12.01.2024

GALACTIC

DECEIT

Project
Documentation

CSE 396

GROUP 8

🎥 Demo-Video
https://www.youtube.com/watch?v=NC79Xh6U8Fg

📂 Source Code
https://github.com/djvra/GalacticDeceit/

Table of Contents

Table of Contents 1

Project Introduction 2

Project Description 3

Infrastructure 4

Development Phases 5

Division of Teams 6

Project Design 8

Project Model 12

Project Requirements 13

Module Progressress 17

Calendar 19

Conclusion and Evaluation 26

Project Introduction

This project aims to develop a multiplayer game similar to Among Us. Players will connect to a central computer from their own smartphones. Electronic cards with joysticks will be linked to their phones to control character movements. The central computer will display all players and the game map, while players connecting from their phones will only see their own perspective.

Project Description

Game Mechanics

Players will find themselves in a space station or a similar environment. Each player will be assigned a random role at the start: "Crewmate" or "Impostor." Crewmates will try to complete tasks, while Impostors will attempt to eliminate other players.

Game Controls

Players will control their characters using joysticks and buttons on electronic cards connected to their phones. Joysticks will be used to control character movement for completing tasks or sabotage actions.

Infrastructure

Central Computer

The central computer serves as the game's central hub, acting as a server to which players connect. It manages player data and displays the game map and all players' locations.

Smartphones

Players will control the game via their smartphones. Phones will connect to the central computer. Joysticks will connect to phones through bluetooth connection which enables players to control character movements and actions.

Development Phases

Design and Concept Development

In this phase, the game's core mechanics, map, and character designs will be developed.

Software Development

The central computer software and phone applications will be created. Communication between players' phones and the electronic cards will be established.

Deployment and Server Setup

The game will run on a server where players can connect. Server setup and player connections will be managed during this phase.

Division of Modules

Desktop Development

EMİRKAN BURAK YILMAZ - 1901042659
UFUKCAN ERDEM - 1901042686
METE GONCA - 161044075

The team is responsible for developing the desktop application which displays the game map with player status.

Game Design

AYŞE ELİF BÜYÜK - 200104004126
UMUT CAN ÖZAY - 1901042641
MUHAMMET FURKAN YILDIZ - 1801042643

The team is responsible for developing the mobile application that players will install on their smartphones to connect to the game.

Game Server

EMİRKAN BURAK YILMAZ - 1901042659
AYŞE ELİF BÜYÜK - 200104004126
UMUT CAN ÖZAY - 1901042641

The team is responsible for enabling the multiplayer functionality of the game by managing the connections.

Hardware Connection

UFUKCAN ERDEM - 1901042686
MUHAMMET FURKAN YILDIZ - 1801042643
METE GONCA - 161044075

The team is responsible for designing and manufacturing the electronic cards with joysticks and buttons that players will use to control the game.

Project Design

Desktop Development Module

The desktop application will be developed utilizing the QT framework and C++ programming language. The project will commence with the creation of a user-friendly home page, where users input the server's IP address. Subsequently, the application will establish a seamless connection with the server using the provided IP address. Once connected, the application will dynamically render and display the game map along with real-time player status information.

Game Design Module

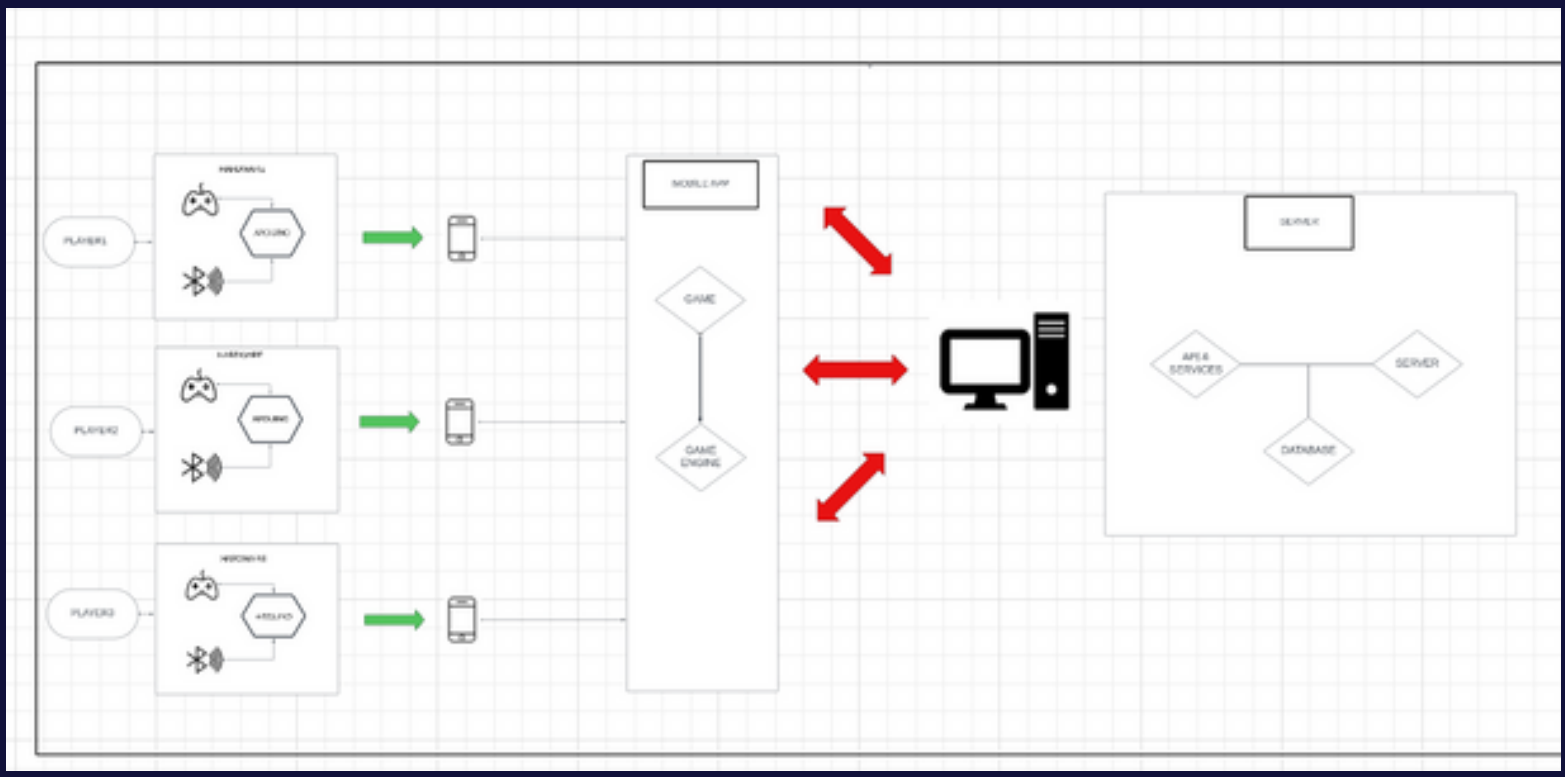
The server module, implemented in C++ as a terminal application manages the game state and establishes connections with both the desktop app and mobile app. The server ensures real-time updates by continuously updating player positions on the dynamic game map and adjusting statuses based on in-game events, allowing the desktop and mobile apps to receive instantaneous information about each player's actions and locations. Networking libraries such as Asio are being considered for handling these connections.

Hardware Connection Module

- The primary hardware components will mainly comprise joysticks, with buttons integrated using joystick shields connected via Arduino. Each joystick will include a single directional analog control along with four functional buttons. These joysticks will establish connections with the mobile application on phones via Bluetooth modules(HC-05). Initial tests will focus on establishing and maintaining seamless communication.
- Ensuring consistent, healthy communication remains the primary objective following successful initial connections. Additionally, minimizing latency and maintaining synchronization between the game and the joystick are crucial aspects. Depending on the project's development, modifications and purposeful enhancements to the joystick hardware may be introduced to optimize functionality.

Game Design

The team will use Unity for mobile app development.
Communication between the app and Arduino will be handled by the Android Bluetooth API. These tools will ensure a robust and efficient application. The aim is to enhance the overall user experience.



overall project model

Project Requirements

Desktop Development

1. Establish and maintain a seamless connection with the server. Implement error handling for connection issues.
2. Retrieve player data using the TCP and UDP protocols server based on the provided IP address.
3. Render the game map with real-time updates.
4. Display player positions on the minimap and in-game events.

Game Server

1. Establish and maintain connections with players.
2. Continuously collect and update player positions on the map.
3. Retrieve and update player status (alive or not).
4. Ensures both the desktop app and other players receive up-to-date information about each player's location and current status during gameplay.
5. Implement error-handling mechanisms for network issues and unexpected events.
6. Manage multiple connections simultaneously

Game Design

1. Render the game state on the mobile device. It should display the player's character and other game elements based on the game state received from the server.
2. Establish and maintain a Bluetooth connection with the Arduino device.
3. Send player actions to the server and receive game state updates.
4. Process the joystick movements received from the Arduino device.

Hardware Connection

1. Construct joystick for playing game on phone.
2. Establish a wireless connection between phone and joystick.
3. Establish real-time synchronization between joystick and phone.
4. Prioritize stable communication to minimize delays in joystick movements on the phone.
5. Improve on joystick for more user-friendly game experience.

Calendar

Desktop Development

- Nov 10 - Nov 19: Create a basic home page layout for obtaining the server IP address.
- Nov 20 - Dec 3: Establish a basic connection mechanism with error handling.
- Dec 4 - Dec 24: Implement retrieving and rendering the game map.
- Dec 25 - Jan 4: Testing and debugging

Game Server

- Nov 10 - Dec 1: Identifying and implementing the most suitable communication protocol for efficient communication between mobile apps and the server using mock data
- Dec 2 - Dec 8: Implementing the connection mechanism between the desktop app and the server using mock data
- Dec 9 - Dec 14: Testing and debugging
- Dec 15 - Jan 4: Integrating modules, testing and debugging

Game Design

- Nov 10 - Nov 21: Build the game and render the game map.
- Nov 22 - Dec 1: Set up connection with server and error handling.
- Dec 2 - Dec 12: Implement Bluetooth communication with Arduino.
- Dec 13 - Dec 26: Create a home page for server IP input.
- Dec 27 - Jan 6: Testing and debugging.

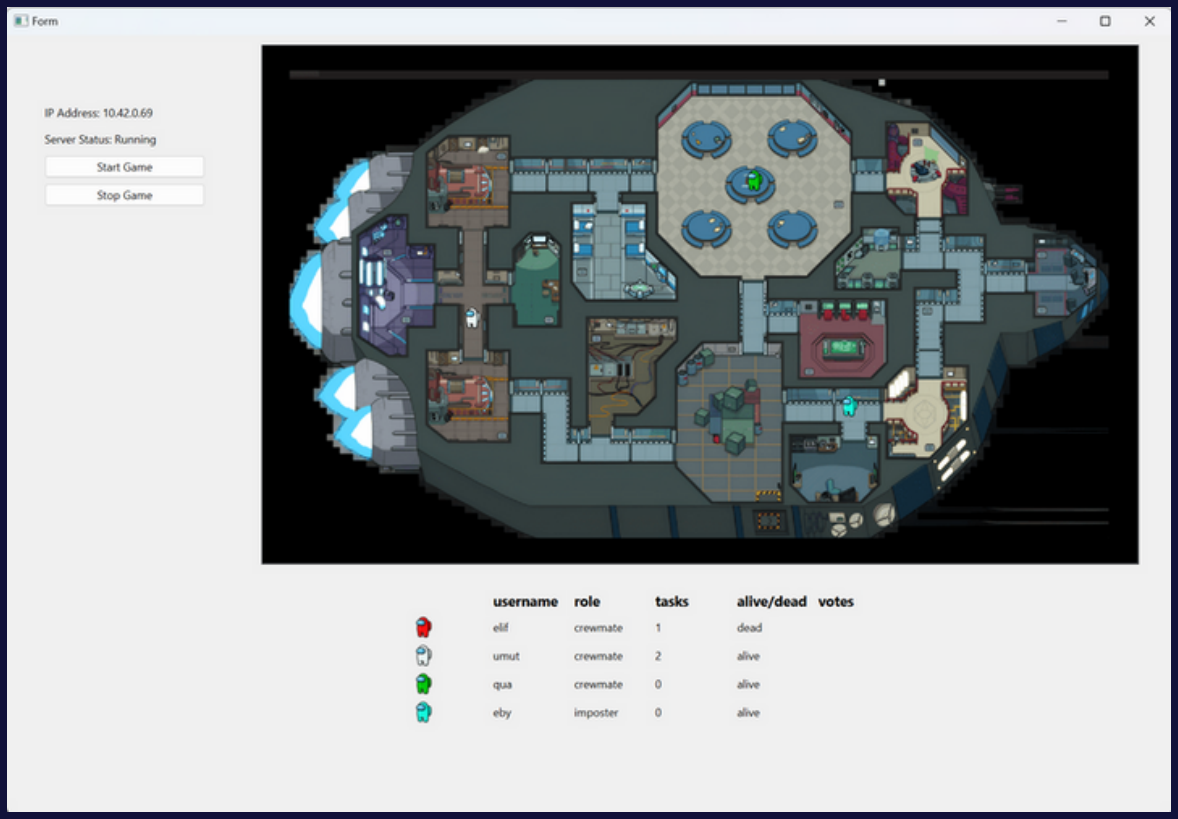
Hardware Connection

- Nov 10 - Dec 1: Construction of Joystick and connection tests
- Dec 2 - Dec 26: Synchronization, Low latency, and Performance Development
- Dec 27 - Jan 6: Developments according to user feedback, Addition to hardware if needed

Module Progresses

Desktop Development

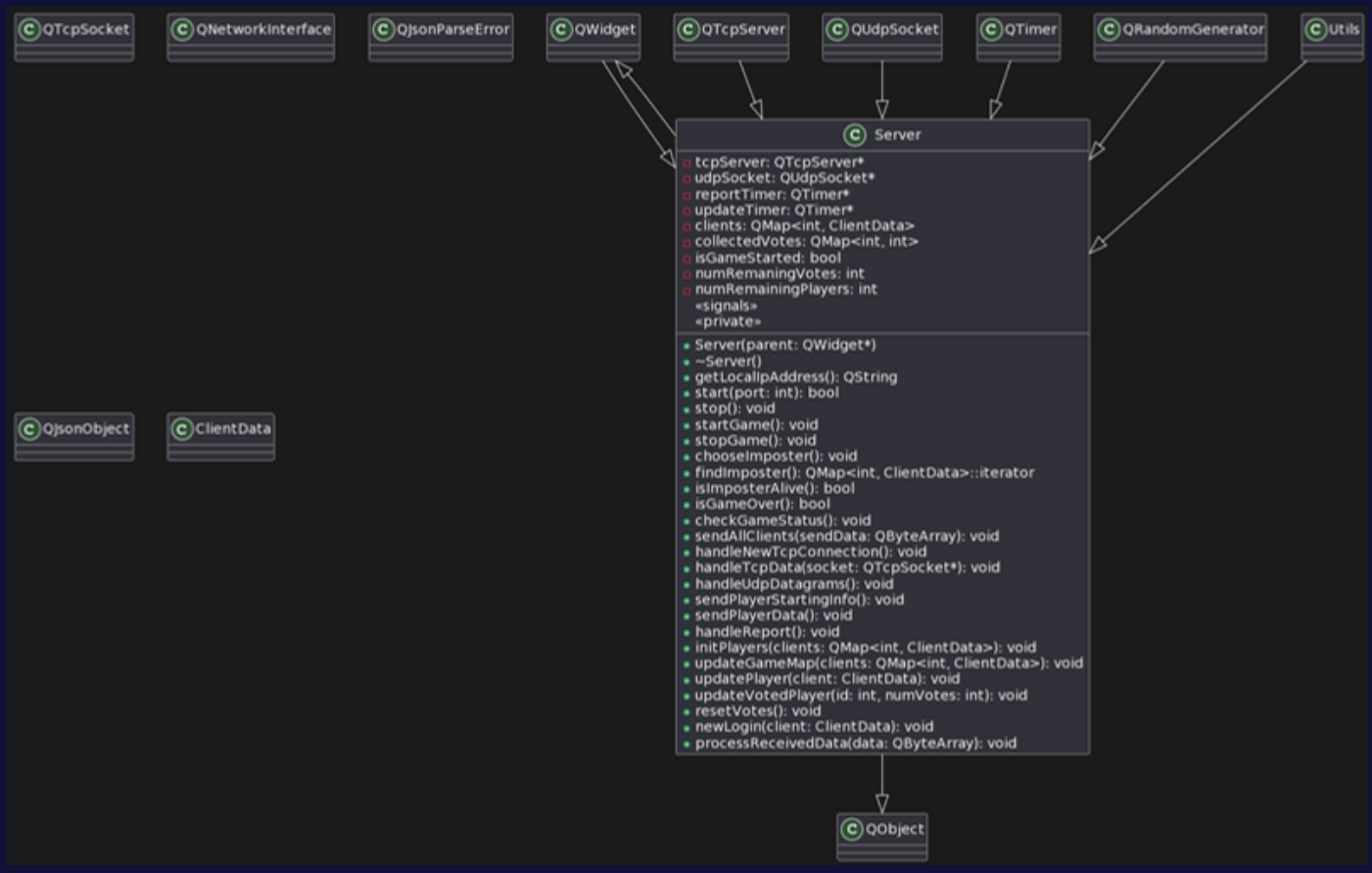
- Successful creation of all planned elements within the application.
- Implementation of the game server initialization process.
- Development of the functionality to showcase the game map.
- Integration of player status display within the application interface.



Displaying the game map with players' status

Game Server

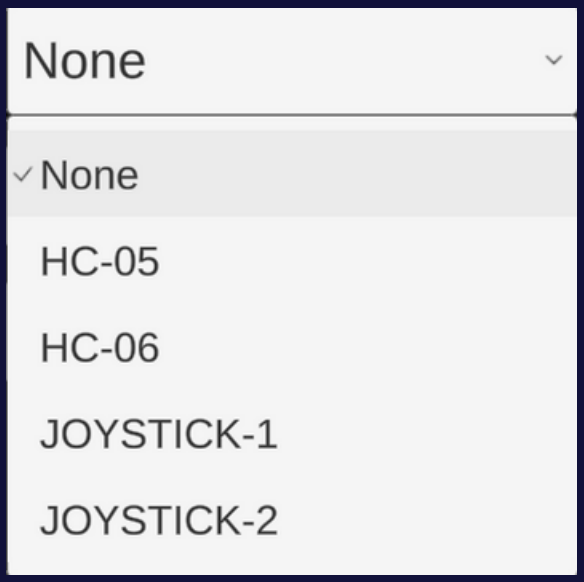
- Configuration of the server with two ports utilizing TCP and UDP protocols.
- Utilization of the TCP port for the game's initial login phase, enabling player registration, and assignment of unique IDs.
- Continued use of TCP within the game for actions like kill confirmations, task acknowledgments, and reporting.
- Leveraging the UDP protocol to distribute player data to all clients in each frame, ensuring accurate rendering within the Unity game environment.



Game Design

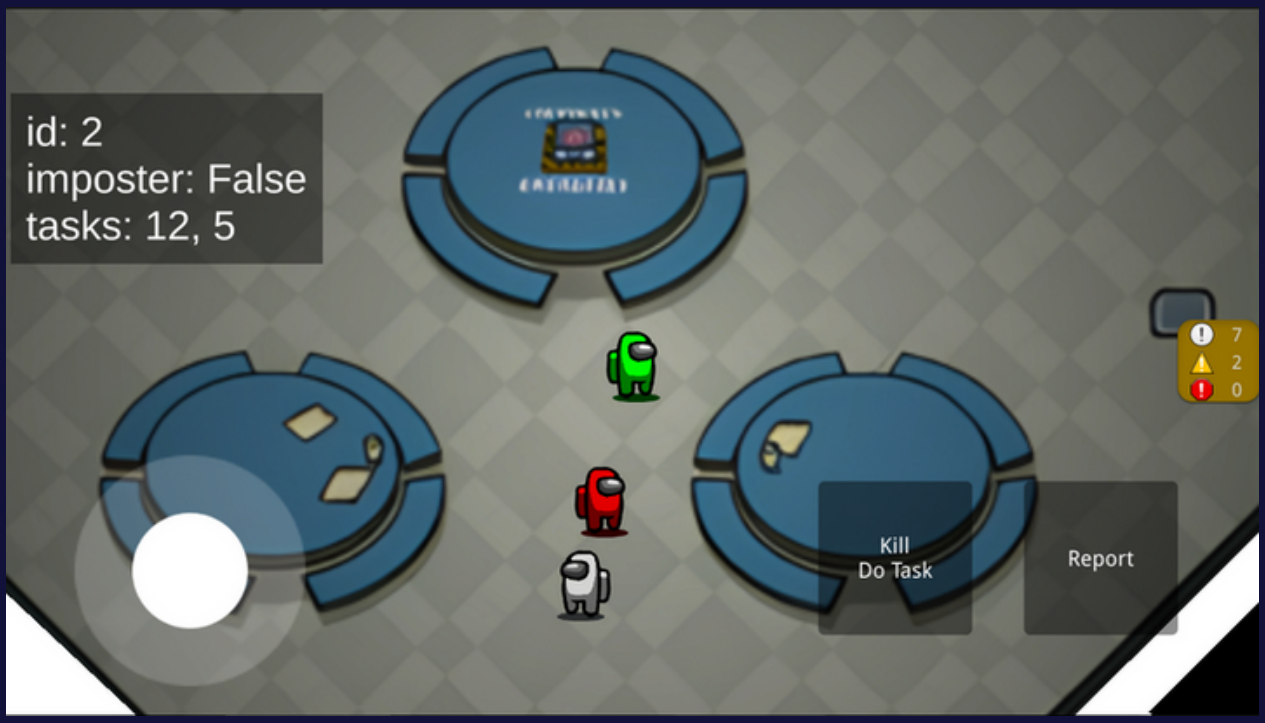
Login screen

The controller name, IP address and username are taken from the user. After filling in the information, the player presses the login button and waits for the game to start on the server side.



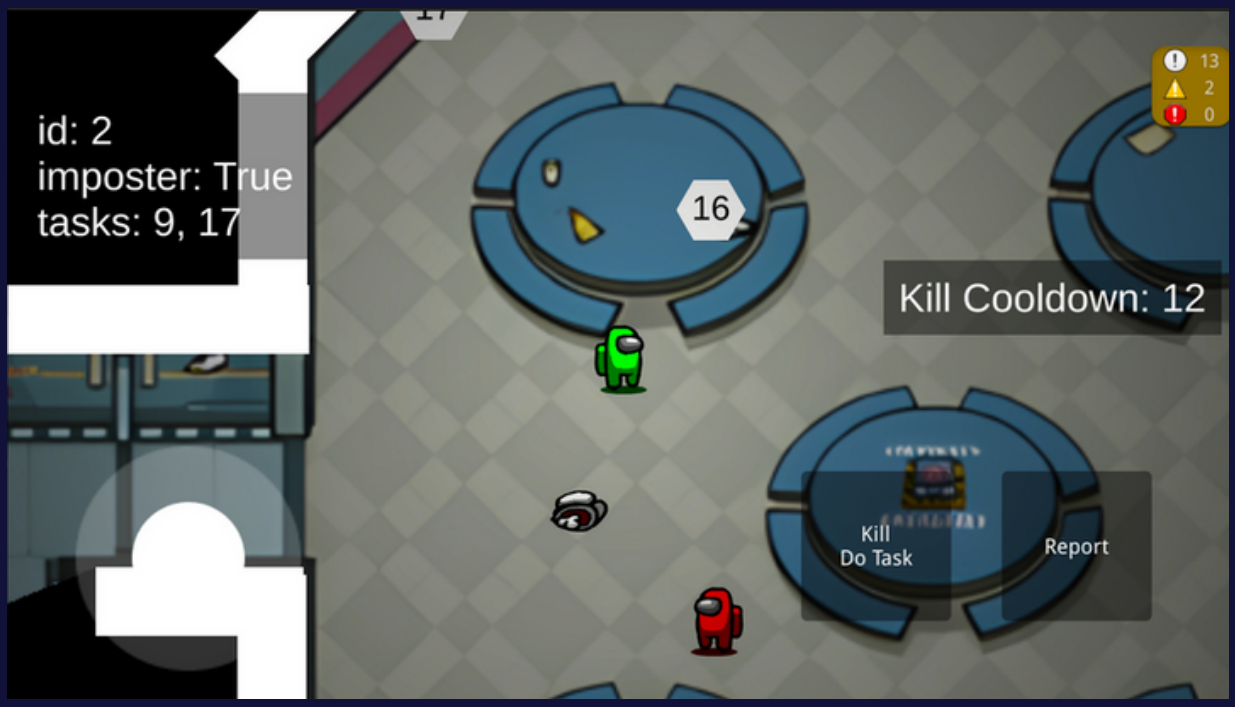
Controller names are predefined. If the user does not want to use controllers, they can select None and use the on-screen controls.

In-game - UI



Each user's ID, whether they are an imposter or not and the IDs of the tasks they have to perform are displayed on the screen. On screen controls consist of a joystick and two buttons. Killing and tasks are done with the same button. If the player's role is imposter, this button will be pressed when approaching a character to perform a kill action. If the player is not an imposter and is near a task, they can use this button to complete the task.

In-game - Killing



When the Impostor kills a player, the player turns into a ghost and leaves his body on the ground. The Impostor must wait 15 seconds before killing another player.

No one else can see the players who turn into ghosts.

In-game - Field of View

White player's field of view

Red player's field of view

Each player has his or her own field of vision. Anything outside the field of view appears as darkness, and the players under this darkness are not visible.

White player's field of view

Red player's field of view

If players want to see each other, they must be in each other's field of view.

In-game - Tasks

Hexagons with a number on them were used to indicate the location of the tasks on the map. Players with the Crewmate role must hover over these tasks and use the "Do Task" button. Which tasks they will do is determined at the beginning of the game and is written in the information box on the top left. There are currently 21 tasks on the map.

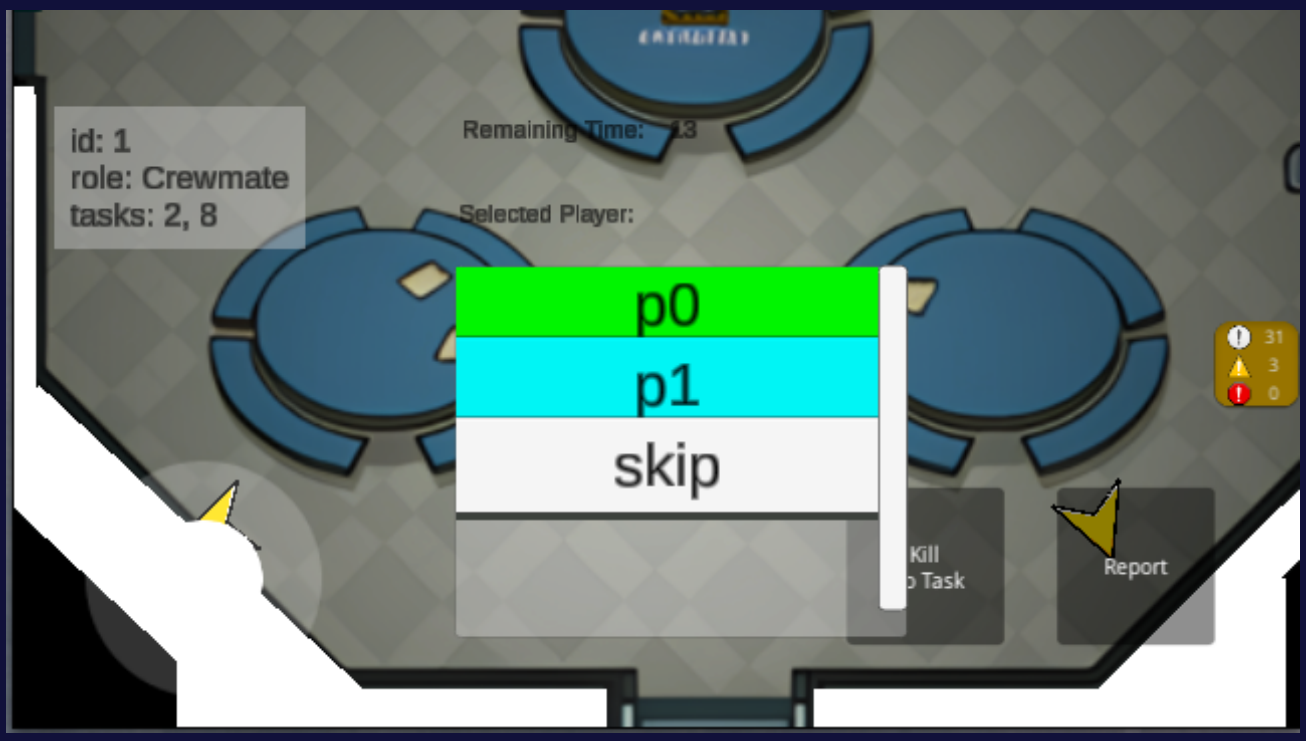
The player found the number 14 tag that had been assigned to him.

In-game - Tasks

The player can follow the arrows on the screen to find the task location.

When the player is close enough to the task location, an arrow different from the direction indicators will appear above the task location. The player can complete the mission by approaching it.

In-game - Reporting



Players can report when they find a dead body on the ground. In the pop-up window they vote for the player they think is an imposter. The player with the most votes is killed. If no player receives the majority of votes, no one is killed. After each voting the players are returned to the starting point.

End game - Crewmate Win

Crewmates win the game if they can finish their mission before the imposter kills them.

If the crewmates can correctly identify the imposter and kill it in the vote, they win the game.

End game - Imposter Win

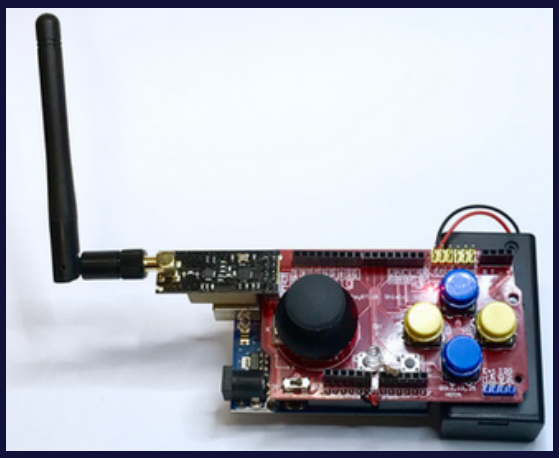
When Imposter and Crewmate are equal, Imposter wins the game. The end of game screen shows who the Imposter is.

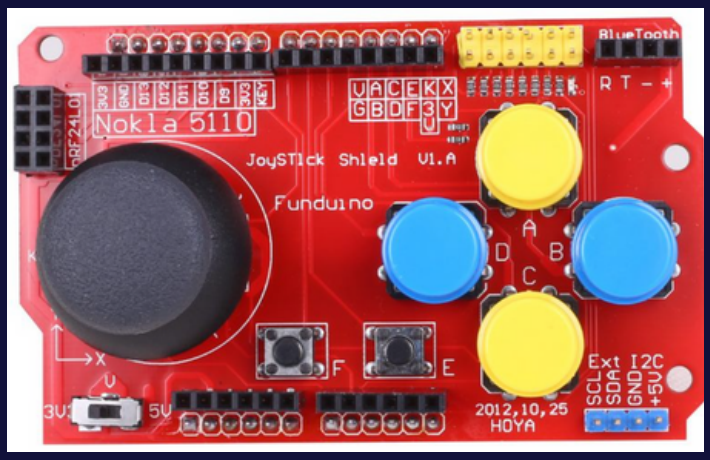
In-game - Task Locations on Map

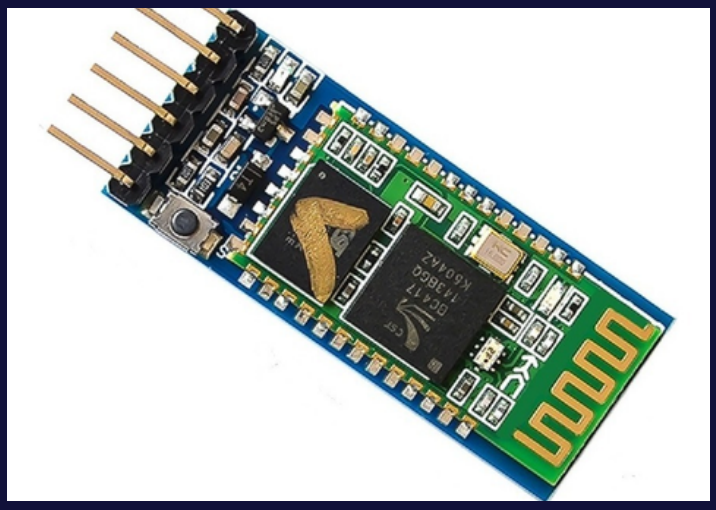
Hardware

Materials:

- Ardunio Uno
- Fundinio Joystick Shield v1.A
- Bluetooth Module(HC-05, HC-06)
- 9v Battery, 9v Battery Adapter







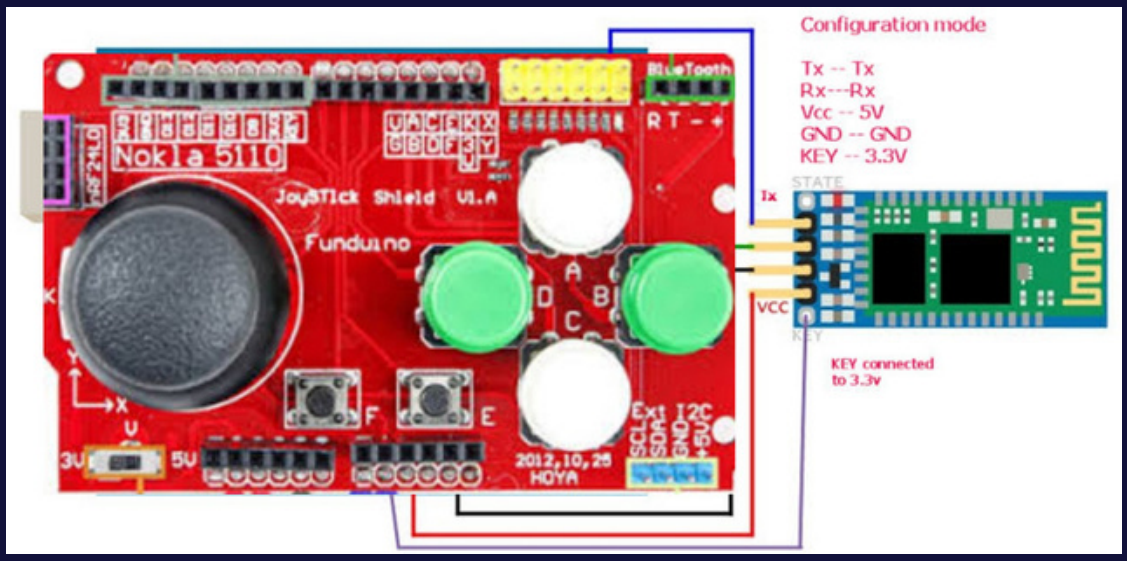
Joystick - Game Controller-HC05

- Consturction of Joystick: Adding Joystick shield on Ardunio UNO

- Setting up: Connecting pins of Joystick and Ardunio, reading meaningful output from Serial Monitor

- Adding Wireless Communication: Adding Bluetooth Module to existing hardware(HC-05 or HC-06), making necessary connections

- Setting Wireless Communication: Adding Bluetooth library into software. Testing and reading the data coming from Joystick in Android phone.



HC-05 Joystick connection scheme

- Connection with Unity Mobile App: Implementing Bluetooth plugin into mobile app. Reading the data coming from bluetooth in mobile side

- Implementing and Processing Data in Mobile App: Meaningful parsing of data to read every button and analog in every message. Parsing this coming data at every iteration. Moving and acting according to this parsed data.

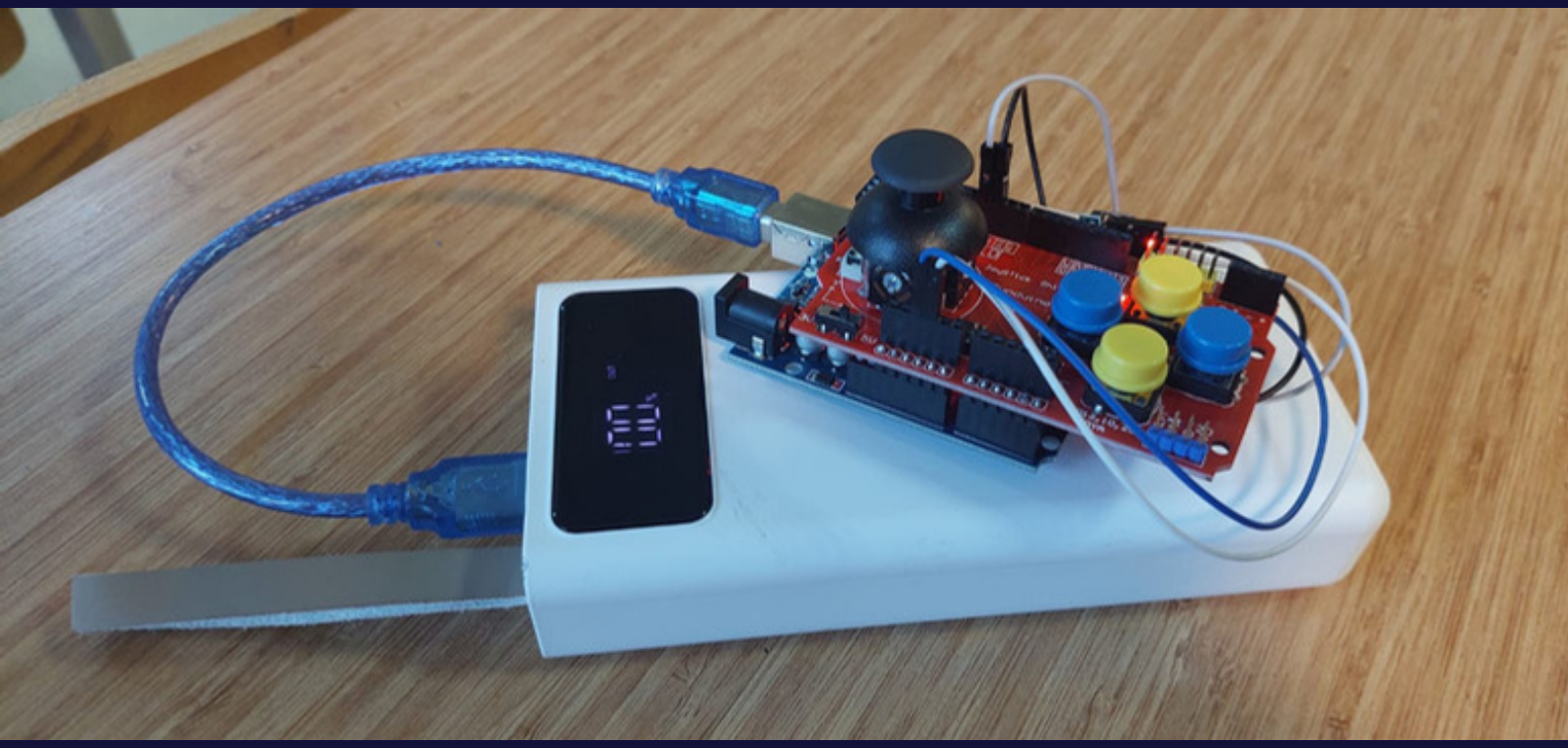
```
if (deviceName != null && deviceName != "")
{
    if (!IsConnected) {
        deviceName = deviceName.Trim();
        Debug.Log("deviceName ->" + deviceName.ToString());

        BluetoothService.CreateBluetoothObject();
        IsConnected = BluetoothService.StartBluetoothConnection(deviceName);
        Debug.Log("FixedUpdate IsConnected ->" + IsConnected.ToString());
    }
```

connection to bluetooth module in mobile app

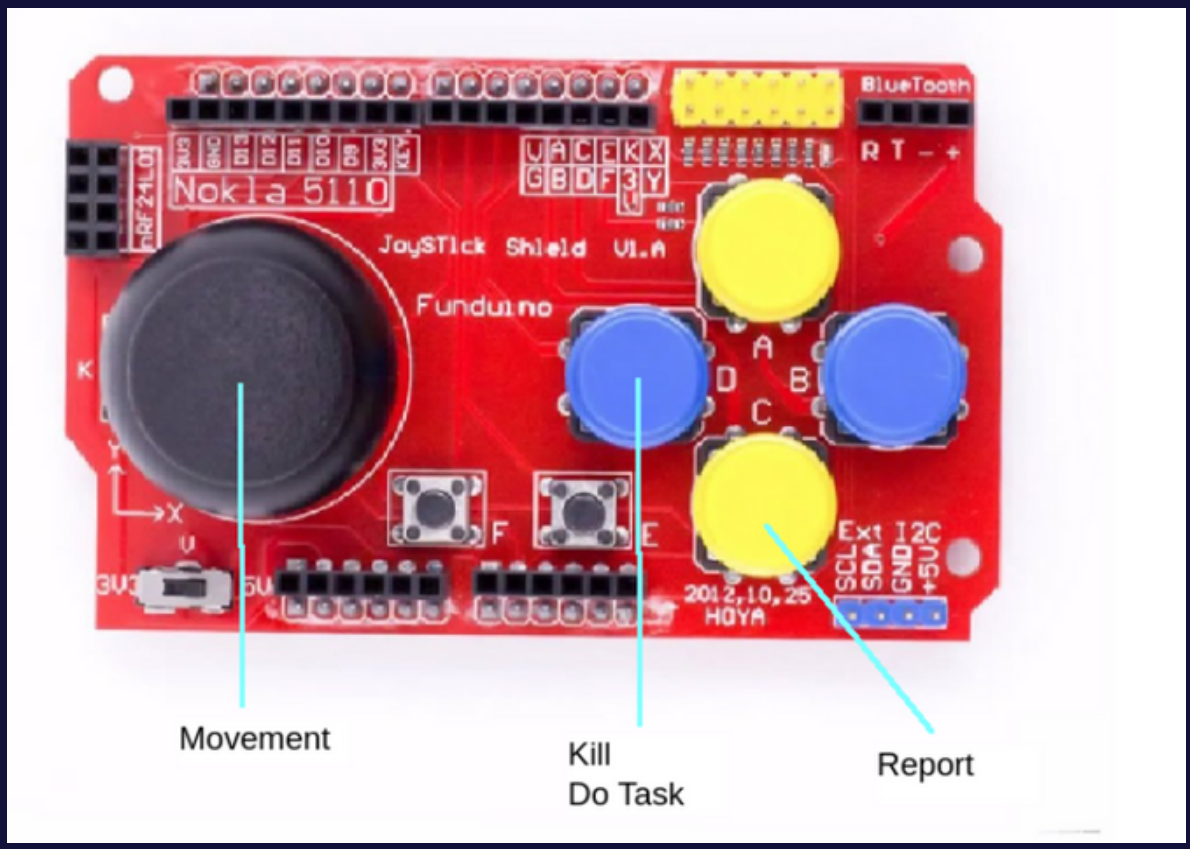
Challenges

We had a big issue that bluetooth module randomly stops after connection established with Unity Mobile app. We spend a lot of time to investigate this issue. We firstly focused on connections. Because the module is randomly stops and restarts at middle of communication. But there wasn't problematic or false connected cable in design. Then we thought it could be software related. We investigate the parsing and processing of data in Mobile side. But there was no big problem that we saw on code. However, we keep check the code and found some little issues that may lead to our problem.

We did the necessary changes but there was no improvement. Finally we think about power consumption of shield and bluetooth module. We find out there is switch on joystick that selects between 3.3v to 5v. This means joystick uses 3.3v or 5v power from ardunio and all of them was 5v. Then we figure it out we also trying to use 5v pin with bluetooth module and this was the main issue. Because both joystick sheild and bluetooth module was trying to use 5v pin. This leads unsuefficient power consumption for bluetooth module so it explain us why module stops middle of communication. Then we switched the joystick shield to 3.3v so 5v pin was free and was there only for powering to bluetooth module. Our tests has no problem after this solution.



Joystick - Game Controller
powered by a powerbank for
Bluetooth Communication



Joystick Keymap

Conclusion and Evaluation

This project aims to deliver a multiplayer experience similar to Among Us. Upon successful completion, the game is intended to garner attention and enjoyment from players. Each project phase will be carefully managed to achieve success.