Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Summer 2022

Team Kratos
Mitsubishi Robotic Arm

Che Justus Kwanga
Nabin Raj Bista
Ayush Shrestha
Yan Wang

Revision History

Revision	Date	Author(s)	Description
0.1	06.23.2022	NRB, CJK, AS, YW	document creation
1.1	07.10.2022	NRB, CJK, AS, YW	complete draft

CONTENTS

List of Figures

List of Tables

1 INTRODUCTION

This robotic arm product is designed to play tic-tac-toe with humans. A gripper is attached to the main body of the robotic arm, which can hold a marker pen to write and play tic-tac-toe. It also holds an eraser so that it can erase the board and start over with a new game.

All the layers of the robotic arm are designed to ensure that it can draw the grid lines required for the game on the glass, playing the game of tic-tac-toe with the user from one side of the glass. During the game, it uses the built-in algorithm to randomly decide the order of tic-tac-toe games, and take strategic actions in turn to play against its human opponent.

2 System Overview

2.1 HMI Layer

The Human-Machine Interaction (HMI) layer is in charge of relaying information between the robot and human commands. It should be able to receive a start command to start the game. It should be able to turn on the LED with an appropriate color to indicate whose turn it is currently. It communicates in a two way form with the logic layer.

2.2 Vision Layer

The Vision layer is a two way communication layer of the robot in charge of translating positions of the board at the request of the logic layer. It will take a picture, convert this picture to text and send the information on the current state of the board to the logic layer for interpretation and for the system to make the next appropriate move.

2.3 Logic Layer

This layer is in charge of the overall game logic in deciding the move to make and the position that the robotic arm should move to. It also in charge of drawing the grid at the start after a command from the HMI layer. It is also in charge of deciding the shapes to be drawn and deciding at what point the game is over. It communicates with the navigation, vision and HMI layer in two way fashion depending on the current demands of the robot.

2.4 Navigation Layer

This is a two way communication layer in charge repositioning the robotic arm based on the command from the logic layer. It should be able to readjust the robotic arm in the x,y and z planes to facilitate three dimensional movement around the board.

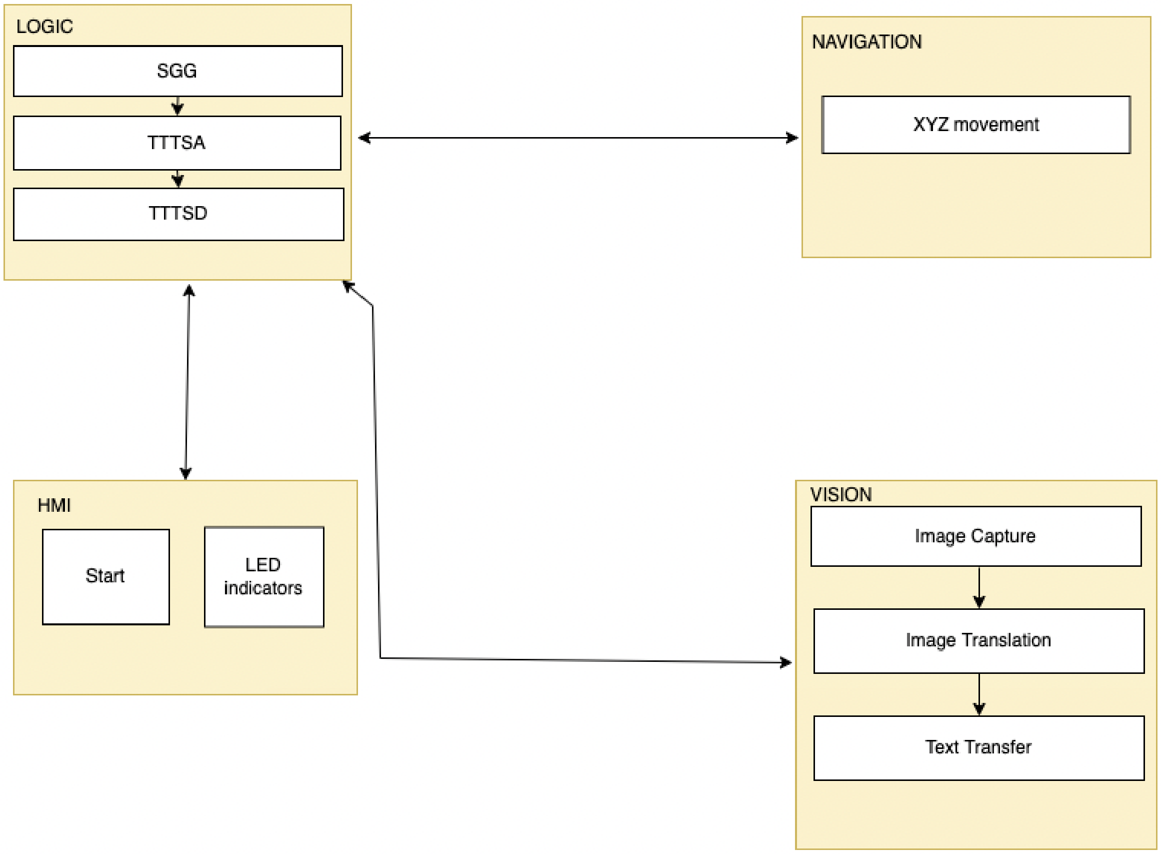

Figure 1: System Architecture

3 HMI Layer

This layer is responsible for the Human-Machine interaction.The HMI layer is very important in this project as the project is mainly based in human-robot interactions and confusion during this process will lead to complications. The robot must be able to determine when it is the humanâs move and when it should be moving. Similarly, it should be able to tell the human that their turn is next and work accordingly.

3.1 Layer Hardware

The hardware consists of cameras as well as lights - green and red to indicate who plays next in the game. The cameras will be detect the presence of a player whereas red light will indicate that it is the robots play and the green light will indicate it is the humans play.

3.2 Layer Operating System

The layer does not require an operating system to operate but solely depend on the program to indicate the turn of plays. The program will decide who will play first randomly and switch between the players.

3.3 Layer Software Dependencies

This layer will depend on the python program to determine the first player randomly then switch between the players.

3.4 Human Detection Subsystem

The purpose of this subsystem is to detect if any human is there to play the game and turn the robot off in case there is no one playing the game.

3.4.1 Subsystem Hardware

This subsystem requires a camera to detect humans. If there is any movements in the playing grid, the robot will start to function and play.

3.4.2 Subsystem Operating System

The subsystem will solely depend on the processing power of the camera and will not require any operating system.

3.4.3 Subsystem Software Dependencies

The subsystem's sole purpose is to detect movements and will not require any software.

3.4.4 Subsystem Programming Languages

The camera will be able to detect the human playing which is followed by the regular game playing algorithm to start.

3.4.5 Subsystem Data Structures

The subsystem camera use infrared to detect movements in the play area. The camera that we will be using detect heat signature change. The Passive Infrared sensor in the camera detect emitted infrared energy from object in the form of heat.

3.5 Human-Machine Indicator Lights

The purpose of this subsystem is to indicate the plays during the play. The red light will indicate the robots play and green will indicate the humans play.

3.5.1 Subsystem Hardware

This subsystem requires a lights to indicate who is supposed to play the game. The system consists of 2 kinds of lights: Green: It is the human's plan and Red: It is the robot's play.

3.5.2 Subsystem Operating System

The subsystem will not require any operating system.

3.5.3 Subsystem Software Dependencies

The subsystem's sole purpose is to indicate and will not require any software.

3.5.4 Subsystem Programming Languages

The camera will use the same programming language that is used on the game playing algorithm. We have planned to use python for the game playing which can also be used to light the red or green lights.

3.5.5 Subsystem Data Structures

The subsystem will indicate the player's turn using the lights. The program will be able to randomly choose the player and indicate the player and then switch between the lights.

4 Vision Layer

This layer deals with the vision of the robotic arm. This layer will be required to handle image capture, image translation and text transfer of the image information to the robotic in order to process and decide the next action.

4.1 Layer Hardware

The hardware will consist of a suitable camera that can be used reliably to enable computer vision and that is able to detect and specify the different ratios of red, blue and green in an image.

4.2 Layer Operating System

This layer will not require an operating system but solely rely on the internal processing power of the selected camera.

4.3 Layer Software Dependencies

This layer will depend on Open CV software which is an open source software that translates images into information that can be used by other systems. Thus Open CV will enable us to incorporate camera vision in our system.

4.4 Image Capture Subsystem

The purpose of this subsystem will be to capture the image of the board state at given intervals. It will be a part of the camera hardware. We will not be required to design it.

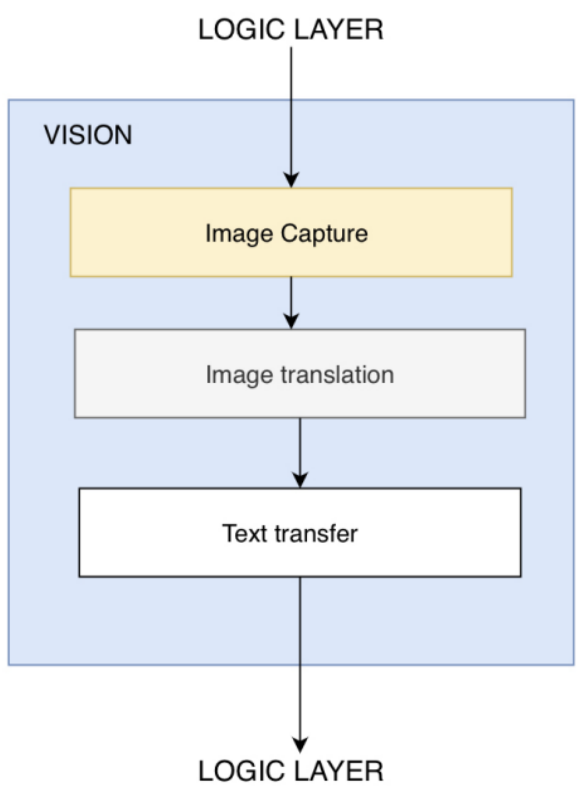

Figure 2: Example subsystem description diagram

4.4.1 Subsystem Hardware

All the hardware needs will be a part of the camera that is used alongside the project. Consists of the lens, diaphragm, and other essential parts of a functional camera.

4.4.2 Subsystem Operating System

This subsystem is within the camera and therefore will not require any operating system and solely depend on the processing power of the camera.

4.4.3 Subsystem Software Dependencies

This subsystem will not require any software as it is only involved in capturing the game state and not any translation or transmission.

4.4.4 Subsystem Programming Languages

No programming language is required for this mechanical piece of hardware.

4.4.5 Subsystem Data Structures

The image sensor of the camera has millions of light capturing wells called photo-sites that each have specific locations and translate that information into pixels. The pixels will then combined into an image that we will translate with the image translation layer.

4.4.6 Subsystem Data Processing

No algorithms aside from the manufacture's preset algorithms in the camera will be used. Therefore, we are not required to curate any algorithms for this subsystem.

4.5 Image Translation Subsystem

The purpose of this subsystem will be to receive and translate an image into useful information for the robotic arm. We are required to optimize the accompanying software to our needs through algorithms.

4.5.1 Subsystem Hardware

This subsystem does not require any hardware design or involvement.

4.5.2 Subsystem Operating System

This subsystem will require a Linux or Windows operating system that will be able to receive the image from the camera.

4.5.3 Subsystem Software Dependencies

This subsystem will depend on Open CV which as an open source software that simplifies computer vision. Open CV will be run on the selected operating system that is compatible.

4.5.4 Subsystem Programming Languages

Open CV is written in C++ but we will be using the bindings in Python for this project.

4.5.5 Subsystem Data Structures

Any data structures that will be used will be a part of the Open CV software that are presets. Such as classes, matrices and buffers.

4.5.6 Subsystem Data Processing

We will curate adequate algorithms that will be able to translate the image of from the camera into useful information for the robotic arm, preferably text.

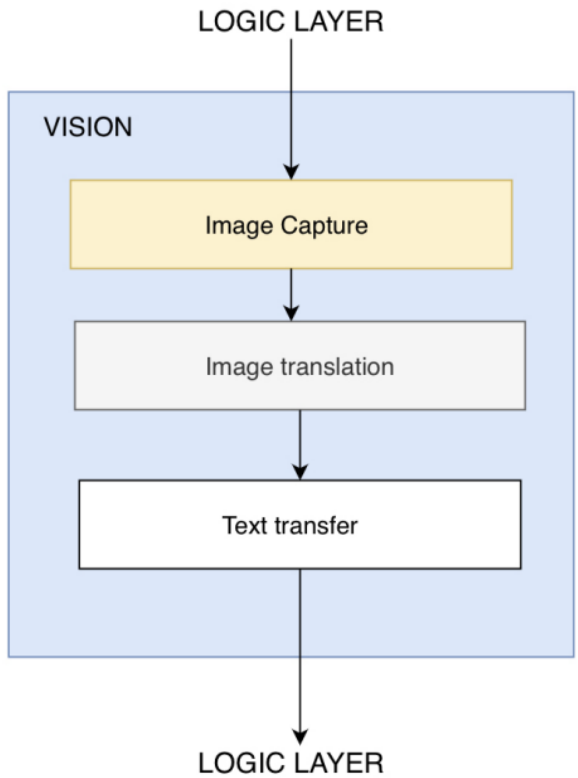

Figure 3: Example subsystem description diagram

4.6 Information Transfer Subsystem

The purpose of this subsystem will be to transmit the information from the open CV software to the robotic arm in order for the robotic arm to make it's next decision based on the game state.

4.6.1 Subsystem Hardware

This subsystem will be dependent on the hardware of the computer that is being used in tandem with the robotic arm for easy transfer.

4.6.2 Subsystem Operating System

This subsystem will depend on the operating system of the computer that was selected for the image translation subsystem.

4.6.3 Subsystem Software Dependencies

This subsystem will require RT Tool Box 3 and it's features that will allow text transfer and translation of text files into suitable information for the robotic arm.

4.6.4 Subsystem Programming Languages

This will depend on assembly programming that is closer to the RT Tool Box 3 software in order to relay information to the arm.

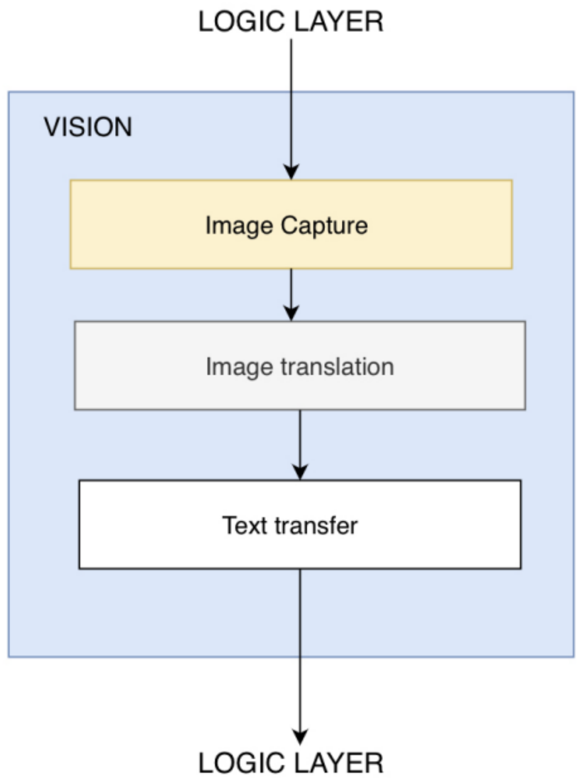

Figure 4: Example subsystem description diagram

4.6.5 Subsystem Data Structures

This subsystem will consist of matrices of information in text form that will be used by the robot to determine the board state.

4.6.6 Subsystem Data Processing

We will be required to curate suitable algorithms that will translate the information from the image capture subsystem into information that will be useful in RT Tool Box 3.

5 Logic Layer

The logic layer is the most basic module among all the modules of the robot arm. It plays the role of the core brain as robot, and is responsible for the tic-tac-toe game algorithm, how to draw game marks, and how to communicate and coordinate the other three layers. The Logic Layer has three subsystems: setup game grid subsystem, tic-tac-toe strategy algorithm subsystem, and tic-tac-toe symbol drawing system.

5.1 Layer Hardware

The hardware set up for the logic layer mainly includes a desktop computer and the controller panel of Mitsubishi Robot RV-8CRL.

5.2 Layer Operating System

The operating systems required by logic layer are Windows and RV-8CRL embedded system since most software operation will be done on a desktop and RV-8CRL control panel.

5.3 Layer Software Dependencies

The software dependencies for logic layer will include but not limited to RT Toolbox3 and OpenCV library.

5.4 SETUP GAME GRID SUBSYSTEM

Setup game grid subsystem obtains permission through the HMI Layer after the robot receives the command from human, and it then starts the Navigation Layer to help robotic arm move to the front of the glass plate in certain location, and uses the algorithm written inside the setting game grid subsystem to draw a game grid consisting of two vertical and two horizontal lines on the glass plate.

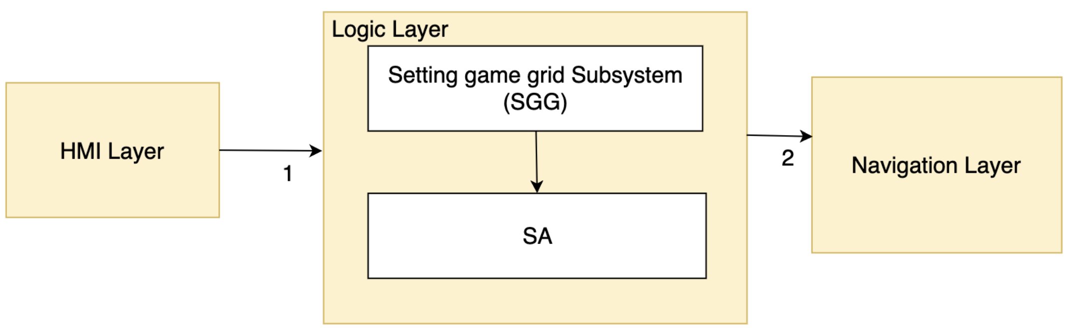

Figure 5: Example subsystem description diagram

5.4.1 Subsystem Hardware

The hardware set up for the setup game grid subsystem includes a RV-8CRL robot arm itself and a gripper attached on it.

5.4.2 Subsystem Operating System

The operating systems required by the setup game grid subsystem are Windows and RV-8CRL embedded system.

5.4.3 Subsystem Software Dependencies

The software dependency for the setup game grid subsystem is all pre-built in RV-8CRL embedded system.

5.4.4 Subsystem Programming Languages

The programming languages used by the setup game grid subsystem will be Python.

5.4.5 Subsystem Data Structures

The data structure involved in the setup game grid subsystem will be an integer array since the game pad will be divided into nine squares and marked individually.

5.4.6 Subsystem Data Processing

No specific data processing strategy is involved in the setup gram grip subsystem at current developing stage.

5.5 Tic-Tac-Toe strategy Algorithm Subsystem

Tic-tac-toe strategy algorithm subsystem is the core subsystem of the Logic Layer. It randomly determines the player of the first round after the game starts, coordinates the Navigation Layer and the Vision Layer, and make marks on board through the tic-tac-toe symbols drawing system to truly complete the task of playing tic-tac-toe with human.

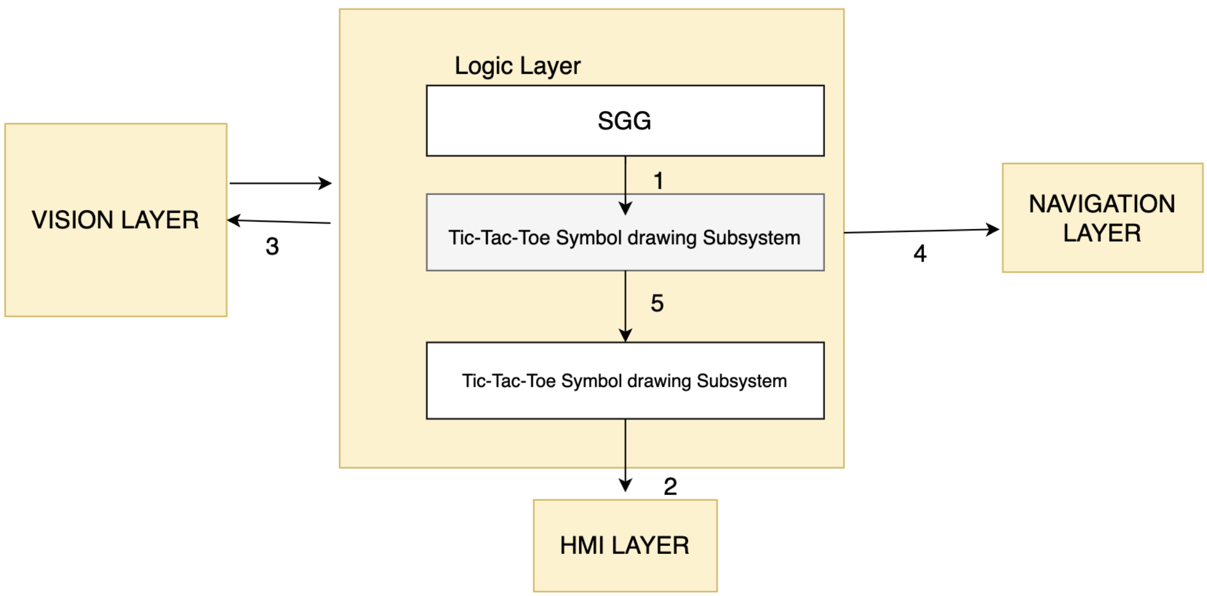

Figure 6: Example subsystem description diagram

5.5.1 Subsystem Hardware

The hardware set up for the tic-tac-toe strategy algorithm subsystem is a desktop computer.

5.5.2 Subsystem Operating System

The operating systems required by the tic-tac-toe strategy algorithm subsystem is Windows.

5.5.3 Subsystem Software Dependencies

No software dependency is mandatorily required for the tic-tac-toe strategy algorithm subsystem, but developers will work it on Visual Studio.

5.5.4 Subsystem Programming Languages

The programming languages used by the tic-tac-toe strategy algorithm subsystem will be Python.

5.5.5 Subsystem Data Structures

The data structure involved in the tic-tac-toe strategy algorithm subsystem will be enumeration which indicates the occupation status of different spots on game pad.

5.5.6 Subsystem Data Processing

The most important algorithm implemented in the tic-tac-toe strategy algorithm subsystem is Minimax algorithm for AI to choose the best move against human player.

5.6 Tic-Tac-Toe Symbol Drawing Subsystem

The tic-tac-toe symbol drawing subsystem is a combination of pixels and algorithms. Its task is only to record the corresponding symbols that should be marked by its own turn at the beginning of each game, and to draw the mark at a specific position when the tic-tac-toe strategy algorithm subsystem sends order.

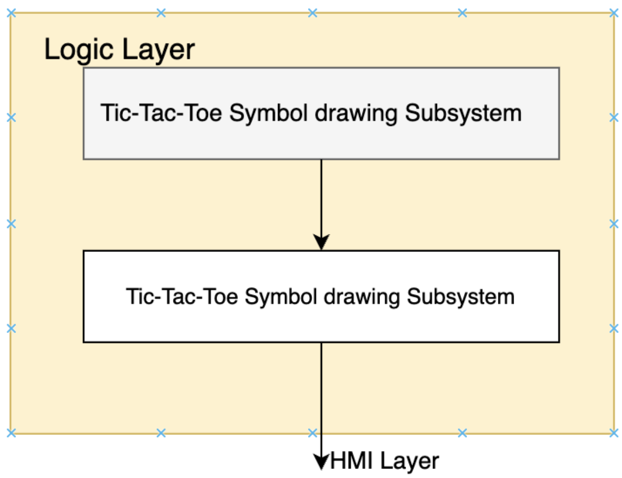

Figure 7: Example subsystem description diagram

5.6.1 Subsystem Hardware

The hardware set up for the tic-tac-toe symbol drawing subsystem includes a RV-8CRL robot arm itself and a gripper attached on it.

5.6.2 Subsystem Operating System

The operating systems required by the tic-tac-toe symbol drawing subsystem is RV-8CRL embedded system.

5.6.3 Subsystem Software Dependencies

The software dependency for the tic-tac-toe symbol drawing subsystem is all pre-built in RV-8CRL embedded system.

5.6.4 Subsystem Programming Languages

The programming languages used by the tic-tac-toe symbol drawing subsystem will be Malfa Basic IV.

5.6.5 Subsystem Data Structures

The data structure involved in the tic-tac-toe symbol drawing subsystem will be enumeration which indicates the occupation status of different spots on game pad.

5.6.6 Subsystem Data Processing

No specific data processing strategy is involved in the tic-tac-toe symbol drawing subsystem at current developing stage.

6 Navigation Layer

This layer mainly deals with navigation of robotic arm. Our robotic arm is able to move in all direction in 3D space. The robotic arm used is Mitsubishi Electric Industrial Robot RV-8CRL.

6.1 Layer Hardware

The main hardware layer involved in navigation layer is RV-8CRL Mitsubishi robotic arm. This robotic arm can be moved for required purpose.

6.2 Layer Operating System

This layer does not require specific operating system. However, better operating system is always preferred.

6.3 Layer Software Dependencies

This layer is depend on RT TOOLBOX3 software for the movements. We use RT TOOLBOX3 software to code the robotic arm so, it can be moved or navigate according to the user requirements.

6.4 Grid Drawing

This subsystem mainly focus on the drawing grid. It involves movements of robotic arm.

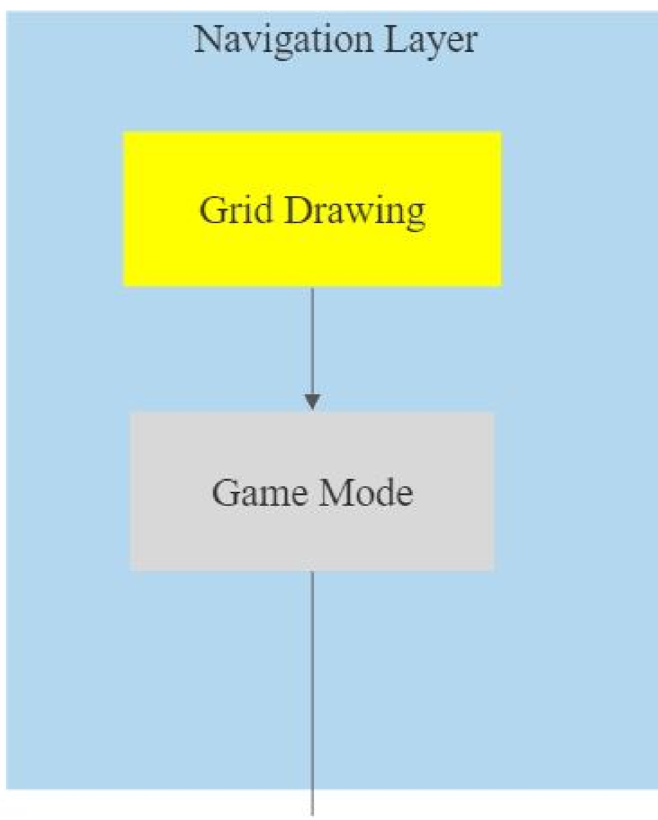

Figure 8: Subsystem description of Navigation Layer

6.4.1 Subsystem Hardware

This subsystem requires gripper which is attached to the front of robotic arm .It can hold pen to draw grid on the required place.

6.4.2 Subsystem Operating System

A specific software is not required for this subsystem.

6.4.3 Subsystem Software Dependencies

This subsystem has software dependencies on RT TOOLBOX3.

6.4.4 Subsystem Programming Languages

This subsystem use assembly programming language.

6.5 Game mode

This subsystem help to navigate robotic arm during game in proper coordinate to draw 'X' or 'O' so that it can play game. Logic layer helps to navigate through out the game.

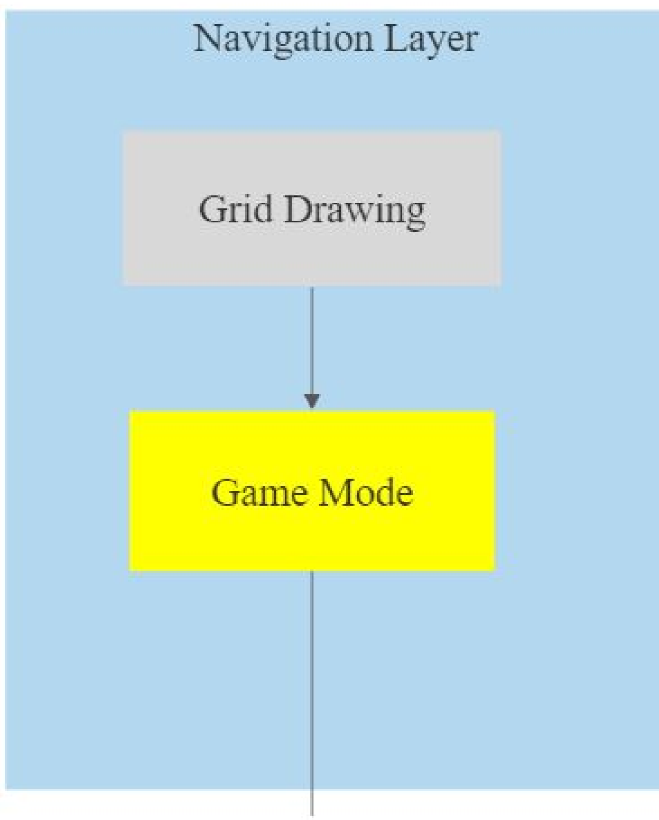

Figure 9: Subsystem description of Navigation Layer

6.5.1 Subsystem Hardware

This subsystem requires gripper which is attached to the front of robotic arm which can hold pen to draw 'X' or 'O' on the grid.

6.5.2 Subsystem Operating System

A specific software is not required for this subsystem.

6.5.3 Subsystem Software Dependencies

This subsystem has software dependencies on RT TOOLBOX3.

6.5.4 Subsystem Programming Languages

This subsystem use assembly programming language and Python.