediction PWM and the collisions with increasing and decreasing kinetic energy. I reworked the task diagram after last-minute realizations on the interactions between some tasks. most of the initial variables into my function. I also got the bounce function working for the I also took into account that the lasers weren't automatically on, but still kept the feature toggleable. I also realized that I don't need to factor in the direction for the PWM of the left LED, so I adjusted that in the code and my functional test. I added a victory and game over screen that limited the LCD activities to just that screen until the program was reset. I replaced time delays with semaphores due to how the delays were potentially causing issues. I reworked the physics task priority to have the shield task take a higher priority. I had implemented the out-of-bounds flag for when the HM was above the screen. Lastly, I reworked some of my physics code after realizing some unit conversions were wrong, but that resulted in the tau physics default value not being implemented in the conversion from cm/s^2 to cm/s/tau and cm/s to cm/tau. In the end, my program had a working demo and was actually winnable for the first time.ing only the collision and LED tasks for implementation. Lastly I got the HM to bounce off the canyons and platform, but the shield boost still requires implementation.

Summary effort & estimate numbers

I have completed **71.25%** of my currently-scoped, estimated work time (85.5 actually spent /120hr total estimate) with **100%** of the initially-estimated work. (120 estimated for the items I have completed, of 120hr total estimate). For the work that has been completed, I took **0.7125x** (85.5/120) as much time as I estimated.

No scope changes have been. My latest scope is my original scope (120 hrs).

In-scope work items

Completed before this week:

- Project Reading and Task Diagram First Draft creation (est 10 hrs) (actually 8 hrs)
- Learn How to Manipulate LCD basic drawings (est 20 hrs) (actually 1 hr)

- Lose Condition LED Control (est 1 hr) (actually 0.5 hr)
- Configure slider to control LED PWM (est 2 hrs)(actually 3.5 hrs)
- Create Tasks and Test on Segger (est 2 hrs)(actually 2 hrs)
- Slider Testing/Fixing (est 6 hrs)(actually 2 hr)
- Task Diagram Revision for clarification and necessary optimization (est 3 hrs) (actually 2 hrs)
- Laser Implementing (est 1 hr)(actually 1 hr)
- Task Integration Testing (est 10 hrs)(actually 4 hrs)
- Platform Motion (est 1 hr)(actually 9 hrs)
- Platform Bounce (est 2 hrs)(actually 4 hour)
- Basic Platform/HM collision (est 2 hrs)(actually 1.5 hrs)

Completed this week:

- HM Physics Unit (est 10 hrs)(12.5 hrs so far)

This took longer than expected mostly because of the larger-than-expected amount of necessary thought required for the physics unit conversions and the need for ensuring that the directional accelerations worked as intended. I added the reduced and increased kinetic energy values and implemented the shield that gives the boost and tied it to the physics timer since the physics task is reliant on the shield to some degree.

- Task Unit Testing (est 10 hrs)(6.5 hrs so far)

I realized upon trying to create the LED prediction that it is too complicated for a unit test and should remain a functional test and no new unit tests are necessary at this stage. This was shorter than expected due to the simple nature of acceleration and velocity.

- Motion/Position Physics Task (est 10hrs)(12 hrs so far)

Physics unit tests passed. I implemented the boost collision physics and confirmed that kinetic energy is lost when the HM mass hits the platform. This took longer than expected mostly because of the larger-than-expected amount of necessary calculations required for the physics to work with the shield and the failing/bounce/laser flagging conditions.

- Platform/HM collision with bounce (est 5 hrs)(5 hours so far)

The bounce collision was added and tested to ensure that kinetic energy increases when the shield is on for when the HM collides during the short period of time after the btn 0 was pressed and that the kinetic energy decreases when colliding with the platform when not in the boosting time window. Default values of collision physics have also been set. implemented. Default values of collision physics have also been set.

- Qualification Check/Debugging (est 15 hrs)(actually 7 hours)

All tests pass. boost physics which is close to completion. I could have spent more time on this, but there would have been diminished returns.

- Commenting and Code Cleanup (est 5 hrs)(actually 1.5 hours)

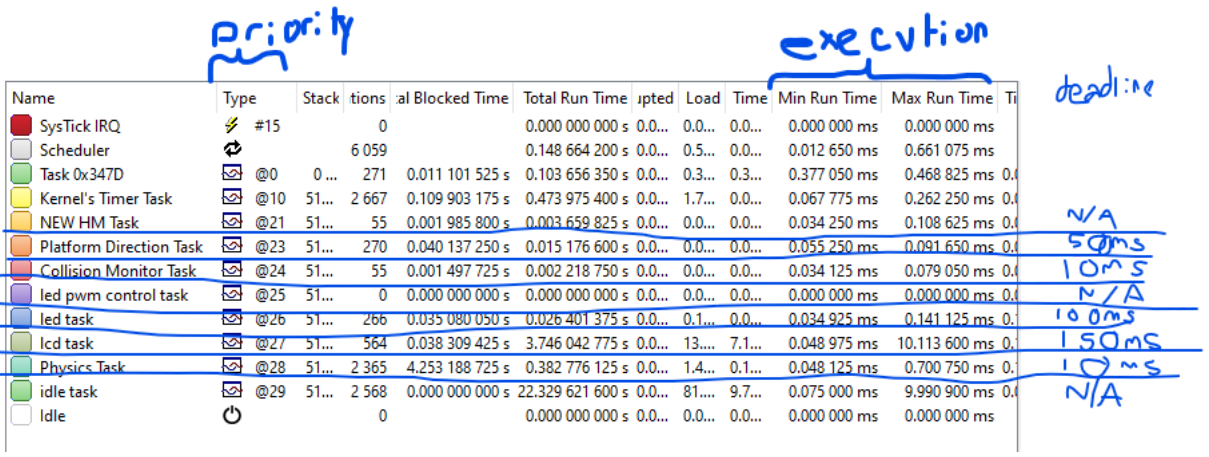
Most of the lines of code were aligned rather than having random indents. Some newline gaps were also shortened. Commenting was provided for most of the unit conversions in the physics

- Platform Interception Prediction LED implementation (est 5 hrs)(actually 2.5 hours)

I received some assistance and guidance in creating the function, which helped shave down some time. I also reminded myself of the kinematic equation and made the function my own. I manipulated the equations to get the acceleration needed for the platform to collide with the HM, then translated the output to a PWM of the right LED.

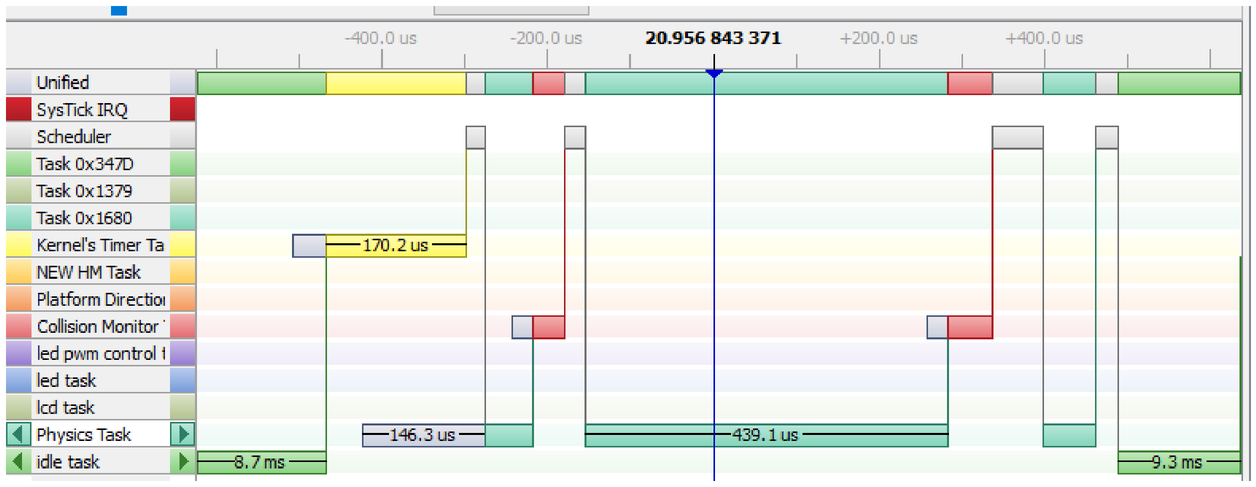
Solution Analysis

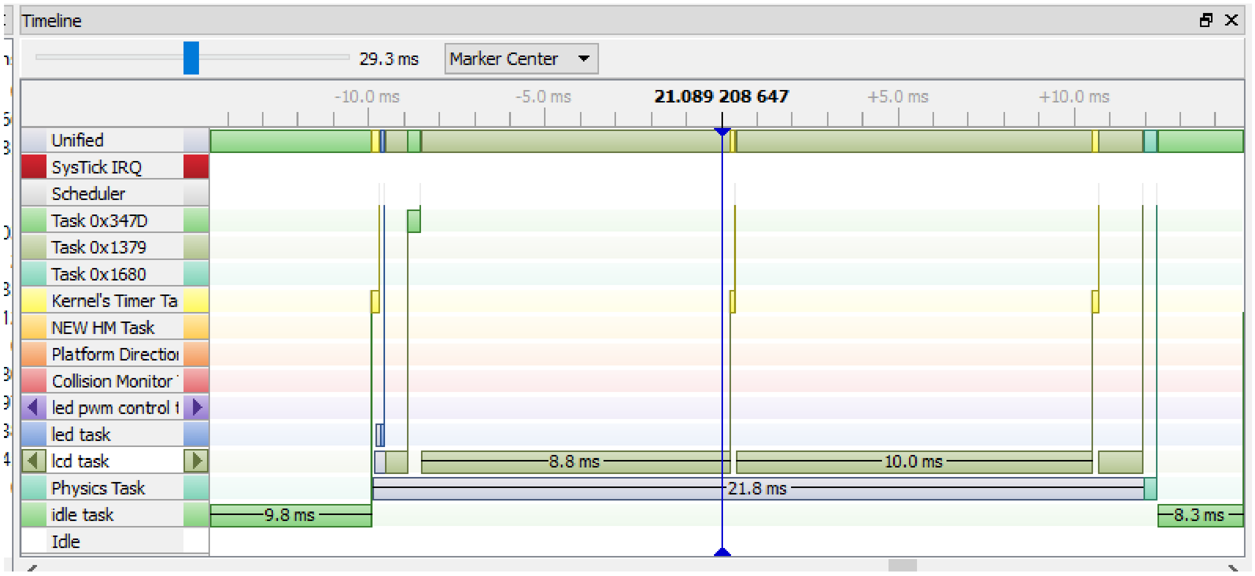
RT Tasks



Name	Type	Stack	tions	al Blocked Time	Total Run Time	ıpted	Load	Time	Min Run Time	Max Run Time	Ti	deadline
SysTick IRQ	#15		0		0.000 000 000 s	0.0...	0.0...	0.0...	0.000 000 ms	0.000 000 ms		
Scheduler			6 059		0.148 664 200 s	0.0...	0.5...	0.0...	0.012 650 ms	0.661 075 ms		
Task 0x347D	@0	0 ...	271	0.011 101 525 s	0.103 656 350 s	0.0...	0.3...	0.3...	0.377 050 ms	0.468 825 ms	0.(	
Kernel's Timer Task	@10	51...	2 667	0.109 903 175 s	0.473 975 400 s	0.0...	1.7...	0.0...	0.067 775 ms	0.262 250 ms	0.(	
NEW HM Task	@21	51...	55	0.001 985 800 s	0.003 659 825 s	0.0...	0.0...	0.0...	0.034 250 ms	0.108 625 ms	0.(	N/A
Platform Direction Task	@23	51...	270	0.040 137 250 s	0.015 176 600 s	0.0...	0.0...	0.0...	0.055 250 ms	0.091 650 ms	0.(	50ms
Collision Monitor Task	@24	51...	55	0.001 497 725 s	0.002 218 750 s	0.0...	0.0...	0.0...	0.034 125 ms	0.079 050 ms	0.(	10ms
led pwm control task	@25	51...	0	0.000 000 000 s	0.000 000 000 s	0.0...	0.0...	0.0...	0.000 000 ms	0.000 000 ms	0.(	N/A
led task	@26	51...	266	0.035 080 050 s	0.026 401 375 s	0.0...	0.1...	0.0...	0.034 925 ms	0.141 125 ms	0.1	100ms
lcd task	@27	51...	564	0.038 309 425 s	3.746 042 775 s	0.0...	13....	7.1...	0.048 975 ms	10.113 600 ms	0.1	150ms
Physics Task	@28	51...	2 365	4.253 188 725 s	0.382 776 125 s	0.0...	1.4...	0.1...	0.048 125 ms	0.700 750 ms	0.1	10ms
idle task	@29	51...	2 568	0.000 000 000 s	22.329 621 600 s	0.0...	81....	9.7...	0.075 000 ms	9.990 900 ms	0.(	N/A
Idle			0		0.000 000 000 s	0.0...	0.0...	0.0...	0.000 000 ms	0.000 000 ms		

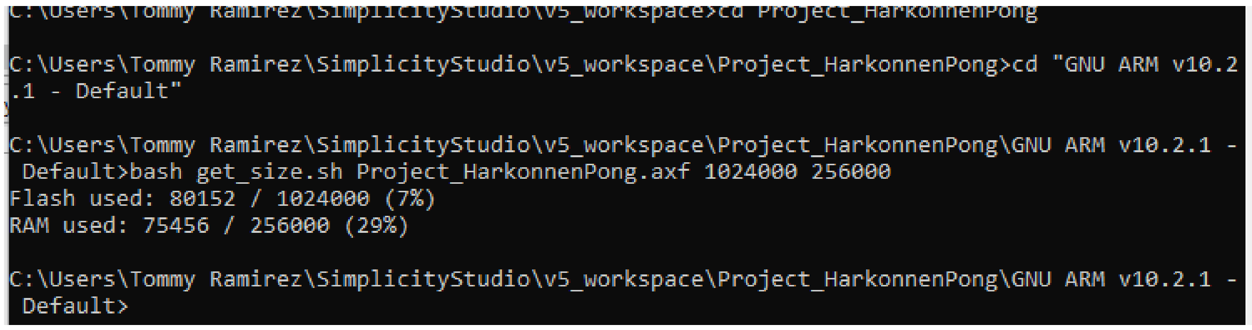
There were issues where the program would inexplicably stop running and others where only the segger would stop working.





Some noticeable details include that the is interrupted by the collision monitor task due to a flag set in the physics task, and that the physics task misses a deadline due to its low priority but its deadline won't cause any major issues so the deadline is a little soft.

Code Space



```
C:\Users\Tommy Ramirez\SimplicityStudio\v5_workspace>cd Project_HarkonnenPong

C:\Users\Tommy Ramirez\SimplicityStudio\v5_workspace\Project_HarkonnenPong>cd "GNU ARM v10.2
.1 - Default"

C:\Users\Tommy Ramirez\SimplicityStudio\v5_workspace\Project_HarkonnenPong\GNU ARM v10.2.1 -
 Default>bash get_size.sh Project_HarkonnenPong.axf 1024000 256000
Flash used: 80152 / 1024000 (7%)
RAM used: 75456 / 256000 (29%)

C:\Users\Tommy Ramirez\SimplicityStudio\v5_workspace\Project_HarkonnenPong\GNU ARM v10.2.1 -
 Default>
```

The high RAM usage is likely from the abundance of floats and the intense calculations needed in the LCD task for drawing and for creating a complex, moving circle. That updates every 150ms.

Physics update approach

For my physics task acceleration calculations, I take the maximum force and divide by the mass to get the platform acceleration and convert to cm/s^2 and scale down the maximum platform acceleration to a more manageable value. Then, depending on finger position on the capsense, I add all (max force/outer capsense), 25% (partial force/inner capsense), or none of the acceleration (finger not on capsense). The acceleration is converted to cm/s/τ and added to the platform velocity (in cm/s) every τ. The x positions for the HM and platform both convert the x velocities to cm/τ and add to the current x position. The y velocity of the HM increases by the gravity converted to from mm/s^2 to cm/s/tau and then the y position is incremented similar to the x position.

For my physics task kinetic energy modifications, I converted the HM's x and y velocities to m/s and added their squares together before square rooting the sum to get the total velocity. That result was multiplied by mass * 0.5 to get the kinetic energy. The kinetic energy is multiplied by (100 + the boost)/100 to get the increased kinetic energy and multiplied by reduction/100 to get the decreased kinetic energy. The new kinetic energy is doubled and divided by the mass and then square rooted to get the new total velocity. The angle of the HM into the platform is found using arctan(velocity y/velocity x) which is then used in cosine and sine functions to multiply with the total velicity and get the new x and y velocities for the HM respectively. The y velocity is also flipped to make the HM go up. This bouncing only happens when above the minimum HM downward velocity and the HM's center of mass is within the platform.

Scaling of variable spaces

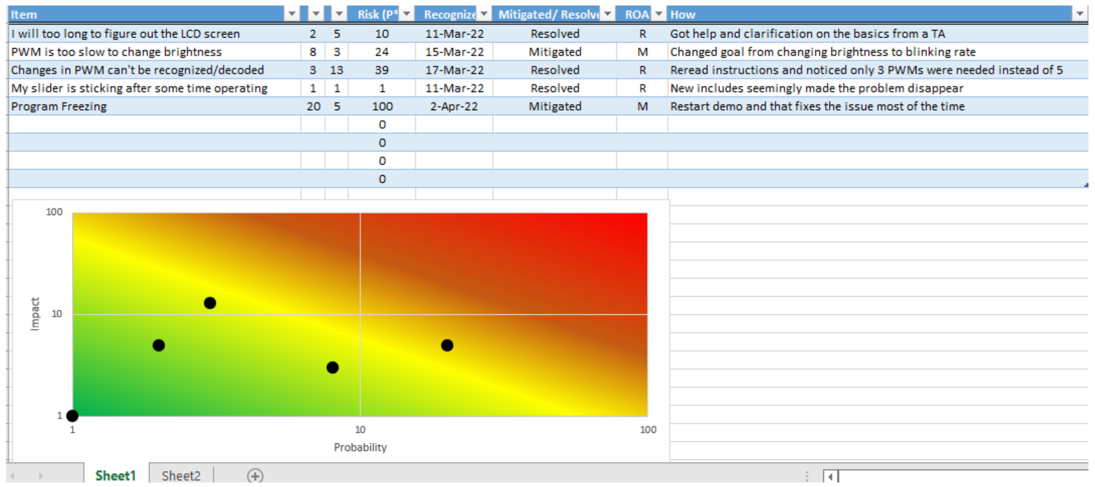
I found that for most of the given variable values, the game is playable with 1 pixel representing 1000 cm and 1 ms representing 1 ms.

An exception is that I had to scale down the acceleration value by a factor of 500 in order to have the platform not bouncing uncontrollably and also not accelerate too slowly to not be able to catch the HM. This resulted from a too high MaxForce, a too little platform mass value, and/or the physics happening too fast. The first two values were default values. The last would have impacted the HM falling if it changes, and the HM's motion is fine as it is. I also had to mess with the new Force and time values for the right LED required force prediction.

Potential next steps

My potential next steps would have been separating the code from my app.c file into separate files to make the 1000 lines of code easier to search through. I also would have made the physics conversions able to be changed by changing the default tau physics value. The right LED prediction was also a little messy, so I could clean that up with more proper declarations. I would also have possibly improved the animations for certain flag conditions like the victory and game over screens and the laser and bounce flags (make the platform blue when shield is active). Lastly I would have created more mutexes for shared values between tasks.

Risks



Item			Risk (P*	Recognize	Mitigated/ Resolve	ROA	How
I will too long to figure out the LCD screen	2	5	10	11-Mar-22	Resolved	R	Got help and clarification on the basics from a TA
PWM is too slow to change brightness	8	3	24	15-Mar-22	Mitigated	M	Changed goal from changing brightness to blinking rate
Changes in PWM can't be recognized/decoded	3	13	39	17-Mar-22	Resolved	R	Reread instructions and noticed only 3 PWMs were needed instead of 5
My slider is sticking after some time operating	1	1	1	11-Mar-22	Resolved	R	New includes seemingly made the problem disappear
Program Freezing	20	5	100	2-Apr-22	Mitigated	M	Restart demo and that fixes the issue most of the time
			0				
			0				
			0				
			0				