Component Test Plan

Unit testing is essential for Testing the correct functioning of each individual component in the software architecture of the Connect-4 robot player, including the initialization procedure, low-level code about different peripherals, and communication between the two cores.

Unit Tests

Builds confidence in the separate blocks/components.

Test Case	Test Conditions
TIM	1. Test basic timer initialization and configuration. 2. Test that the timer generates interrupts at the expected intervals. 3. Test the use of multiple timers simultaneously.
NVIC	4. Test NVIC initialization and configuration. 5. Test that NVIC correctly prioritizes and handles multiple simultaneous interrupts. 6. Test the enabling and disabling of individual interrupts. 7. Check the functionality of interrupt masking and unmasking. 8. Test that the NVIC can properly clear pending interrupts. ***()HAL_NVIC_ClearPendingIRQ()***
EXTI	9. Test EXTI initialization and configuration for specific GPIO pins. ***HAL_EXTI_GetConfigLine()*** 10. Test whether the EXTI triggers the desired action upon detecting an external event. ***HAL_EXTI_RegisterCallback() ???*** 11. Test that the EXTI can clear the interrupt flag correctly Pending Register (EXTI_CnPRIx) ***HAL_EXTI_ClearConfigLine()*** 12. Test the proper handling of EXTI interrupts by the NVIC.
UART	13. Test UART initialization and configuration with various baud rates, data bits, stop bits, and parity settings. 14. Test that the UART can transmit and receive data correctly. 15. Test that the UART can operate at different baud rates. 16. Test that the UART can properly handle errors. 17. Test that the UART can properly handle interrupts related to data transmission and reception.
I2C	18. Test whether the I2C address of the Connect-4 is correctly configured. 19. Test that the I2C can properly transmit and receive data. 20. Test that the I2C can properly handle errors. 21. Test that the I2C can operate at different speeds. 22. Test that the I2C can properly handle interrupts related to data transmission and reception.
LED	23. Test LED initialization and configuration. 24. Test that the LED individually can turn on and off correctly. 25. Test that the LED can blink at different frequencies and duty cycles. 26. Check the proper integration of LED control with other peripherals, such as timers.
GPIO	27. Test whether the GPIO pins on the robot player are properly configured and controlled by the software. 28. Test whether the GPIO pins have the correct input/output state. 29. Test GPIO interrupt generation and handling (EXTI).
ADC	30. Test whether the ADC is properly configured and enabled.

	31. Test whether the ADC is able to accurately measure the sensor input voltage and convert it to a digital signal.
ETH	32. Establish ETH initialization and configuration with different network settings (IP, MAC, etc.). 33. Test that ETH can send and receive data packets correctly. 34. Test ETH functionality with various frame sizes and protocols. 35. Check ETH error handling and recovery.
HSEM	36. Test HSEM initialization and configuration. 37. Test that HSEM can manage access to shared resources between the two cores. 38. Test HSEM functionality with different resources and priority settings.

Functionality Tests ?

Vacuum Pump	1. Test vacuum pump initialization and control. 2. Test that the vacuum pump can generate sufficient vacuum for token handling.
Vacuum Sensor	3. Test vacuum sensor initialization and readout. 4. Test that the vacuum sensor can accurately detect the presence of a token.
Vacuum Valve	5. Test vacuum valve initialization and control. 6. Test that the vacuum valve can effectively control the vacuum flow.
RGB Sensor	7. Test RGB sensor initialization and configuration. 8. Test that the RGB sensor can accurately detect token colours.
IR Sensor	9. Test IR sensor initialization and readout. 10. Test that the IR sensor can accurately detect token drop events. 11. Test the integration of the IR sensor with the game logic.
Proximity Sensor	12. Test proximity sensor initialization and readout. 13. Test that the proximity sensor can accurately detect the position of objects. 14. Test the integration of the proximity sensor with the motor control and end-switches.
End-Switches	15. Test end-switch initialization and readout. 16. Test that end-switches can accurately detect the end of the motion range for motors. 17. Test the integration of end-switches with motor control and proximity sensors.
Home Switches	18. Test home switch initialization and readout. 19. Test that home switches can accurately detect the home position of motors. 20. Test the integration of home switches with motor control and encoders.
Encoder Readout	21. Test encoder initialization and readout. 22. Test that encoders can accurately measure motor position and speed. 23. Test the integration of encoders with motor control and PID calculations.
PID calculations	24. Test PID algorithm implementation and tuning. 25. Test that PID calculations can effectively control motor position and speed. 26. Test the integration of PID calculations with encoder readout and motor control.
PWM Signal	27. Test PWM signal generation and configuration. 28. Test that the PWM signal can effectively control motor speed and servo position. 29. Test the integration of the PWM signal with motor control and servo control.

Motor control – X	30. Test X-axis motor initialization and control. 31. Test that the X-axis motor can accurately move to desired positions. 32. Test the integration of the X-axis motor control with encoder readout, end-switches, and PID calculations.
Motor control – Z	33. Test Z-axis motor initialization and control. 34. Test that the Z-axis motor can accurately move to desired positions. 35. Test the integration of the Z-axis motor control with encoder readout, end-switches, and PID calculations.
Servo – End-effector	36. Test end-effector servo initialization and control. 37. Test that the end-effector servo can accurately rotate the end-effector. 38. Test the integration of the end-effector servo with the token picker master.
Servo – Board opener	39. Test board opener servo initialization and control. 40. Test that the board opener servo can accurately open and close the game board. 41. Test the integration of the board-opener servo with the master block
Token picker master	42. Test token picker master initialization and control. 43. Test that the token picker master can accurately manage the token picking process. 44. Test the integration of the token picker master with the vacuum pump, vacuum sensor, vacuum valve, and end-effector servo.
Token colour separator master	45. Test token colour separator master initialization and control. 46. Test that the token colour separator master can accurately sort tokens by colour. 47. Test the integration of the token colour separator master with the RGB sensor and flipper/solenoid control.
Motor master	48. Test motor master initialization and control. 49. Test that the motor master can accurately manage the X and Z-axis motors. 50. Test the integration of the motor master with motor control, encoder readout, end-switches, home switches, and PID calculations.
Flipper/Solenoid control	51. Test flipper/solenoid initialization and control. 52. Test that the flipper/solenoid can accurately separate tokens based on color. 53. Test the integration of the flipper/solenoid control with the token color separator master and RGB sensor.
Emergency Stop	54. Test emergency stop initialization and functionality. 55. Test that the emergency stop can halt all system operations safely and immediately. 56. Test the integration of the emergency stop with motor control, servo control, and error handling.
Dual-core communication	57. Test dual-core communication initialization and data exchange. 58. Test that Cortex-M7 and Cortex-M4 cores can communicate effectively and share tasks. 59. Test the integration of dual-core communication with the HSEM and other system modules.
Error handling	60. Test error handling implementation and functionality. 61. Test that the error handling system can detect, report, and recover from errors in various modules. 62. Test the integration of error handling with motor control, sensor readout, dual-core communication, and emergency stop.

Test Case	Test Conditions
Level 1	
CM4	**CM4 Initialization and Configuration:** a. Test the initialization and configuration of Cortex-M4. **CM4 Peripheral Integration:** a. Test if the core can initialise all peripherals and communicate/control them. **CM4 Subsystem Integration:** a. Test if the core can communicate with the subsystems and control them. **CM4 Dual-core Communication:** a. Test is CM4 can receive tasks from CM7 and if it responds accordingly while providing feedback on its status. **CM4 Real-time Performance:** a. Test if the Cortex-M4 core can manage multiple tasks. **CM4 Error Handling:** a. Test that the core can detect, report, and recover from any errors related to task execution, peripheral communication, or subsystem interaction.
CM7	**CM7 Initialization and Configuration:** a. Test the initialization and configuration of Cortex-M7. **CM7 Peripheral Integration:** a. Test if the core can initialise all peripherals and communicate/control them. **CM7 Subsystem Integration:** a. Test if the core can communicate with the subsystems and control them. **CM7 Dual-core Communication:** a. Test is CM7 can send tasks to CM4 and check if it receives feedback about the status of the tasks. **CM7 High-level Performance:** a. Verify that the Cortex-M7 core can manage multiple tasks. **CM7 Error Handling:** a. Test that the core can detect, report, and recover from any errors related to task execution, peripheral communication, or subsystem interaction.
Level 2 – M4	
Task Manager	**Task Manager Initialization:** a. Test the initialization and configuration of the Task Manager module.

	Task Delegation Functionality: a. Test the Task Manager module to give tasks to the appropriate subsystems, such as the Motor Controller, Token Picker Controller, Token Color Separator Controller, User Detector, etc. b. Verify that the Task Manager module can maintain a queue of tasks and manage their execution in the correct order. **Task Synchronization and Prioritization:** a. Test the ability of the Task Manager module to prioritize tasks based on their urgency or importance. b. Verify that the Task Manager module can synchronize the execution of tasks to avoid conflicts and ensure a smooth game flow. **Task Manager - Subsystem Interface:** a. Test the communication between the Task Manager and various subsystems. b. Verify that the Task Manager can effectively delegate tasks to the subsystems and receive feedback on the status of the tasks. **Task Manager Timing:** a. Test the response time of the Task Manager module during task delegation and synchronization operations. b. Verify that the Task Manager module can perform its functions within the expected time frame to ensure a smooth game flow. **Task Manager - Error Handling:** a. Verify that the Task Manager module can detect, report, and recover from any errors or malfunctions related to task delegation, synchronization, or subsystem communication. **Task Manager - Dual-core Communication:** a. Test the communication between the Task Manager module and the Cortex-M7 core. b. Verify that the Task Manager can effectively delegate tasks from the Cortex-M7 core to the appropriate subsystems within the Cortex-M4 core.
Motor Controller	**Motor Controller Initialization:** a. Test the initialization and configuration of the Motor Controller module for both X and Z motors and for their respective PID controllers. b. Verify that the Motor Controller module is correctly set up to control the motors, encoders, and home/end switches. **Motor Movement Functionality:** a. Test the ability of the Motor Controller module to accurately move the motors in both X and Z directions. b. Verify that the Motor Controller module can control the motors to achieve the desired position and speed. **Encoder Readout Functionality:**

	a. Test the ability of the Motor Controller module to accurately read the position data from the encoders. b. Verify that the Motor Controller module can use encoder data to control the motors' position and speed. **Home/End Switch Functionality:** a. Test the ability of the Motor Controller module to detect the home and end positions using the home/end switches. b. Verify that the Motor Controller module can prevent the motors from going beyond their limits by using the home/end switches. **PID Calculation Functionality:** a. Test the PID calculations in the Motor Controller module for maintaining the desired position and speed of the motors. b. Verify that the Motor Controller module can effectively use the PID calculations to control the motors' performance. **Motor Controller Timing:** a. Test the response time of the Motor Controller module during motor movement operations. b. Verify that the Motor Controller module can perform motor movements within the expected time frame to ensure a smooth game flow. **Motor Controller - Task Manager Interface:** a. Test the communication between the Task Manager and Motor Controller modules. b. Verify that the Task Manager can effectively transfer motor movement tasks to the Motor Controller module. **Motor Controller - Error Handling:** b. Verify that the Motor Controller module can detect, report, and recover from any errors or malfunctions related to the motors, encoders, or home/end switches.
Token colour separator controller	**Token Color Separator Controller Initialization:** a. Test the initialization and configuration of the Token Color Separator Controller module. b. Verify that the Token Color Separator Controller module is correctly set up to control the RGB sensor and the flipper/solenoid. **Token Color Detection Functionality:** a. Test the ability of the Token Color Separator Controller module to accurately detect the color of tokens (red and yellow). b. Verify that the Token Color Separator Controller module can read data from the RGB sensor and correctly identify the color of the tokens. **Token Separation Functionality:** a. Test the ability of the Token Color Separator Controller module to separate tokens based on their color.

	b. Verify that the Token Color Separator Controller module can control the flipper/solenoid to successfully sort and distribute the tokens to the correct sides. **Token Color Separator Controller Timing:** a. Test the response time of the Token Color Separator Controller module during token color detection and separation operations. b. Verify that the Token Color Separator Controller module can perform token color detection and separation actions within the expected time frame to ensure a smooth game flow. **Token Color Separator Controller - Task Manager Interface:** a. Test the communication between the Task Manager and Token Color Separator Controller modules. b. Verify that the Task Manager can effectively delegate token color detection and separation tasks to the Token Color Separator Controller module. **Token Color Separator Controller - Error Handling:** b. Verify that the Token Color Separator Controller module can detect, report, and recover from any errors or malfunctions related to the RGB sensor or flipper/solenoid.
Token picker controller	**Token Picker Controller Initialization:** a. Test the initialization and configuration of the Token Picker Controller module. b. Verify that the Token Picker Controller module is correctly set up to control the vacuum pump, vacuum sensor, and vacuum valve. **Token Picker Functionality:** a. Test the ability of the Token Picker Controller module to accurately pick up tokens. b. Verify that the Token Picker Controller module can control the vacuum pump, vacuum sensor, and vacuum valve to successfully pick up and hold tokens. **Token Release Functionality:** a. Test the ability of the Token Picker Controller module to release tokens accurately. b. Verify that the Token Picker Controller module can control the vacuum pump and vacuum valve to release tokens at the desired position. **Token Picker Controller Timing:** a. Test the response time of the Token Picker Controller module during token picking and releasing operations. b. Verify that the Token Picker Controller module can perform token picking and releasing actions within the expected time frame to ensure a smooth game flow. **Token Picker Controller - Task Manager Interface:**

	a. Test the communication between the Task Manager and Token Picker Controller modules. b. Verify that the Task Manager can effectively delegate token picking and releasing tasks to the Token Picker Controller module. **Token Picker Controller - Error Handling:** b. Verify that the Token Picker Controller module can detect, report, and recover from any errors or malfunctions related to the vacuum pump, vacuum sensor, or vacuum valve.
User Detector	**User Detector Initialization:** a. Test the initialization and configuration of the User Detector module. b. Verify that the User Detector module is correctly set up to read data from the photodiode circuit. **User Detection Functionality:** a. Test the ability of the User Detector module to accurately detect the presence of a user. b. Verify that the User Detector module can differentiate between user presence and absence based on the readings from the photodiode circuit. **User Detector Timing:** a. Test the response time of the User Detector module upon detecting a user. b. Verify that the User Detector module can send signals to other modules in a timely manner once a user is detected. **User Detector - Task Manager Interface:** a. Test the communication between the Task Manager and User Detector modules. b. Verify that the Task Manager can receive user detection signals from the User Detector module and act accordingly. **User Detector - Board Opener Interface:** b. Verify that the User Detector module can send a signal to the Board Opener module to initiate the board opening process when a user is detected. **User Detector - Error Handling:** b. Verify that the User Detector module can detect, report, and recover from any errors or malfunctions related to the proximity sensor or user detection logic.
Board Opener	**Board Opener - Task Manager Interface:** b. Verify that the Task Manager can effectively delegate board opening and closing tasks to the Board Opener module. **Board Opener - User Detector Interface:** b. Verify that the Board Opener module can receive the signal from the User Detector module when a user is detected, initiating the board opening process.

	Board Opener - Error Handling: b. Verify that the Board Opener module can detect, report, and recover from any errors or malfunctions during the opening and closing process.
Level 2 – M7	
Initialization	**Initialization Module Functionality:** a. Test the ability of the Initialization module to correctly initialize and configure the system hardware and software components. b. Verify that the Initialization module sets up the system to be ready for operation, including configuring the peripherals, initializing the modules, and setting the system clock. **Initialization Module - Game Controller Interface:** a. Test the communication between the Initialization module and the Game Controller module. b. Verify that the Initialization module correctly sets up the Game Controller module to manage the overall game logic and flow. **Initialization Module - CM4 Task Generator Interface:** a. Test the communication between the Initialization module and the CM4 Task Generator module. b. Verify that the Initialization module correctly sets up the CM4 Task Generator module to generate and delegate tasks to the M4 core. **Initialization Module - Game End Block Interface:** a. Test the communication between the Initialization module and the Game End Block module. b. Verify that the Initialization module correctly sets up the Game End Block module to detect and handle game-ending conditions. **Initialization Module - UART Controller Interface:** a. Test the communication between the Initialization module and the UART Controller module. b. Verify that the Initialization module correctly sets up the UART Controller module to handle UART communication with external devices. **Initialization Module - M4 Core Interface:** a. Test the communication between the Initialization module and the M4 core. b. Verify that the Initialization module correctly initializes the M4 core, including setting up its peripherals and modules. **Initialization Module - Error Handling:** a. Test the integration of the Initialization module with the Error Handling module. b. Verify that the Initialization module can detect, report, and recover from any errors or malfunctions during the initialization process.
Game controller	**Game Controller Initialization:** a. Test the initialization and configuration of the Game Controller module.

	b. Verify that the Game Controller module is correctly set up to manage the overall game logic and flow. **Game State Management:** a. Test the Game Controller's ability to maintain and update the game state, including the board state and player turns. b. Verify that the Game Controller module accurately represents the game state at all times. **Next-Move Decision-Making:** a. Test the Game Controller's ability to decide the next move for the robot player based on the current game state and difficulty setting. b. Verify that the Game Controller module can generate appropriate next moves according to the difficulty setting. **Win/Loss/Draw Detection:** a. Test the Game Controller's ability to detect win, loss, or draw conditions. b. Verify that the Game Controller module can accurately identify the game outcome and perform the necessary actions, such as displaying results and resetting the board. **Game Controller - Initialization Module Interface:** a. Test the communication between the Game Controller and the Initialization module. b. Verify that the Game Controller module receives correct initial settings and configuration from the Initialization module. **Game Controller - Game End Block Interface:** a. Test the communication between the Game Controller and the Game End Block module. b. Verify that the Game Controller module can trigger the Game End Block module when win, loss, or draw conditions are detected. **Game Controller - Error Handling:** a. Test the integration of the Game Controller module with the Error Handling module. b. Verify that the Game Controller module can detect, report, and recover from any errors or malfunctions related to game management.
CM4 Task Generator	
Game end block	
UART controller	

Test Data	Test Set-Up

Happy-Path Test

The intended way of using the system.

Acceptance Test