Murfin ECEN3753 Project Unit Testing Plan

See Task Flow Diagram

UNIT TESTS

**1. Theta Boundary Values**

Test: Test to make sure that theta is within boundary values of -90 to +90 degrees. Anything outside of that should flag a “Pendulum Fell” violation

Status: Implemented, (currently main.c ctrl+f “UNIT TEST 1”)

**2. Xmax Boundary Values**

Test: make sure that a physics update doesn’t push the pendulum’s base outside of the constant boundary values. If it does, it should flag an “Xmax” violation.

Status: Implemented, but with currently unknown boundary values being compared. (currently main.c ctrl+f “UNIT TEST 2”)

**3. Force Gain Boundary Values**

Test: make sure that a button push doesn’t update past the maximum or minimum force gain values.

Status: Implemented. (currently main.c ctrl+f “UNIT TEST 3”)

**4. Force Boundary Values**

Test: Make sure that capsense inputs only adjust force to expected/controllable values.

Status: Implemented (currently main.c ctrl+f “UNIT TEST 4”)

**5. Buttons Adjust Force Gain Value**

Test: Make sure that button pushes actually adjust force gain value in shared data struct.

Status: Implemented (currently main.c ctrl+f “UNIT TEST 5”)

**6. Capsense Measurements Adjust Force Value**

Test: Make sure that capsense inputs actually adjust force value in shared data.

Status: Implemented (currently main.c ctrl+f “UNIT TEST 6”

**7. Proper Physics Calculations**

Test: Make sure that physics is being calculating properly.

Status: Temporarily Implementation. Need to fully figure out physics calculations and actually implement them to develop a proper test. (currently main.c ctrl+f “UNIT TEST 7”

**8. Buttons Adjust LED0 PWM**

Test: Make sure that button inputs adjust LED0 PWM freq or duty cycle value.

Status: Implemented. (currently main.c ctrl+f “UNIT TEST 8”)

**9. Confirm OS Timers Started Properly**

Test: Make sure OS Timers are started properly at their individual time of start. I differentiate this from an OS function test, because it would frankly be a little risky not to test that they were started since each timer is critical.

Status: Implemented. (main.c ctrl+f “UNIT TEST 9” and there will be a few results)

**10. Capsense Task Update if Value Changed**

Test: Make sure the MeasureCapsenseTask updates the shared data if, and only if, the measurement is different from the previous one. This is important because this will be a frequent periodic task, which most of the time will not change (sample rate currently ~10Hz, far beyond human max input rate).

Status: Implemented. (main.c ctrl+f “UNIT TEST 10”)

FUNCTIONAL TESTS

**1. LED1 Handler “Rod Fallen” Flag**

Test: Set a breakpoint in the LED1OutTask under the “FUNCTIONAL TEST 1” comment Play the game for a little bit and after a bit, do nothing and let the rod fall. Ensure code goes to breakpoint.

Status: Implemented (main.c ctrl+f “FUNCTIONAL TEST 1”)

**2. LED1 Handler “X-Max” Violation Flag**

Test: Set a breakpoint in the LED1OUTTASK under the “FUNCTIONAL TEST 2” comment. Play the game for a little bit all the way to the edge of the screen, until you push the cart outside the bounds of the screen. Ensure code goes to breakpoint.

Status: Implemented (main.c ctrl+f “FUNCTIONAL TEST 2”)

**3. LED0 Operation**

Test: Increase and Decrease the force gain with btn0 and 1. Ensure you can see a visible change in the LED0 brightness.

Status: Pass

**4. Force Gain Operation**

Test: Play the game for a bit without adjusting the force gain, then adjust it up with BTN0 until it is at its max (LED0 not getting any brighter), ensure that force adjustments on the Capsense slider have a much more significant effect than before. Next, adjust it down all the way with BTN1, until the LED0 no longer gets dimmer. Ensure that force adjustments on the capsense slider have a much less significant effect than before.

Status: Not Implemented

**4. LCD Operation**

Test: Play the game for a bit. Ensure it doesn’t look too glitchy, there are no weird bugs with the display.

Status: Not Implemented

**5. Reliable Capsense Measurements**

Test: Set breakpoint under “FUNCTIONAL TEST 5” in main.c. Keep re-F8’ing code while holding your finger steady on various parts of the capsense pad. When breakpoint triggers (should be instant from human reference) ensure that “currPendulumForce” value matches the logically correct macro (HARD\_LEFT, SOFT\_RIGHT, etc)

Status: Pass (main.c, ctrl+f “FUNCTIONAL TEST 5”)

**6. Reliable Physics**

Test: Play the game for a bit. Ensure that the game physics are acceptably realistic.

Status: Not Implemented