Murfin ECEN3753 Project Unit Testing Plan

See Task Flow Diagram

UNIT TEST UPDATES FROM LAST WEEK

*Overall, I had a flawed understanding of what unit tests vs functionality tests vs OS operation tests. Because of this, I am only keeping two of the twelve tests, and coming up with eight more new ones. Directly below you’ll see my explanation for deleting some tests, and keeping others. Keep scrolling and there’s a new list.*

**Below: deleted because not unit tests: functional tests. Will implement when implement LCD graphics functionality (wk3)**

**1. LCD Graphics Display Functionality**

Test that graphics can be displayed properly on LCD screen. Implement a pendulum rod, and shift the base at a periodic rate such that we can visually see it traverse the entire needed range of motion.

**2. Reliable Capsense Measurements**

Since some implementations of Capsense measurements can be “glitchy” (you know what I mean), test to find a specific implementation that allows reliable control of at least 4 positions, over a period of 1 minute.

**Below: deleted because not really unit tests, they’re basic OS functionality tests = unnecessary**

**3. Periodic Execution of “Service Physics”**

Ensure task is entered every period “Tphy”.

**4. Periodic Execution of “LCD Graphics Display”**

Ensure task is entered every period “Tlcd”

**5. Periodic Execution of “Measure Capsense”**

Ensure task is entered every period “Tcap”

**6. GPIO ISR Triggers “LED0 & Button Handler” via Semaphore**

Ensure task is entered following GPIO ISR

**Below: Kept, now Unit Test # 5**

**7. “LED0 & Button Handler” adjusts gain value in “Pendulum Parameters” struct**

Ensure that when executed, task can successfully acquire lock of the data struct and edit its gain value.

**Below: Deleted because it is functionality test**

**8. “LED0 & Button Handler” LED0 PWM Adjustable Operation**

Ensure that task can adjust and run a dedicated PWM timer operation with LED0 – specifically, it can do that while the RTOS is running and the 3 periodic tasks (as described in 3, 4, and 5) are being entered periodically by the RTOS, and the RTOS tick nor any delays, etc. are affecting the PWM.

**Below: Test Kept, now Unit Test # 7**

**9. “Service Physics” Properly Calculating**

Analyze individual execution instances of the tasks’ calculations, ensuring that the rod position values in the data struct are adjusted properly based on physics formulas and initial values, and compare it to a pre-calculated expected value.

**Below: deleted because functionality test**

**10. “Service Physics”, “Measure Capsense”, and “LCD Display Graphics” Appears Acceptable**

Test the basic operation of the game. The capsense slider should visually adjust position of the rod base.

**11. “LED1 Handler” Rod Fallen Flag**

Test the basic operation of the game, but let the pendulum simply fall. Ensure that “LED1 Handler” task is awoken and the LED1 is de-illuminated, and the game freezes.

**12. “LED1 Handler” Xmax Violation**

Test the basic operation of the game, but push the base to one side keeping the top from falling. Push so far as to push the base outside of boundaries. Ensure that the LED1 Handler task is awoken. Ensure LED blinks on and off at 1Hz and the game does nothing else. Can use OS timer for blink implementation (probably easier than any sort of dedicated timer and the OS will be doing nothing else anyway).

NEW STATUS OF 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/line 580, or 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/line 594, or 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, but with currently unknown upper boundary condition. (currently main.c/line 824, or 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/line 736, or 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/line 820 or 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/line 763, or 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/line 575, or 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: Temporary Implementation. Need to implement dedicated LED0 PWM. (currently main.c/line 843, or 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. (on multiple lines, 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. (currently main.c/line 750, or ctrl+f “UNIT TEST 10”)