Software Requirements Document

Designing an Autonomous Robot Player for Connect-4

Version: 0.2

Date: 6-3-2023

ALTEN

Boris Ivanov

<Confidential, Restricted, Internal>

< Draft, For Review or Released>

Version History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date | State | Author | Remark |
| 0.1 |  |  |  | First draft of the document |
|  |  |  |  |  |

Acronyms and Abbreviations

|  |  |
| --- | --- |
| Term | Explanation |
| <ABC> | <First three letters of the alphabet> |
|  |  |

Referenced documents

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Reference | Title | Date | Author |
| 1 |  |  |  |  |
|  |  |  |  |  |

Contents

1. Introduction 4

1.1 Background 4

1.2 Purpose 4

2. User requirements 5

2.1 User roles **Error! Bookmark not defined.**

2.2 User stories **Error! Bookmark not defined.**

3. Functional system requirements 7

4. Non-functional system requirements **Error! Bookmark not defined.**

5. Project requirements 8

Appendix A. Traceability 9

Appendix B. Verification 10

# Introduction

## Background

*The 4-in-a-row (Connect4, Four Up) robot was developed for demos at trade fairs and open days at universities. The robot game is meant to demonstrate the knowledge of the consultants of ALTEN, and it is therefore developed with industrial components.*

*The game is simple, there is a seven-by-six rack board, with slots at each spot for two coloured tokens. A red one and a yellow one. The first player to Connect 4 tokens in any direction wins. In our case, one player is a human, the other one is a robot. It is a completely autonomous process. After a token has been placed in the idle robot, the machine can calculate its next move based on a difficulty setting. To be able to execute everything, the 4-in-a-row robot is equipped with ‘X’ and ‘Z’ plane motors, a rotating vacuum gripper, and a routine to clear the board and reset the tokens.*

## Purpose

*Implement the previously designed software architecture for the new STM32H755ZIT6U controller.   
My task will involve designing the modules to make the system reliable and functional to the best of its capacity. That will involve writing code for the needed modules, improving and adapting flowcharts/logic, and redesigning modules that do not function as expected from the software architecture. Further steps will include the further optimization of the BSP and testing on the robot itself. Moreover, research on ethernet communication with the robot will also be investigated, as a secondary goal.*

# User requirements

These are acquired from two documents. One is Pascal’s graduation report and the other one is an SRD for the Connect-4 from 24-06-2019 from Jeroen Grollenbeek, Ralph Lentink and Arjan Verboord.  
They were slightly modified to reflect current progress.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirement | Description / explanation | Priority |
|  | The user should be able to start and play a game of Connect-4 against the robotized opponent without operator intervention. |  | Must |
|  | The user shall be notified when the game ends. | The system could communicate somehow who won (sounds, visual, giving a token). A future upgrade to add a screen to show winner, top scores, etc. | Could |
|  | The robot should detect a cheating player and respond by resetting the game. | A cheating player is someone who plays out of their turn, or someone who inserts two coins or more at once in one or several columns. | Must |
|  | A Board Support Package (BSP) must be made of the operating system with which the necessary hardware components of the robot can be controlled. |  | Must |
|  | The insertion of a game token in an arbitrary column shall be detected by the photodiodes and IR sensors. |  | Must |
|  | The system is able empty the playfield, separate the tokens by colour and prepare itself for the next game. | A total reset of the play environment. | Must |
| UR6.1 | After a game, the tokens must move to the sorting base, by emptying the game board column by column. | In order to avoid obstruction during clearing the board game and make the token checking principle easier. | Must |
| UR6.2 | From the sorting base, the yellow and red tokens shall be sorted and returned to their belonging base. |  | Must |
| UR6.3 | A flipper will shoot the human (yellow) tokens back to their base. |  | Must |
|  | The robot head should be controlled to the desired X and Z position within 1.5mm accuracy |  | Should |
|  | The robot end effector should suck up tokens by actuating the pressure air pump. | Research needs to be done on the sucking power w.r.t. the tokens. | Must |
|  | The robot end effector must release the token at a given position to insert the token into board. |  | Must |
|  | The algorithm running on the Raspberry Pi could be integrated on the new STM32H7 dual-core. |  | Could |

## 

These were acquired from another document. A master-test plan by Jasper Jansen from 09-04-2019.

They were slightly modified to reflect current progress.

## Unit Test Plan

* Timing interrupt
* Vacuum components
* End-switches
* Home-switches
* Encoder readout
* PID calculations
* Motor control
  + X-axis
  + Z-axis
* Servo control
  + End-effector rotation
  + Board clean-up piece
* Software-driven movements
* Token detector – entry point
  + Multiple tokens at once in a single column
  + Multiple tokens at one in different columns
* Flipper control
* RGB sensor
* Emergency stop
* Power/Reset button

## Integration Test Plan

* Initialization sequence
* Home procedure
* Normal play sequence (put token inside column)
* Token separation sequence: column by column (new)
* Token sorting sequence (new)
* Cheating Procedure (worst case scenarios)(new)
* Emergency stop and recovery procedure

## System Test Plan (use cases?)

* Starting the system
* Playing of multiple games  
  🡪 winning  
  🡪 losing  
  🡪 different difficulties
* Shutdown
* System is able to play for multiple hours (new)
* Maybe start with sorting procedure (the robot has to know which colour is his)(new)