	Software Architecture
Title:	**Software Architecture ROS TicTacToe**

Maturity:	First Draft
Author(s):	Alfonso, Rudy Manalo
Version:	0.1
Release Authority:	
Distribution:	
Security Classification:	
Number of pages	24

Table of Contents

1. Introduction

"ROS TicTacToe" is basically the famous tic-tac-toe game. It develops under ROS environment. The system can handle two players. Players can decide on how they want to play the game by changing the game settings.
This document elaborates the software architecture document for the system "ROS TicTacToe". The system architecture is abstracted into many views and components which are explained in detail. The document follows the 4+1 view model as the reference model for this document.

1.1 Purpose

This document provides a comprehensive architecture overview of the system, using a number of different architecture views to depict different aspects of the system. It is intended to capture and convey the significant architecture decisions which have been made on the system.
This document elaboarates the architecture of the system in 5 different views. Both static and dynamic behavior of the system is described in this document. All the required diagrams and their descriptions are available in this document.
Using 4+1 view model makes it possible to depict the software as accurarely as possible. It allows a wide range of stakeholders to find what they require in the architecture document.

1.2 Scope

The software architecture document applies to each static and dynamic aspect of the system. Since 4+1 view model is used as the reference model, it incorporates many view of the system, thus makes the document complete and consistent.
Under the static behavior of the system, the document discusses the class diagrams and other static architecture designs. Dynamic aspects of the system are elaborated using use case realizations and system sequence diagrams.

1.3 Definitions, Acronyms, and Abbreviations

Abbreviation	Definition
ROS	Robot Operating System
OOP	Object Oriented Programming
T3	TicTacToe
CPU	Central Processing Unit
BCU	Base Control Unit

1.4 Overview

The report will present a detailed analysis of the architecture of ROS TicTacToe system. Further sections cover the architectural representation of the project including architectural goals and constraints and use case realizations. The later sections cover the detailed specific details of the 4 main views (logical view, process view, deployment view and implementation view) of the system.

2. Architecture Representation

This section details the architecture using the views defined in the 4+1 model. The views used to document the ROS TicTacToe application are:

2.1 Scenarios

Audience:
All the stakeholders of the system, including the end-users.

Area:
Describes the set of scenarios and/or use cases that represent some significant, central functionality of the system. Describes the actors and use cases for the system.

2.2 Logical view

Audience:
Designers, Programmers, Testing staff

Area:
Functional requirements, object hierarchy, system layers
Describes the design of object model. Also, described the subsystems of the system and their relationships.

2.3 Process view

Audience:
Integrators, Programmers

Area:
Non-functional requirements, describes the design’s concurrency and synchronization aspects. Elaborates the run time behavior of the system.

2.4 Development view

Audience:
Programmers, Code testers

Area:
Software components: describes the modules and subsystem divisions of the system.

2.5 Physical view

Audience:
System engineers

Area:
Persistence: describes the architecturally significant persistent elements in the data model. Describes the mapping of the software onto the hardware and shows the system’s distributed aspects.

3. Architectural Goals and Constraints

3.1 ROS

As the system is a ROS-based system, the roscore must be running in order for the ROS nodes to communicate. By running the roscore, it will start-up the ROS Master, ROS Parameter server and rosout logging node. A launch file to run roscore and system related ROS nodes is available in the project.

3.2 Reliability / Availability

The system will be subjected to several testing operations before being deployed in order to make sure that the system is reliable.

3.3 Performance

The system responds to any request in real-time. It makes sure that the maximum execution time of each task especially the tasks executed in Base Control Unit is within the ideal 50% of the minimun execution time.

3.4 Portability and reuse

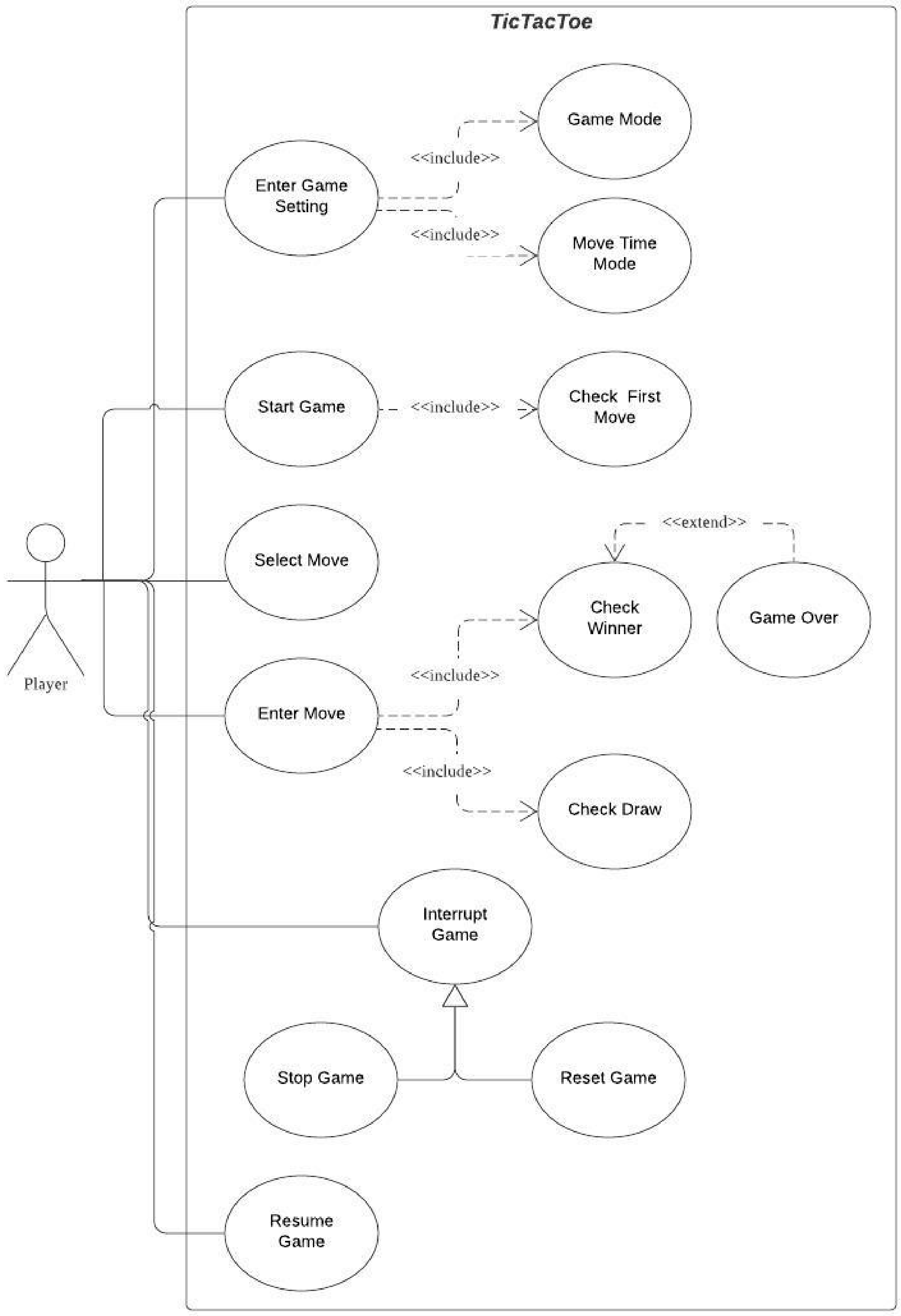
All functionalities are very well structured and layered in order to maintain the reusability and minimize the design and implementation changes. Best practices are followed throughout the project.

3.5 Development tools

Programming: Sublime Text
Diagrams: Lucidchart, Microsoft Excel

4. Use-Case View

4.1 Use-case diagram


TicTacToe
Player
Enter Game Setting
<<include>>
Game Mode
<<include>>
Move Time Mode
Start Game
<<include>>
Check First Move
Select Move
<<extend>>
Check Winner
Game Over
Enter Move
<<include>>
<<include>>
Check Draw
Interrupt Game
Stop Game
Reset Game
Resume Game

4.2 Use-Case Realizations

4.2.1 User Management related use cases

4.2.1.1 Player enters into game setting

Use case name	Enter Game Setting
Scenario	The system is inside the game setting.
Triggering event	Player triggers the enter game setting condition.
Brief description	When Player is inside the game setting, the Player can possibly change the game configuration(e.g. game mode, move time mode).
Actor	Player
Related use cases	None.
Preconditions	Game is not on-going.
Post conditions	The system should inform the Player that it is already inside the game setting.
Flow of events	Player enters into game setting. While inside the game setting, the Player can possibly change the game configuration. After game configuration is done, system will move out from game setting and display the new game configuration.
Exception conditions	None

4.2.1.2 Player starts a new game

Use case name	Start Game
Scenario	New game has been requested.
Triggering event	The Player or system triggers the start a new game condition.
Brief description	If Player or system starts a new game, the new game sequence will start.
Actor	Player
Related use cases	None
Preconditions	Game is not on-going.
Post conditions	The system should inform the Players that a new game has just begun.
Flow of events	The Player or the system starts a new game. A new game sequence will start to inform the Players that a new game is about to start. A new game is started.
Exception conditions	None

4.2.1.3 Player selects move

Use case name	Select Move
Scenario	The game is on-going and Player starts selecting his/her move.
Triggering event	Player pressed and released the select button.
Brief description	If Player pressed and released the select button, the led board nearest vacant slot will be selected.
Actor	Player
Related use cases	None.
Preconditions	The game is on-going.
Post conditions	The system updates the led board information according to slot availability.
Flow of events	The game is on-going. Player starts to select his/her move. System updates the status of the game according to Player's movement.
Exception conditions	If game has stopped, the Player can't select his/her move.

4.2.1.4 Player enters the selected move

Use case name	Enter Move
Scenario	Player enters/registers the selected move.
Triggering event	Player pressed and released the enter button or move time has expired.
Brief description	If Player pressed and released the enter button or move time has expired, the currently selected led board slot will be registered.
Actor	Player
Related use cases	None.
Preconditions	The game is on-going.
Post conditions	The system updates the led board information according to Player's registered move.
Flow of events	The game is on-going. Player registered the selected move or move time has expired. System checks for the game status(winner , game over or draw).
Exception conditions	If game has stopped, the Player can't register the selected move.

4.2.1.5 Player interrupts the game

Use case name	Interrupt Game
Scenario	The game is on-going and it is being interrupted.

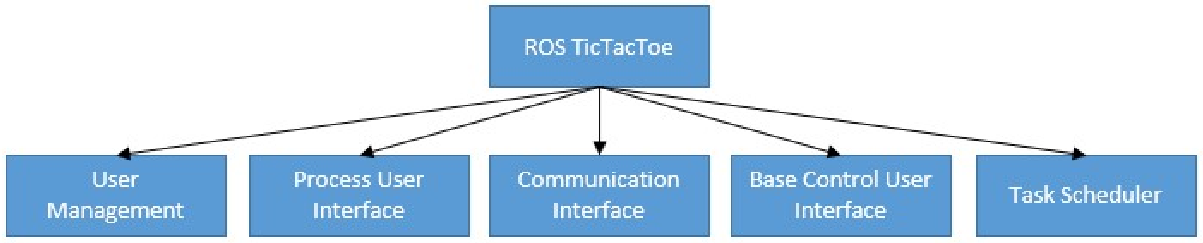
Triggering event	Player stops or resets the game.
Brief description	If game is on-going, Player can possibly interrupt it by stopping or resetting.
Actor	Player
Related use cases	None
Preconditions	The game is on-going.
Post conditions	The game is interrupted.
Flow of events	The game is on-going. Player stops/resets the game. The system informs the Player regarding the game interruption. The Player can possibly resume back the game or start a new game.
Exception conditions	None

4.2.1.6 Player resumes the game

Use case name	Resume Game
Scenario	The game has been previously stopped and Player resumes it back.
Triggering event	Player triggers the resume game condition.
Brief description	If game has stopped, Player can resume back the game.
Actor	Player
Related use cases	None
Preconditions	The game has stopped.
Post conditions	The game has resumed.
Flow of events	The game has stopped. Player resumes the game. The system informs the Players that the game has been resumed.
Exception conditions	None

5. Logical View

5.1 Overview



5.1.1 Subsystems

ROS TicTacToe can be divided into 5 main sub systems.

1. User Management Subsystem
2. Process User Interface Subsystem
3. Communication Interface Subsystem
4. Base Control User Interface Subsystem
5. Task Scheduler Subsystem

5.1.1.1 User Management Subsystem

This is the subsystem that incorporates most of the game algorithms and processing power. Main use cases that comes under this subsystem includes

- Evaluation of Player’s action
- Changing game setting
- Starting new game
- Stopping and resuming game
- Resetting of game
- Checking game results
- Updating game information

5.1.1.2 Process User Interface Subsystem

This is the subsystem where the user interfaces are being processed based on the actual status of the game and Player input.
Main use cases that comes under this subsystem includes

- Switch state information
- Game board LED display
- Score boards and timer board display
- LED indicators
- Buzzer indicator

5.1.1.3 Communication Interface Subsystem

This subsystem involves the ROS message publishing and subscribing of the system.
Main use cases that come under this subsystem includes

- ROS nodes decalaration
- Publish ROS message
- Subscribe ROS message

5.1.1.4 Base Control User Interface Subsystem

This is the subsystem that incorporates all of the game control to deliver the information to the Players. Main use cases that come under this subsystem includes

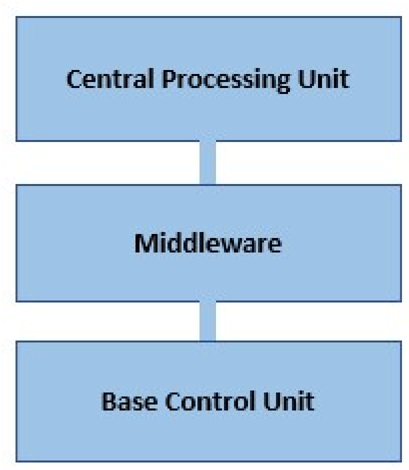
- Reading of switch input status
- Updates the game board LED dispay
- Updates the score boards and timer board display
- Updates LED indicators
- Updates Buzzer indicator

5.1.1.5 Task Scheduler Subsystem

This is the subsystem that handles the task scheduling of the system.

5.1.2 Layering

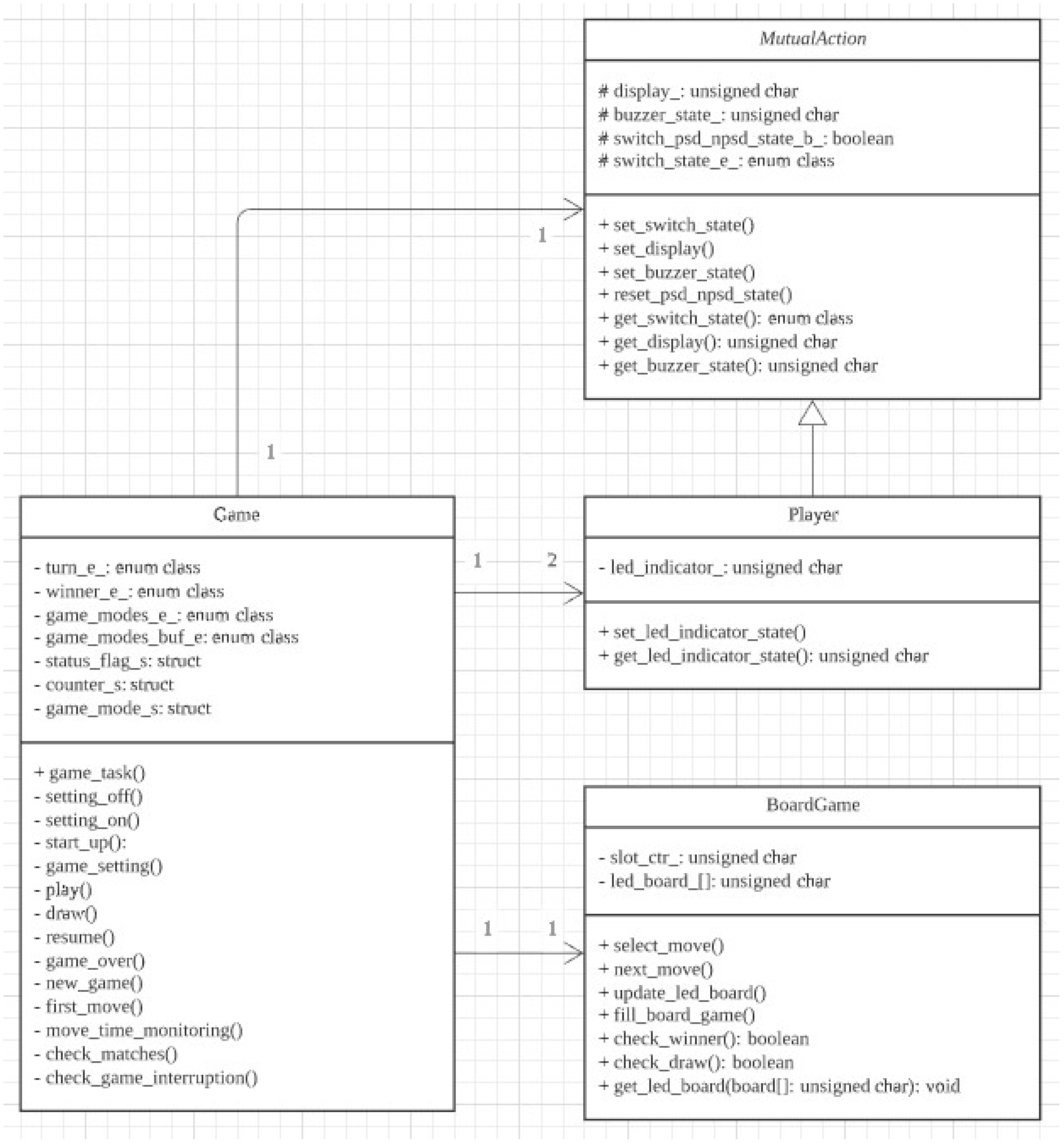
ROS TicTacToe is divided into 3 layers. The layering model of the ROS TicTacToe application is based on a responsibility layering strategy that associates each layer with a particular responsibility. This strategy has been choosen because it isolates various system responsibilities from one another, so that it improves both system development, reusability and maintenance.



Subsystems	**Layers**
User Management Process User Interface	Central Processing Unit
Communication Interface	Middleware
Base Control User Interface Task Scheduler	Base Control Unit

5.2 Architecturally Significant Design Packages

5.2.1 Class diagram

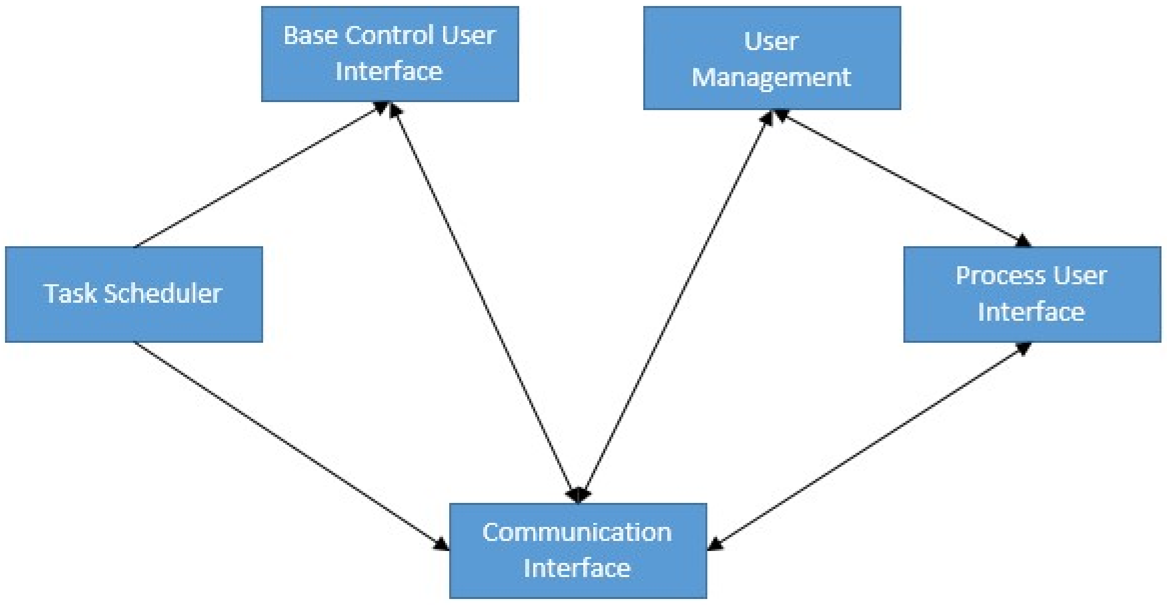

MutualAction
# display_: unsigned char
# buzzer_state_: unsigned char
# switch_psd_npsd_state_b_: boolean
# switch_state_e_: enum class
+ set_switch_state()
+ set_display()
+ set_buzzer_state()
+ reset_psd_npsd_state()
+ get_switch_state(): enum class
+ get_display(): unsigned char
+ get_buzzer_state(): unsigned char
1
1
Game
- turn_e_: enum class
- winner_e_: enum class
- game_modes_e_: enum class
- game_modes_buf_e: enum class
- status_flag_s: struct
- counter_s: struct
- game_mode_s: struct
+ game_task()
- setting_off()
- setting_on()
- start_up():
- game_setting()
- play()
- draw()
- resume()
- game_over()
- new_game()
- first_move()
- move_time_monitoring()
- check_matches()
- check_game_interruption()
1
2
Player
- led_indicator_: unsigned char
+ set_led_indicator_state()
+ get_led_indicator_state(): unsigned char
1
1
BoardGame
- slot_ctr_: unsigned char
- led_board_[]: unsigned char
+ select_move()
+ next_move()
+ update_led_board()
+ fill_board_game()
+ check_winner(): boolean
+ check_draw(): boolean
+ get_led_board(board[]: unsigned char): void

Especial design decisions

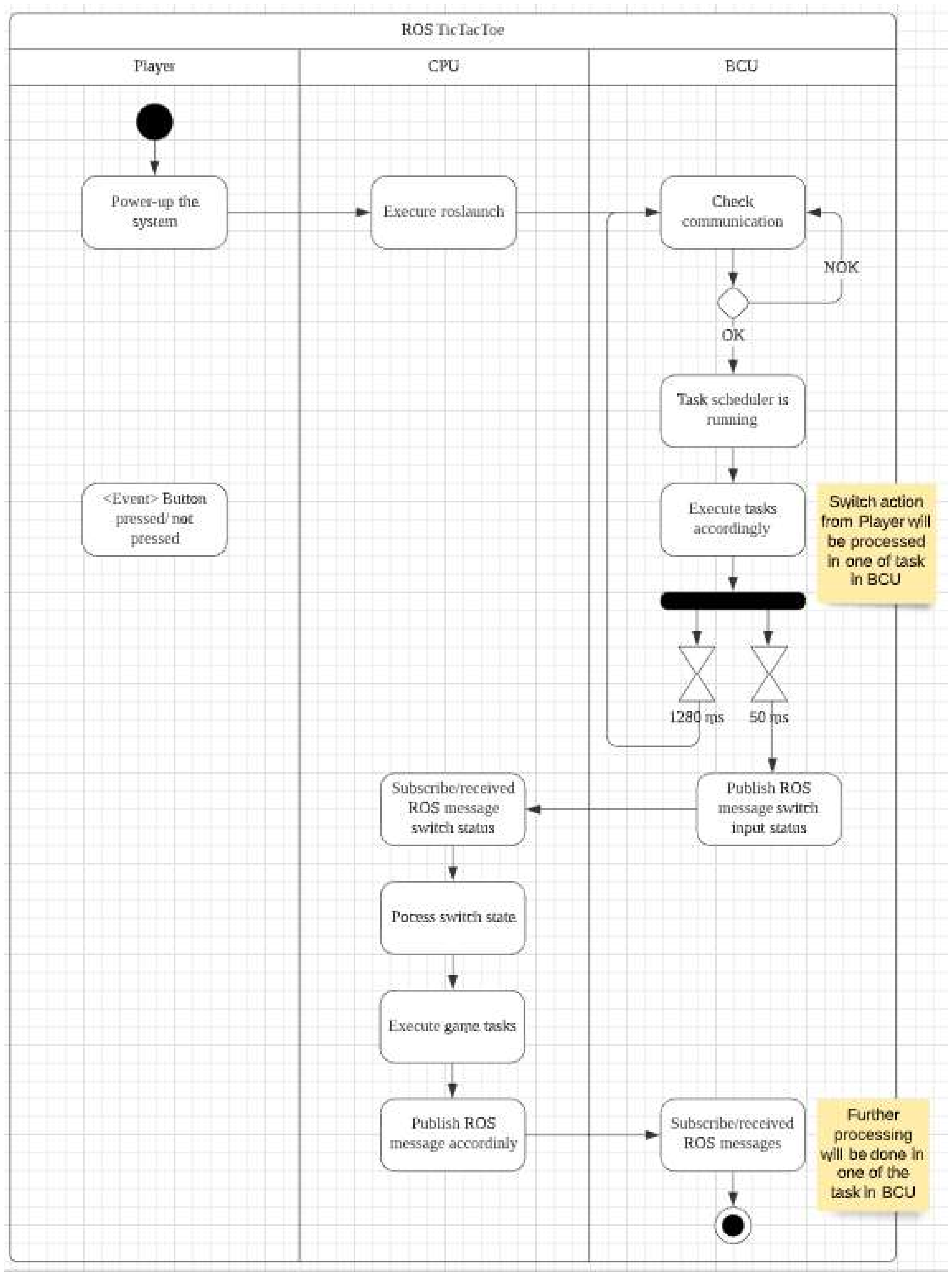
- All Players are derived from the super class called “MutualAction”. It is because each Player of the system has common attributes and behavior such as switch state, display, etc.
- The “Game” class contains the main game application. Each task is executed according to game condition and status.
- Game class uses MutualAction and BoardGame classes to execute common tasks for all Players and having a relationship of one to one.
- Game class uses Player class to execute specific task for each Player. The tictactoe game has two Players and that is the reason for having a one to two relationship between Game and Player.

6. Process View

6.1 Overview



6.2 System activity diagram

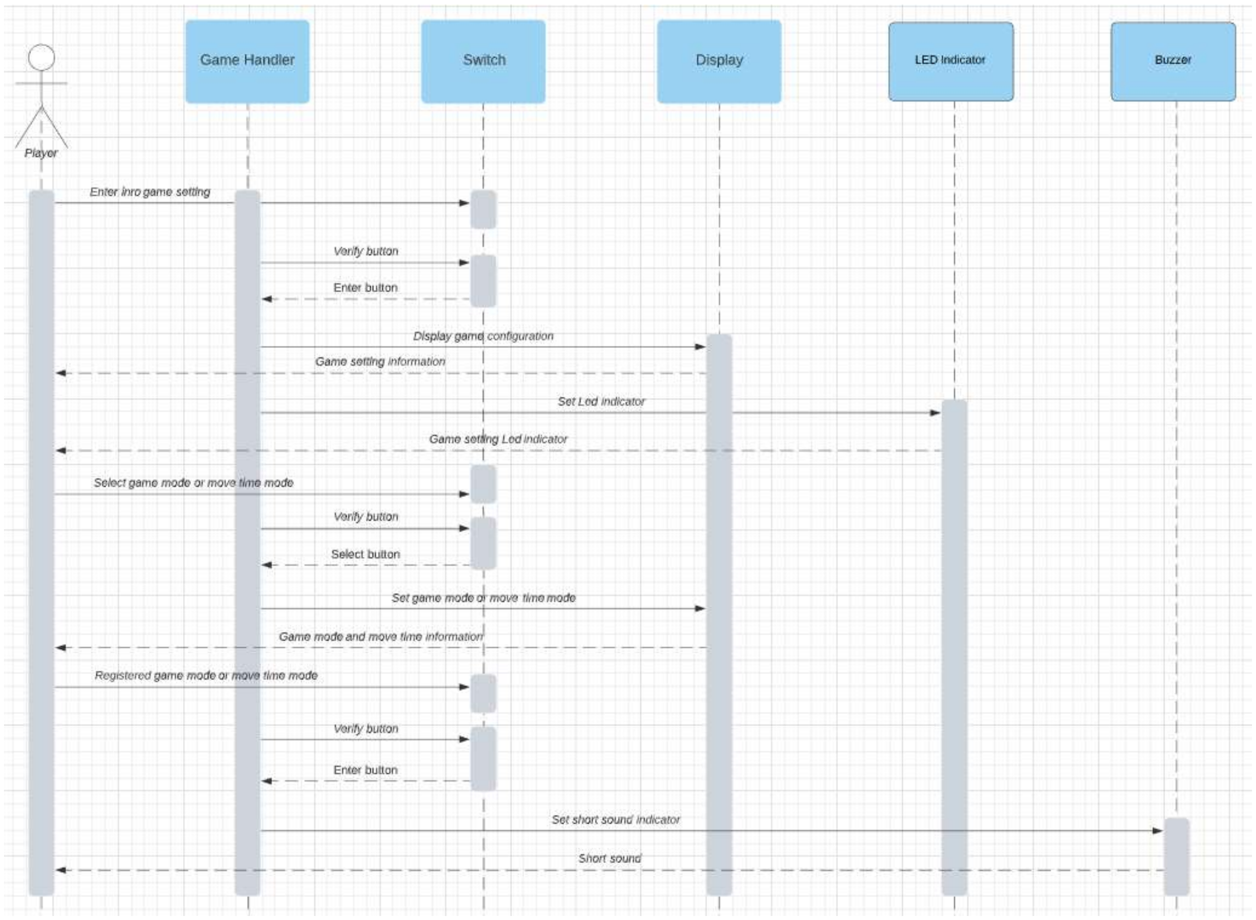

ROS TicTacToe
Player
CPU
BCU
Power-up the system
Execure roslaunch
Check communication
NOK
OK
Task scheduler is running
<Event> Button pressed/ not pressed
Execute tasks accordingly
Switch action from Player will be processed in one of task in BCU
1280 ms
50 ms
Subscribe/received ROS message switch status
Publish ROS message switch input status
Pocess switch state
Execute game tasks
Publish ROS message accordinly
Subscribe/received ROS messages
Further processing will be done in one of the task in BCU

6.3 System sequence diagrams

When Players started the sytem, the system starts the startup sequence. After successful startup, system always checks the status of the game and monitor the input from Players.
The system executes specific task based on the game status and input from Players. Some of these tasks are shown in Chapter 4.1 Use-case diagram.

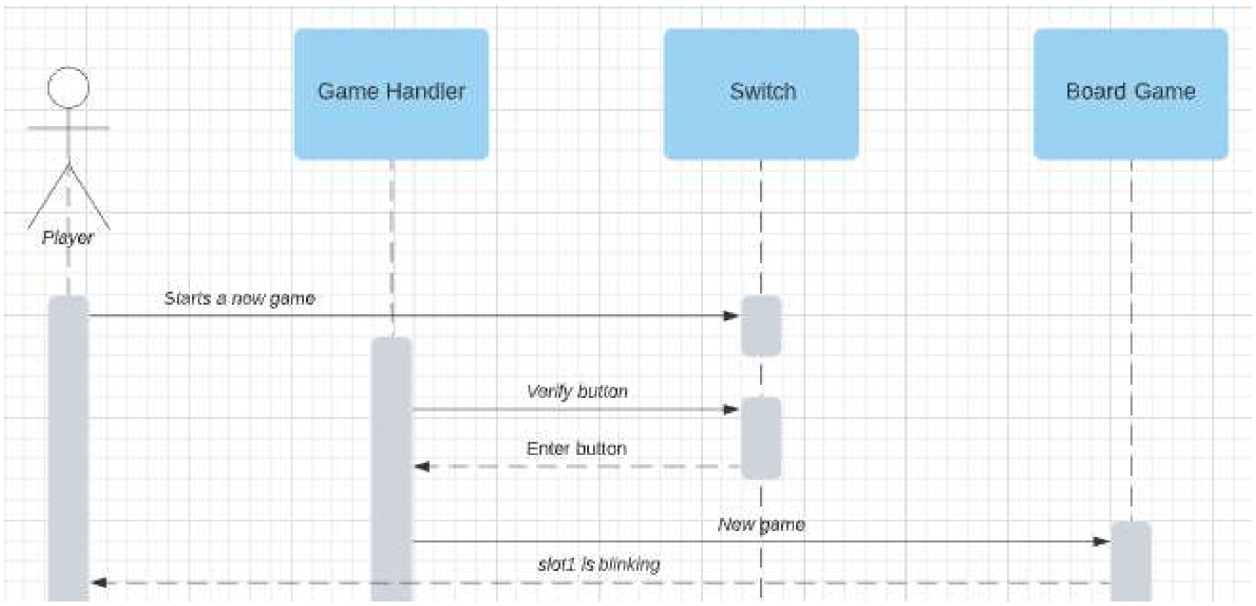
6.3.1 Player enters into game setting

When Player pressed the enter button continously for at least 3 seconds, the system enters into game setting. The current game configuration will be displayed. Led indicators will start to blink to inform Player that the system is already inside the game setting. The Player can select new game mode and move time mode configuration by momentarily pressed the select button. And when Player momentarily pressed the enter button, the selected game mode or move time mode will be registered. Buzzer will produce short-sound to inform Player about the successfull registration.


Game Handler
Switch
Display
LED Indicator
Buzzer
Player
Enter inro game setting
Verify button
Enter button
Display game configuration
Game setting information
Set Led indicator
Game setting Led indicator
Select game mode or move time mode
Verify button
Select button
Set game mode or move time mode
Game mode and move time information
Registered game mode or move time mode
Verify button
Enter button
Set short sound indicator
Short sound

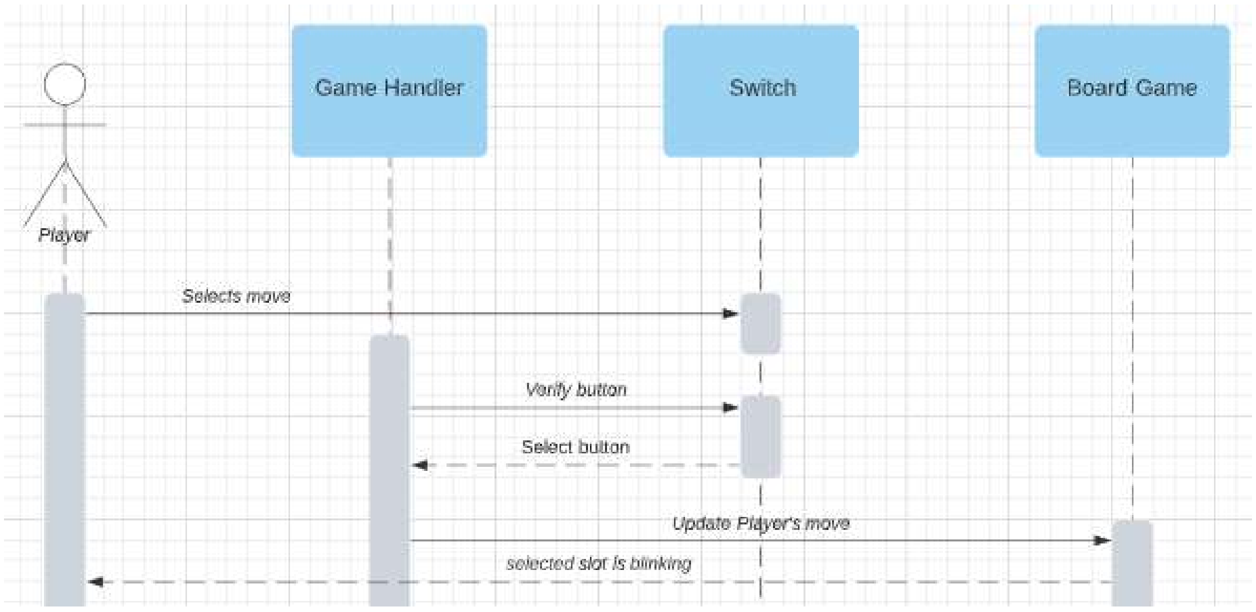
6.3.2 Player starts a new game

When Player pressed momentarily the enter buton, a new game will be started. The led board row0/column0 starts to blink. This informs the Player that the game has just begun.



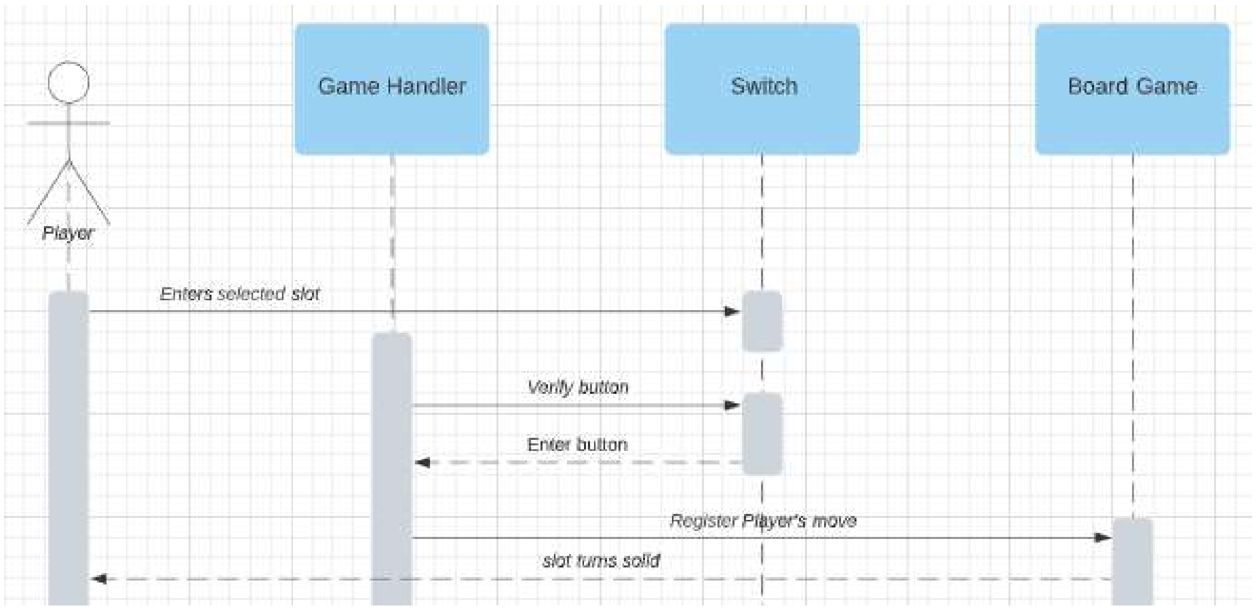
6.3.3 Player selects move

When Player pressed momentarily the select button, the led board game slot will move from one vacant slot to another. The selected slot starts to blink to inform Player about the currently selected slot.



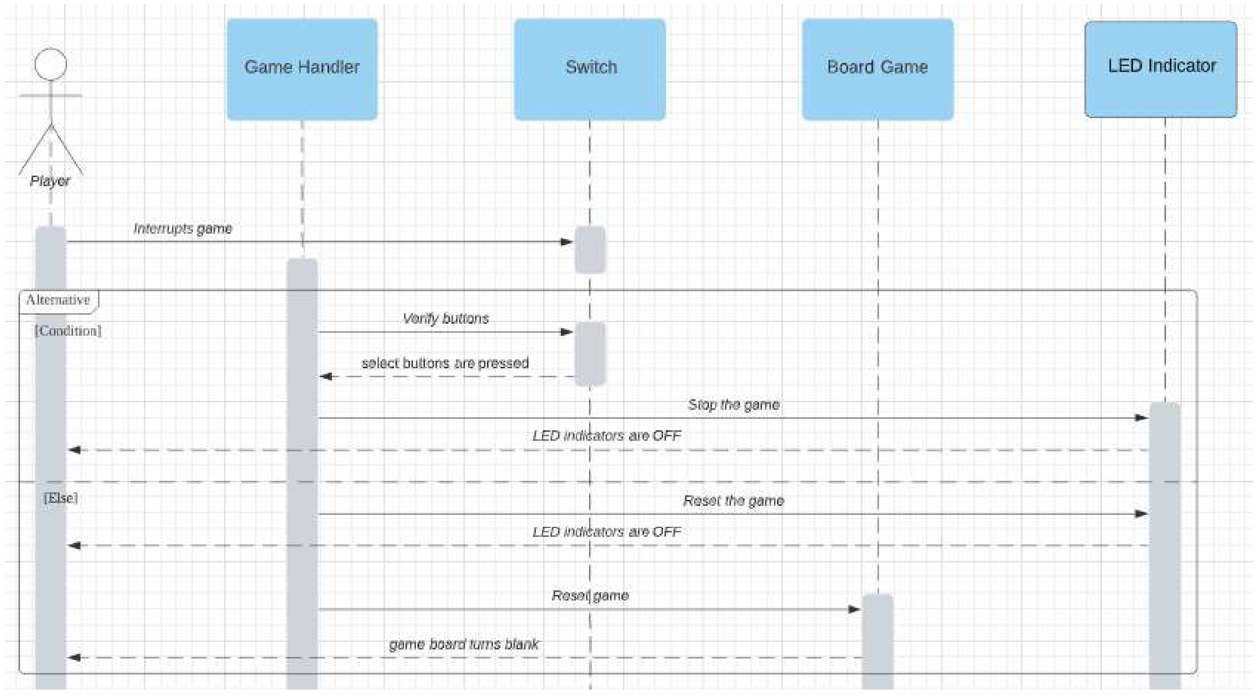
6.3.4 Player enters the selected move

When Player pressed momentarily the enter button, the selected slot will be registered. The selected slot will stop blinking and turn into solid light. This informs the Player that the selected move has been succesfully registered.



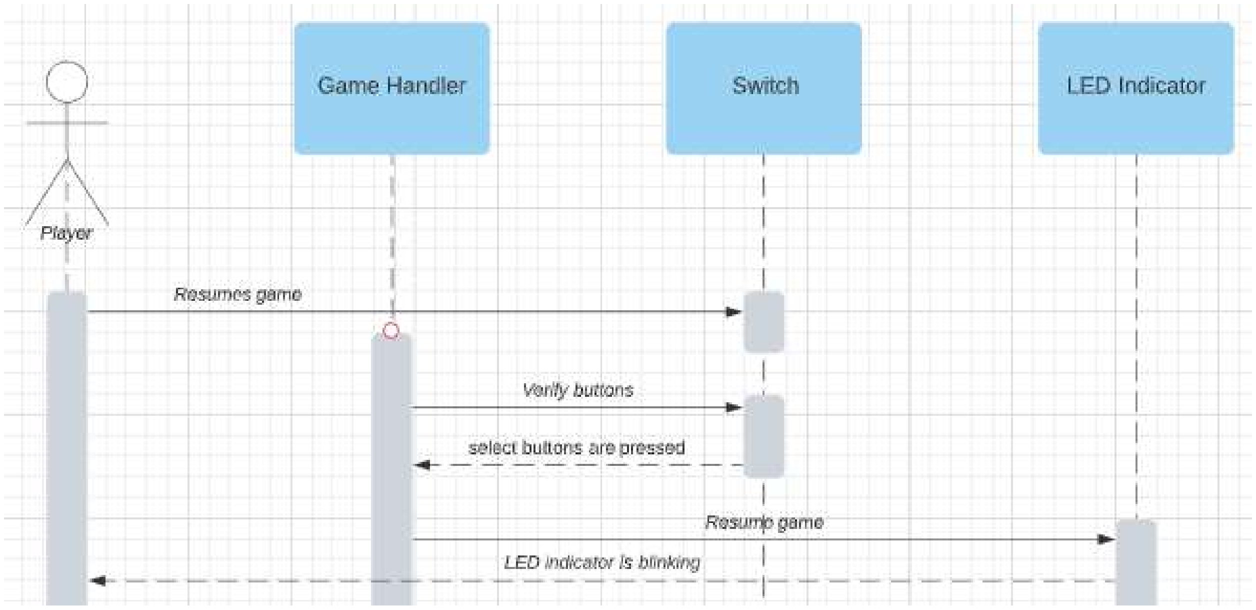
6.3.5 Player interrupts the game

When Player pressed both select buttons or at least one select button and enter button simultaneously for at least 1 second, the game will be interrupted. The game is either stop or reset. The LED indicators will turn OFF to inform Player about the game interruption.
And if game is reset, the led board will turn blank.

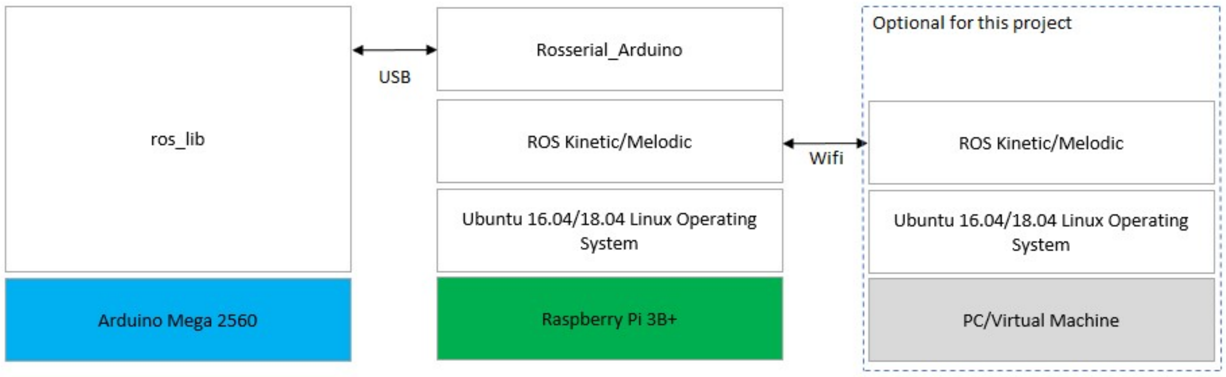


6.3.6 Player resumes the game

When Player pressed the select buttons simultaneously for at least 1 second, the game will resume back. The LED indicator will start to blink accordingly.



7. Deployment View

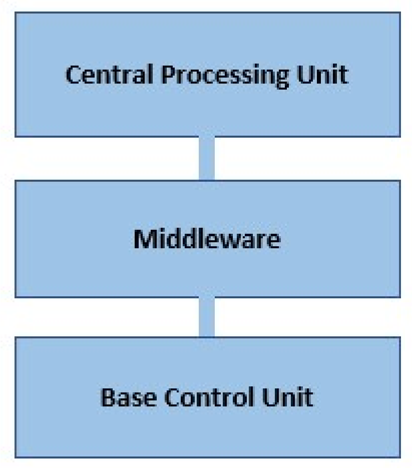


The arduino mega 2560 microcontroller is responsible in controlling the hardware components like LEDs, displays, buzzer and reading of input switches. It is responsible also for the task scheduler of the overall system.
The Raspberry PI 3B+ on the other hand is responsible on the overall game algorithms and CPU processing. It acts as the brain of the system. It has Linux Operating System and ROS (Robot Operating System) installed on it.

From Software Architecture and Requirement point of view, the arduino mega 2560 microcontroller is called T3_BCU /BCU and Raspberry PI 3B+ is called T3_CPU /CPU.

One of the main reason for having this kind of architecture design is to minimize the changes needed on the overall sytem in case microcontroller or ARM board computer is changed.
Like for an instance, if arduino mega 2560 is changed to other microcontroller, there is no need to change anything in the ARM board computer and vice-versa.

8. Implementation View



As mentioned above, the functions or subsystems are separated into layers to improve the maintainability and reusability of the system thus the software will be developed in this way.

9. Size and Performance

Since the system is ROS-based system, the computer/ARM board computer needs to have a valid IP address. It doesn't matter whether it is connected to internet or not. The main algorithms and processing will be done in Central Processing Unit. And all control that has direct connection to hardware components will be processed in Base Control Unit.
The size of the software in Base Control Unit is expected to be a little bit higher since the ROS library has to be integrated. The library is trimmed down to remove unneccessary modules that not needed or related to the system. The overall size is still within the 80% maximum resource consumption at the end of development.

10.Quality

Architecture goals and constrains topic covers the quality parameters for the ROS TicTacToe system.