# Week 3:

**Describe at least 2 desirable unit tests and what "cuts" would be needed (see lecture topic 2.5), that would cover main paths and/or boundary condition handling, explaining how you could unit test utilizing these cuts.**

## 1. Unit Test for Drone's Energy Consumption by Disruptor Activation

Desired Unit Test (UT): Verify that the energy consumption when activating the quantum disruptor deducts the correct amount of energy from the drone's energy store.

- **Cut Identified:** The cut isolates the energy deduction logic upon disruptor activation from the actual user input and physical energy store update mechanism. This cut allows testing the logic that calculates and applies the energy deduction without external dependencies.
- **Using the Cut:** The unit test would mock the disruptor activation process, providing a predefined energy level in the drone's energy store and simulating the activation. The test checks if the energy store's value decreases by the expected amount upon activation.
- **Isolation:** This UT isolates the disruptor's energy consumption calculation from the physical button press for activation and the real-time energy recharge mechanism. It specifically targets the logic that handles energy deduction to ensure accuracy and reliability in energy management.

## 2. Unit Test for Wall Collision Detection

Desired Unit Test (UT): Ensure the collision detection system accurately identifies when the drone collides with a wall within the maze.

- **Cut Identified:** The cut for this test isolates the collision detection logic from the actual movement and position update system of the drone. By focusing on collision detection, we avoid the complexities of real-time drone movement and environmental interactions.
- **Using the Cut:** This unit test would simulate the drone being at various positions relative to a wall (including edge cases like barely touching or just missing the wall) and verify that the collision detection logic correctly identifies whether a collision has occurred.
- **Isolation:** The UT isolates the wall collision detection logic from the rest of the drone's navigation and movement control system. It aims to ensure that the

method responsible for determining collisions is accurate, regardless of how the drone's position is updated in the broader system.

## Where the project stands:

This week I continued project implementation, identified more risks for my risk register, and tested some of the 8 functional tests. Last week, I had some trouble understanding the Physics task implementation, specifically the logic to convert from velocity to positioning. I found out this is crucial for the MVP because the gyro data only recognizes rotation of the board, which when tied to the ball will not give me desired functionality. I have almost completed implementing the Physics task this week which I estimated to take the most amount of time. At first I struggled with the logic of getting the Physics logic but now after I have understood the requirements I feel I can get it to work properly. The implementation is taking longer than anticipated because my programming skills are not the best so I have to constantly stop and debug. This week I also got some ITC implementation going as I wanted to get the logic down before adding that. I wanted to compartmentalize the work a bit more to make debugging simpler. I estimated that the project will take me about **57.5** hours in total to complete. Each week I estimate that I have **11.5** hours of work to stay on track. So far I have completed **39%** of my estimated work in **100%** of the budgeted total project time. For the work that has been completed, I took **1 x** (**22.5**/**57.5**) as much time as I estimated for the total and **1 x** (**11**/**11.5**) as much time as I estimated weekly. Next week I would like to get started on the Generate Map task and Disruptor task as I predict these will be the most important next step. I also would like to get collisions working as I have the ball movement working. I just need to make sure my physics logic is on par with the MVP. Then I can implement the full LCD task and LED task. I got a basic LCD task up and running as I just to see what is happening on the display to help test my Physics task. I was able to get a ball on the screen and get it to roll around. Since my physics task is basically complete, minus collisions, I need to get the maze task working next week. I think I will start with a set maze generating algorithm to meet the MVP then I will try to get it to be randomly generated.

## Functional Tests:

### 1. Gyroscope Angle Detection and Drone Movement

**Setup:** STM32F429i-DISC1 board flat on a stable surface with the labyrinth game initialized and displayed on the LCD.

**Triggering Stimulus:** Tilt the board gradually in a specific direction (e.g., forward).

**Expected Results:** The system should interpret gyroscope data to accurately calculate the tilt angle and direction. The drone on the LCD should start moving in the corresponding direction, with the speed of movement increasing as the tilt angle increases.

**Status:** Passing

## 2. Quantum Disruptor Activation and Energy Consumption

**Setup:** Game in progress with the drone approaching a wall, sufficient energy in the energy store.

**Triggering Stimulus:** Press the user button to activate the quantum disruptor.

**Expected Results:** The disruptor activates, allowing the drone to pass through the wall for a temporary period. The energy store decreases by the expected amount defined for a disruptor activation. The activation duration matches the configured maximum or the button press duration, whichever is shorter.

**Status:** Not Tested

## 3. Energy Recharge Rate Verification

**Setup:** Drone stationary in the maze with depleted energy stored below the minimum activation threshold.

**Triggering Stimulus:** Wait without any input or disruptor activation.

**Expected Results:** The energy store recharges at the rate specified in the configuration. The red LED's flashing rate and pattern change according to the recharge status, slowing down as the energy level approaches the minimum activation threshold.

**Status:** Not Tested

## 4. Maze Generation and Obstacle Placement

**Setup:** Game initialization sequence.

**Triggering Stimulus:** Start a new game.

**Expected Results:** A maze is randomly generated, adhering to the obstacle probability configurations. Walls and holes are placed according to specified probabilities, ensuring no immediate traps are near the starting waypoint. The generated maze is displayed on the LCD.

**Status:** Failing

## 5. Collision Detection Without Disruptor

**Setup:** Drone navigating towards a wall without the disruptor activated.
**Triggering Stimulus:** Control the drone to move directly into a wall.

**Expected Results:** The game detects the collision accurately, stopping the drone's movement at the wall, without allowing it to pass through, simulating a realistic collision response.

**Status:** Not Tested

## 6. Waypoint Achievement Validation

**Setup:** Drone near a waypoint in the maze.

**Triggering Stimulus:** Navigate the drone so that its center overlaps with the waypoint's center.

**Expected Results:** The system recognizes the drone has reached the waypoint, updating the game state to reflect this progress. If all required waypoints are reached within the time limit, the game indicates a win scenario.

**Status:** Not Tested

## 7. LED Status Indicators for Energy and Game Status

**Setup:** Game in progress with varying energy levels.

**Triggering Stimulus:** Observe the LED behaviors as the game progresses through different energy states.

**Expected Results:** The green LED adjusts its brightness relative to the energy store's fullness. The red LED flashes at a rate indicative of the time left to recharge to the minimum activation energy, turning off once the threshold is met.

**Status:** Not Tested

## 8. Game Completion: Win/Loss Conditions and Scoring

**Setup:** Drone approaching the final waypoint with varied game outcomes (e.g., within the time limit for a win or with the disruptor inactive above a trap for a loss).

**Triggering Stimulus:** Reach the final waypoint or fall into a trap.

**Expected Results:** The game accurately determines and displays a win or loss based on the condition met. For a win, the score calculated as waypoints/second is shown; for a loss, no score is shown. If configuration values were altered, this is also indicated alongside the game completion status.

**Status:** Not Tested