CPIT 490

(**Arduino**)

Instructor:

Dr. Adil Khadidos

**Final group project report**

**Group members:**

|  |  |
| --- | --- |
| **Name** | **KAU ID** |
| Faris Al-Nahari | 1846765 |
| Ammar Al-Bar | 1847572 |
| Abdulmajeed Alahmadi | 1845390 |
| Ahmed Al-Osaimi | 1845361 |
| Adel Mansho | 1847026 |

Table of content

[**Introduction:** 3](#_Toc57633860)

[**Tasks:** 4](#_Toc57633861)

[**Diagrams:** 6](#_Toc57633862)

[**Tools and Implementation:** 9](#_Toc57633863)

# **Introduction:**

Among Us is video game about a group of individuals who are working on a broken ship and they have certain tasks to do to succeed in fixing the ship and win.

all good right? Well no, the only problem is that there is an imposter among them who does not want the ship fixed and his only goal is to sabotage and stop players.

Our main project idea is to use what we have learned about microcontrollers and sensors to take the game to another level and implement it in real life.

**Game workflow**:

There will be 5 players [One of them will be randomly chosen to be the imposter], each player will be carrying an ESP8266, RFID Scanner, and a card with a button [See diagram 1, last page].

* Why?
  + RFID Scanner will be used to either scan the tasks RFID or disable players [Disabling Players is authorized only if you’re an imposter]
  + RFID card, so that the imposter knows which player to disable
  + Button for reporting

There’s also a Mobile application that all players and the admin have to use.

It will have the following functionality.

Admin page:

* Start or reset the game
* View the status of the players and who is the imposter
* Reset meeting, resume game

Player page:

* Check whether the player the imposter or not once the game starts
* See the shared tasks bar with all players
* See the location of his tasks
* Display Tasks once a Card / FOB is scanned

# **Tasks:**

We have implemented 5 tasks in total, which are distributed into 20 different locations. So, each crewmate can do his tasks in different location than the other 3 players. This is done to balance the game, and give the imposter a chance to disable players without increasing his chance of being spotted by other crewmates.

We thought distributing the tasks across multiple locations increases intensity of the game and evens out the playing field so crewmates don’t have an advantage over imposters.

3 of the tasks will be physical, 2 are completely in the application.

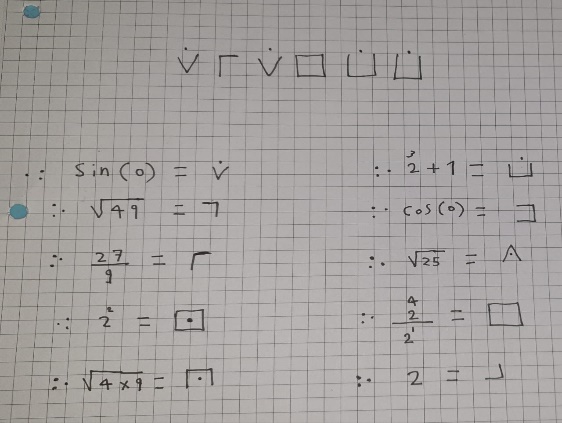
All tasks will have an RFID card that the player can scan so the application can load the appropriate screen related to the task.

* First task:

[Physical] Guess the tone. Once the player reaches the location of the task there will be an Arduino with a button and a buzzer [See diagram 2].

The player has to scan the code of the task, press the physical button then a song of certain game will play [Mario game] the player has to guess it and put the answer in the application.

* Second task:

[Physical] Decrypt the code. The main idea of this task is decipherment, the player is asked to solve a problem to get the password that he has to enter in the application for doing this task , in the lab the player will find a paper that include a symbols and he is asked to decode the symbols into a number (instruction will be provided in the paper as it shown) , after decoding and getting the number the player have to enter the value into the Arduino , a keypad has been placed for this purpose , if the player entered the correct code a green led will be turned on and he will get the password to be entered in the application , but if the value was wrong a red led will be turned on and he must wait 5 minutes before he gain access to enter another code , also keep in mind that the imposter can sabotage this task by entering a special code for the imposter , a yellow led will be turned on and it indicate that an imposter has sabotaged this task and the player have to wait till it goes off ( it will take 30 seconds) .

[See diagram 3, last page].

* Third task:

[Physical] Joystick collaboration. The idea of this task is very simple, we’re using two joysticks to complete the task, the user needs to hold both joysticks and tries to navigate both of them in a specific direction that we provide for him. After he manages to do the task correctly, a password will be displayed on the LCD. By writing that password on the application the task will be completed and he will process to the next task or he will be finished with all of them.

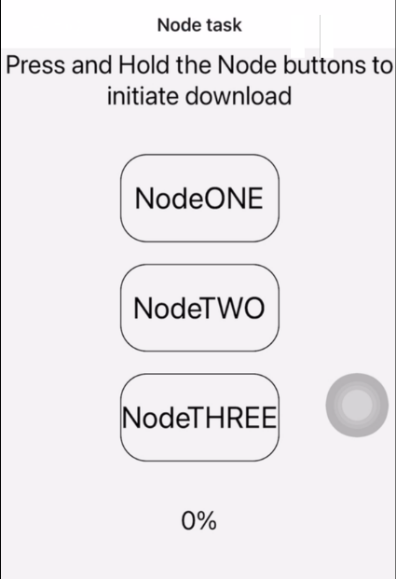
[See diagram 4].

* صورة تحتوي على إلكترونيات, لوحة المفاتيح

  تم إنشاء الوصف تلقائياًFourth task:

Password typing. The simplest task. The application will generate a random password and the player has to re-enter it.

* Fifth task:

Data download. The player activates the three nodes, by long pressing on each one of them until it turns green. Then the download is initiated and after the download is completes the player is navigated back to the home page.

# **Diagrams:**

In this section we will put all the diagrams to the devices and how it will be connected.

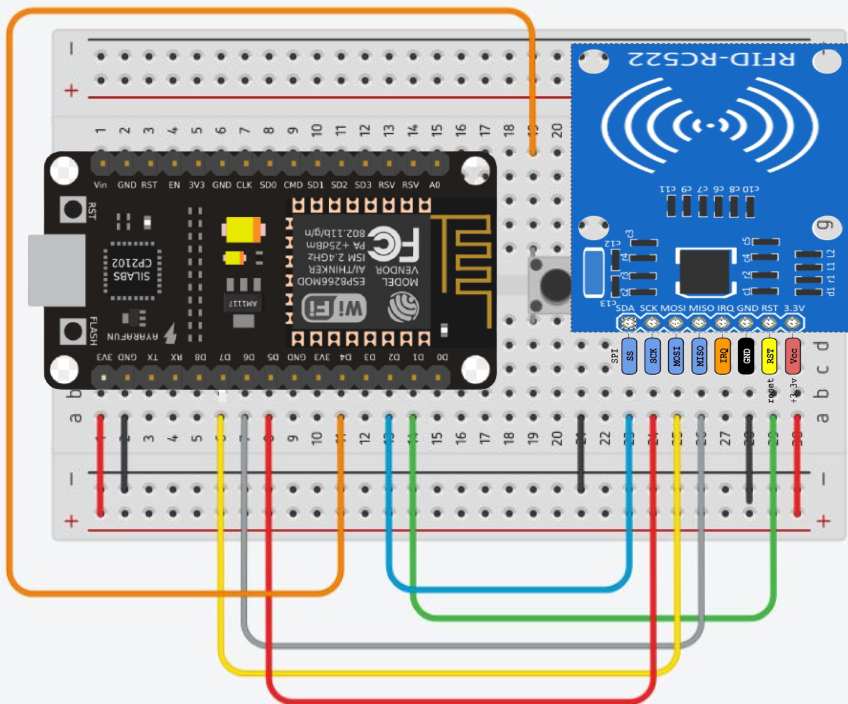
* Diagram 1

Diagram 2:

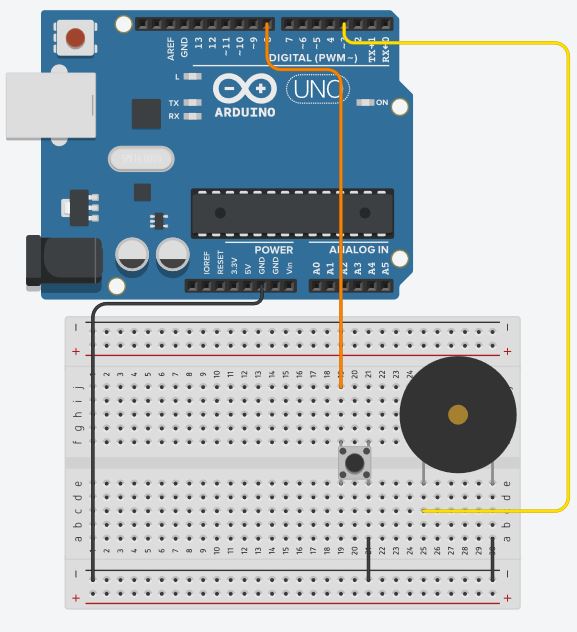


Diagram 3:

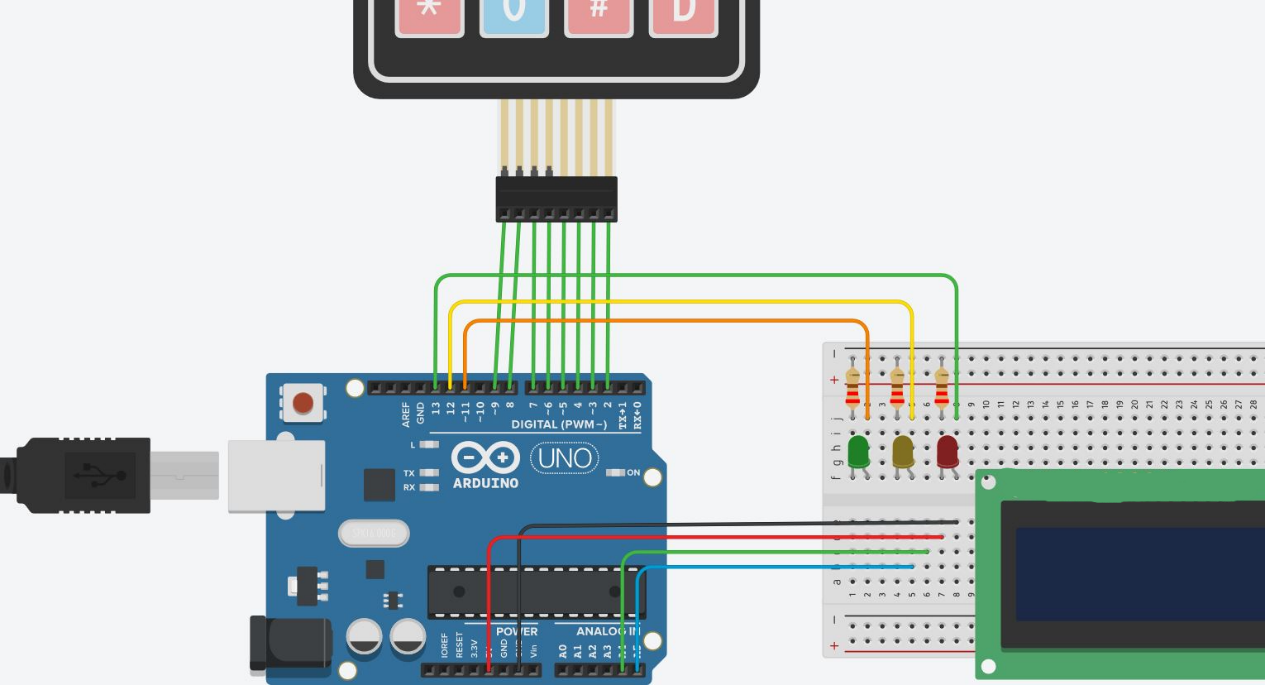
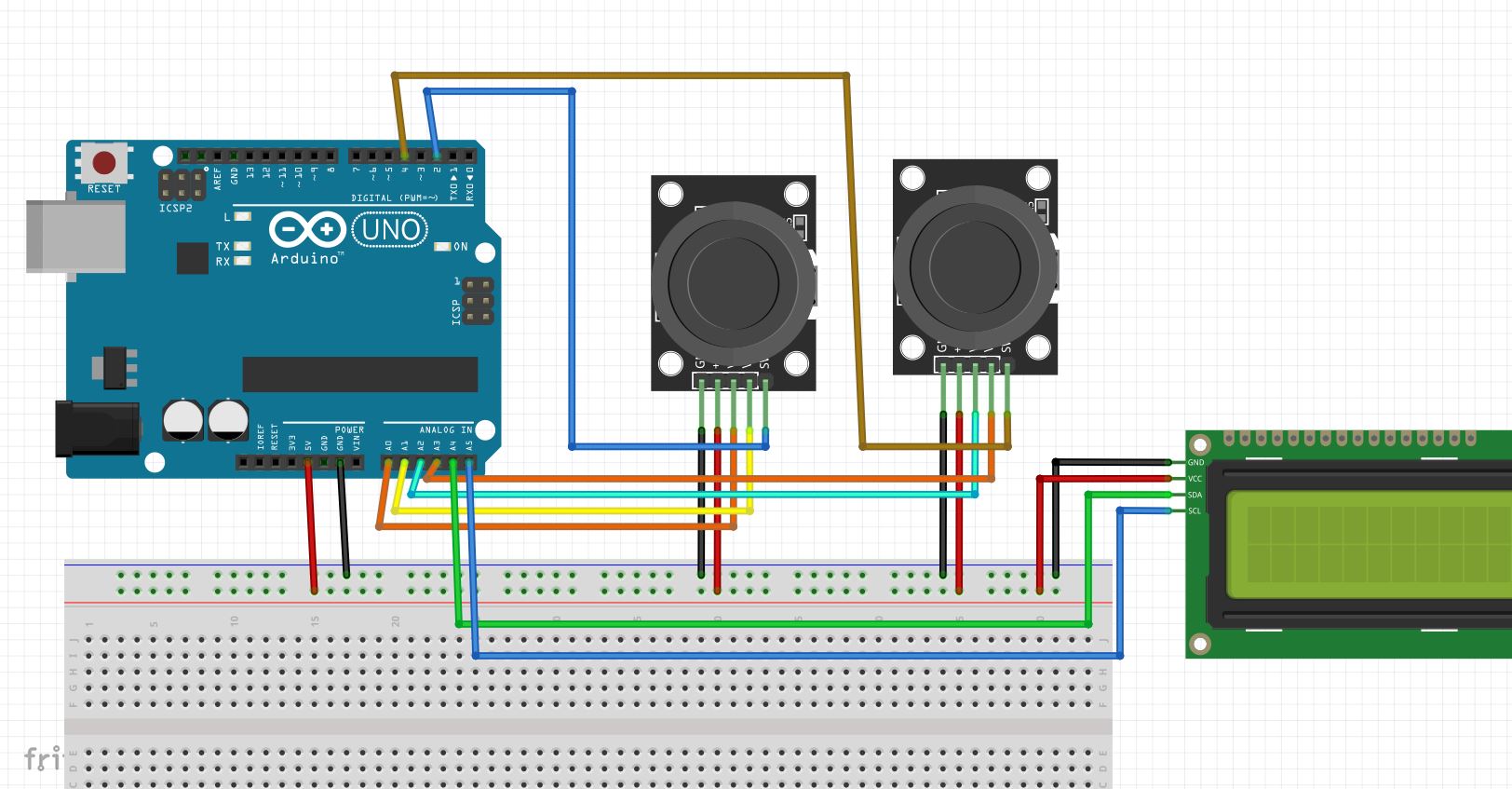


Diagram 4:



# **Tools and Implementation:**

* All the related codes will be attached with the report.

For this project, we used three main components to deliver the physical implementation of Among Us:

1. NodeMCU ESP8266
2. Firebase
3. React Native

NodeMCU ESP8266:

Our main usage of the NodeMCU ESP8266 relies in its portability, TCP/IP Protocol Stack, and the tools we can use with it to implement a physical version of the Popular Among Us Game. Using an RFID Scanner, the player can scan cards / FOBs which are distributed across multiple locations, with each card / FOB representing a specific task, as well as disable players and prevent them from further participation in the game session.

Firebase:

Due to its popularity, ease of use, accessibility, and gearing towards mobile platforms, we have chosen Firebase as our de facto backend infrastructure. Session information is kept in the cloud, this includes Game Settings, Player Information, Admin Information, and Tasks. Using Firebase and its Realtime database, we were able to play the game and update information related to the session (e.g. Number of tasks completed) effortlessly and smoothly, without impacting the flow of play.

React Native:

React Native was chosen as a Framework for this project mainly due to its cross-platform native deployment, as well as its incredible community support and state-of-the-art state management system, which renders elements on the page and updates them whenever changes are made to the database (i.e. Firebase). Due its state-focused design, React Native was the most suitable candidate for this project as it is able to immediately reflect the changes made to the Database on the User Interface.