Functionality Tests

The tests in this document are grouped by the level in accordance with the software architecture designed for the Connect-4 robot. The tests for Low-Level (peripherals of STM32H) unit and component testing were omitted from this plan due to time limitations. However, a basic draft was created for future reference.

Test Cases	Test Conditions	Done
Level 2 – Cortex-M4		
Initialization	**1.** Verify that the initialization sets up the system to be ready for operation, including configuring the peripherals, initializing the modules.	1.
Task Manager	1. Test the initialization and configuration of the module. 1.1. Test if the module can read the tasks from memory and trigger the state transition. 2. Verify that the task manager can detect, report, and recover from errors.	1. 1.1 2.
Motor Controller	**1.** Test the initialization and configuration of the module and both motors. 1.1. Verify that the module is set up to receive signals from the encoders, and home/end switches. 1.2. Test if the module moves the motors in both X and Z directions. 1.3. Test if the PWM signal controls the motors effectively 1.4. Test if the module can read the position from the encoders. 1.5. Test that the home/end-switches send the correct interrupt and stop the motor. **2.** Verify that the Motor Controller module can detect, report, and recover from errors.	1. 1.1 1.2 1.3 1.4 1.5 2.
Token colour separator controller	**1.** Test the initialization and configuration of the module. 1.1. Test the module is correctly set up to control the RGB and proximity sensor and the flipper. 1.2. Test if the module detects the colour of tokens (red and yellow). 1.3. Test if the module detects the proximity of the token. 1.4. Test the activation of the flipper. **2.** Verify that the Token Colour Separator Controller module can detect, report, and recover from errors.	1. 1.1 1.2 1.3 1.4 2.
Token picker controller	**1.** Test the initialization and configuration of the controller. 1.1. Test if the controller can move the end-effector servo, read the vacuum sensor, and control the vacuum valve. 1.2. Test if the vacuum pump generates enough pressure to pick up a token and transport it. 1.3. Test if the positions of all different pick-up/drop-off points are correct. **2.** Verify that the module can detect, report, and recover from errors.	1. 1.1 1.2 1.3 2. 3.
User Detector	**1.** Test the initialization and configuration of the module.	1.

	1.1. Verify that the module can read data from the light-gate circuit. **2.** Verify that the module can detect, report, and recover from errors.	1.1 2.
Board Opener	**1.** Test the initialization and configuration of the module. 1.1. Test that the servo motors can open the board column by column. 1.2. Test that the Task Manager can send commands for opening and closing the board. **2.** Verify that the Board Opener module can detect, report, and recover from any errors.	1. 1.1 1.2 2.
Level 2 – Cortex-M7		
Initialization	**1.** Verify that the initialization sets up the system to be ready for operation, including configuring the peripherals, initializing the modules.	1.
Game controller	**1.** Verify that the Game Controller module is correctly set up to manage the overall game logic and flow. 1.1. Test the module's ability to keep and update the game state, including the board state and player turns. 1.2. Test the module's ability to detect win, loss, or draw conditions. 1.3. Test the state transitions of the controller. **2.** Verify that the Game Controller can detect, report, and recover from any errors.	1. 1.1 1.2 1.3 2.
CM4 Task Generator	**1.** Test the initialization and configuration of the module. 1.1. Test the CM4 Task Generator's ability to create tasks based on the game state and requests from other modules. 1.2. Verify that the CM4 Task Generator module receives correct game state updates and next-move decisions from the Game Controller module. **2.** Verify that the CM4 Task Generator module can detect, report, and recover from any errors	1. 1.1 1.2 2.
Game end block	**1.** Test the initialization and configuration of the module. 1.1. Test if the module is able to and handle a win/lose/draw condition for either the human player or the robot player. **2.** Verify that the module can detect, report, and recover from any errors.	1. 1.1 2.
UART controller	**1.** Test the initialization and configuration of the UART Controller module. 1.1. Test the UART's ability to transmit/receive data to and from external blocks. 1.2. Test the debug environment created through UART * **2.** Verify that the UART controller can detect, report, and recover from any errors.	1. 1.1 1.2 2.

*If created and discussed that it is reasonable to do so.

System Timing

It is important for future improvements of the system and separate modules to know the timing of specific operations and action. It would be wise for them to be divided into the same structure as the one set up in the above tests and the architecture. This would be more useful for the Cortex-M4 at the moment, and therefore I have compiled a brief list of timings that would be useful to know.

- Record how long picking the tokens takes.
- Record how long releasing the tokens takes.
- Record how fast a token dropping is recognized.
- Test reaction to multiple tokens insertion in the same column (cheat move).
- Test reaction to multiple tokens insertion in multiple columns (cheat move).
- Test reaction to when a token is inserted in the wrong player state (cheat move).
- Test the time needed to run the length of the axis of the X and Z motors **pre-PID** controller implementation (current system).
- Test the time needed to run the length of the axis of the X and Z motors **post-PID** controller implementation.
- Record how long it takes to clear the full board.

Happy-Path Test

This is a full test of the system that involves the whole gameplay loop as one would normally (without cheat moves) expect it to run.

This includes verifying that the robot moves the tokens accurately, the sensor detect the correct positions when tokens are inserted and the correct colour when cleaning the board, the game logic (FSM) functions properly and transitions correctly based on the current state, and the communication between the two cores.

Acceptance Test

Made against initial requirements.