**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KỸ THUẬT MÁY TÍNH**

**ĐỖ VĂN KHẢI**

**NGUYỄN THỊ THANH THỦY**

**PHẠM NGUYỄN THIỆN MINH**

**ĐỒ ÁN MÔN HỌC**

**GAME BẮN TÀU**

**TP. HỒ CHÍ MINH, 2018**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KỸ THUẬT MÁY TÍNH**

**ĐỖ VĂN KHẢI – 15520331**

**NGUYỄN THỊ THANH THỦY - 15520865**

**PHẠM NGUYỄN THIỆN MINH - 15520491**

**ĐỒ ÁN MÔN HỌC**

**GAME BẮN TÀU**

**GIẢNG VIÊN HƯỚNG DẪN**

**TRẦN NGỌC ĐỨC**

**TP. HỒ CHÍ MINH, 2018**

MỤC LỤC

[Chương 1. TỔNG QUAN 8](#_Toc533091939)

[1.1. Giới thiệu về game bắn tàu 8](#_Toc533091940)

[1.2. Phân công công việc. 9](#_Toc533091941)

[1.3. Vi điều khiển NUC140 10](#_Toc533091942)

[1.3.1. Sơ lượt chung về NUC140 10](#_Toc533091943)

[1.3.2. Đặc tính nổi bật NUC140 10](#_Toc533091944)

[1.4. Module wifi ESP 8266 v12 11](#_Toc533091945)

[1.4.1. Sơ lượt chung về ESP8266 v12 11](#_Toc533091946)

[1.4.2. Sơ lượt về tập lệnh AT Command ESP8266 v12 12](#_Toc533091947)

[Chương 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 14](#_Toc533091948)

[2.1. Phân thích mô tả và thiết kế phần cứng cho gamepad 14](#_Toc533091949)

[2.2. Phân thích mô tả và thiết kế phần server AWS. 19](#_Toc533091950)

[2.2.1. Giao diện hệ thống đăng nhập và đăng kí. 19](#_Toc533091951)

[2.2.2. Cơ sở dữ liệu để lưu trữ thông tin người dùng. 22](#_Toc533091952)

[2.2.2.1. Giới thiệu về MySQL 22](#_Toc533091953)

[2.2.2.2. Cách tạo một database đơn giản với MySQL 23](#_Toc533091954)

[2.2.3. Sơ đồ chức năng hệ thống đăng nhập và đăng kí 24](#_Toc533091955)

[Chương 3. KẾT QUẢ THỰC NGHIỆM 24](#_Toc533091956)

[3.1. Tính năng đăng nhập vào hệ thống. 24](#_Toc533091957)

[3.1.1. Các trường hợp thử kiểm tra. 24](#_Toc533091958)

[3.1.2. Video kiểm tra thực tế các trường hợp trên 25](#_Toc533091959)

[Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 25](#_Toc533091960)

DANH MỤC HÌNH VẼ

[Hình 1.1: Ví dụ đặt tàu hợp lệ 10](#_Toc533016143)

[Hình 2.2: Tên hình 1 **Error! Bookmark not defined.**](#_Toc533016144)

DANH MỤC BẢNG

[Bảng 1.1: Tên bảng 1 14](#_Toc367742567)

[Bảng 2.1: Tên bảng 1 **Error! Bookmark not defined.**](#_Toc367742568)

DANH MỤC TỪ VIẾT TẮT

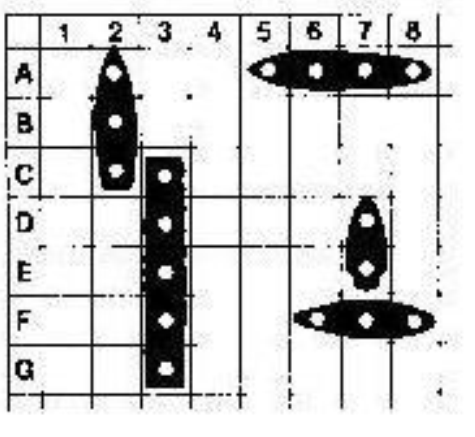
# TỔNG QUAN

## Giới thiệu về game bắn tàu

Hai người chơi truy cập vào trang web trò chơi tiếp hành đăng nhập tài khoản đã đăng kí để vào trò chơi. Sau đó tiến hành đặt 2 bản đồ sao cho không ai có thể nhìn thấy bản đồ của đối phương. Bí mật đặt 5 tàu chiến của bạn trong vùng biển của chính mình. Đối phương làm tương tự.

**Luật đặt tàu chiến:**

* Đặt tàu theo vị trí ngang hoặc dọc, không được chéo.
* Không được đặt các tàu chồng lên nhau để mà nó bị trùng số, chữ.
* Không được thay đổi vị trí của tàu một khi trò chơi bắt đầu.



Hình 1.1: Ví dụ đặt tàu hợp lệ

**Cách chơi**

Dùng tay cầm được thiết kế sẵn kết nối vào hệ thống để chọn phòng rồi chọn tay cầm để vào xếp tàu tiến hành chơi bắn tàu. Ai kết nối trước sẽ được bắn trước.

**Bắn trúng!**

Nếu bạn bắn trúng tàu đối phương thì thiết bị tay cầm bắn trúng rung mạnh, tay cầm người bị bắn trúng rung nhẹ hơn. Và tàu bị bắn trúng xuất hiện một viên đạn mà đỏ tại vị trí bắn trúng. Và tiếp tục lượt bắn đến khi bắn trật tàu của đối phương.

**Bắn trật!**

Tương tự bạn bắn trúng nhưng tay cầm sẽ không rung. Và có viên đạn màu đen tại vị trí bắn trật. Bắn trật sẽ mất lượt

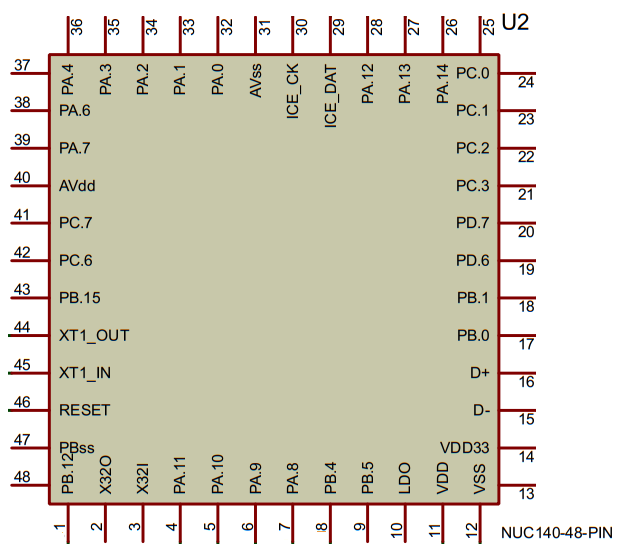
Ngoài ra trò chơi còn chế độ sau 30s người tới lượt không bắn thì mất lượt.

**Chiến thắng!**Bạn chiến thắng khi đã đánh chìm tất cả 5 chiến hạm của đối phương.

## Phân công công việc.

## Vi điều khiển NUC140

### Sơ lượt chung về NUC140



Hình 1.2: Sơ đồ chân NUC140

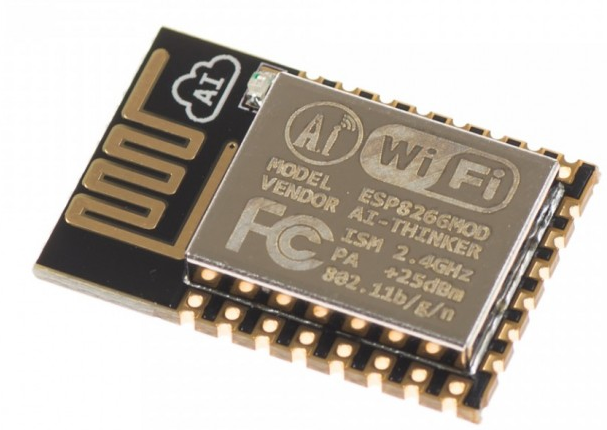
NUC140 là sản phẩm của hang Nuvoton là công ty con được tách ra từ Tập đoàn Điện tử Winbond – một hãng điện tử bán dẫn đứng hàng đầu Đài Loan. Hãng này có 3 dòng chip vi điều khiển (MCU) 4-bit, 8-bit và 32-bit (ARM Cortex). Dòng vi điều khiển ARM Cortex-M được thiết kế nhúng tối ưu hóa cho các ứng dụng vi xử lý MCU. Dòng ARM Cortex-M0 là dòng vi điều khiển lõi ARM có kích thước nhỏ nhất, tiêu thụ điện năng thấp nhất và có kiến trúc được sắp xếp hợp lý tương thích với việc sử dụng tools nạp của các hãng khác để phát triển các ứng dụng

### Đặc tính nổi bật NUC140

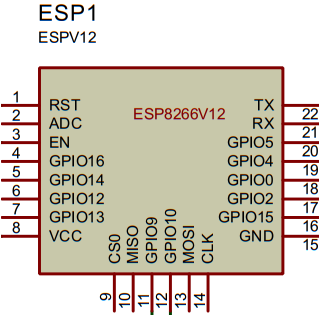
* NUC140 là vi điều khiển 32-bit lõi ARM Cortex-M0, trình đơn phần cứng 32 bit, chạy lên tới 50MHz.
* Có 4 mức ưu tiên ngắt đầu vào, có 128 KB flash ROM cho bộ nhớ chương trình. 16KB SRAM, 4KB bộ nhớ flash cho nạp chương trình trong hệ thống.
* Giao tiếp thiết bị ngoại vi: 8 kênh 12bit ADC, UART nối tiếp tốc độ cao, SPI lên đến 32MHz, I2C lên đến 1MHz; kết nối thiết bị ngoại vi USB 2.0, CAN, LIN…
* Thiết bị ngoại vi có tính năng phong phú: PWM, RTC, bộ ngắt nhận dạng Brownout, GPIO, PDMA và 4 bộ Timer 32 bit.
* Dải điện áp hoạt động rộng từ 2,5V~5,5V, chống nhiễu tiếng ồn tốt, tích hợp dữ liệu flash, dao động thạch anh nội chính xác ±1% với nhiệt độ phòng, có khả năng bảo mật trên chip, điện áp reset lại mạch thấp.

## Module wifi ESP 8266 v12

### Sơ lượt chung về ESP8266 v12



Hình 1.3: Module wifi ESP 8266 v12



Hình 1.4: Sơ đồ chân module wifi ESP 8266 v12

* Tiêu chuẩn wifi : 802.11b/g/n, với tần số 2.4GHz,và hổ trợ bảo mật WPA/WPA2
* Mạch nhỏ, gọn (24.75mm x 14.5mm)
* Tích hợp 10-bit ADC
* Tích hợp giao thức TCP/IP (hiện tại thời điểm này hỗ trợ ipv4)
* Tích hợp năng lượng thấp 32-bit MCU
* SDIO 2.0, SPI, UART, I2C
* STBC, 1x1 MIMO, 2x1 MIMO
* Điện áp làm việc 3.3v
* Có các chế độ: AP, STA, AT + STA
* Bộ nhớ Flash: 4MB
* Lệnh AT rất đơn giản, dễ dàng sử dụng
* Lập trình trên các ngôn ngữ: C/C++, Micropython, NodeMCU – Lua

### Sơ lượt về tập lệnh AT Command ESP8266 v12

ESP8266 sử dụng tập lệnh AT Command của riêng nó để có thể giao tiếp và lập trình.

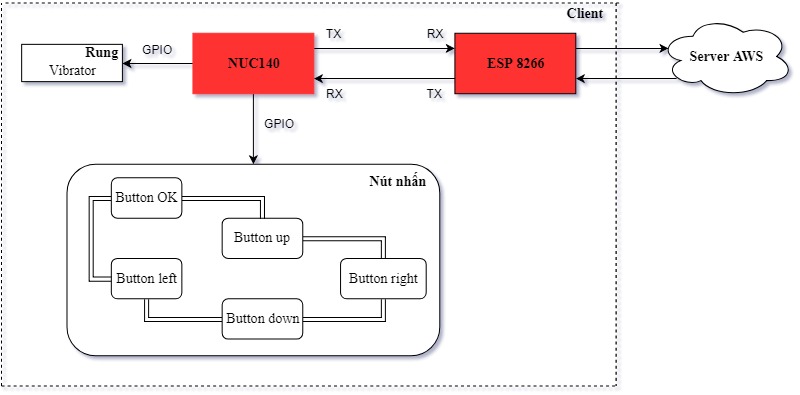
|  |  |  |
| --- | --- | --- |
| Lệnh AT Command | Mô tả chức năng | Ví dụ |
| AT+CWMODE = <mode> | Cài đặt chế độ  1 = Station  2 = Access Point  3 = Both | AT+CWMODE=1 |
| AT+CIPMUX = <mode> | Cài đặt số lượng các kênh kết nối  0 = 1 kênh kết nối  1 = Nhiều kênh kết nối | AT+CIPMUX=1 |
| AT+CWLAP | Truy vấn các mạng wifi có thể kết nối | AT+CWLAP |
| AT+CIPAPMAC? | Nhận địa chỉ mac của ESP8266 softAP. | AT+CIPAPMAC? |
| AT+CWJAP= <ssid>, <password> | Kêt nối một mạng wifi với SSID và password | AT+CWJAP = "abc", "5678" |
| AT+CIPSTART=  type, addr, port | Kết nối vào 1 TCP/UDP server của 1 server trên internet | AT+CIPSTART=  "TCP","34.205.32.160",  3333 |
| AT+CIPSEND=length | Dùng để gửi dữ liệu có độ dài length | AT+CIPSEND=1 |

Bảng 1.1: Một số lệnh AT Command cho ESP8266

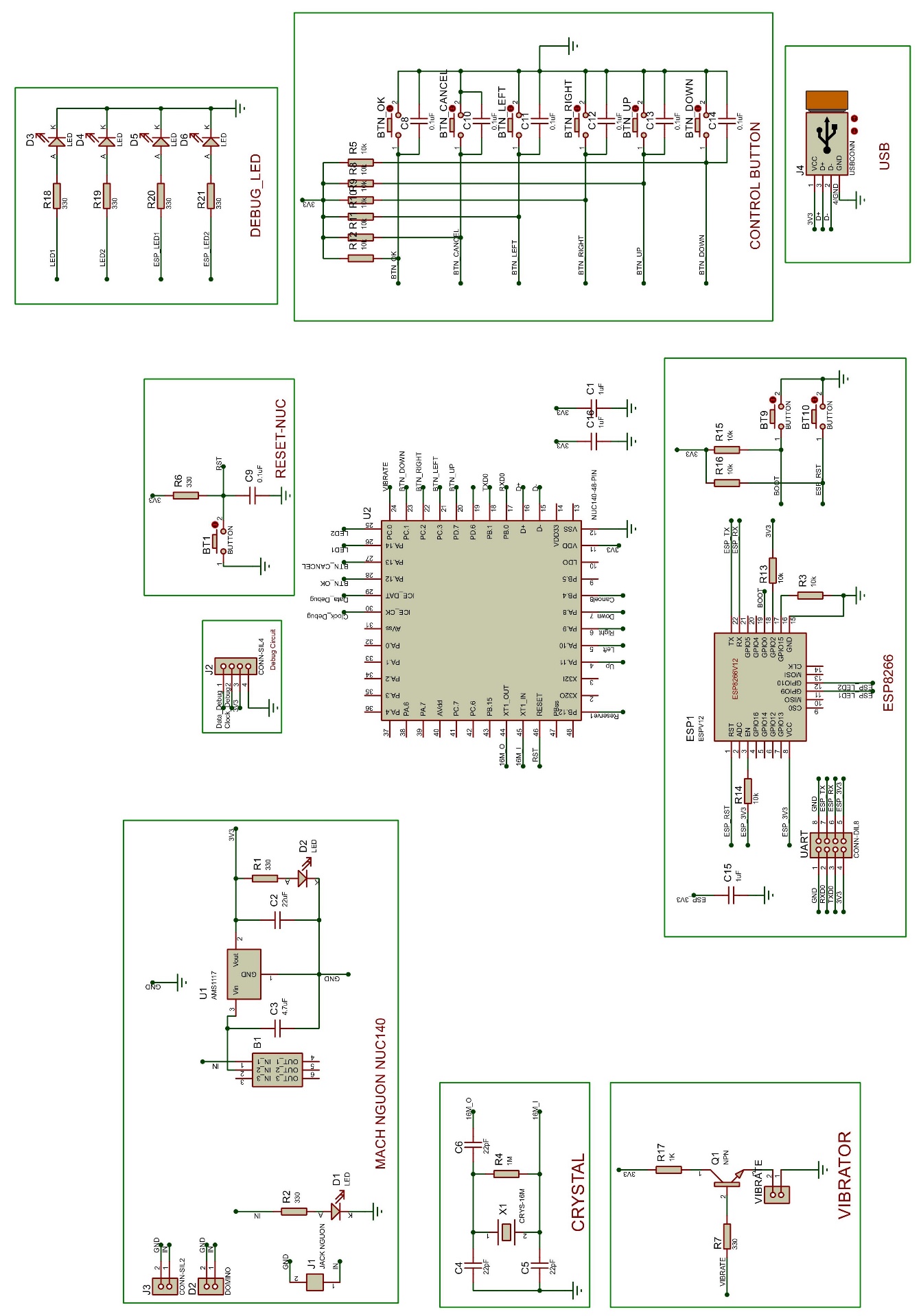
# PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## Phân thích mô tả và thiết kế phần cứng cho gamepad

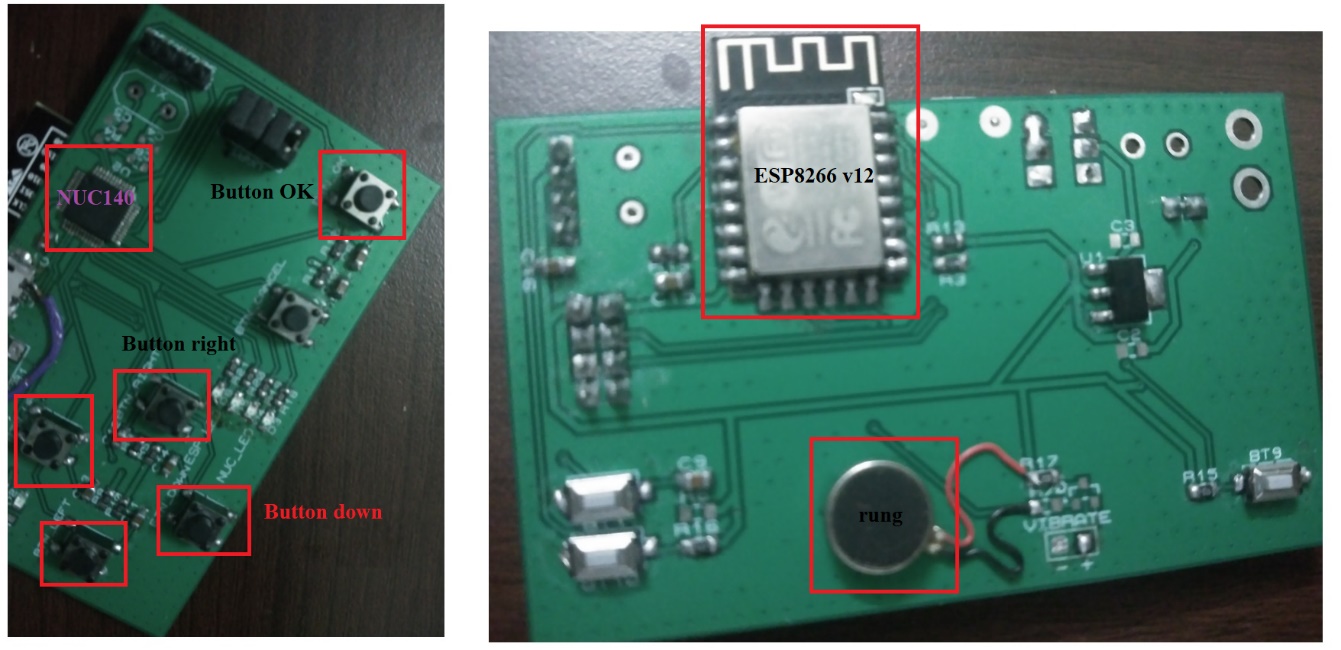
Tay cầm gọi chung là thiết bị phía client gồm có 4 nút nhấn để di chuyển lên (button up), xuống (button down), trái (button left) và phải (button right) như hình mô tả bên dưới. Cùng với một nút nhấn khác để thực hiện thao tác bắn tàu (button ok) của đối phương. Mạch bao gồm NUC140 kết nối với ESP8266 v12 thông qua UART. NUC140 là xử lý trung tâm điều khiển các thao tác di chuyển, nhận tín hiệu từ server truyền về mỗi khi bắn trúng để rung gamepad đồng thời NUC140 gửi tín hiệu để ESP8266 kết nối đến server và nhận tín hiệu từ server truyền về thông qua ESP8266.



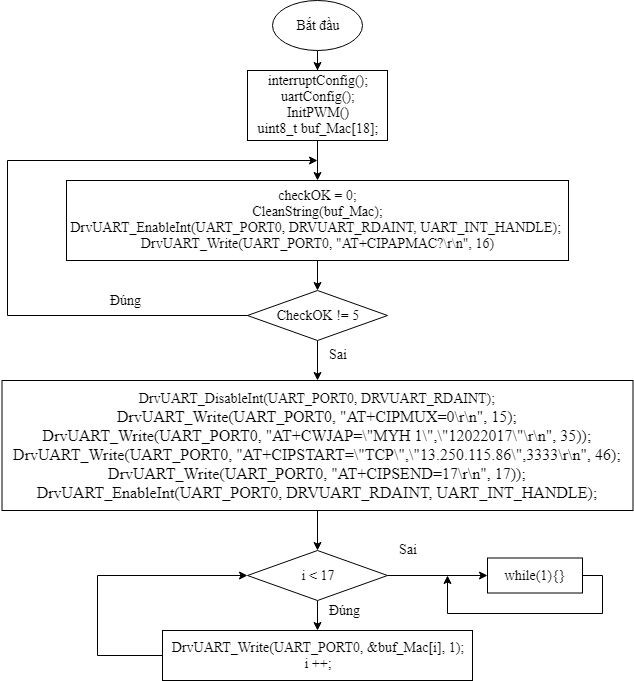
Hình 2.1: Sơ đồ khối chức năng của gamepad

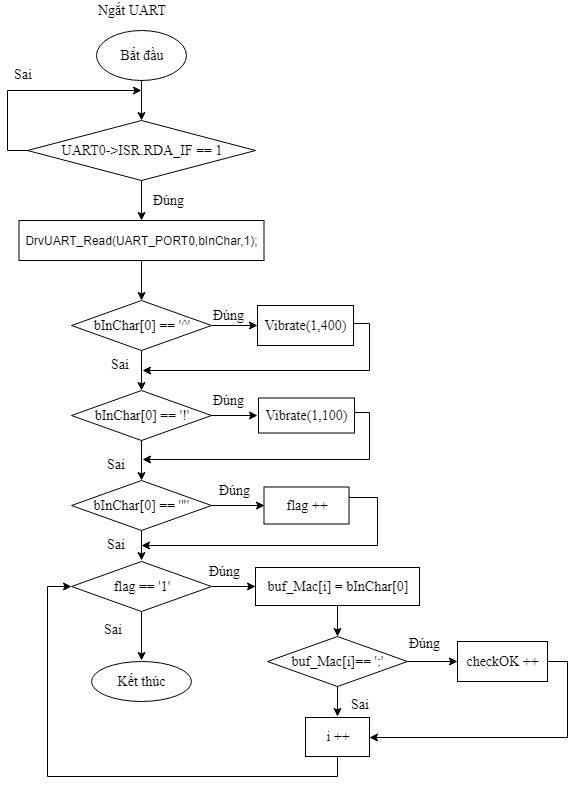


Hình 2.2: Mạch nguyên lý của gamepad

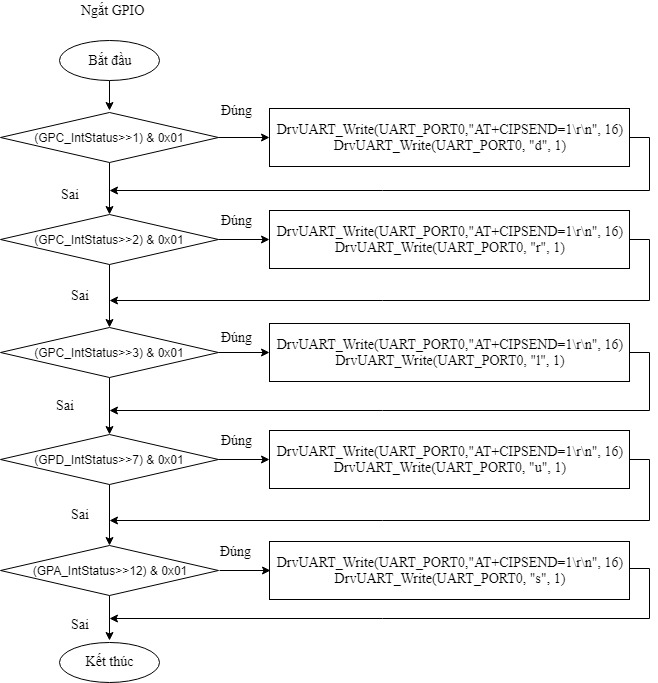


Hình 2.3: Hình ảnh gamead thực tế

Phân tích thiết kế phần mềm cho gamepad. Hình 2.3: Sơ đồ thuật toán tay cầm



Hình 2.3: Sơ đồ giải thuậ hàm ngắt UART



Hình 2.3: Sơ đồ giải thuậ hàm ngắt GPIO

## Phân thích mô tả và thiết kế phần server AWS.

### Giao diện hệ thống đăng nhập và đăng kí.

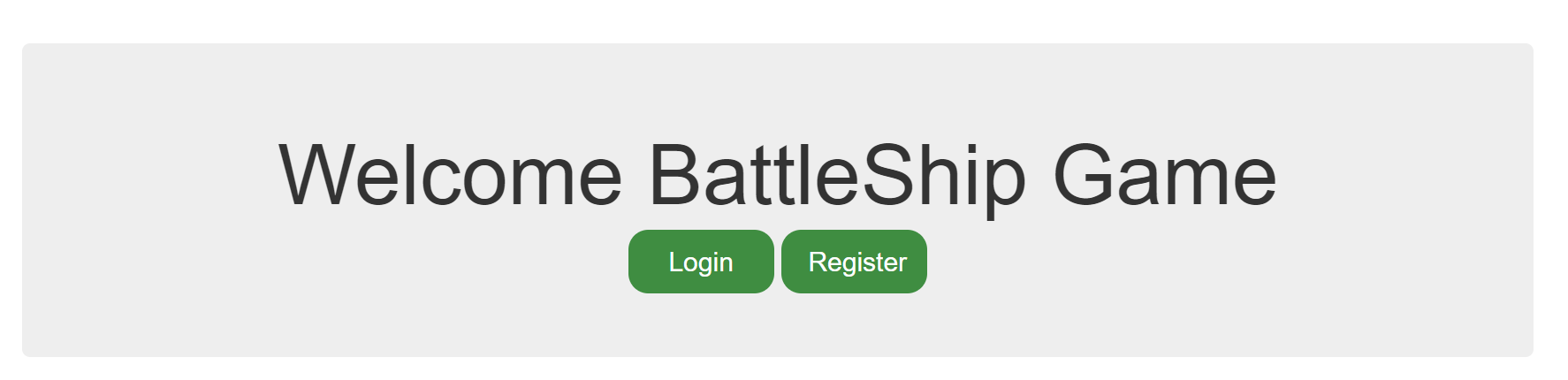
#### Giới thiệu về bootstrap

Bootstrap là một nền tảng (framework) miễn phí, mã nguồn mở, dựa trên HTML, CSS & Javascript, nó được tạo ra để xây dựng các giao diện Website tương thích với tất cả các thiết bị có kích thước màn hình khác nhau.

Bootstrap bao gồm những cái cơ bản có sẵn như: typography, forms, buttons, tables, navigation, modals, image carousels và nhiều thứ khác. Nó cũng có nhiều Component, Javascript hỗ trợ cho việc thiết kế Reponsive của bạn dễ dàng, thuận tiện và nhanh chóng hơn.

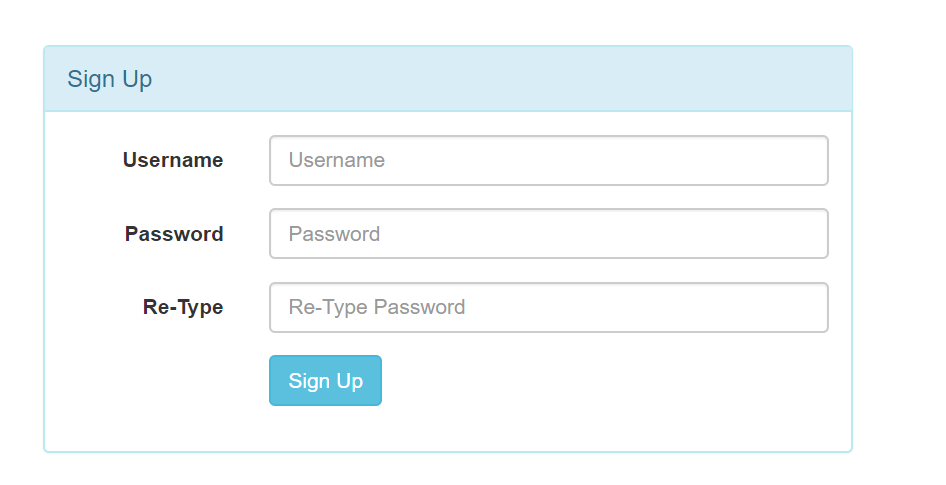
#### Thiết kế giao điện hệ thống đăng nhập đăng ký

* Màn hình chính gồm có hai nút nhấn để lựa chọn đăng nhập và đăng ký

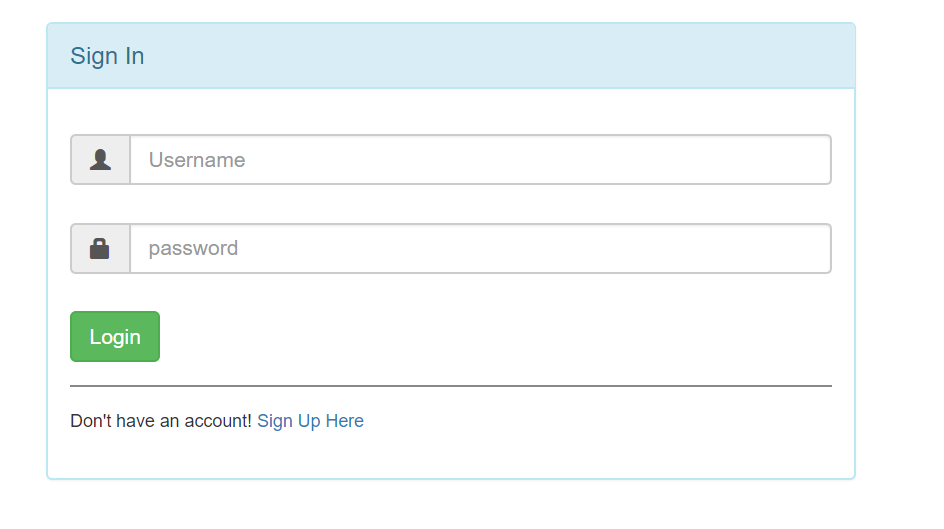


Hình 2.3: Giao diện màn hình chính

* Màn hình đăng kí gồm 3 trường yêu cầu nhập thông tin là username, password và repassword.

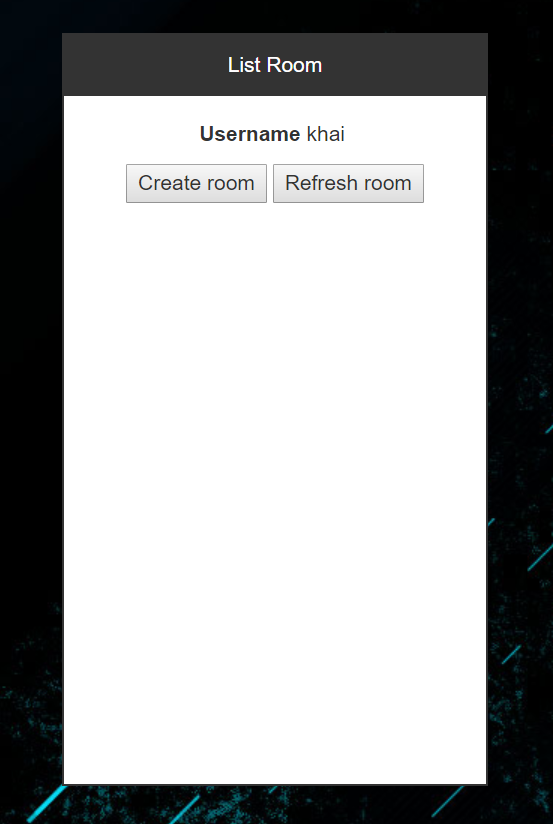


Hình 2.3: Giao diện màn đăng kí

* Màn hình đăng nhâp gồm 2 trường yêu cầu nhập thông tin là username, password.

Hình 2.3: Giao diện màn hình đăng nhập

* Sau khi đăng nhập thì sẽ chuyển tới giao diện tạo phòng bắt đàu chơi game



Hình 2.3: Giao diện màn tạo phòng chơi game

### Cơ sở dữ liệu để lưu trữ thông tin người dùng.

#### Giới thiệu về MySQL

Được phát hành từ giữa thập niên 90s (sau đó bị thâu tóm bởi Oracle), MySQL ban đầu là một database mã nguồn mở và cũng vẫn mở cho tới tận bây giờ. Vì là mã nguồn mở, MySQL có rất nhiều phiên bản khác dựa trên nó. Sự khác biệt giữa các biến thể này là không lớn; cấu trúc và chức năng cơ bản tương đương nhau.

Một điều đã trở thành đặc tính riêng của MySQL là nó cực kỳ phổ biến trong cộng đồng startup. Vì nó là mã nguồn mở và miễn phí, lập trình viên có thể dễ dàng bắt đầu với MySQL, và chỉnh sửa code nếu họ cần làm vậy. MySQL thường được dùng đồng thời với PHP và Apache Web Server, trên một bản Linux distribution, bộ tứ này đã trở thành một tên gọi nổi tiếng và quyền lực: LAMP (Linux, Apache, MySQL, PHP).

#### Cách tạo một database đơn giản với MySQL

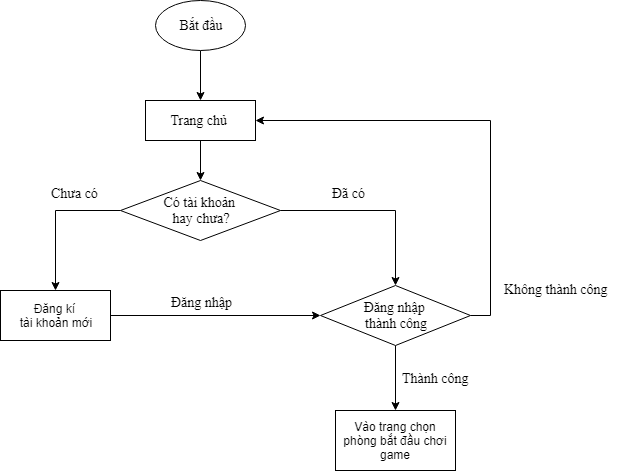
* Tạo một database với tên là nodejs\_game\_v1

1. var mysql = require('mysql');
3. var con = mysql.createConnection({
4. host: "localhost",
5. user: "root",
6. password: ""
7. });
9. con.connect(function(err) {
10. **if** (err) **throw** err;
11. console.log("Connected!");
12. con.query("CREATE DATABASE Nodejs\_Game\_v1", function (err, result) {
13. **if** (err) **throw** err;
14. console.log("Database created");
15. });
16. });

* Tạo một bản có tên user trong database nodejs\_game\_v1

1. var mysql = require('mysql');
3. var con = mysql.createConnection({
4. host: "localhost",
5. user: "root",
6. password: "",
7. database: "Nodejs\_Game\_v1"
8. });
10. con.connect(function(err) {
11. **if** (err) **throw** err;
12. console.log("Connected!");
13. var sql = "CREATE TABLE users (id INT primary key auto\_increment,username VARCHAR(255),password  VARCHAR(255), scores INT)";
14. con.query(sql, function (err, result) {
15. **if** (err) **throw** err;
16. console.log("table created");
17. });
18. });

### Sơ đồ chức năng hệ thống đăng nhập và đăng kí



Hình 2.3: Sơ đồ chức năng hệ thống đăng nhập đăng kí

### Giới thiệu nodejs ,socket io,net

#### Nodejs

NodeJS là một mã nguồn mở được xây dựng dựa trên nền tảng Javascript V8 Engine, nó được sử dụng để xây dựng các ứng dụng web như các trang video clip, các forum và đặc biệt là trang mạng xã hội phạm vi hẹp.

NodeJS có thể chạy trên nhiều nền tảng hệ điều hành khác nhau từ WIndow cho tới Linux, OS X .NodeJS cung cấp các thư viện phong phú ở dạng Javascript Module khác nhau giúp đơn giản hóa việc lập trình và giảm thời gian ở mức thấp nhất.

Khi nói đến NodeJS thì phải nghĩ tới vấn đề Realtime. Realtime ở đây chính là xử lý giao tiếp từ client tới máy chủ theo thời gian thực.

+Các đặc tính của nodejs

* **Không đồng bộ**: Tất cả các API của NodeJS đều không đồng bộ (none-blocking), nó chủ yếu dựa trên nền của NodeJS Server và chờ đợi Server trả dữ liệu về. Việc di chuyển máy chủ đến các API tiếp theo sau khi gọi và cơ chế thông báo các sự kiện của Node.js giúp máy chủ để có được một phản ứng từ các cuộc gọi API trước (Realtime).
* **Chạy rất nhanh**: NodeJ được xây dựng dựa vào nền tảng V8 Javascript Engine nên việc thực thi chương trình rất nhanh.
* **Đơn luồng nhưng khả năng mở rộng cao**: Node.js sử dụng một mô hình luồng duy nhất với sự kiện lặp. cơ chế tổ chức sự kiện giúp các máy chủ để đáp ứng một cách không ngăn chặn và làm cho máy chủ cao khả năng mở rộng như trái ngược với các máy chủ truyền thống mà tạo đề hạn chế để xử lý yêu cầu. Node.js sử dụng một chương trình đơn luồng và các chương trình tương tự có thể cung cấp dịch vụ cho một số lượng lớn hơn nhiều so với yêu cầu máy chủ truyền thống như Apache HTTP Server.
* **Không đệm**: NodeJS không đệm bất kì một dữ liệu nào và các ứng dụng này chủ yếu là đầu ra dữ liệu.
* **Có giấy phép**: NodeJS đã được cấp giấy phép bởi MIT License

#### Socket io

Là một bộ thư viện dành cho các ứng dụng web, mobile realtime. Với đặc trưng mạnh mẽ và dễ sử dụng, [**Socket.IO**](http://Socket.IO) đang dần trở nên quen thuộc với các nhà phát triển (Từ Microsoft Office, Yammer, Zendesk, Trello… tới những đội hackathon, những start up trẻ).

Thư viện này gồm 2 phần:

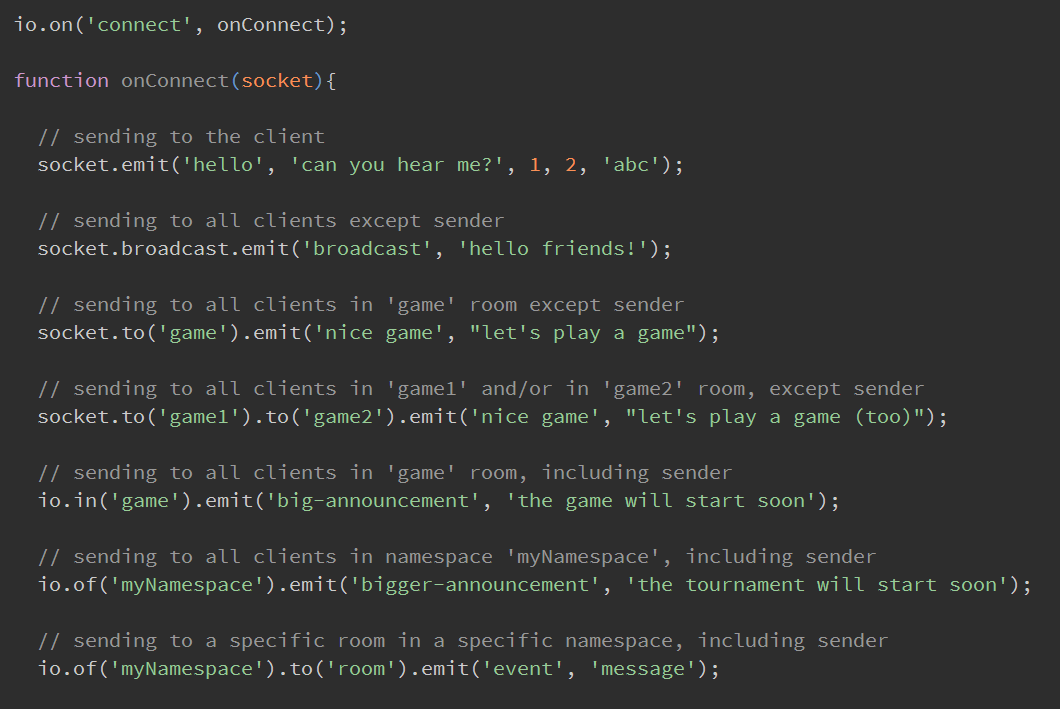
* **Phía client**: gồm bộ thư viện viết cho web(JavaScript), iOS, Android
* **Phía server**: viết bằng JavaScript và dùng cho các máy chủ node.JS

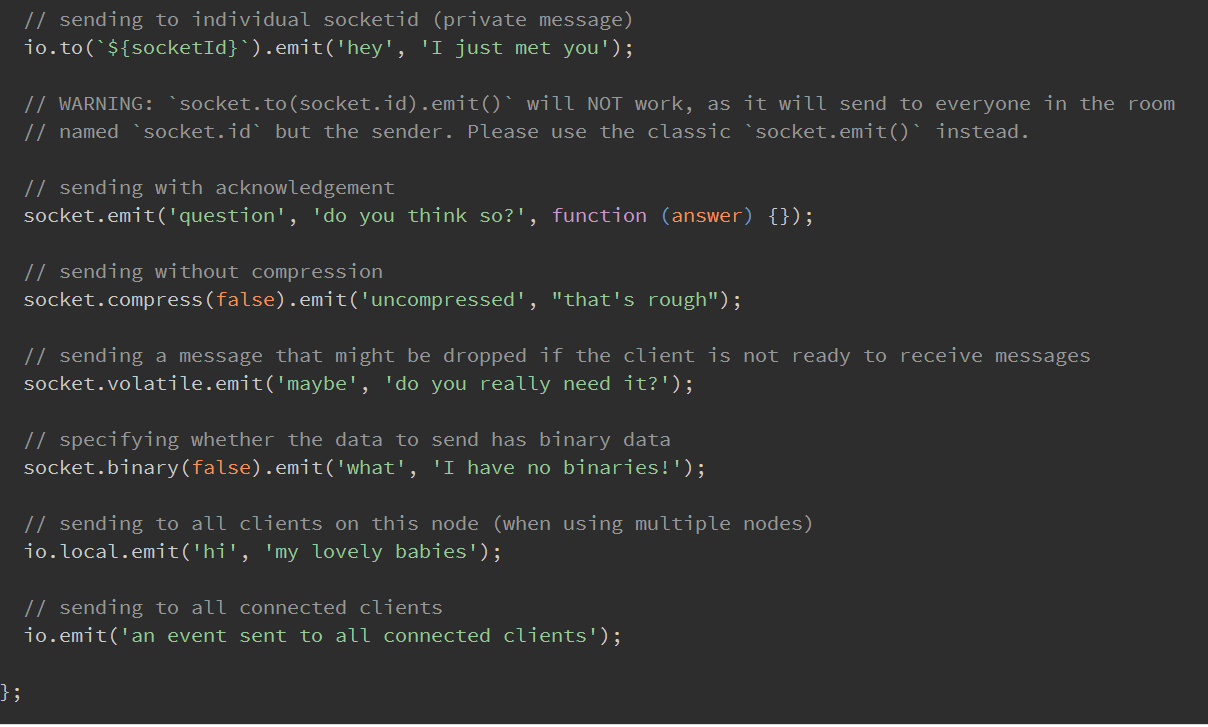
[**Socket.IO**](http://Socket.IO) cung cấp cho các nhà phát triển một cách đơn giản và thuận tiện để xây dựng một ứng dụng realtime đa nền tảng (web và mobile). Với bộ thư viện này, làm việc với socket trở nên đơn giản hơn rất nhiều.

Thư viện [Socket.IO](http://Socket.IO) trên Android cung cấp những **hàm cơ bản** sau:

* **connect()**: kết nối với server socket
* **on(event\_name, listener)**: đăng kí lắng nghe sự kiện từ server trả về
* **emit(event\_name, data)**: gửi một sự kiện lên server
* **off(event\_name)**: ngừng lắng nghe một sự kiện nào đó

**Các lệnh emit thường sử dụng:**



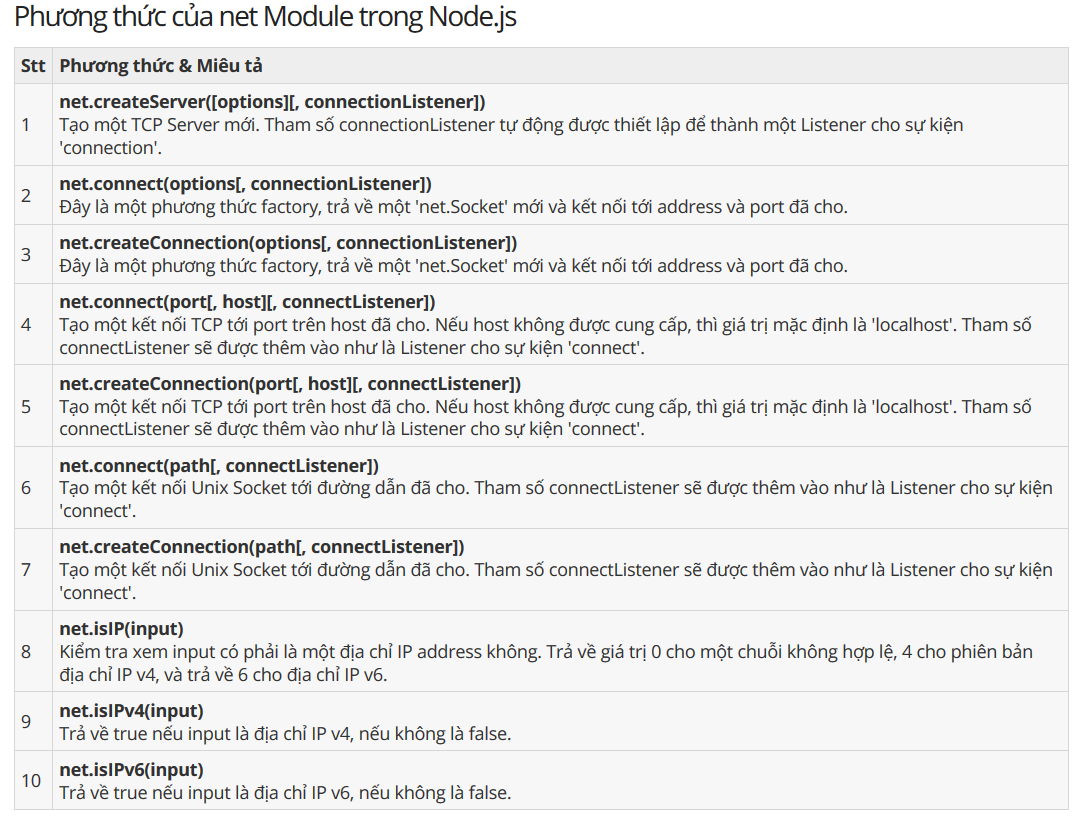


#### Net

**net** Module trong Node.js được sử dụng để tạo Server và Client. Module này cung cấp một Network Wrapper không đồng bộ và có thể được import với cú pháp:

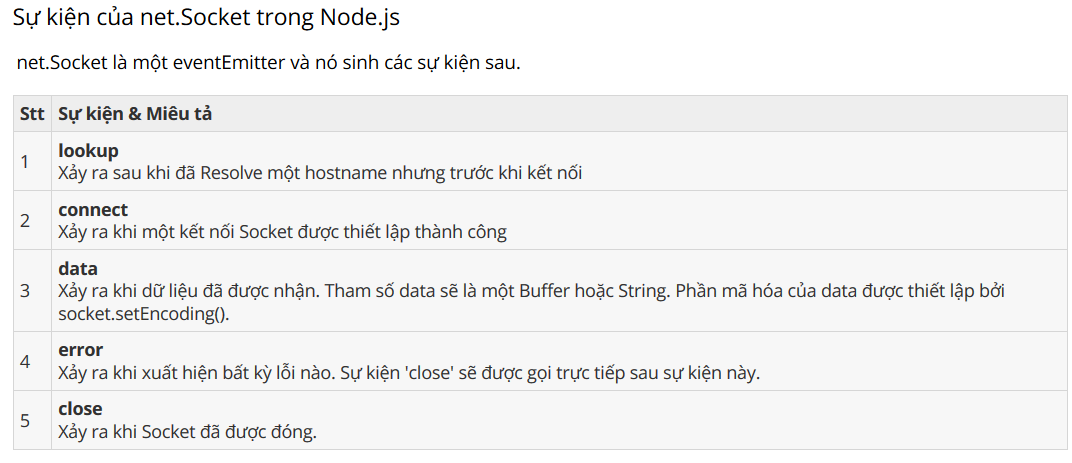


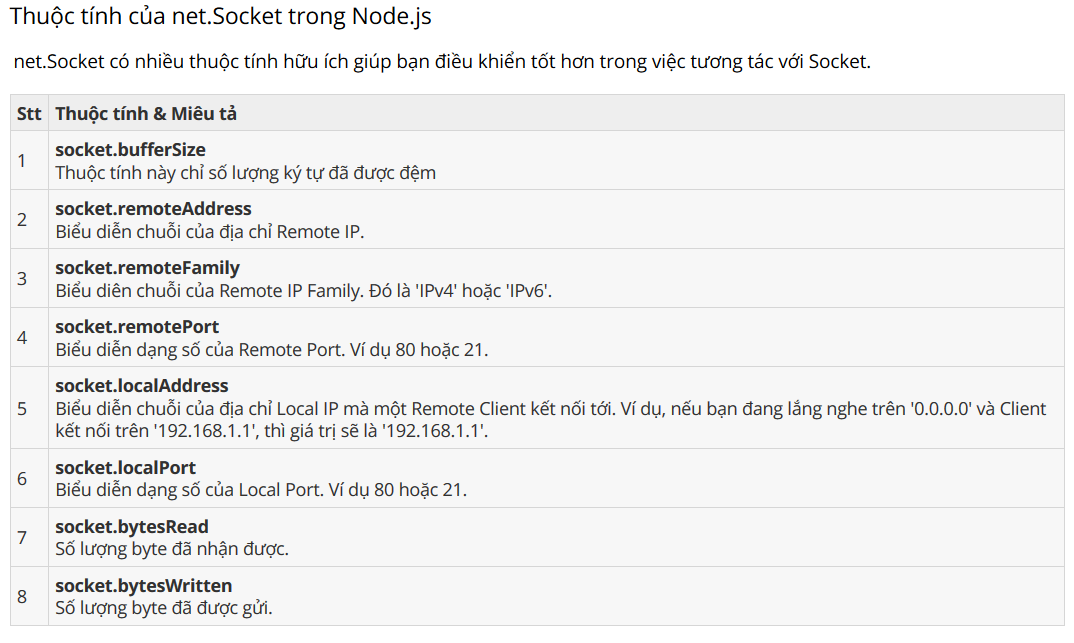
Một số phương thức của net:

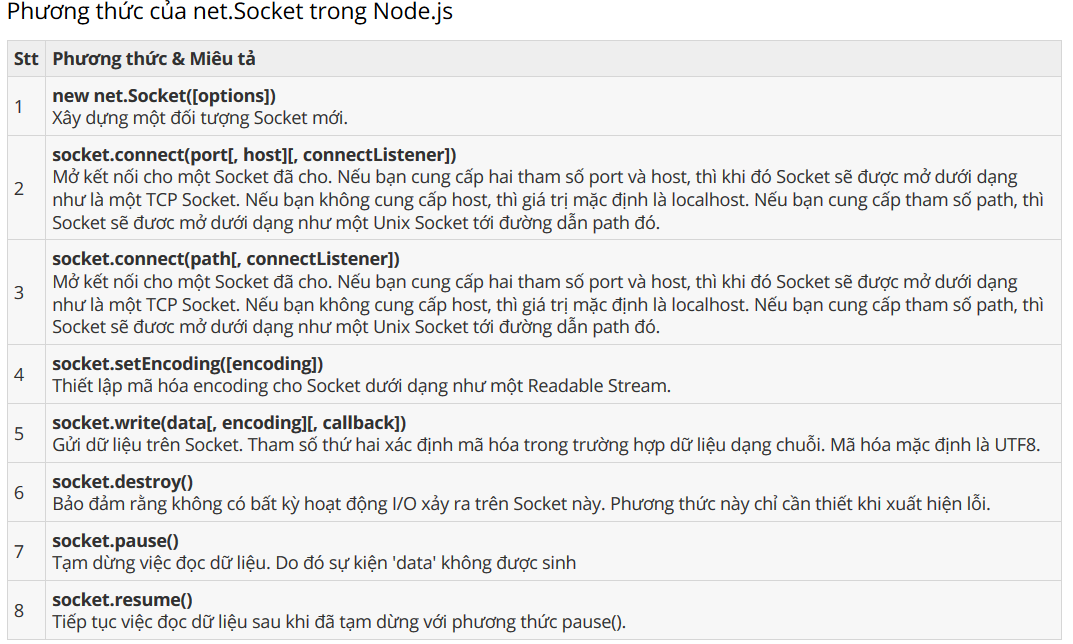


Lớp net.Socket trong Nodejs:

Đối tượng này là lớp trừu tượng của TCP hoặc Local Socket. net.Socket kế thừa duplex Stream interface. Chúng có thể được tạo bởi người dùng hoặc bởi một Client (bởi phương thức connect()) hoặc có thể được tạo bởi Node.js và được truyền tới người dùng thông qua sự kiện 'connection' của một Server.

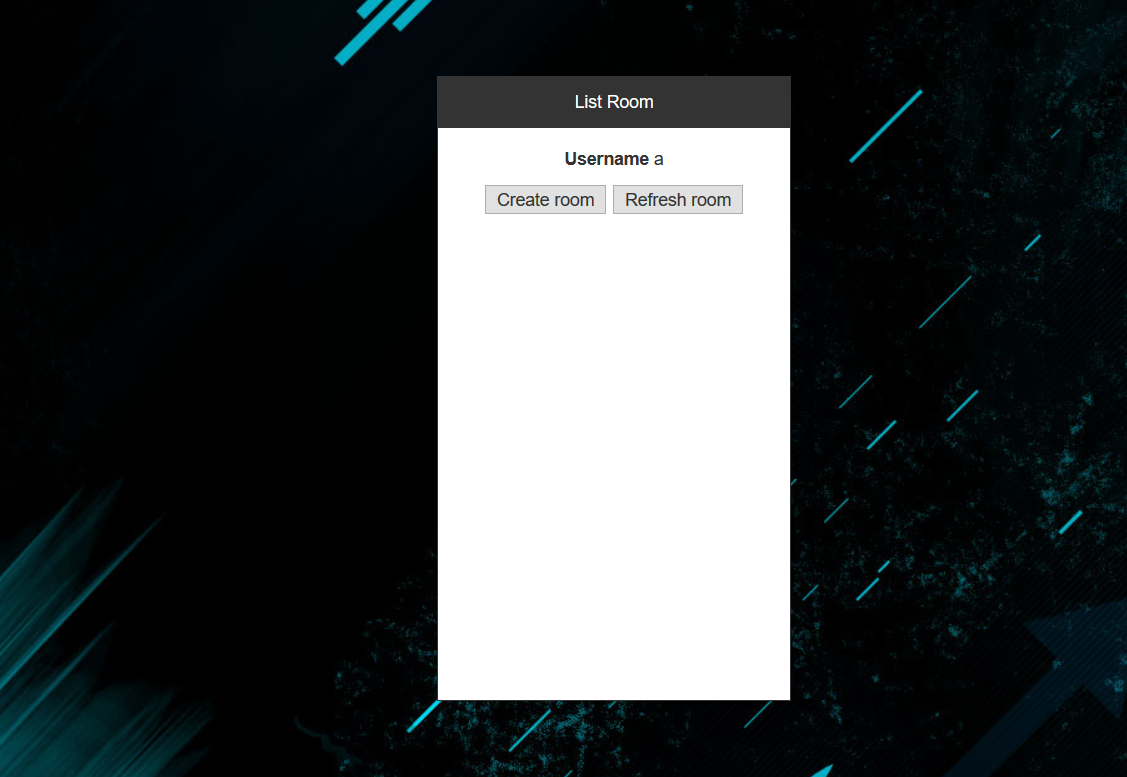






### Thiết kế danh sách phòng chơi

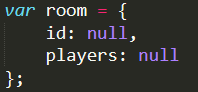
Sau khi đăng nhập thành công thì ta sẽ vào giao diện như hình:



Khi nhấn nút Create room thì chạy :

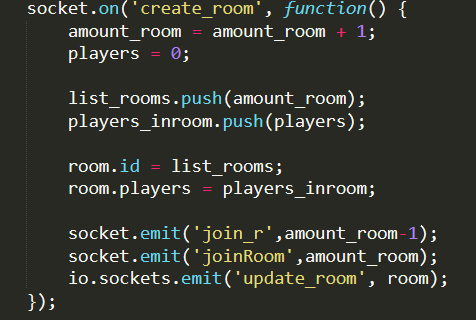
1. Từ phía **client**: (web đang đăng nhập sẽ có socket riêng biệt) sẽ dùng **socket.emit('create\_room')** để thông báo về server có sự kiện tạo phòng
2. Từ phía **server: socket.on('create\_room',function(){})** để lắng nghe sự kiện tạo phòng từ client ,hàm function sẽ được thực thi.

* Ta phải tạo trước 1 đối tượng:



|  |  |
| --- | --- |
| **id** | Mảng tên phòng |
| **players** | Mảng số lượng người trong phòng |

* Hàm function sẽ thực hiện:

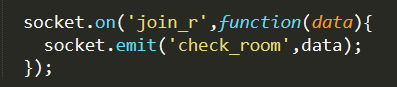


-Ban đầu **amount\_room** =0,khi tạo phòng thì tăng 1 và cũng là tên room rồi thêm vào mảng **list\_rooms** .Người chơi trong phòng =0 rồi thêm vào mảng **player\_inroom**. Đối tượng **room** lưu 2 mảng trên để cập nhật mới nhất

-Tiếp theo **socket.emit('join\_r',amount\_room-1)**

Để thông báo cho **phía client** sẽ vào phòng vừa tạo. Data của emit **join\_r** là **chỉ số phòng trong mảng** trong đối tượng **room.** Do ban đầu amount\_room luôn tăng 1 trước khi đưa vào mảng nên muốn có được chỉ số thì trừ -1 sẽ ra được

-**Phía client** lắng nghe sự kiện **join\_r**:



**Data** là chỉ số phòng nhận được,ta sẽ thông báo lại về **phía server** sự kiện check\_room đó

-**Phía server** lắng nghe sự kiện **check\_room:**



+Kiểm tra nếu người trong phòng lớn hơn 2 thì báo cho **client** sự kiện not\_joinRoom không cho vào phòng

+Nếu ngược lại thì

+Báo cho **client** sự kiện **joinRoom** với **data truyền đi là id của phòng** đó,thực hiện việc chuyển sang **giao diện** phòng chờ đối thủ

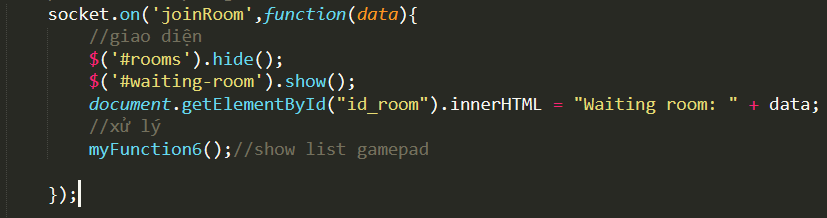
+Tăng số lượng người trong phòng,thêm thuộc tính **socket.room** để biết **socket** hiện tại ở **room** nào

+Dùng **io.sockets.emit('update\_room', room) :**thông báo cho **tất cả client** về sự kiện update\_room,cập nhật hiển thị lại những thay đổi của danh sách room

+ **socket.join('room' + socket.room):** vào phòng có tên là chuỗi **'room' + socket.room**

**-Phía client:**

+Lắng nghe sự kiện **not\_joinRoom**,**data nhận được là chỉ số mảng trong room**

+Lắng nghe sự kiện **joinRoom**,**data nhận được là id phòng**

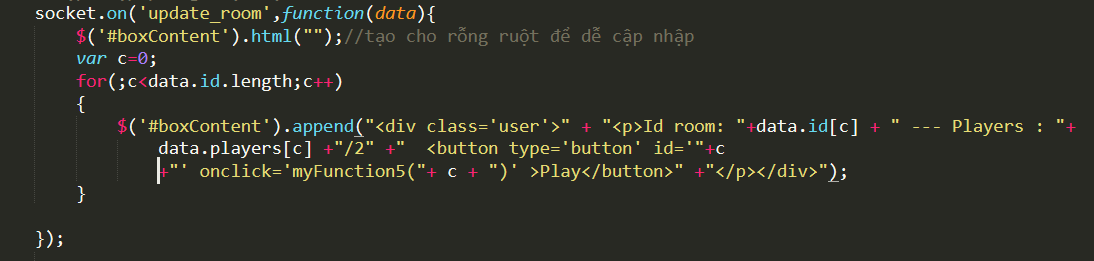
+Giao diện thì ẩn rooms (dnah sách phòng)đi,hiện waiting-room(phòng chờ đối thủ)

+Hiển thị tên phòng

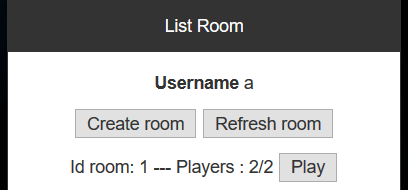
+Hàm hiển thị danh sách tay cầm hiện có



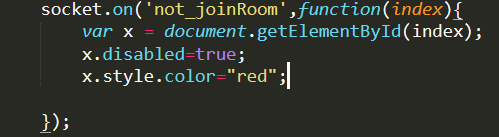
+Lắng nghe sự kiện **update\_room, data nhận được là đối tượng room**



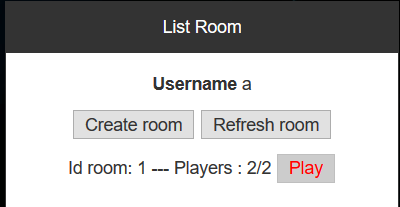
+Có 1 div với id boxContext để hiện danh sách phòng.Mỗi lần cập nhật thì xóa hết nội dung cũ trong div đó

+Dùng for duyệt đối tượng room,mỗi lần duyệt sẽ thêm 1 **div**, gồm **tên phòng**,**số người** trong phòng, **nút play có id** với **data truyền vào** là chỉ số mảng khi duyệt để khi nhấn **nút play** thì nó chuyển qua giao diện vào phòng chờ đối thủ

+Lắng nghe sự kiện **not\_joinRoom**,**data nhận được là chỉ số mảng trong room**



+do khi update room có tạo ra các div gồm nút play có id là chỉ số của mảng nên ta lấy id và disable nút đó,chuyển sang màu đỏ để cho biết phòng đã đầy



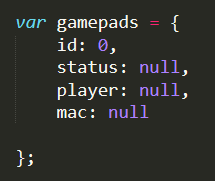
### Thiết kế phòng chờ

Sau khi vào phòng chờ,sẽ hiển thị danh sách tay cầm để chọn



Tay cầm sử dụng tcp để kết nối,do vậy trên server thì nodejs có gói **net** hỗ trợ về tcp

Đầu tiên tạo 1 đối tượng gamepads như sau:



|  |  |
| --- | --- |
| id | Mảng chứa địa chỉ ip và port của tay cầm khi kết nối |
| status | Mảng Trạng thái tay cầm đã có người chọn hay chưa.0 là chưa chọn,1 là chọn rồi |
| player | Mảng Socket.id của người chơi |
| mac | Mảng Địa chỉ mac |

Ta khai báo gói net :



Rồi dùng hàm createServer trong gói net để tạo tcp server



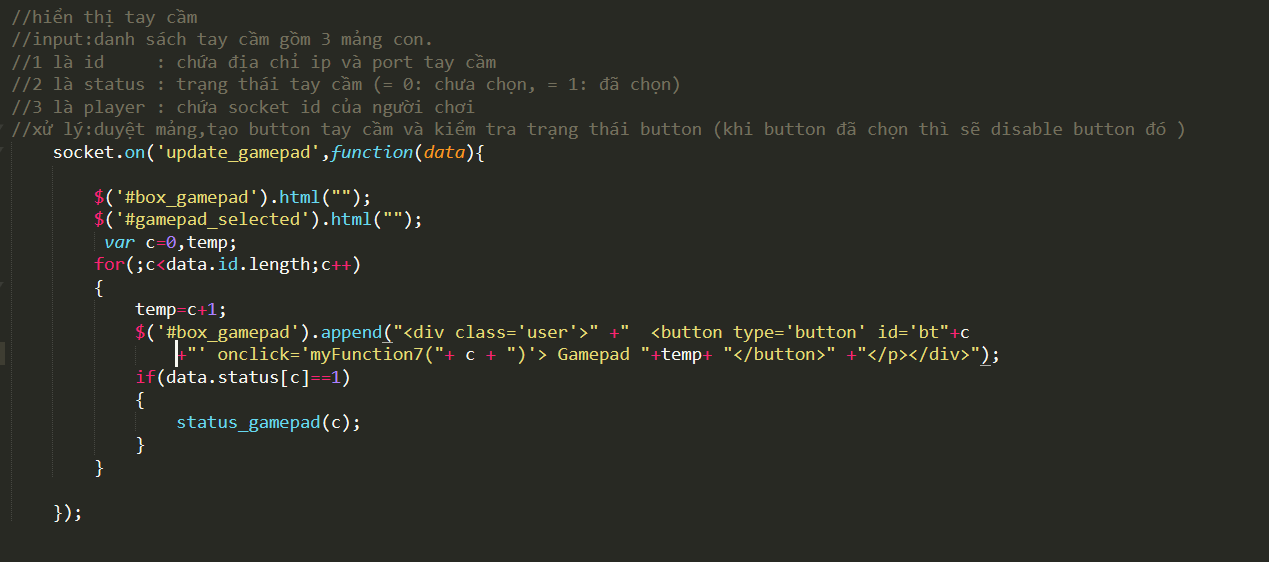
+Khi có tay cầm kết nối thì ta nhận được đối tượng socket

+ta thêm 1 thuộc tính name vào socket gồm địa chỉ ip và port rồi lưu nó vào mảng clients. Ta cũng khởi tạo các giá trị ban đầu của **status,player**.Cuối cùng đối tượng **gamepads** sẽ lưu các mảng vừa tạo ở trên.

+Tiếp theo từ **server tcp** sẽ thông báo cho tất cả **các client( web )**về sự kiện có tay cầm

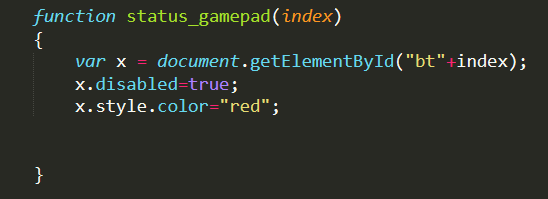


+Phía **client** lắng nghe sự kiện **update\_gamepad**:



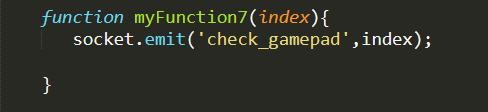
Ta duyệt mảng **gamepads** nhận được,từng phần tử sẽ tạo 1 nút nhấn có **id = chuỗi (“bt”+chỉ số phần tử)** và đồng thời kiểm tra tình trạng của nút đó.

+nếu **status =1** thì tức tay cầm đã có người chọn.Ta gọi tới hàm **status\_gamepad**(chỉ số phần tử) để **disable** nút đó không cho những người khác chọn và hiện màu đỏ làm dấu



+Sau khi hiển thị,ta tới phần **chọn tay cầm**:





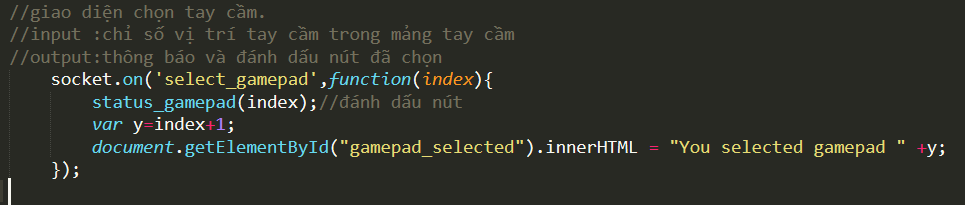
Khi nhấn thì sự kiện nhấn nút sẽ chạy hàm myFunction7,data là chỉ số của phần tử trong mảng gamepads.Hàm nãy sẽ thông báo một sự kiện check\_gamepad tới server để kiểm tra có được chọn hay không.

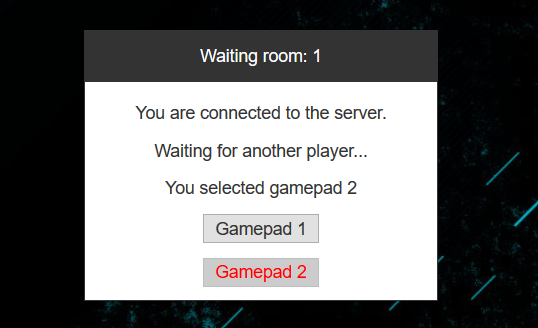
+Phía server sẽ lắng nghe sự kiện như sau:





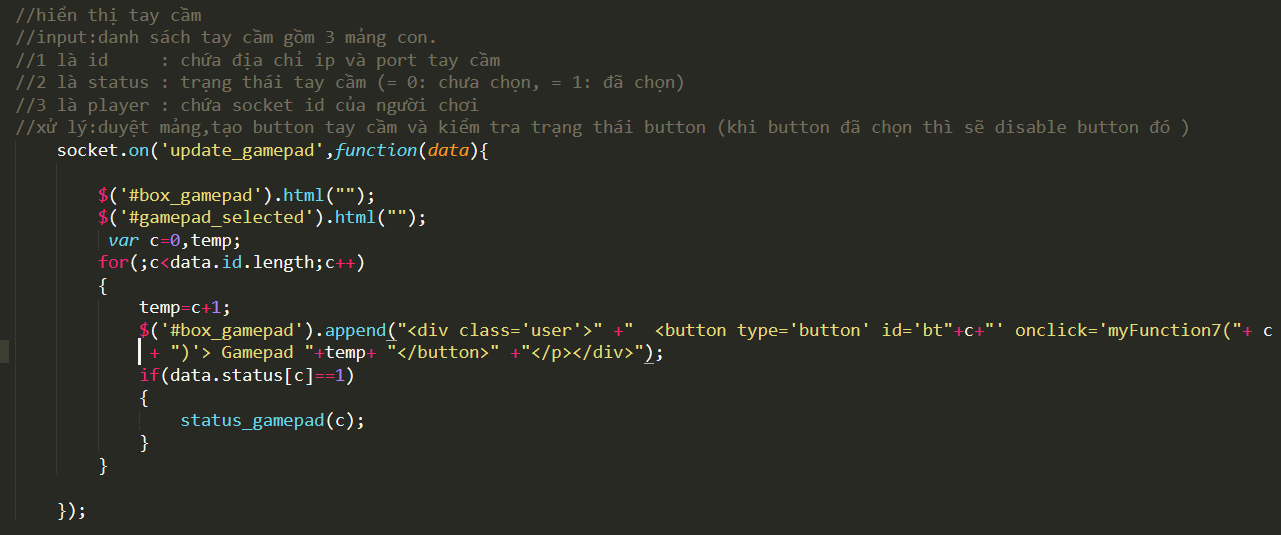
+kiểm tra nếu status của gamepad và socket.gamepad chưa chọn thì thực hiện:

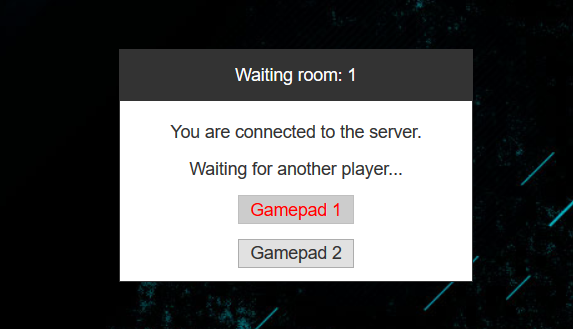
+Thông báo cho client sự kiện **select\_gamepad** chọn tay cầm đó, Phía client sẽ lắng nghe sự kiện và thay đổi giao diện 



+Thiết lập trạng thái tay và lưu thông tin socket người chơi trong gamepads. Gán địa chỉ mac của tay cầm thêm cho socket thuộc tính socket.gamepad

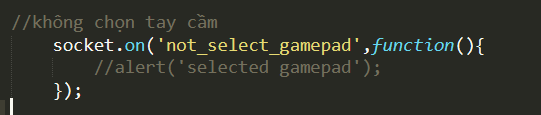
+Thông báo cho các client còn lại sự kiện update\_gamepad để cho biết rằng có tay cầm đã chọn. Phía client sẽ lắng nghe sự kiện và thay đổi giao diện





Khi đã có người chọn tay cầm đó thì sẽ disable nút đó,các người chơi còn lại sẽ không chọn tay cầm đó được nữa

+kiểm tra nếu status của gamepad và socket.gamepad đã chọn thì báo cho client đó sự kiện **not\_select\_gamepad**



Do ở trên đã xử lý disable nút đó rồi nên hàm này trống

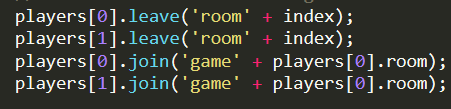
+Việc cuối cùng là kiểm tra trong phòng đó có đủ 2 người và 2 người đó đã chọn tay cầm hay chưa,nếu rồi thì sẽ vào map game,còn không thì sẽ chờ cho đến khi điều kiện trên đúng



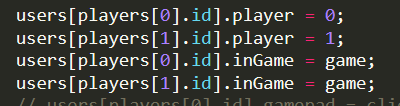
+Hàm này sẽ lấy các socket đã vào phòng rồi kiểm tra số lượng và thuộc tính gamepad của socket đã có ip mac hay chưa

+Nếu điều kiện đúng nó sẽ tạo game. Khởi tạo game cần 3 đầu vào theo thứ tự:phòng chơi,socket id người 1,socket id người 2

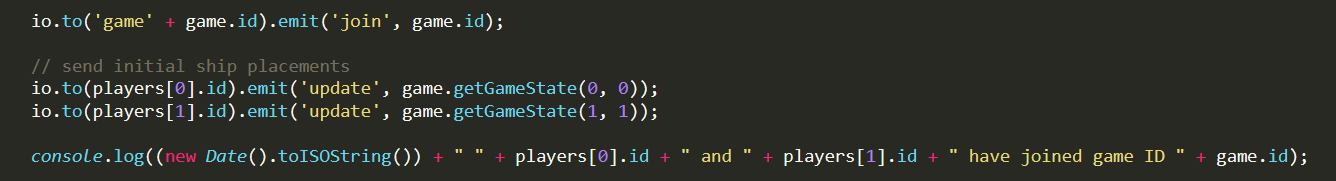
+Cả hai người sẽ thoát khỏi phòng chờ và nhảy vào phòng game.

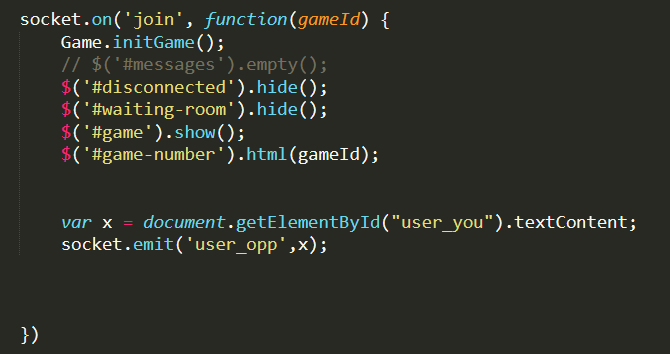


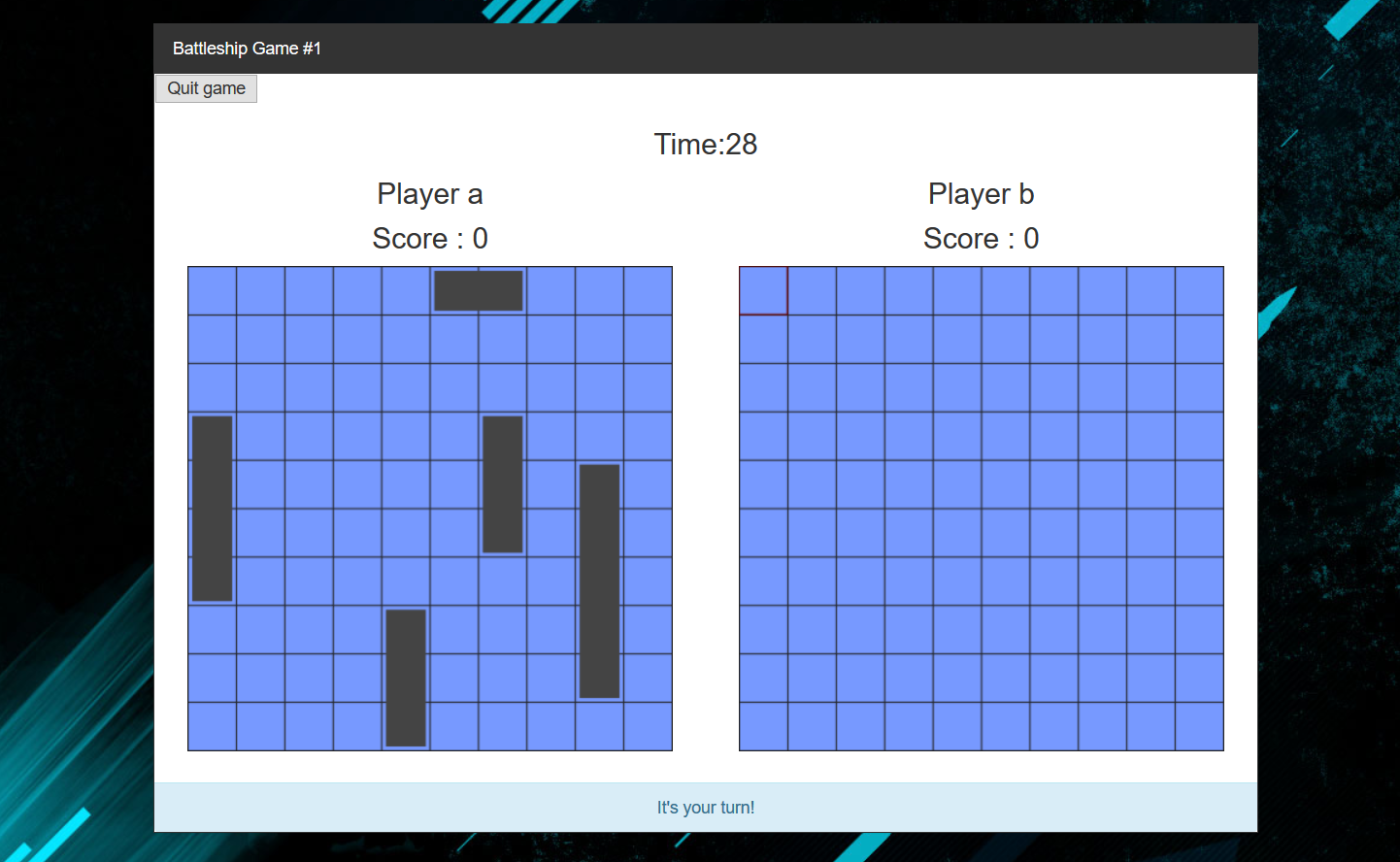
+Thiết lập của người chơi:



+Server sẽ thông báo cho các clietn trong phòng game đó sự kiện **join** thiết lập map game ban đầu,và thuyền của cả hai người chơi:







### Nhận thông tin từ tay cầm và xử lý

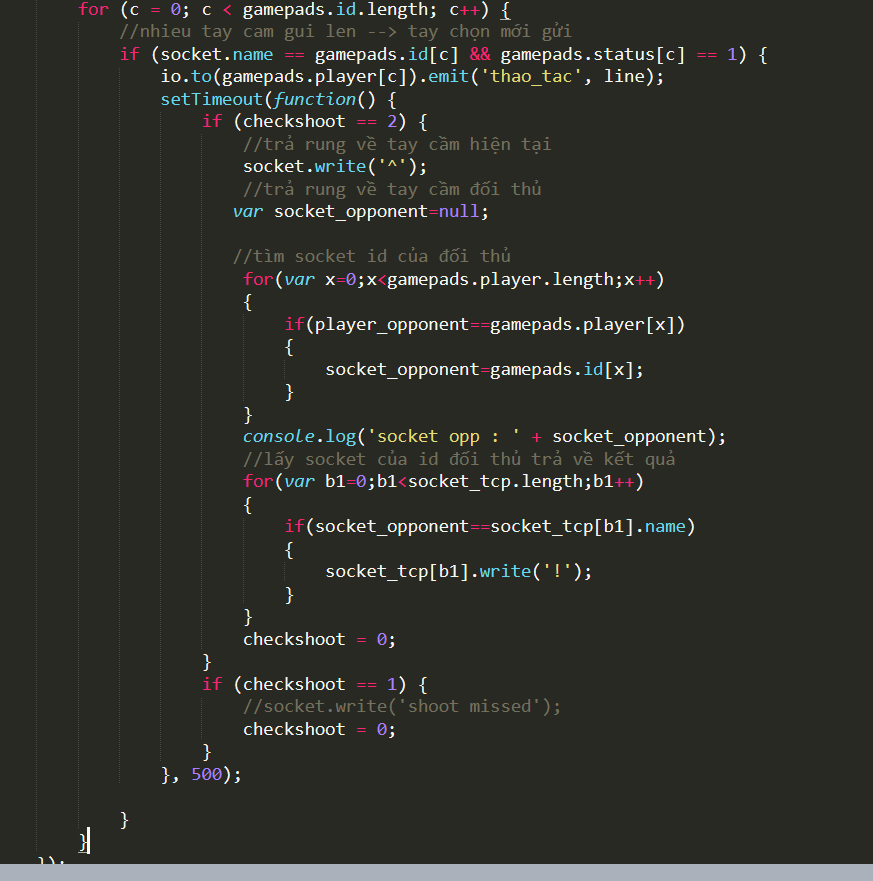
+Hàm nhận data trong server tcp sẽ có 2 trường hợp:

+**data là ip mac gồm 17 ký tự:** dùng để xác định ai đang chơi tay cầm nào,dù người chơi có rút tay cầm ra thì vẫn có thể kết nối lại nhờ ip mac

+**data là 1 ký tự**: dùng để di chuyển hoặc bắn

+ code trên là nhận **ip mac**.

+hình dưới là di chuyển hoặc bắn



+**xử lý nhận dữ liệu 1 ký tự** như sau:

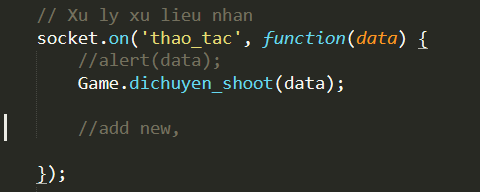
+xét trong **mảng id gamepads** với **socket.name (chứa ip : port)** **và trạng thái** tay cầm của nó nếu đã chọn thì **server io** sẽ thông báo sự kiện **thao\_tac** tới socket id đã lưu trong mảng **gamepads.player**

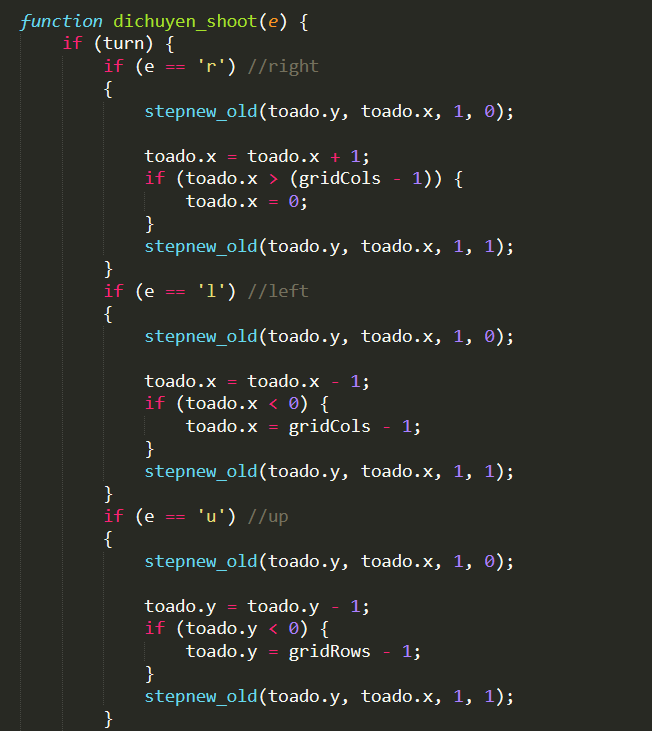
**+**Có một biến toàn cục tên checkshoot để kiểm tra nó đã bắn trúng hay trật ,khi ta chạy **riêng** các dòng lệnh của hàm trong **setTimeout** mà không có **time out** thì biến checkshoot nó sẽ không cập nhập kết quả mới vì nguyên tắc thực thi chạy lệnh của nodejs.

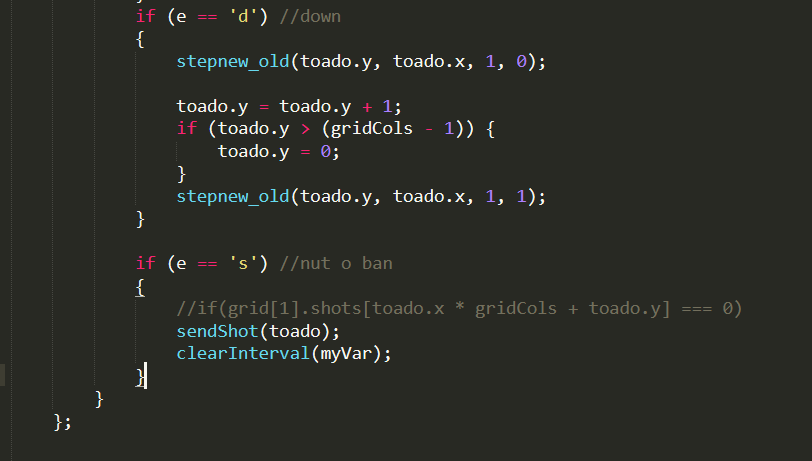
Các lệnh trong hàm nó sẽ chạy các lệnh đồng bộ trước (tức là theo lệnh từ trên xuống dưới nhưng không phụ thuộc vào kết quả ) rồi mới chạy các lệnh bất đồng bộ sau.

Do lệnh io .emit đã là bất đồng bộ nên ta cần đặt hàm checkshoot trong setTimeout để trở thành bất đồng bộ luôn. Thì kết quả là **io emit** sẽ **thực thi trước** xong rồi thì mới tới lệnh trong setTimeout sau **khoảng thời gian đủ** để io emit **thực thi xong**.

+Phía **client** nhận sự kiện **thao\_tac**:



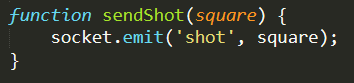




Hàm này sẽ thực hiện di chuyển lên xuống qua lại và bắn.

+Khi di chuyển qua vị trí mới thì phải trả về trạng thái cũ của vị trí cũ. Hàm **stepnew\_old** thực hiện việc này.Các tham số theo thứ tự là 2 tọa độ vị trí , vị trí map (có 0 là bên trái và 1 là bên phải),bước mới(=1) hoặc cũ (=0)

+Khi bắn thì gửi tọa độ về cho **server** xử lý:



**Phía server** sẽ lắng nghe sự kiện **shot:**



+Đầu tiên lấy game của socket đang chơi thông qua socket id

+Kiểm tra đến lượt của mình chưa nếu đúng thì thực hiện việc bắn bằng hàm game.shoot(vị trí). Kết quả trả về là true nếu bắn được,false nếu không bắn được

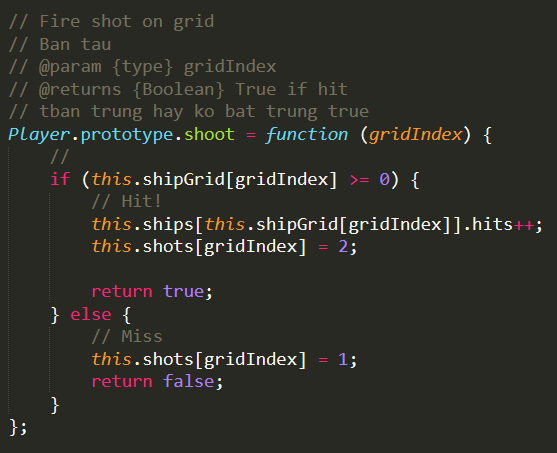


+Hàm này sẽ kiểm tra vị trí của đối thủ đã bắn hay chưa và trạng thái của game đang chơi hay kết thúc. Nếu đúng thì sẽ thực hiện bắn. Mỗi người chơi có 2 mảng :

|  |  |
| --- | --- |
| Mảng shots | Giá trị ban đầu là 0,nếu bắn trúng =2,bắn trật =1 |
| Mảng shipGrid | Giá trị ban đầu là -1,nếu có tàu sẽ set >=0 |

+khi chạy tới lệnh thì player sẽ thực hiện hàm của nó:





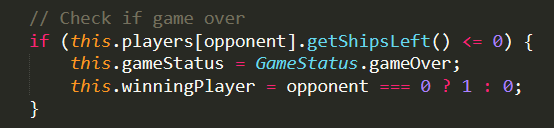
Kiểm tra mảng shipGrid có tàu hay không.Nếu có thì tăng biến bắn trúng tàu lên 1 và gán trong mảng shots tại vị trí đó =2 (tức bắn trúng).Ngược lại thì gán trong mảng shots tại vị trí đó =1 (tức bắn trật)

+Sau khi thực hiện bắn song thì sẽ check



Nếu bắn trật sẽ set biến check\_shoot =1 và đổi lượt,ngược lại nếu trúng thí check\_shoot =2 và tăng số điểm hiện tại của người chơi

+sau khi bắn sẽ kiểm tra đã thua hay chưa:

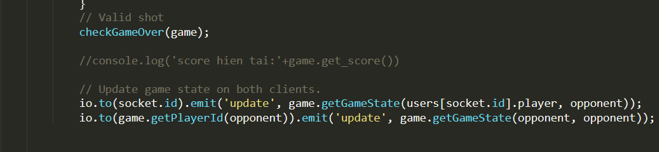


+sau đó nó sẽ trả về true nếu bắn được và false nếu không bắn được.Quay về tiếp tục phía server sự kiện **shot.**



+sau khi thực thi lệnh **game.shoot(position)** ,ta kiểm tra nếu biến **check\_shoot** = 2 tức là trúng thì ta sẽ cập nhập biến **checkshoot( dùng để trả về tín hiệu cho tay cầm bắn )** và biến toàn cục **player\_opponent**= **socket id** của đối thủ (**để trả về tín hiệu tay cầm đối thủ )**

+các câu lệnh tiếp là theo kiểm tra game kết thúc hay chưa và update lại mapgame của cả hai người chơi



+Sau khi xử lý xong sự kiện **shot** thì hàm trong Settimeout sẽ thực thi



Mức thời gian ví dụ như 500 trong hàm này tức là sau 500 ms thì hàm này sẽ được đưa vào hàng chờ để thực thi.

+Lúc này biến **checkshoot và player\_oppenent(socket id đối thủ)** đã được cập nhập xong.

+nếu **checkshoot =2** thì sẽ gửi **về tay cầm hiện tại** dấu **^ (socket.write('^');) .** Để gửi về **tín hiệu !** cho phía **tay cầm đối thủ** thì ta dựa vào **player\_oppenent** để duyệt mảng gamepads tìm được **socket.name (địa chỉ ip:port)** trong **gamepads.id[].** Tiếp theo ta dựa vào **socket.name** để tìm trong **mảng socket\_tcp có thuộc tính name** khớp với nó thì trả về **!** 

**+ xử lý nhận data là ip mac gồm 17 ký tự:**

+Mục đích:trong trường hợp rút tay cầm ra thì cắm lại vẫn kết nối với người chơi cũ



**+ý tưởng**:khi vừa kết nối tay cầm sẽ gửi thêm địa chỉ **mac**,cũng lưu các thông tin rồi cập nhập vào gamepads như thường. Tiếp theo lấy ngay mac **vị trí cuối** của **mảng mac trong gamepad**(tức cái **vừa kết nối**). Ta duyệt mảng mac **từ đầu đến vị trí kế cuối**,nếu **đã có mac** thì sẽ bật flag =1 và lưu **id (ip :port)** tại **vị trí** cuối vào **phần tử cùng mac** trong **mảng id** của **gamepads**(tức địa chỉ ip :port mới của tay cầm cũ). Xét nếu flag bằng 1 thì xóa phần tử vừa thêm vào đối tượng gamepads.Sau đó sẽ set trạng thái và socket id của người cũ.

# KẾT QUẢ THỰC NGHIỆM

## Tính năng đăng nhập vào hệ thống.

### Các trường hợp thử kiểm tra.

* Kiểm tra trường hợp 1: Đăng nhập không có username password 🡪 Hệ thống báo lỗi.
* Kiểm tra trường hợp 2: Đăng nhập nhập username không nhập password 🡪 Hệ thống báo lỗi.
* Kiểm tra trường hợp 3: Đăng nhập không nhập username nhập password 🡪 Hệ thống báo lỗi.
* Kiểm tra trường hợp 4: Đăng nhập nhập username sai nhập password đúng 🡪 Hệ thống báo lỗi.
* Kiểm tra trường hợp 5: Đăng nhập nhập username đúng nhập password sai 🡪 Hệ thống báo lỗi.
* Kiểm tra trường hợp 6: Đăng nhập nhập username đúng nhập password đúng 🡪 **Hệ thống đăng nhâp thành công.**
* Kiểm tra trường hợp 7: Đăng kí không nhập username password và repassword 🡪 Hệ thống báo lỗi.
* Kiểm tra trường hợp 8: Đăng kí nhập username không nhập password và repassword 🡪 Hệ thống báo lỗi.
* Kiểm tra trường hợp 9: Đăng kí nhập username nhập password và không nhập repassword 🡪 Hệ thống báo lỗi.
* Kiểm tra trường hợp 10: Đăng kí nhập username không nhập password và nhập repassword 🡪 Hệ thống báo lỗi.
* Kiểm tra trường hợp 11: Đăng kí nhập username nhập password và repassword nhưng không trùng khớp🡪 Hệ thống báo lỗi.
* Kiểm tra trường hợp 12: Đăng kí nhập username nhập password và repassword khớp nhưng usenaem đã tồn tài 🡪 Hệ thống báo lỗi.
* Kiểm tra trường hợp 13: Đăng kí nhập username nhập password và repassword khớp và username chưa tồn tài 🡪 **Hệ thống đăng nhâp thành công.**

### Video kiểm tra thực tế các trường hợp trên

<https://bit.ly/2BusElN>

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN