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 | 06-03 |  |  | First draft of the document |
| 0.2 | 16-03 |  |  | Revision with Michael |
| 0.3 | 20-03 |  |  | Revision with Gwen |

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

Unit Test Plan 6

Integration Test Plan 7

System Test Plan (use cases?) 7

# 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 | Explanation | Priority |
| UR.1 | The user should be able to start and play a game of Connect-4 against the robotized opponent without operator intervention. | The system should be fully automated, allowing the user to start and play a game of Connect-4 against the robot without any human intervention. The robot should be able to detect the user's moves and respond accordingly with its own moves. | Must |
| UR.2 | The user shall be notified when the game ends. | The system should provide a clear indication to the user when the game has ended, either because one player has won, or because the game has ended in a draw. | Could |
| UR.3 | The robot must 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. Or one who tries to input a wrong object. | Must |
| UR.4 | A Board Support Package (BSP) must be made of the operating system with which the necessary hardware components of the robot can be controlled. | BSP must be developed for the operating system, which will allow the necessary hardware components of the robot to be controlled. This will ensure that the system is able to operate reliably and consistently. | Must |
| UR.5 | The insertion of a game token in an arbitrary column shall be detected by the photodiodes and IR sensors. | The system should be able to detect the insertion of a game token in any column using photodiodes and IR sensors. This will ensure that the robot is able to accurately detect the user's moves and respond accordingly. | Must |
| UR.6 | The system is able empty the playfield, separate the tokens by colour and prepare itself for the next game. | The system should be able to automatically empty the playfield at the end of each game, separate the tokens by colour, and prepare itself for the next game. This could involve moving the tokens to a sorting base, as specified in the following sub-requirements. | 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 – on the user side. | The tokens must be sorted by colour at the sorting base and returned to their belonging base on the user side. This will ensure that the system is ready for the next game. | Must |
| UR6.3 | A flipper will shoot the human (yellow) tokens back to their base. | This sub-requirement specifies that a flipper must be used to shoot the yellow tokens back to their base. This will ensure that the system is fully automated, and the user does not need to manually retrieve the tokens. | Must |
| UR.7 | The robot head should be controlled to the desired X and Z position within 1.5mm accuracy | This requirement specifies that the robot head must be able to move to the desired X and Z position with a high degree of accuracy. | Should |
| UR.8 | 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 |
| UR.9 | The robot end effector must release the token at a given position to insert the token into board. | The robot must be capable of precise positioning and releasing of the token to  ensure it goes into the correct slot. | Must |
| UR.10 | The algorithm running on the Raspberry Pi could be integrated on the new STM32H7 dual-core. | This means that the software running on the robot could be optimized for performance and power efficiency using the new hardware. The integration of the algorithm on a new platform may require modifications to the code and additional testing to ensure proper functioning. | Could |

## 

# Test Cases

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
  + Vacuum Pump
  + Vacuum Sensor
  + Vacuum Valve
* End-switches
* Home-switches
* Encoder readout
* PID calculations
* Motor control and **accuracy**
  + 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
  + Single tokens at once in different columns
  + Multiple tokens at once 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 with detection and response by robot
* Emergency stop and recovery procedure
* Integration of the new dual-core architecture with the existing system

## 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).
* • Ability to switch between different difficulty levels
* • Ability to play against a human opponent
* • Testing of different play scenarios with multiple games
* • Testing of overall system reliability over extended period of time
* • System response to unexpected events such as power loss or component failure.