Week 3 Work Report

Goals for the Week

1. Intermediate Tasks
    a. Complete a minimum of 20% or more of the items in your scoped work list each week
2. Test Plan and Results
    a. Augment your unit tests by writing your actual test code and summarizing execution results.
3. Current Project Status
    a. Accurate summary statement of your functionality deliverables and usability so far.
    b. Summary effort and estimate numbers.
4. List of in-scope work items
    a. Indicate complete or not-yet-complete, along with your estimates of how long you think they will take in total for each.
5. Update Risk Register

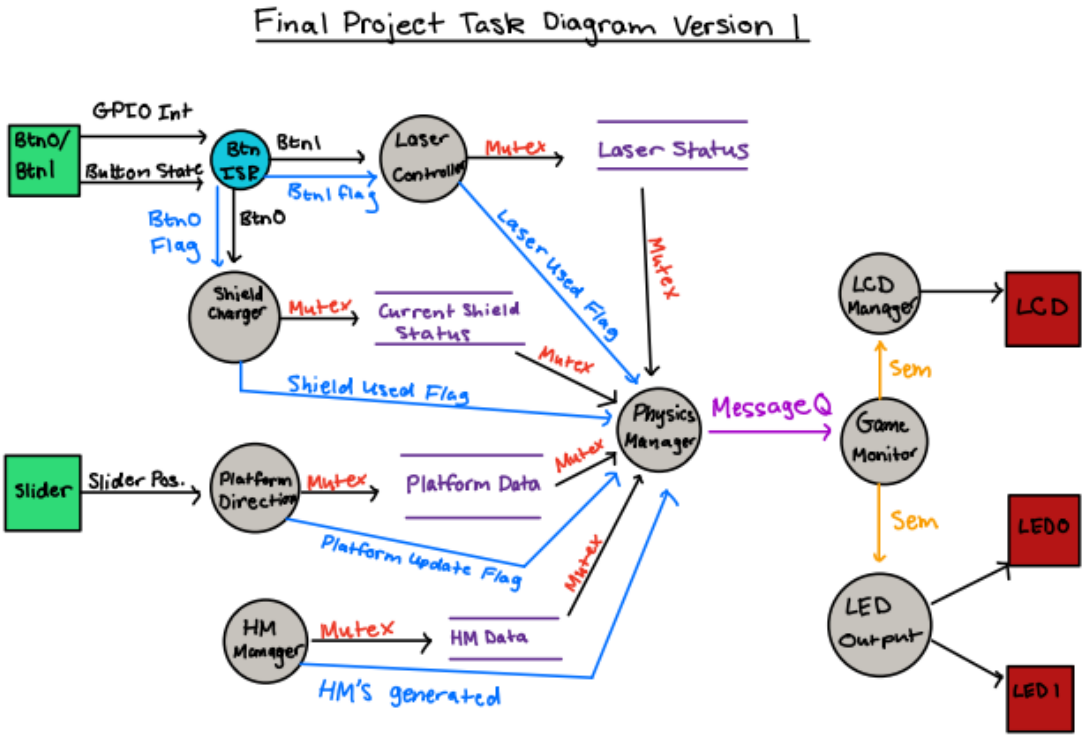
Week 3 Work Time Estimation

1. Intermediate Tasks: 7 - 9 Hours
    a. I didn't get as much progress done last week as I had liked to. I need to reassess how long each task is going to take because I am not seeing the results I would like to. This week I need to make significant progress on task code along with ITC implementations.
2. Test Plan and Results: 2 - 5 Hours
    a. Implementation of my unit tests will be a large undertaking. I don't think that by the end of the week that everything will be complete or set in stone. I think the testing process will be evolving as the project continues. For now I need to lay solid groundwork for future testing by implementing base tests.
3. Current Project Status: 15 - 30 Min
    a. Simple couple sentence to paragraph update on my progress.
4. List of in-scope work items: 1 Hours

a. This will take some more time this week as I need to reevaluate my time estimates for each item. My timings need to change because I haven't been able to complete the tasks within estimated time.

5. Updating the risk register: 30 - 45 Min
    a. Thinking of risks should take most of the time here. Inputting them should take 5 - 10 minutes.

Task Diagram



Test Plan and Results (Cutting Points)

This week I needed to complete a testing plan for going forward utilizing at least 3 "cutting points". I need to have a list of at least 10 tests. This will include Unit Tests and Functional Tests. I will create a list of each type below along with a short description of how they work. This is the initial plan going forward and is subject to change drastically upon new information or design changes.

Unit Tests

Unit tests will cover the functionality of the code within tasks. They need to be isolated in order to unit test them, so shared data structures with other tasks will pose a challenge. The main idea behind these tests is to make sure, assuming that the ITC is working correctly, that the jobs of each task are working as expected. The tests will likely be conducted on a task by task basis with some assumptions on data structures.

1. Input Unit Test
    a. Code to verify that inputs are being read and sent to the correct tasks
    b. Test will consist of monitoring the inputs and verifying the correct task activating in reaction.
2. ISR Unit Test
    a. Verification of ISR's being called upon button pushes and calling the appropriate functions with the expected output
    b. Test will consist of verification of ISR entry and function calls via tracking code flow and marking areas of code that are activated in response to the ISR's
3. Shield Charger Task Unit Test
    a. Verification of the inner-task behavior based on the shield button input and activation of the task via button ISR
    b. This test will utilize one cutting point
        i. Cutting off the Shield Charger from the physics manager
4. Laser Controller Unit Test
    a. Verification of the inner-task behavior based on the laser button input
    b. This test will utilize one cutting point
        i. Cutting off the Laser Controller from the physics manager
5. Platform Direction Unit Test
    a. Verification of the inner-task behavior based on the capsense input
    b. This test will utilize one cutting point
        i. Cutting off the Platform Direction task from the Physics Manager
6. HM Manager Unit Test
    a. Verification of the creation of our HM's
    b. This test will utilize one cutting point
        i. Cutting off the HM Manager from the Physics Manager
7. Given Dummy Data - Physics Manager Unit Test
    a. Verification of Physics Manager manipulating our dummy data.

Functional Tests

Functional tests will cover the behavior of the outputs based on a specific set of inputs. These tests will involve testing the behavior of the system when taking inputs. These tests will be periodically performed as the project progresses. This way we can make sure that future changes are not affecting already existing code.

1. Platform Movement
    a. We want to observe that utilizing the slider we can see accurate and responsive movement of the platform
2. Shield Behavior
    a. We want to observe the shield utilization via the button press along with the charging behavior
3. HM Behavior
    a. We want to observe the creation and launching of HM's according to the parameters created by the HM. This will have less inputs than the other functional tests.
4. Final Game Behavior
    a. Here we want to test a combination of all our previous tests. We need to rigorously test our final project with every piece working together.

Testing Plan Summary

As of this week, I am implementing the general plan of what needs to be tested as I develop the project. I will include more detailed explanations of each test step-by-step once we get to the implementation stage. For now, none of the tests are implemented. Tests may be added or removed as the circumstances and specifications of my design change. As of the end of Week 3, I have a solid understanding of what modules need to be tested and a general idea of how.

Test Implementation

I did not get to implement most of my testing plan this week. I am currently struggling to keep up with my set deadlines and need to reevaluate my approach to this project. I have decided to implement compile-time switches to enable individual unit tests. There will be basically two sections to each testable module: the actual code for that section and a testing section that will try to replicate the behavior of the actual code and run tests on it.

Current Project Status Week 3

Most of Week 3's time went into code and test implementation. I decided to take the development in a new direction. Instead of batching all of the work for some tasks into one, I have started doing partial completion based on the flow of what I am working on. This makes sure that I have all of the relevant information in front of me when I am implementing new features or data instead of having to guess what I will need in the future. This change definitely set me back temporarily, but I believe will be very beneficial in the future weeks. Because of this change in direction, I have not reached my desired targets for testing implementation and my in-scope work items will have to be broken up into smaller chunks.

I have completed 37.6**%** of my currently-scoped, estimated work (20.7 actually spent 55/hr total estimate) in 89.2**%** of the initially-estimated time. (23.07 estimated for the items I have completed, of 55hr total estimate). For the work that has been completed, I took **1.12x** (9.16/8.2) as much time as I estimated.

List of In-Scope Work Items

Here is my organized list of all of the things that need to be done in order to complete this project along with documentation. Some are vague temporarily so when they appear in the upcoming weeks I can update them with specifics. I am going to be giving each item a generous amount of time since humans are terrible at predicting how much time a task will take.

Total List

Programming

1. Create Project and File Structure (30 Min - 1 Hour) - Complete
2. Setup Button Inputs and ISR's (15 - 30 Min) - Incomplete
3. Initialize Data Structures for the game (45 Min - 1.5 Hours) - Incomplete
4. Initialize ITC's and MutEx's (30 Min - 1 Hour) - Incomplete
5. Complete Shield Task (1 - 2 Hours) - Incomplete
6. Complete Laser Task (1 - 2 Hours) - Incomplete
7. Complete Platform Task (1 - 2 Hours) - Incomplete
8. Complete HM Task (1 - 1.5 Hours) - Incomplete

9. Complete Physics Task (6 - 8 Hours) - Incomplete
10. Complete Game Monitor Task (2 - 4 Hours) - Incomplete
11. Complete LCD Task (2 - 6 Hours)- Incomplete
12. Complete LED Task (1 - 2 Hours) - Incomplete

Reports

1. Task Diagram (1.5 - 2.5 Hours) - Complete
2. Week 1 Report (1.5 Hours) - Complete
3. Week 2 Report (1.5 Hours) - Complete
4. Week 3 Report (1.5 Hours) - Complete
5. Week 4 Report (1.5 Hours) - Incomplete

Risk Registers

1. Week 1 Risk Register Update (10 - 20 Min) - Complete
2. Week 2 Risk Register Update (10 - 20 Min) - Complete
3. Week 3 Risk Register Update (10 - 20 Min) - Incomplete
4. Week 4 Risk Register Update (10 - 20 Min) - Incomplete

Status Updates

1. Week 1 Status Update (10 - 20 Min) - Complete
2. Week 2 Status Update (10 - 20 Min) - Complete
3. Week 3 Status Update (10 - 20 Min) - Complete
4. Week 4 Status Update (10 - 20 Min) - Incomplete

Testing

1. Week 1 Testing (10 - 20 Min) - Complete
2. Week 2 Testing (3 - 4 Hours) - Complete
3. Week 3 Testing (3 - 4 Hours) - Incomplete
4. Week 4 Testing (3 - 4 Hours) - Incomplete

Week 3 List

1. Setup Button inputs and ISR's (15 - 30 Min) - Complete

    a. Took a lot longer than expected because I changed how the ISR's handle flagging the respective tasks. It was also made more complex because each button will flag a different task so not only did the ISR handling change, but so did the flag grouping. 2 hours used.
2. Initialize data structures (45 Min - 1.5 Hours) - Complete
    a. I have decided that the data structures will be implemented as I program the tasks that need them. This way I have all of the information I need to fill the data structures instead of just guessing what parameters will be needed in each structure. I have made some temporary structure definitions that will be updated as I get to more tasks. 1.5 hours used.
3. Initialize ITC's and MutEx's (30 Min - 1 Hour) - Complete
    a. I have initialized the relevant data pertaining to the shield task as that is the one I was focused the most on this week. There will be more ITC added as I need them in the future. 1.5 hours used.
4. Complete Shield Task (1 - 2 Hours) - Almost complete
5. Week 3 Report (1.5 Hours) - Complete
    a. 1.5 Hours
6. Week 3 Risk Register (10 - 20 Min) - Complete
7. Week 3 Status update (10 - 20 Min) - Complete
8. Week 3 Testing (3 - 4 Hours) - In Progress

   Total Time Estimate: (8.2 - 12.66)

   Total Time Used: 9.16 Hours

Updating Risk Register

This week for the risk register, I updated my falling behind risk because it has started to manifest. I also resolved my file structure risk from last week since I did some reviewing of my past projects and other documentation to verify my structure.