Test Plan – Master Plan

**4 in a row robot**

*Version: 01*

*Date: 9-4-2018*

ALTEN

Jasper Jansen

Internal

Draft

Version History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **State** | **Author** | **Remark** |
| ***01*** | 9-4-2018 | Draft | Jasper Jansen | First draft |
|  |  |  |  |  |

Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Explanation** |
| For the acronyms and abbreviations of the concerned project, see Internal Project Plan (IPP). | |

Referenced documents

*< This table shall list the number, title, revision, and date of all leading documents referenced in this document. >*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Reference | Title | Date | Author |
| 01 |  |  |  |  |

Contents

1. Introduction 5

1.1 Test Objectives 5

1.2 Extent of tests 5

1.3 Scope 5

2. Target Test Items 6

3. Outline of planned tests 7

3.1 Test inclusions 7

3.2 Test exclusions 7

4. Test approach 8

4.1 Testing techniques 8

5. Pass/Fail criteria 9

5.1 Entry Criteria 9

5.1.1 Unit Testing 9

5.1.2 Integration Testing 9

5.1.3 System Testing 9

5.1.4 Acceptance Testing 9

5.2 Exit Criteria 9

5.2.1 Unit Testing 9

5.2.2 Integration Testing 9

5.2.3 System Testing 10

5.2.4 Acceptance Testing 10

5.3 Suspension Criteria 10

5.4 Resumption Requirements 10

6. Test Deliverables 11

7. Environmental needs 12

7.1 Hardware Resources 12

7.2 Software Resources 12

8. Responsibilities 13

9. Schedule 14

10. Risks and contingencies 15

Appendix A: Test Traceability Matrix (TTM) 16

Appendix B: <Heading> 17

Notes 18

Unit Test Plan 18

Integration Test Plan 18

System Test Plan (use cases?) 18

Acceptance Test Plan 18

# Introduction

The software controlling the 4 in a row robot consists out of two main components. Firstly, the high-level part that runs on a raspberry pi to control the AI of the game. The second part is the low-level software running on a microcontroller that controls all peripherals such as motors and sensors. These subsystems communicate via UART to work together.

This test plan describes the required tests to be performed for both the low- and high-level software layer of the 4 in a row robot.

## Test Objectives

The objective of the described tests is to validate requirements, system architecture and object design. The final product of the tests is a demo-ready software together with a set of test scripts that can be reused for Functional Test execution.

The following objectives are supported by this document:

* Define the tools to be used throughout the testing process.
* Inform all involved stakeholders with information about the items to be tested and environmental needs.
* Defining the test methods and pass/fail conditions.

## Extent of tests

The tests referenced in this document are meant to validate use-cases, requirements and system architecture. Testing is started by verifying all sub components and modules that are developed in this project. The tests to validate the system architecture are run next after integration is done as integration tests.

## Scope

This test plan describes the unit, subsystem integration, system level and acceptation tests that shall be performed on the components of the 4 in a row robot.

The unit tests are done as clear and black-box checking, to verify boundaries and conditions. Subsystem tests are used to verify the interfaces between various components of the robot system.

# Target Test Items

The following subsystems are identified as targets for testing:

Raspberry Pi

* High-level AI software

STM32 mainboard

* Low-level software
* Hardware interfaces
* Sensors

Mechanics

* Integration of mechanics with control

# Outline of planned tests

This section describes the type of tests that are included or excluded from this test plan.

## Test inclusions

The following tests are included within the 4 in a row project:

* Unit tests
* Integration tests
* System tests
* Acceptance tests

## Test exclusions

The following tests are excluded within the 4 in a row project:

* Security testing
* Stress testing
* Failure testing

# Test approach

Testing of submodules will be automated if possible, this could be done with a testing framework if needed. Before testing of a submodule is commenced, it is assumed to be considered done by the programmer.

Any bugs that are found during testing phase, shall be posted in the ALTEN Redmine of the 4 in a row project. Information to be posted shall at least contain a description of the problem together with who it found and steps to reproduce.   
Eventually this report shall contain information by whom and when it was fixed. After this step the software shall be subjected to testing again.

## Testing techniques

There will be four testing techniques used within this project:

* Unit testing
* Integration testing
* System testing
* Acceptance testing

Unit tests are to be executed on the lowest level of the software and modules. Sub-functions that are used within the project should be verified with this method, most of the automated tests can be done here.

Integration testing is done to identify problems of interfaced between the individual modules.

System testing is required for testing the entire cooperation between modules, together with hardware and a user. The system testing is mostly done with use-cases of the complete product.

Acceptance testing shall be done as last step to find problems with the compliance of requirements of the system.

It should be noted that not all possible combinations of playing moves can be tested. To overcome the necessity of doing this, limit testing shall be applied during unit and integration testing. This means that functions and actions shall be tested near their boundaries of operation.

# Pass/Fail criteria

This section defines the entry and completion criteria for their specific test phase. It provides the criteria when testing may commence and criteria that indicate testing of an item or feature is complete.

## Entry Criteria

The entry criteria are defined to make sure that the application is ready for acceptance into the testing phase. Within the 4 in a row project, the following entry criteria are specified per test type:

### Unit Testing

* Functional requirements and specification is complete.
* High level design documentation is complete.

### Integration Testing

* The project is complete, there is no missing code.
* All previously found priority bugs are fixed, retested and closed.
* Documentation is up to date to current state of the project.
* All integration test plans, scenarios and cases are available.

### System Testing

* Successful completion of integration testing.
* All previously found priority bugs are fixed, retested and closed.
* System testing environment is available.
* System testing plans, scenarios and cases are available.

### Acceptance Testing

* Successful completion of acceptance testing.
* All previously found priority bugs are fixed, retested and closed.
* All Functional Requirements are met.
* Acceptance testing environment is ready.
* Acceptance testing plan is available.

## Exit Criteria

The exit criteria are defined to ensure that the tested component is ready for the next phase. Again, these are subdivided per test type:

### Unit Testing

* The unit tests are passed successfully.
* The code is compliant with the specification.
* Documentation of the code is complete and up to date.
* No major or blocking issues are found.
* Unused or redundant code is removed.
* Code compiles without warnings/errors.

### Integration Testing

* The integration tests are passed successfully.
* All found priority bugs are fixed, retested and closed.
* Documentation of the code is complete and up to date.

### System Testing

* The system tests are passed successfully.
* No major or blocking issues are found.
* The system meets all functional requirements.
* The system supports all used hardware.
* Documentation of the code is complete and up to date.

### Acceptance Testing

* The acceptance tests are passed successfully.
* No major or blocking issues are found.
* User acceptance test closing meeting is done.

## Suspension Criteria

If any defects which severely impact testing progress are found, the test lead may choose to suspend testing. This section outlines the circumstances that would result in a partial or complete suspension of testing.

Criteria that are considered for suspension of testing are:

* Assigned resources for testing are not available when needed.
* The amount of serious errors in the software build is too high.
* The number of open incidents becomes so high that testing has no value
* System downtime

## Resumption Requirements

If testing is suspended based on any of the before mentioned items, the resumption of testing requires resolving of the found issue(s).   
When the cause for suspension of testing is a critical defect, the fix must be verified by the testing engineers before the testing process may be resumed.

# Test Deliverables

The testing phase should generate several documents (plans, reports and logs) together with output data. This data shall be made into the deliverables mentioned below and should be available to all stakeholders within this project.

* Master Test Plan (This document)
* Test strategies
* Test cases
* Test scripts
* Test data
* Incident/bug reports
* Test results/reports
* Test summary

# Environmental needs

This section describes the environmental resources to start and complete the testing phase of the four in a row robot. It is divided in hardware based resources and software based resources.

## Hardware Resources

* Test development PC/laptop
* STM32 Discovery board (STLink to connect to mainboard)
* Oscilloscope to verify output timing

## Software Resources

* Windows 7/10 Operating System
* System Workbench for STM32
* Redmine project entry

# Responsibilities

Unit testing of code will be done by the developer who is assigned to a specific module. This shall be done during development phase itself.

Integration, system and acceptance tests design and execution require at least one engineer to satisfy testing capacity within this project.

# Schedule

Unit tests 🡪 possible when hardware is available

Rpi is tested with mock-up of hardware

Integration tests 🡪 only with hardware

System testing 🡪 once integration testing is done

Table 1: Milestone dependancy

|  |  |  |
| --- | --- | --- |
| Milestone |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Risks and contingencies

This section describes potential risks to the overall testing process, any preferred options and mitigations are included.

Table 2: Risks and Mitigations

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Probability | Impact | Mitigation |
| Developer leaves ADC to start at customer. | High | Medium | Maintaining high documentation standards. |
| Delayed delivery of required hardware for integration. | Medium | High | Reschedule testing according hardware delivery schedule.  Start tests on available hardware subsystems in case not all components have delay (relax entry criteria). |
| Changes to original requirements are made. | Medium | High | Reschedule testing process. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Appendix A: Test Traceability Matrix (TTM)

*<Test and requirements tracing, documenting the links between the requirements and the defined tests. An optional overview of the defined tests and to verify related requirements.*

*To check every requirement has a test and vice versa. This could lead to identify unnecessary requirements or tests.>*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Id | Test id | Description | Requirement id | Note | |
| 01 |  |  |  | |  |

# Appendix B: <Heading>

*<Appendixes may be used to provide information published separately for convenience in document maintenance (e.g., charts, classified data). As applicable, each appendix shall be referenced in the main body of the document where the data would normally have been provided. Appendixes may be bound as separate documents for ease in handling.>*

# Notes

## Unit Test Plan

* AI’s
* Endswitches
* Homeswitches
* Encoder readout
* Motor control
* Timing interrupt
* PID calculations
* Movements
* Vacuum components
* Coin detector
* Multiple coins after another
* Multiple coins in different columns
* Servo control
* Flipper control
* RGB sensor
* Emergency stop
* Difficulty setting
* Reset button
* Subfunctions?

## Integration Test Plan

* Initialization sequence
* Home procedure
* Normal play sequence (put coin inside column)
* Game end sorting sequence
* Emergency stop and recovery procedure

## System Test Plan (use cases?)

* Starting the system
* Playing of multiple games  
  🡪 winning  
  🡪 losing  
  🡪 different difficulties
* Shutdown

## Acceptance Test Plan

* Testing against requirements