Design Manual

20th November 2022

Group 18 :

M.I.Rishard	E/17/005
A.M.F.Shalha	E/17/327
S.P.D.D.S.Weerasinghe	E/17/379

Table of contents

Introduction

"Remote Billiard" is a project which provides usual Billiard game experience via an online platform.Not all the players could be in the same place to play a billiard game nowadays. Project "Remote Billiard" solves this problem. Players can play their game physically at their own places individually.

The following diagram shows the high-level architecture for our solution.

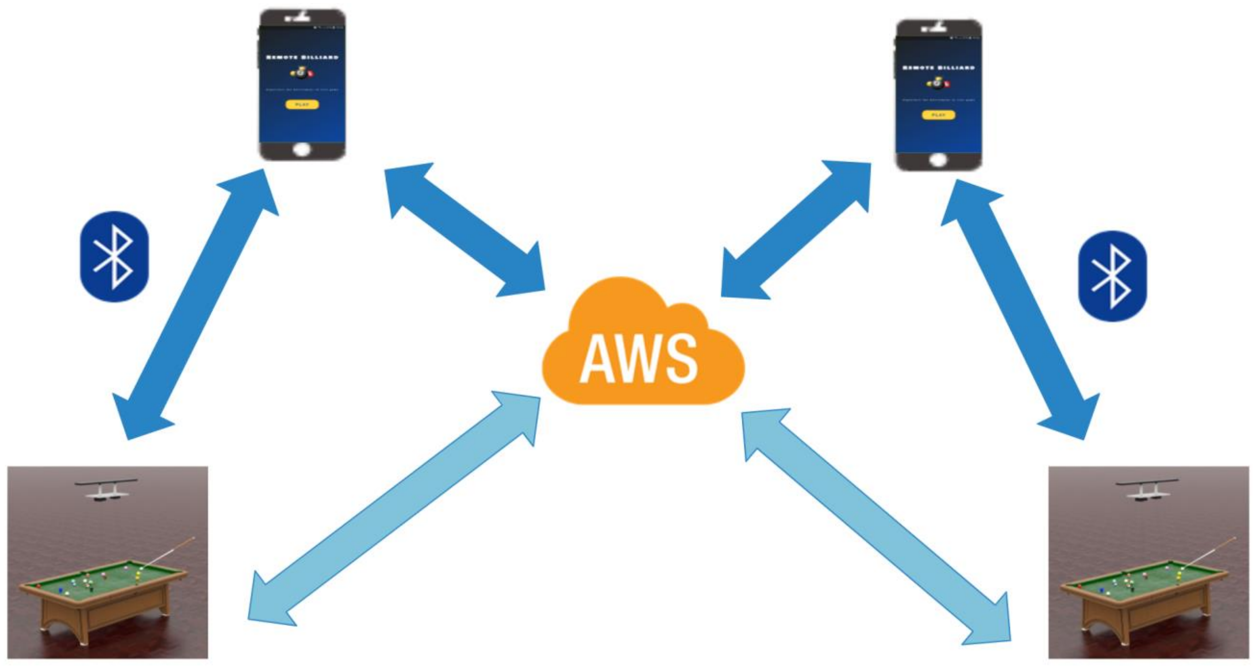


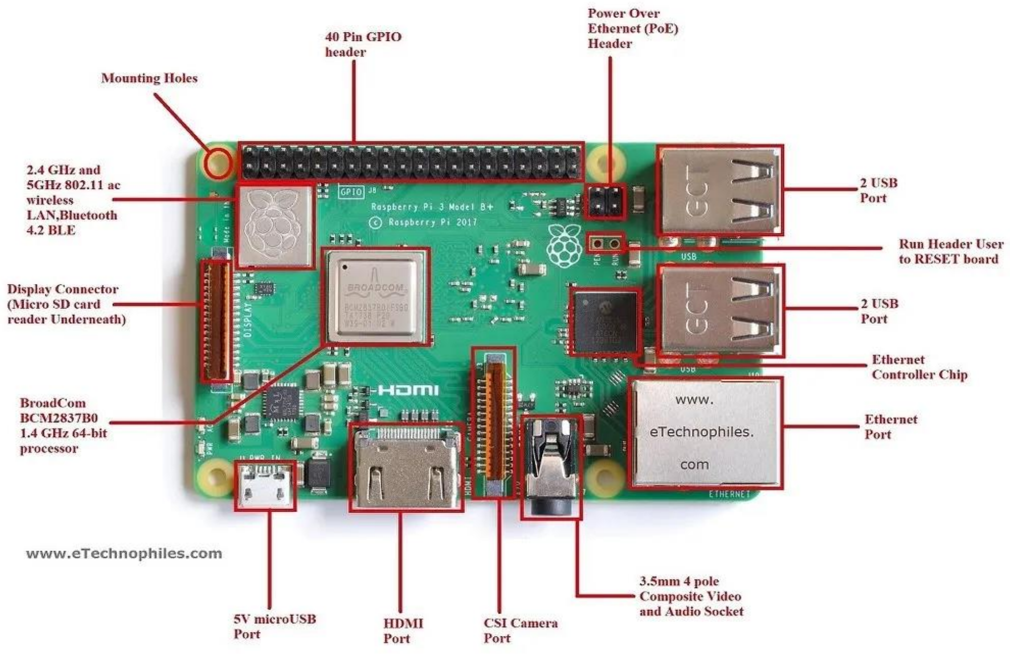
Figure 01:High level Architecture Diagram

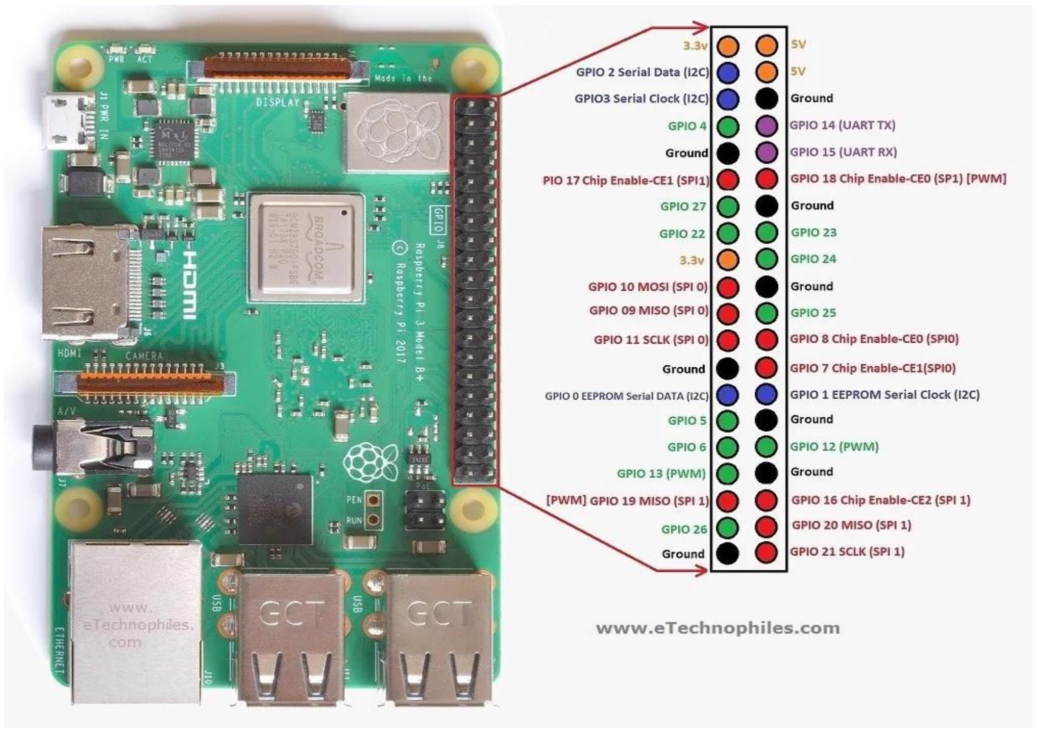
Hardware Information

Basic Components

Raspberry Pi 3 Model B+ microcontroller

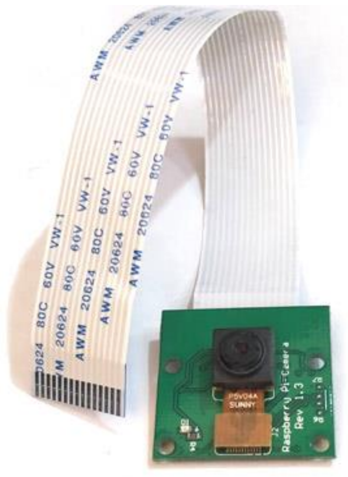
Like every other Raspberry Pi board, R-Pi 3 B+ is a single-board computer. But it has a fast and power-efficient 1.4 GHz processor (1.2GHz in model B) and a faster gigabit Ethernet (it’s limited to 300 Mbit/s by the internal USB 2.0 connection) or dual-channel 2.4 / 5 GHz 802.11ac Wi-Fi (100 Mbit/s).


40 Pin GPIO header
Power Over Ethernet (PoE) Header
Mounting Holes
2.4 GHz and 5GHz 802.11 ac wireless LAN,Bluetooth 4.2 BLE
2 USB Port
Run Header User to RESET board
Display Connector (Micro SD card reader Underneath)
2 USB Port
Ethernet Controller Chip
BroadCom BCM2837B0 1.4 GHz 64-bit processor
Ethernet Port
www.eTechnophiles.com
3.5mm 4 pole Composite Video and Audio Socket
5V microUSB Port
HDMI Port
CSI Camera Port



Raspberry pi camera module

High Definition camera module compatible with all Raspberry Pi models. Provides high sensitivity, low crosstalk and low noise image capture in an ultra small and lightweight design. The camera module connects to the Raspberry Pi board via the CSI connector designed specifically for interfacing to cameras.



In our product, this is used to capture the ball arrangement of the pool board once a player finishes his shot.
Quantity needed :1

16GB micro SD card

Micro SD card is used to provides the initial storage for the Operating System and files.

Cooling Fan

A Cooling fan is used to reduce the heat inside the main device.

LEDs

3 LED indicators are used as,

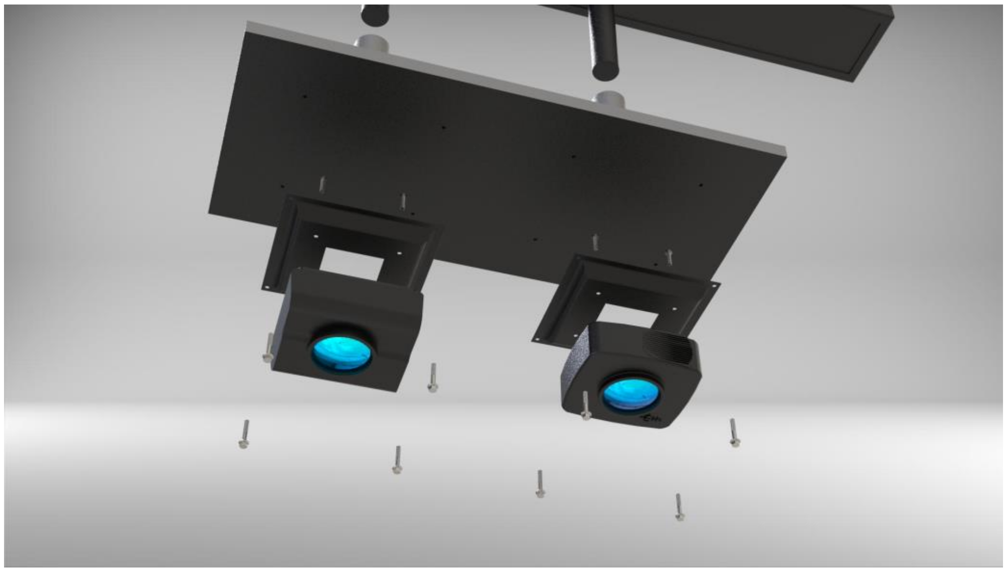
- Height Indicator
- Arrangement Indicator
- Projection area Indicator

Designs

3D Designs

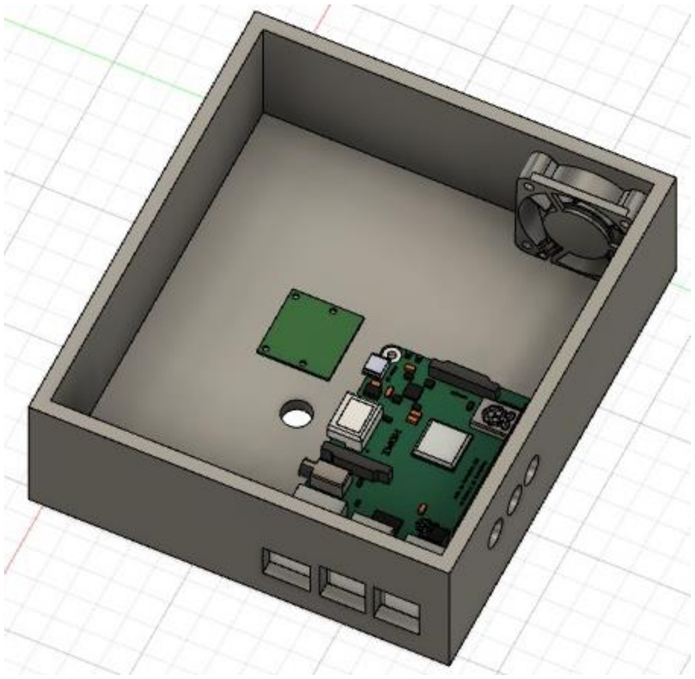
3D overview of the remote Billiard Device

Main device and the projector with height adjustable supporter

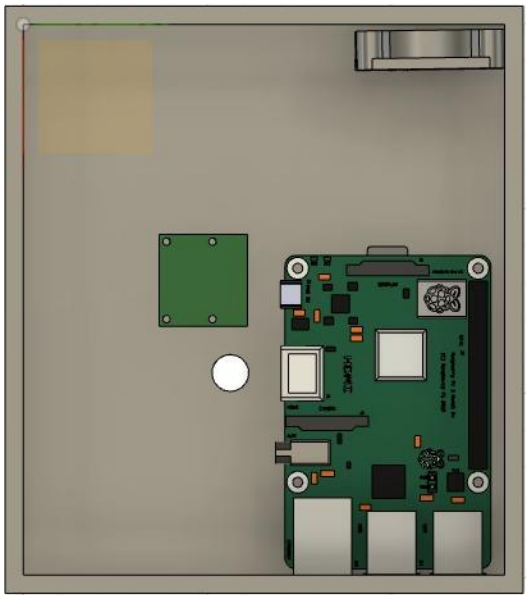


Projector

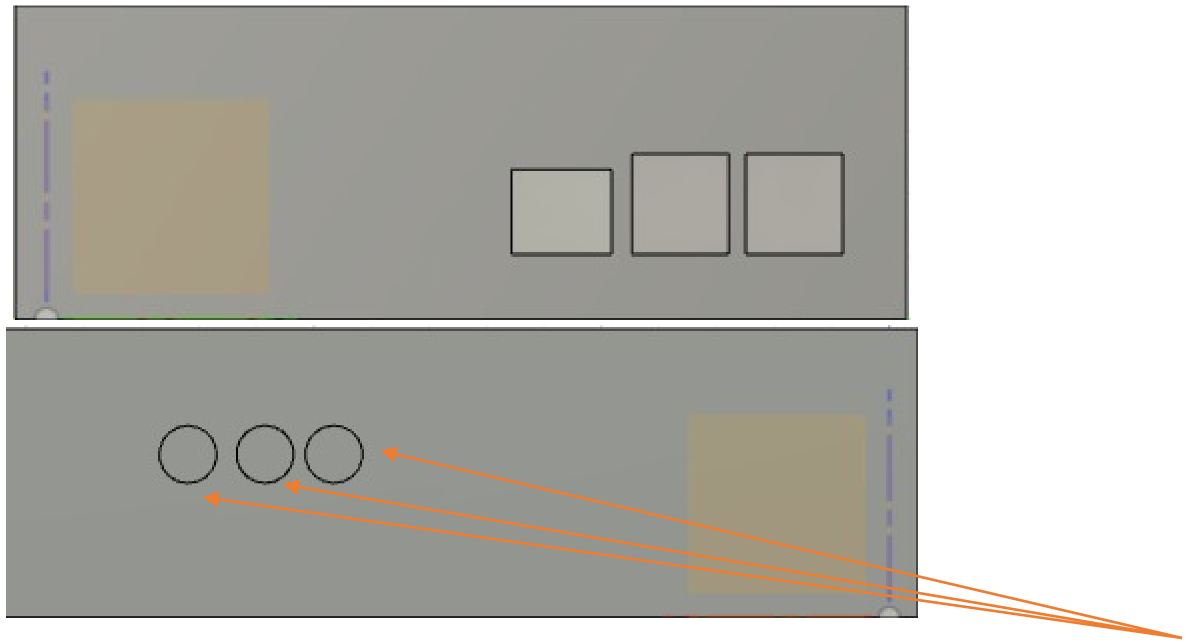
Main Device



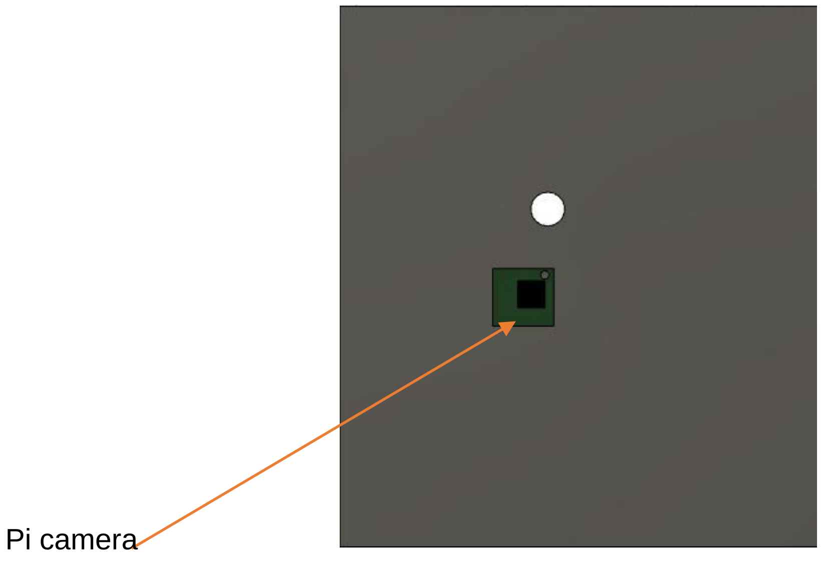
Top view of the main device



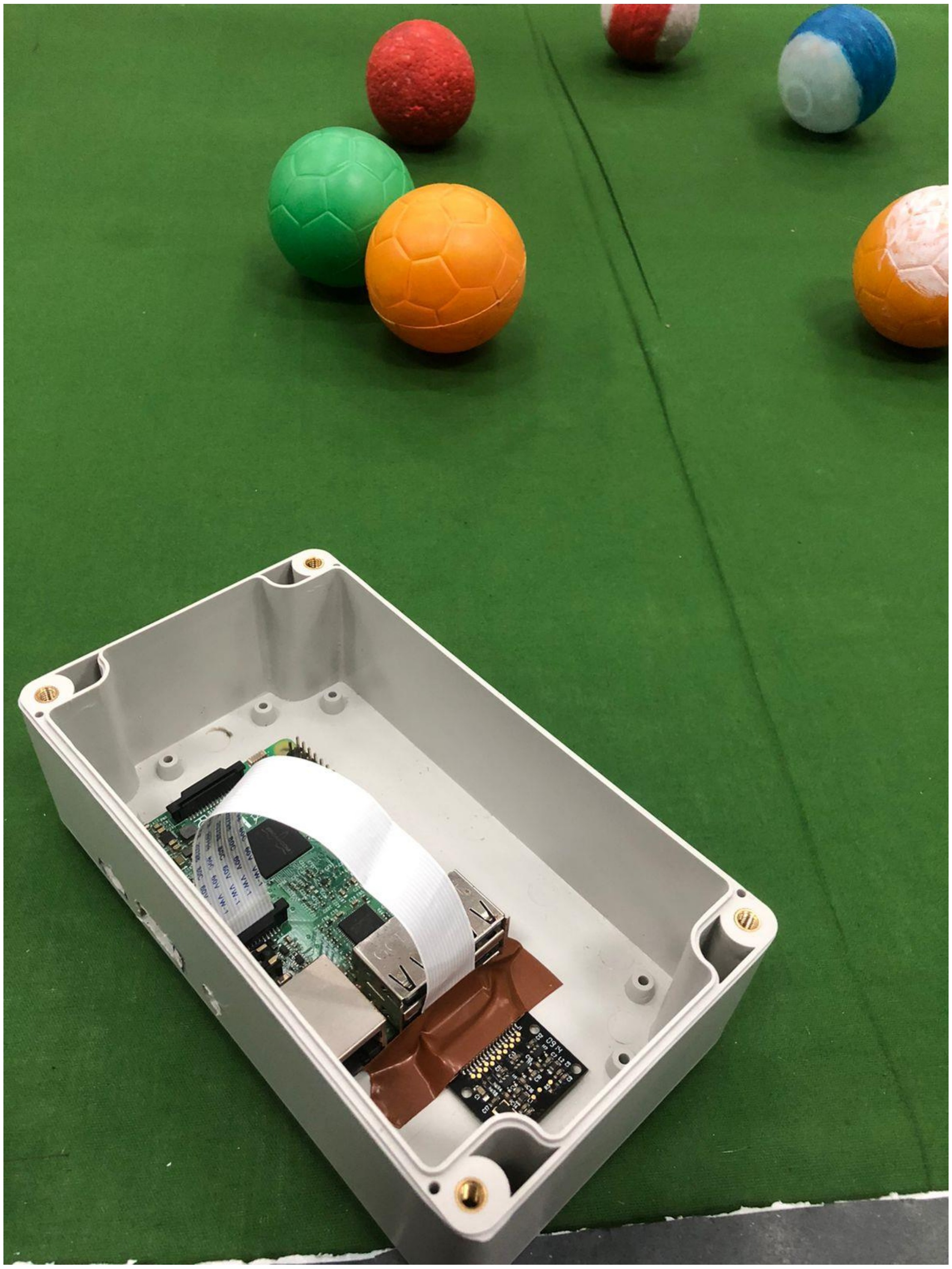
Side views of the device



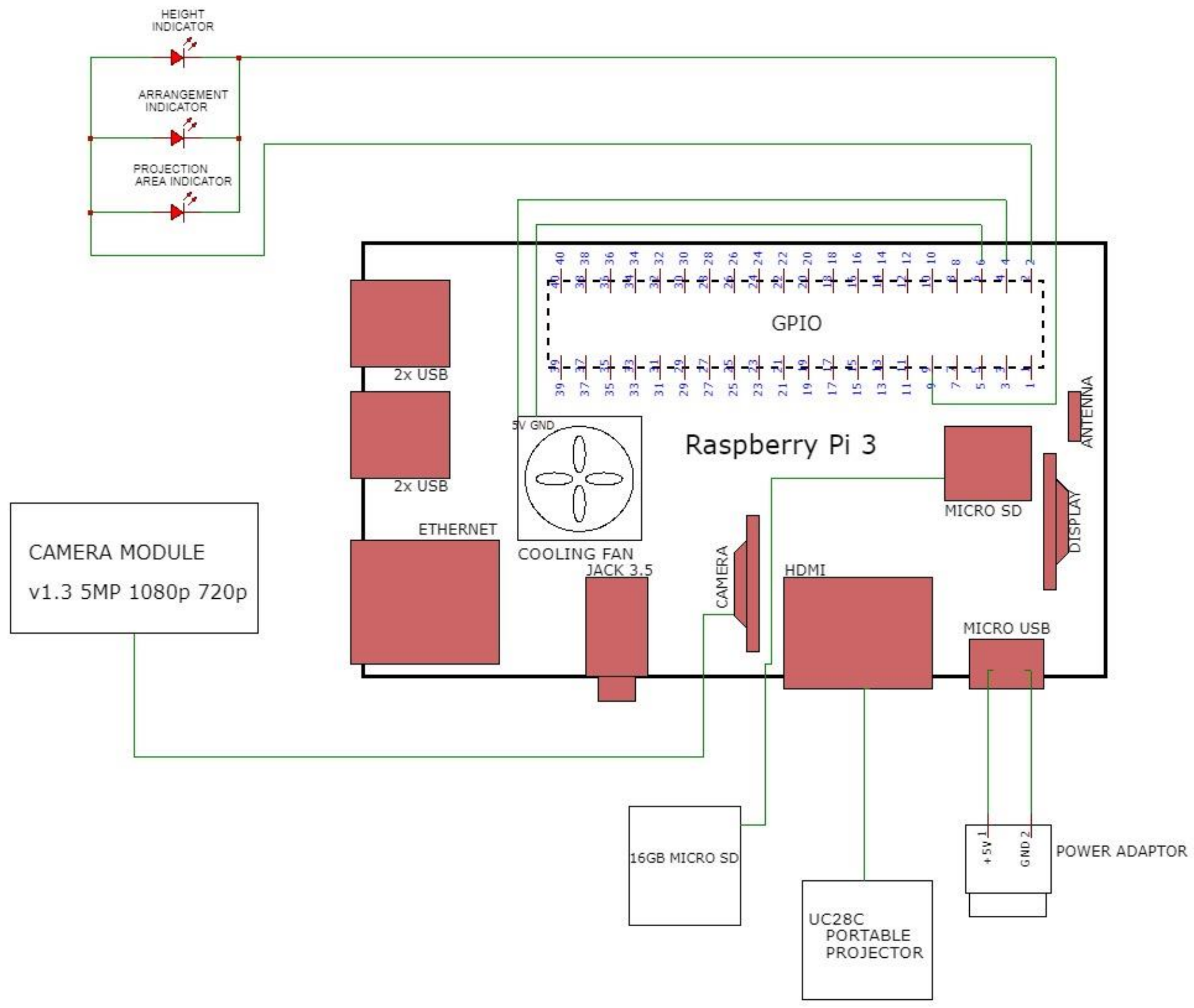
Under View of the device



Final Product



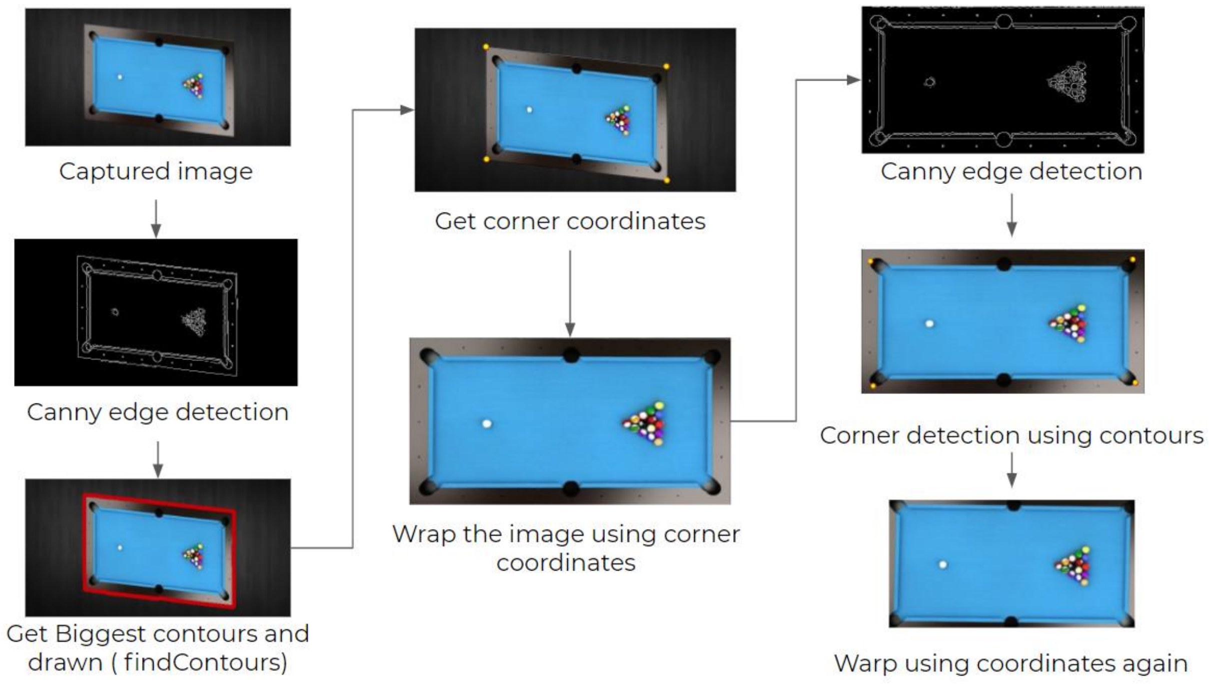
Circuit Diagram



Algorithms

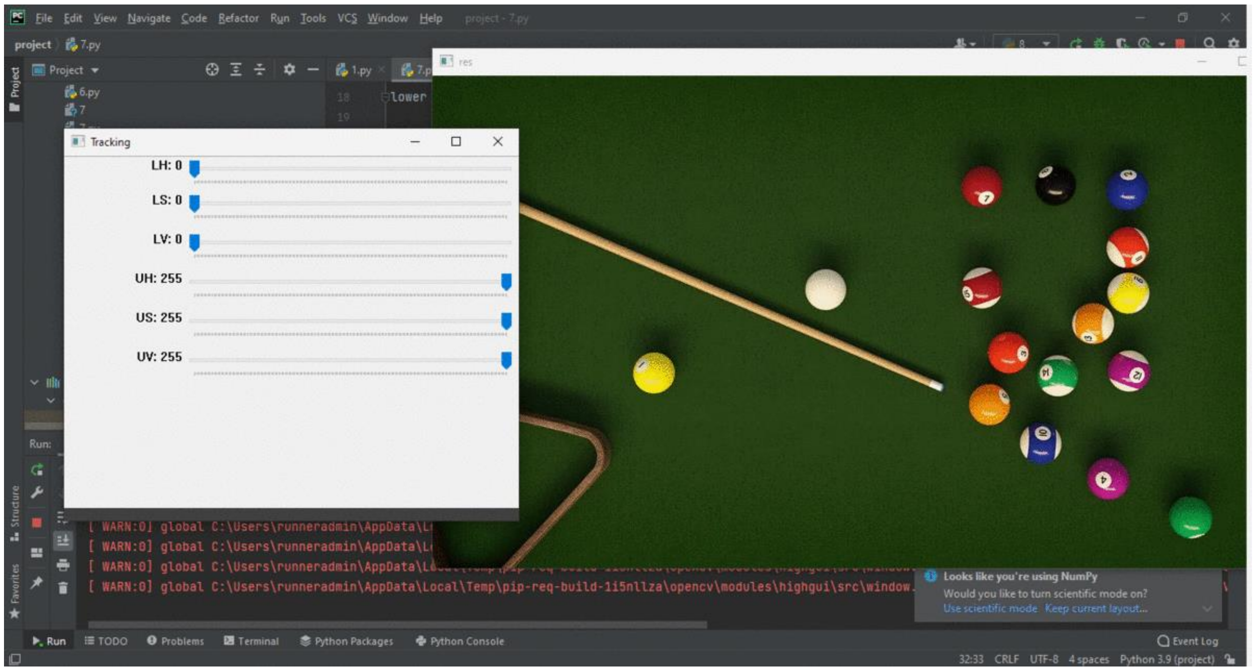
Board area isolation and wrapping

To get the board area from the captured image. Corners were identified and using corner coordinates image was wrapped. Link to the implementation


Captured image
Canny edge detection
Get Biggest contours and drawn (findContours)
Get corner coordinates
Wrap the image using corner coordinates
Canny edge detection
Corner detection using contours
Warp using coordinates again

Ball color detection isolation and generate processed image

OpenCV trackbars are used to get the accurate color levels of balls. Link to the implementation

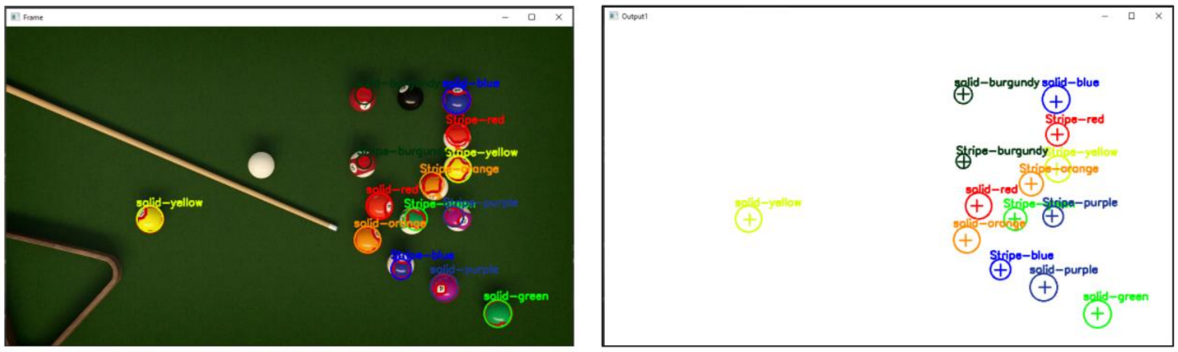


```
LH: 0
LS: 0
LV: 0
UH: 255
US: 255
UV: 255
```

```
[ WARN:0] global C:\Users\runneradmin\AppData\L
[ WARN:0] global C:\Users\runneradmin\AppData\L
[ WARN:0] global C:\Users\runneradmin\AppData\L
[ WARN:0] global C:\Users\runneradmin\AppData\Local\Temp\pip-req-build-1i5nllza\opencv\modules\highgui\src\window.
```

Looks like you're using NumPy
Would you like to turn scientific mode on?
Use scientific mode Keep current layout...

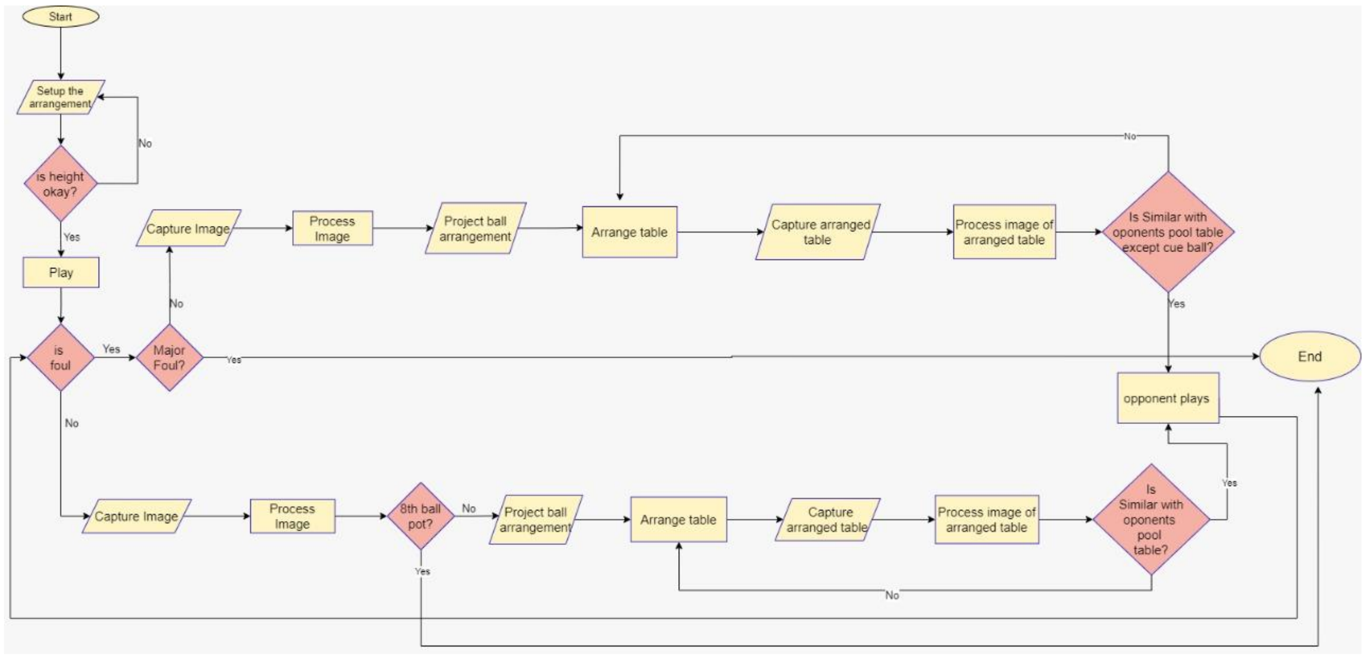
Using those HSV values ball locations were identified and locations are drawn on a white background. The final processed image is ready to send. Link to the implementation



Ball Movement Detection

To identify whether the balls still moving or stopped moving first the video was captured from OpenCV and then grabbed the video frame by frame. If the previous frame is exact with the current frame this means the video vision is still. A ball moving or not moving signal is generated using this concept.Link to the implementation

Overall Remote Billiard Device Flow Diagram



Software Information

Server

The server side was designed using node JS and the database is a Atlas No SQL MongoDB DataBase. The server is deployed on the AWS EC2 instance and the database is on Atlas Cloud. The server uses web sockets to communicate with clients. Web sockets provide a full-duplex communication channel over TCP connection.
Code for the server can be obtained from :

https://github.com/cepdnaclk/e17-3yp-remote-billiard/tree/main/server

Data Base

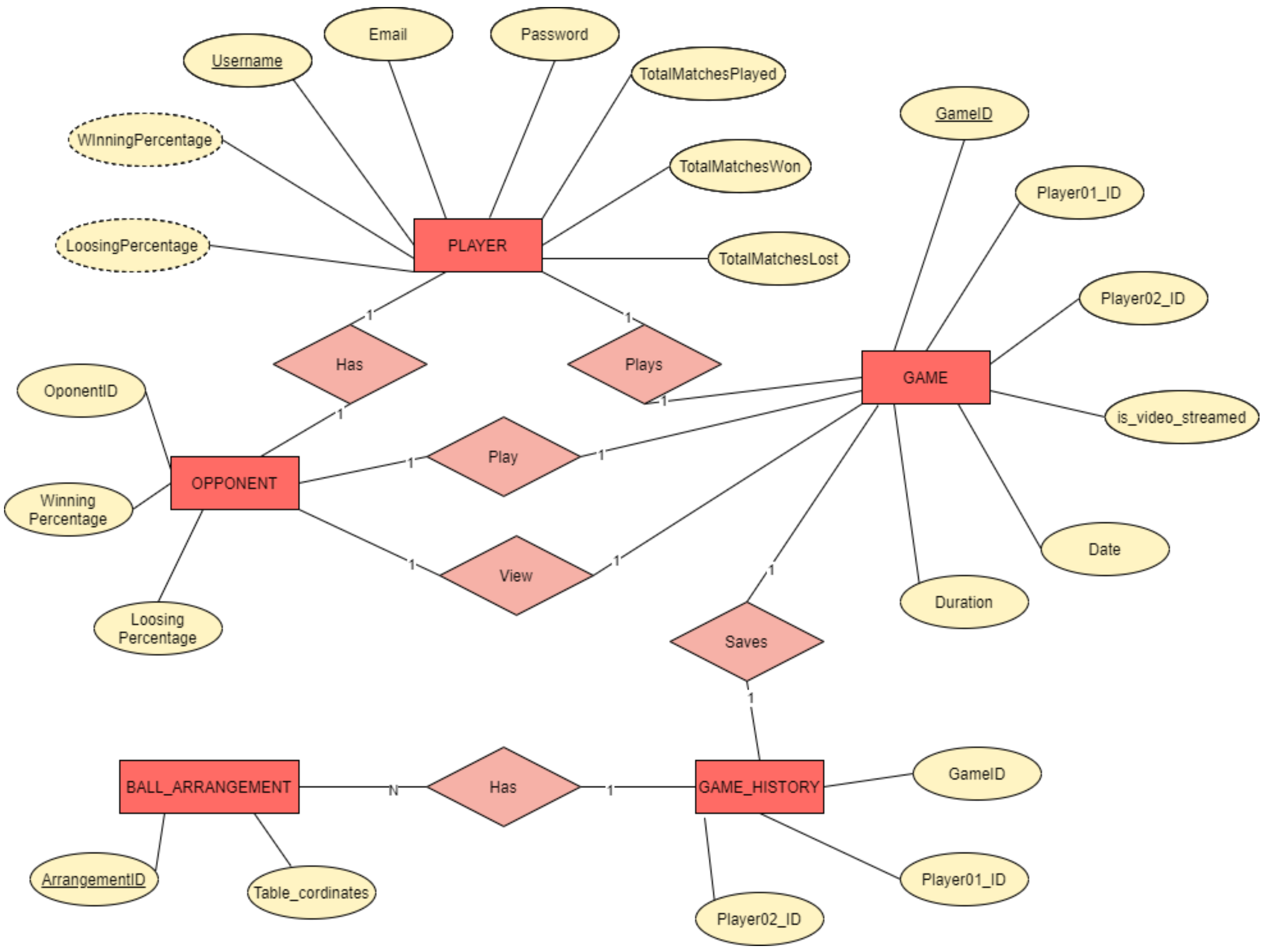
The entity-relationship model (or ER model) is shown below.

Main Entities

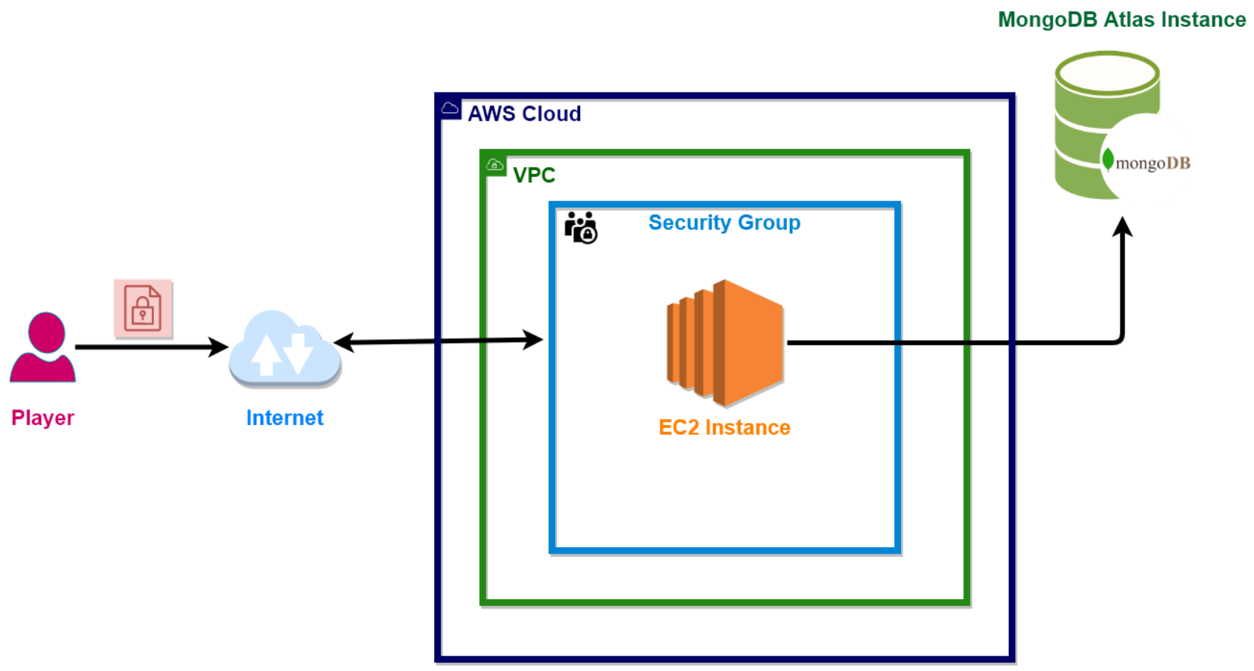
- Player
- Opponent
- Game
- Game History
- Ball Arrangement

Main Relationships

- Play
- View
- Save
- Has


Username
Email
Password
TotalMatchesPlayed
WInningPercentage
TotalMatchesWon
LoosingPercentage
PLAYER
TotalMatchesLost
GameID
Player01_ID
Player02_ID
1
1
Has
Plays
GAME
OponentID
1
1
is_video_streamed
1
Play
1
OPPONENT
Winning Percentage
1
View
1
1
Date
Loosing Percentage
Duration
Saves
1
BALL_ARRANGEMENT
N
Has
1
GAME_HISTORY
GameID
ArrangementID
Table_cordinates
Player01_ID
Player02_ID

Cloud Deployment



- Server is deployed in an EC2 instance with in a security group by creating Virtual Private Cloud.
- MongoDB database is in a MongoDB Atlas Instance

Mobile Application

The mobile application was designed using the flutter framework. It plays a huge role in our product. Users sign in, log in, connect device, registered players,find players and send invitations to the opponent, chat option,toss, choose pocket, send wait / your turn signals and call for the foul option are done with the mobile application.

The following plugins were used:

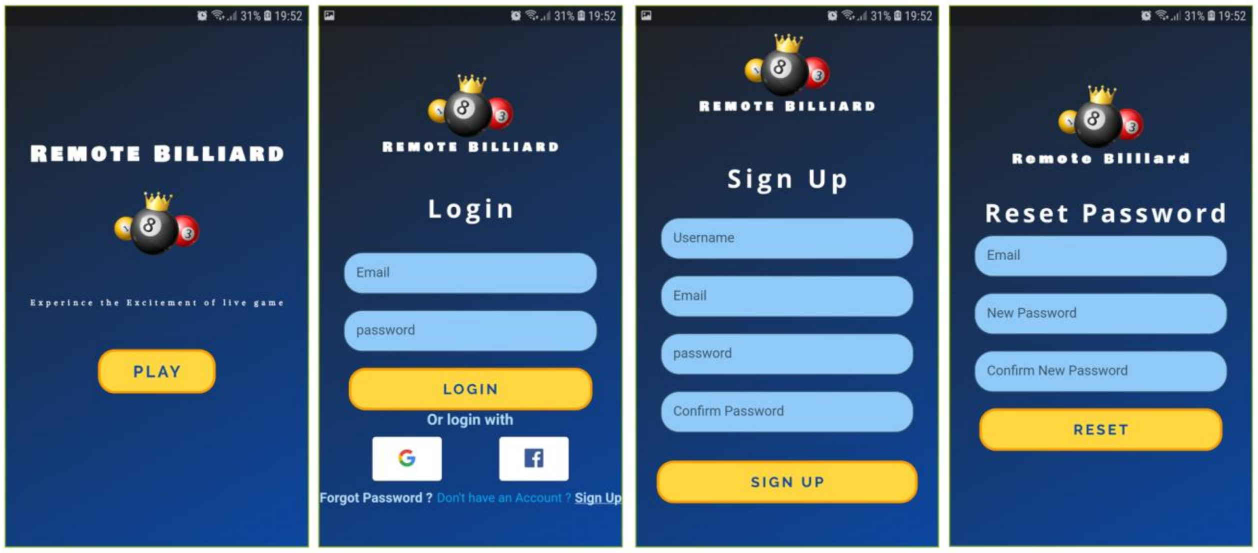
- Google people API

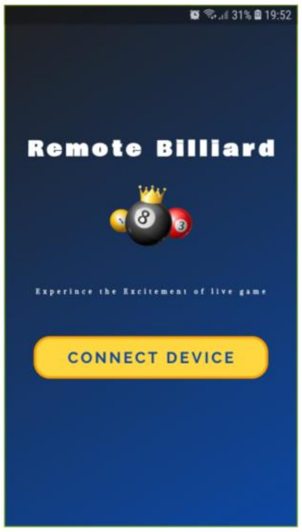
Also, the following flutter packages were used:

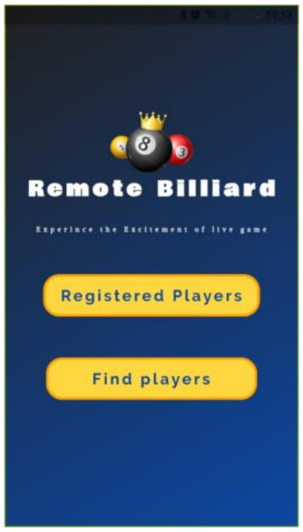
- Web_socket_channel package version 1.0.8
- Email_validator package version 2.0.1
- Flutter_bluetooth_serial package version 0.2.2
- Google_sign_in package version 5.1.0
- flutter_bluetooth_serial: ^0.4.0
- Http package version : ^0.12.1

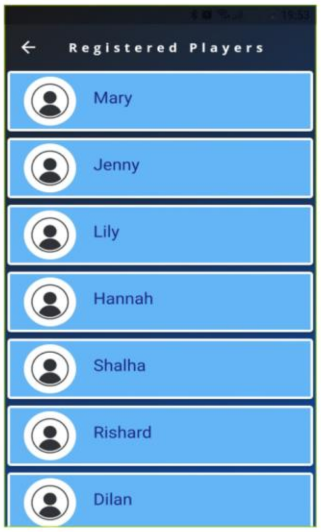
Code of the mobile App can be obtained from:

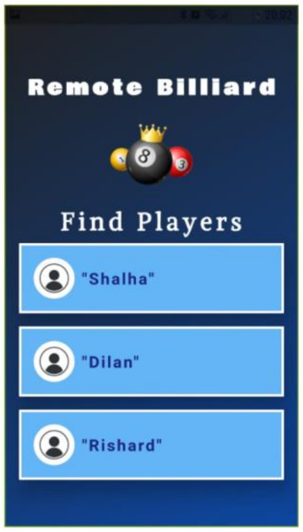
https://github.com/cepdnaclk/e17-3yp-remote-billiard/tree/main/remote-billiard

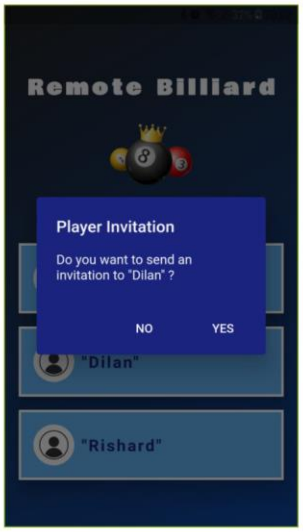




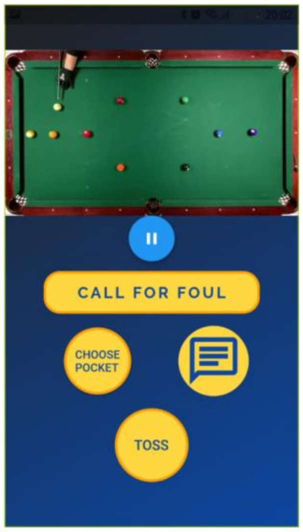


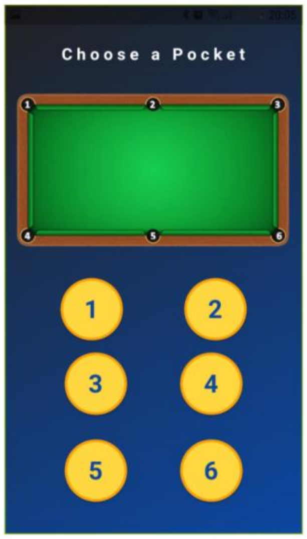


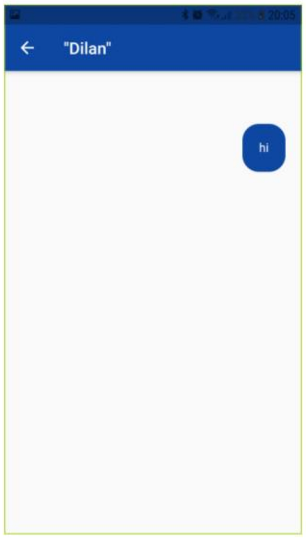




Do you want to send an invitation to "Dilan" ?

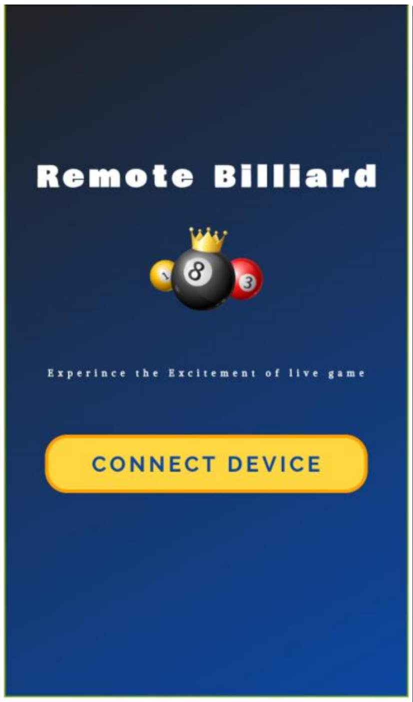


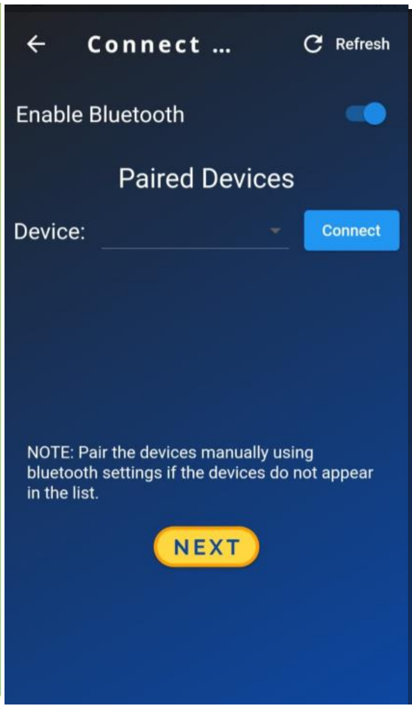




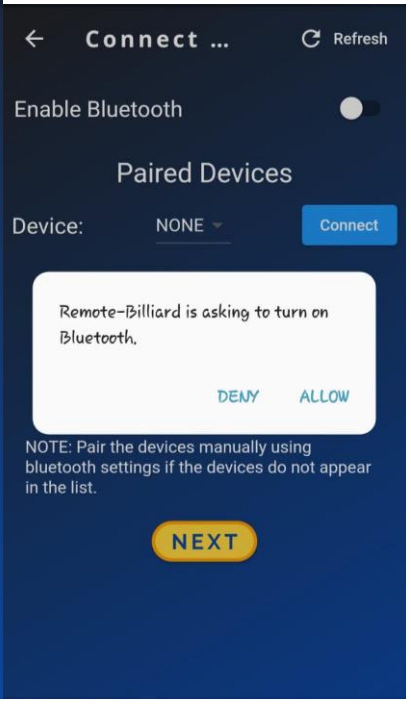
Connection between mobile application and Hardware

The connection between the mobile application and raspberry pi microcontroller is made using bluetooth.For that flutter bluetooth serial package is used in the mobile application.





NOTE: Pair the devices manually using bluetooth settings if the devices do not appear in the list.



Remote-Billiard is asking to turn on Bluetooth.

NOTE: Pair the devices manually using bluetooth settings if the devices do not appear in the list.

Tests Carried out

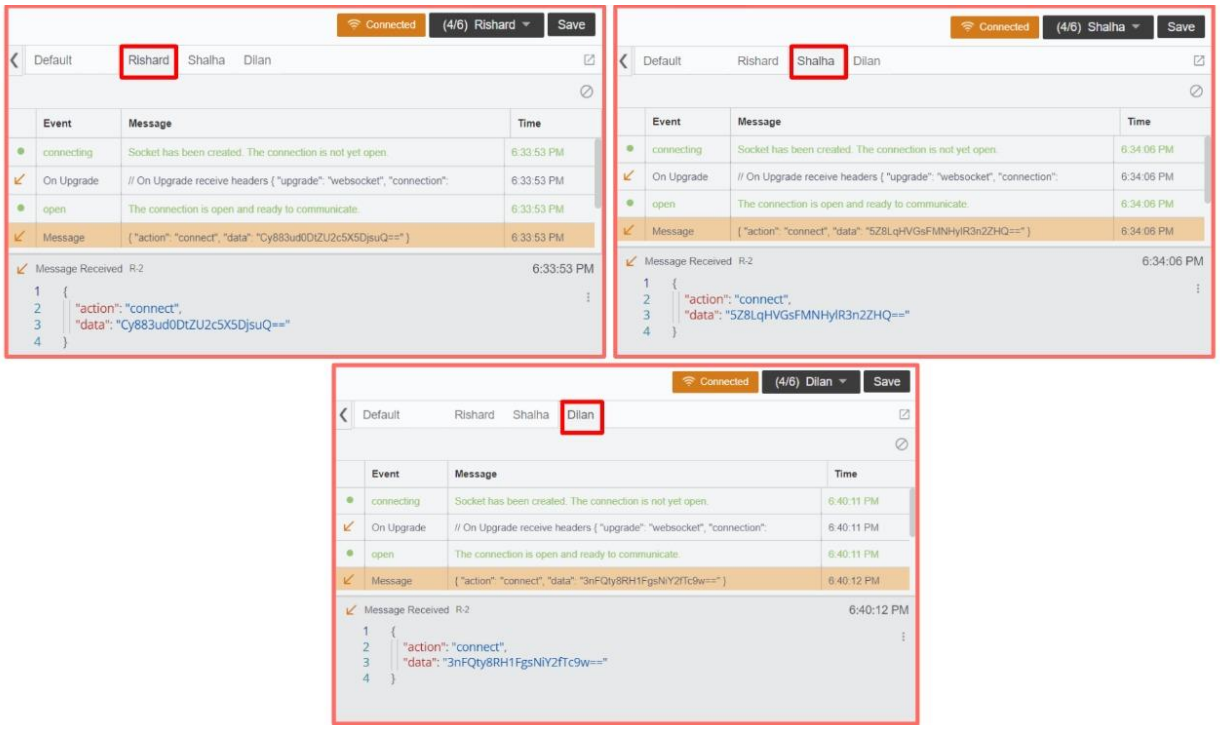
Software Testing

Mobile app

- Check the Validity of login details(email, password)

Database and Server

- Client and server connection establishment
    - **Integration Testing**



	Event	Message	Time
●	connecting	Socket has been created. The connection is not yet open.	6:33:53 PM
	On Upgrade	// On Upgrade receive headers { "upgrade": "websocket", "connection":	6:33:53 PM
●	open	The connection is open and ready to communicate.	6:33:53 PM
	Message	{ "action": "connect", "data": "Cy883ud0DtZU2c5X5DjsuQ==" }	6:33:53 PM

```
Message Received R-2                                6:33:53 PM
{
  "action": "connect",
  "data": "Cy883ud0DtZU2c5X5DjsuQ=="
}
```

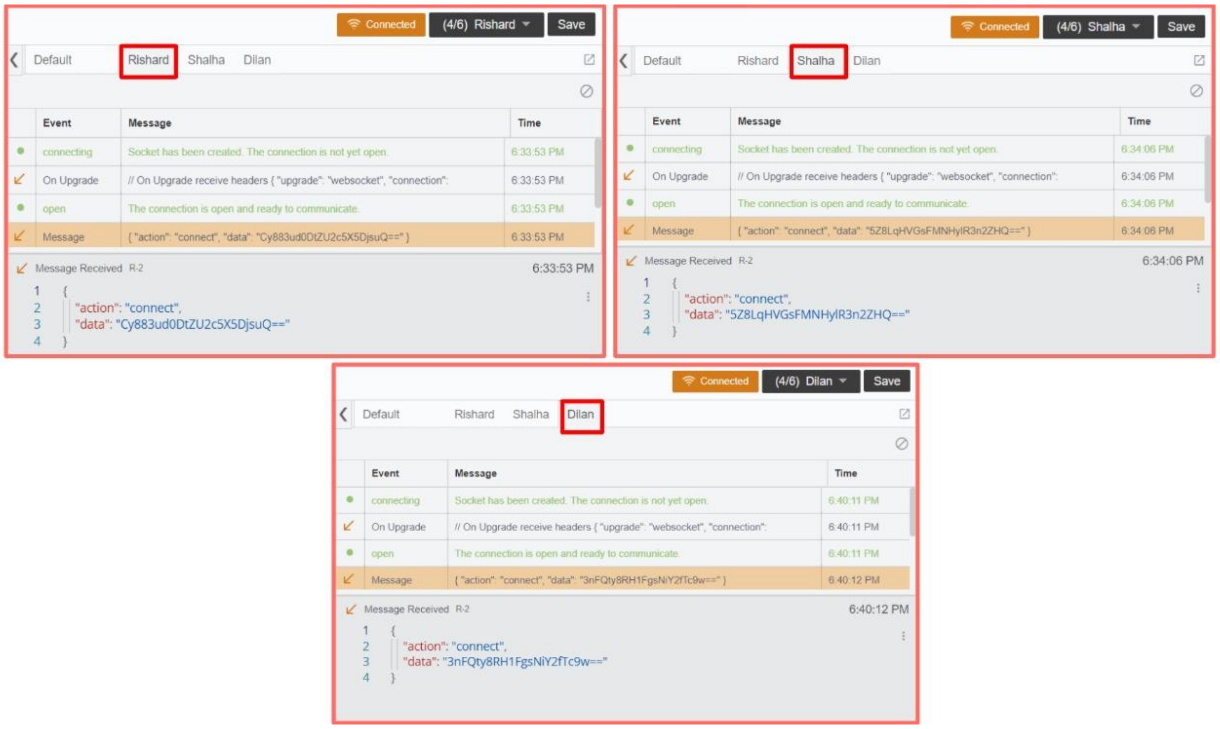
	Event	Message	Time
●	connecting	Socket has been created. The connection is not yet open.	6:34:06 PM
	On Upgrade	// On Upgrade receive headers { "upgrade": "websocket", "connection":	6:34:06 PM
●	open	The connection is open and ready to communicate.	6:34:06 PM
	Message	{ "action": "connect", "data": "5Z8LqHVGsFMNHylR3n2ZHQ==" }	6:34:06 PM

```
Message Received R-2                                6:34:06 PM
{
  "action": "connect",
  "data": "5Z8LqHVGsFMNHylR3n2ZHQ=="
}
```

	Event	Message	Time
●	connecting	Socket has been created. The connection is not yet open.	6:40:11 PM
	On Upgrade	// On Upgrade receive headers { "upgrade": "websocket", "connection":	6:40:11 PM
●	open	The connection is open and ready to communicate.	6:40:11 PM
	Message	{ "action": "connect", "data": "3nFQty8RH1FgsNiY2fTc9w==" }	6:40:12 PM

```
Message Received R-2                                6:40:12 PM
{
  "action": "connect",
  "data": "3nFQty8RH1FgsNiY2fTc9w=="
}
```

1. <u>Multiple client connection testing</u>



```
Event        Message                                                             Time
connecting   Socket has been created. The connection is not yet open.            6:33:53 PM
On Upgrade   // On Upgrade receive headers { "upgrade": "websocket", "connection":  6:33:53 PM
open         The connection is open and ready to communicate.                    6:33:53 PM
Message      { "action": "connect", "data": "Cy883ud0DtZU2c5X5DjsuQ==" }         6:33:53 PM

Message Received R-2                                                             6:33:53 PM
{
  "action": "connect",
  "data": "Cy883ud0DtZU2c5X5DjsuQ=="
}
```

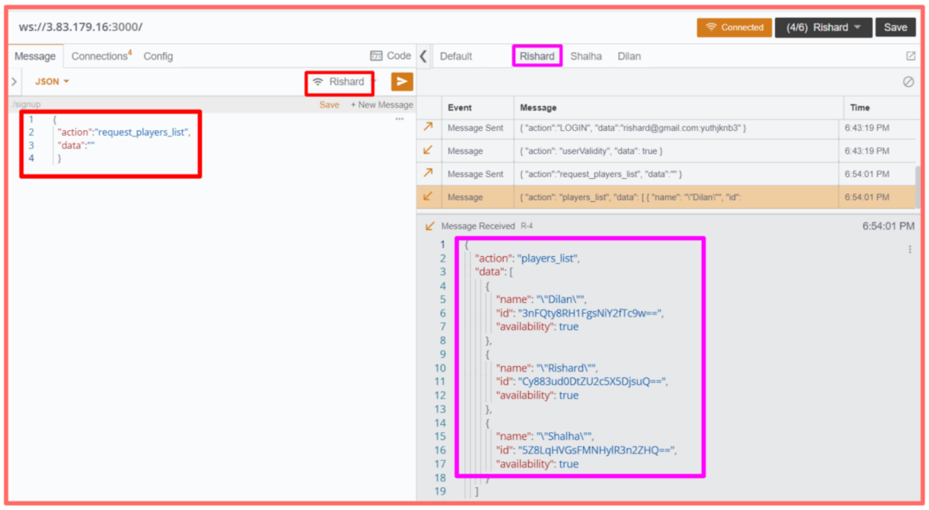
```
Event        Message                                                             Time
connecting   Socket has been created. The connection is not yet open.            6:34:06 PM
On Upgrade   // On Upgrade receive headers { "upgrade": "websocket", "connection":  6:34:06 PM
open         The connection is open and ready to communicate.                    6:34:06 PM
Message      { "action": "connect", "data": "5Z8LqHVGsFMNHyIR3n2ZHQ==" }         6:34:06 PM

Message Received R-2                                                             6:34:06 PM
{
  "action": "connect",
  "data": "5Z8LqHVGsFMNHyIR3n2ZHQ=="
}
```

```
Event        Message                                                             Time
connecting   Socket has been created. The connection is not yet open.            6:40:11 PM
On Upgrade   // On Upgrade receive headers { "upgrade": "websocket", "connection":  6:40:11 PM
open         The connection is open and ready to communicate.                    6:40:11 PM
Message      { "action": "connect", "data": "3nFQty8RH1FgsNiY2fTc9w==" }         6:40:12 PM

Message Received R-2                                                             6:40:12 PM
{
  "action": "connect",
  "data": "3nFQty8RH1FgsNiY2fTc9w=="
}
```

2. <u>SignUp and Login testing</u>



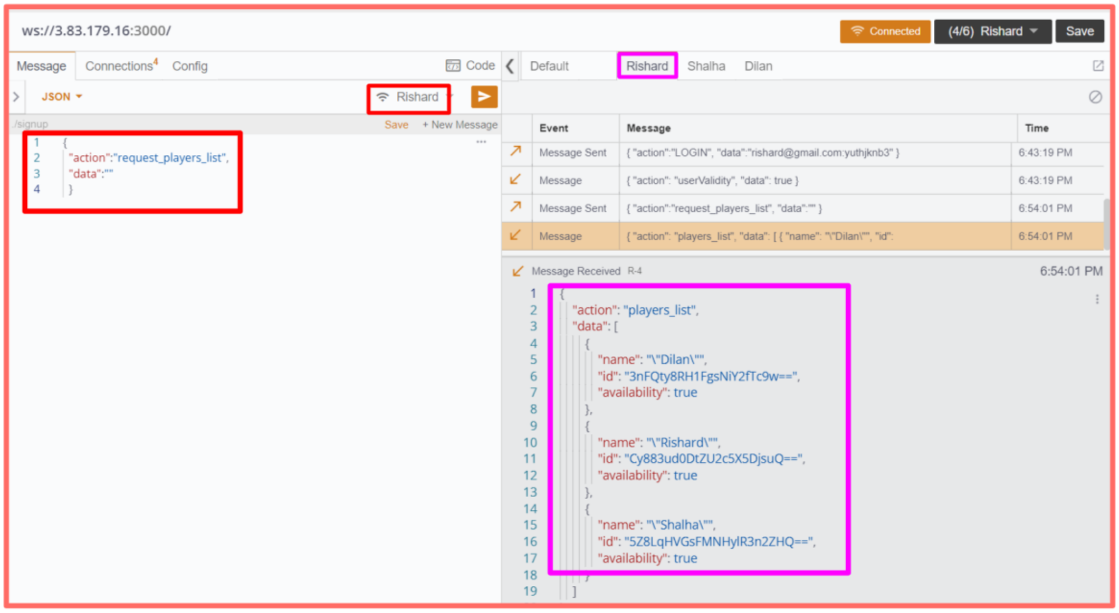
```
ws://3.83.179.16:3000/

./signup
{
"action":"request_players_list",
"data":""
}
```

```
Event         Message                                                                Time
Message Sent  { "action":"LOGIN", "data":"rishard@gmail.com:yuthjknb3" }             6:43:19 PM
Message       { "action": "userValidity", "data": true }                             6:43:19 PM
Message Sent  { "action":"request_players_list", "data":"" }                         6:54:01 PM
Message       { "action": "players_list", "data": [ { "name": "\"Dilan\"", "id":     6:54:01 PM

Message Received R-4                                                                 6:54:01 PM
{
  "action": "players_list",
  "data": [
    {
      "name": "\"Dilan\"",
      "id": "3nFQty8RH1FgsNiY2fTc9w==",
      "availability": true
    },
    {
      "name": "\"Rishard\"",
      "id": "Cy883ud0DtZU2c5X5DjsuQ==",
      "availability": true
    },
    {
      "name": "\"Shalha\"",
      "id": "5Z8LqHVGsFMNHyIR3n2ZHQ==",
      "availability": true
    }
  ]
```

<u>3.Currently online players list</u>



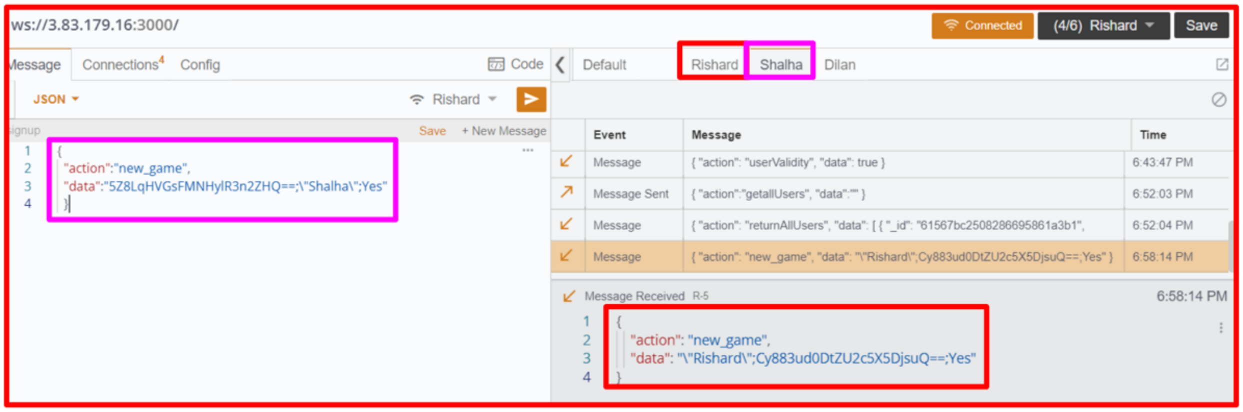
```
{
"action":"request_players_list",
"data":""
}
```

Event	Message	Time
Message Sent	{ "action":"LOGIN", "data":"rishard@gmail.com:yuthjknb3" }	6:43:19 PM
Message	{ "action": "userValidity", "data": true }	6:43:19 PM
Message Sent	{ "action":"request_players_list", "data":"" }	6:54:01 PM
Message	{ "action": "players_list", "data": [{ "name": "\"Dilan\"", "id":	6:54:01 PM

Message Received R-4 6:54:01 PM

```
{
  "action": "players_list",
  "data": [
    {
      "name": "\"Dilan\"",
      "id": "3nFQty8RH1FgsNiY2fTc9w==",
      "availability": true
    },
    {
      "name": "\"Rishard\"",
      "id": "Cy883ud0DtZU2c5X5DjsuQ==",
      "availability": true
    },
    {
      "name": "\"Shalha\"",
      "id": "5Z8LqHVGsFMNHylR3n2ZHQ==",
      "availability": true
    }
  ]
```

4.Displays all registered users



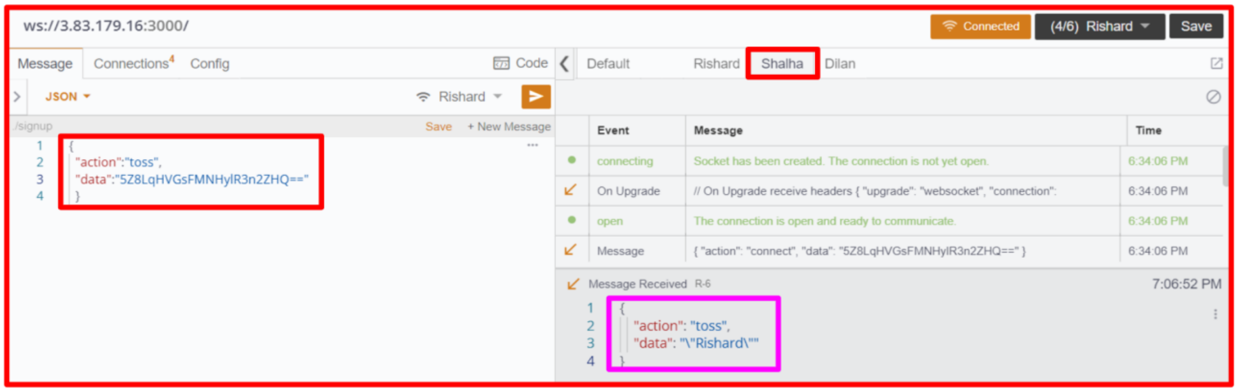
```
{
"action":"new_game",
"data":"5Z8LqHVGsFMNHylR3n2ZHQ==;\"Shalha\";Yes"
}
```

Event	Message	Time
Message	{ "action": "userValidity", "data": true }	6:43:47 PM
Message Sent	{ "action":"getallUsers", "data":"" }	6:52:03 PM
Message	{ "action": "returnAllUsers", "data": [{ "_id": "61567bc2508286695861a3b1",	6:52:04 PM
Message	{ "action": "new_game", "data": "\"Rishard\";Cy883ud0DtZU2c5X5DjsuQ==;Yes" }	6:58:14 PM

Message Received R-5 6:58:14 PM

```
{
  "action": "new_game",
  "data": "\"Rishard\";Cy883ud0DtZU2c5X5DjsuQ==;Yes"
}
```

5. Sends game invitation to another player who is online



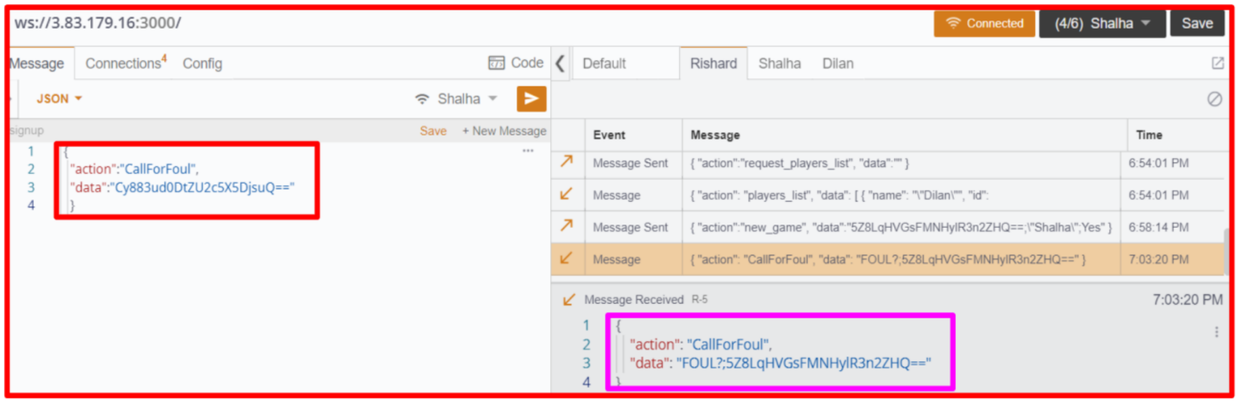
```
{
"action":"toss",
"data":"5Z8LqHVGsFMNHylR3n2ZHQ=="
}
```

	Event	Message	Time
●	connecting	Socket has been created. The connection is not yet open.	6:34:06 PM
↙	On Upgrade	// On Upgrade receive headers { "upgrade": "websocket", "connection":	6:34:06 PM
●	open	The connection is open and ready to communicate.	6:34:06 PM
↙	Message	{ "action": "connect", "data": "5Z8LqHVGsFMNHylR3n2ZHQ==" }	6:34:06 PM

Message Received R-6 7:06:52 PM

```
{
"action": "toss",
"data": "\"Rishard\""
}
```

6.Random Toss generation



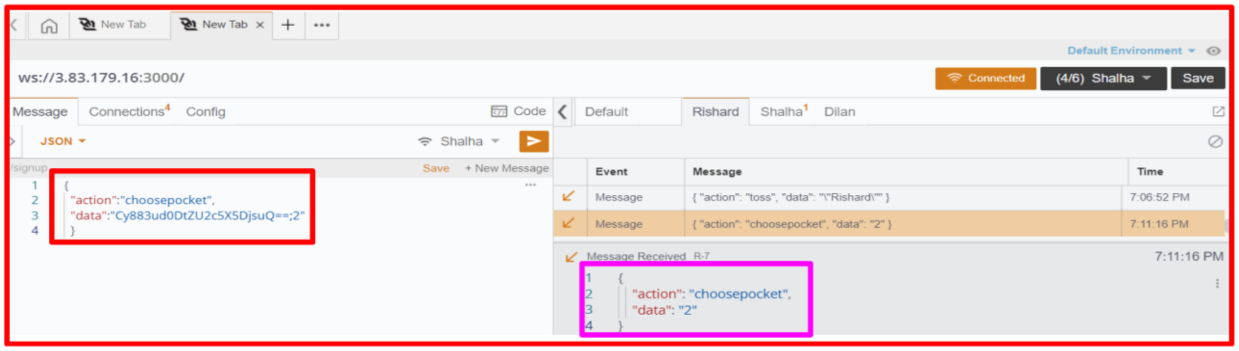
```
{
"action":"CallForFoul",
"data":"Cy883ud0DtZU2c5X5DjsuQ=="
}
```

	Event	Message	Time
↗	Message Sent	{ "action":"request_players_list", "data":"" }	6:54:01 PM
↙	Message	{ "action": "players_list", "data": [{ "name": "\"Dilan\"", "id":	6:54:01 PM
↗	Message Sent	{ "action":"new_game", "data":"5Z8LqHVGsFMNHylR3n2ZHQ==;\"Shalha\";Yes" }	6:58:14 PM
↙	Message	{ "action": "CallForFoul", "data": "FOUL?;5Z8LqHVGsFMNHylR3n2ZHQ==" }	7:03:20 PM

Message Received R-5 7:03:20 PM

```
{
"action": "CallForFoul",
"data": "FOUL?;5Z8LqHVGsFMNHylR3n2ZHQ=="
}
```

7. Foul call



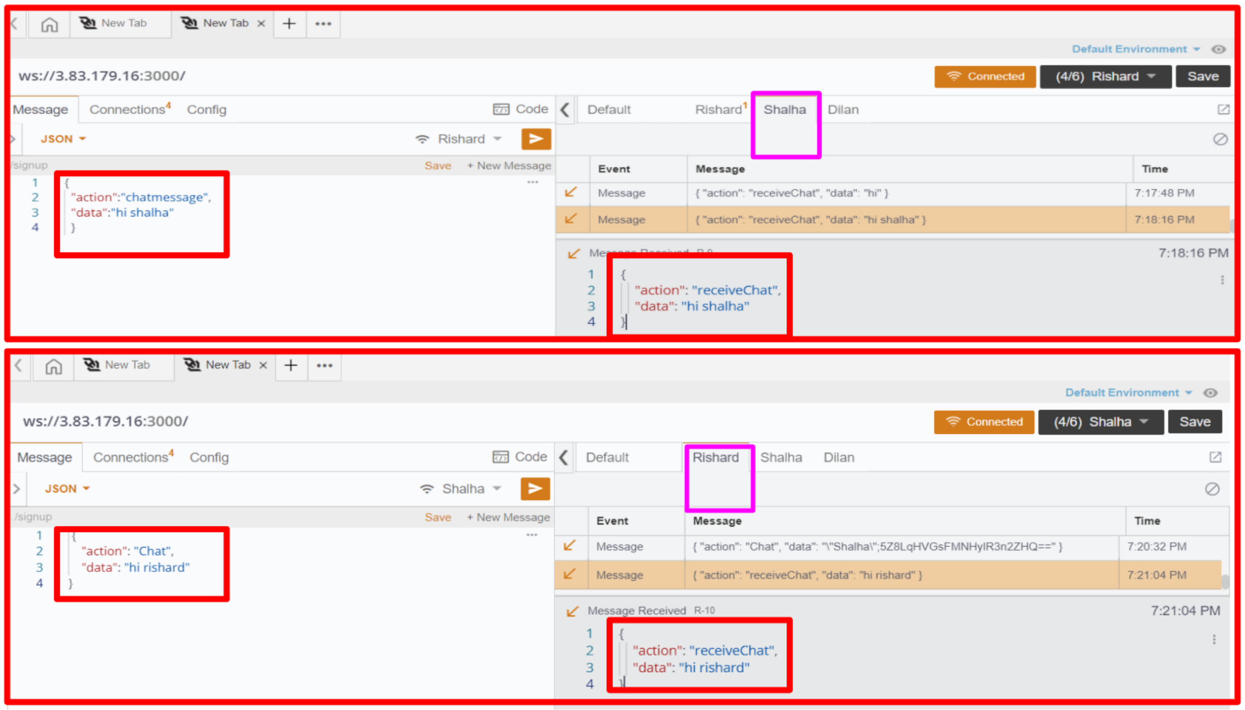
```
{
"action":"choosepocket",
"data":"Cy883ud0DtZU2c5X5DjsuQ==;2"
}
```

	Event	Message	Time
↙	Message	{ "action": "toss", "data": "\"Rishard\"" }	7:06:52 PM
↙	Message	{ "action": "choosepocket", "data": "2" }	7:11:16 PM

Message Received R-7 7:11:16 PM

```
{
"action": "choosepocket",
"data": "2"
}
```

8.Pocket selection



ws://3.83.179.16:3000/

```
{
"action":"chatmessage",
"data":"hi shalha"
}
```

Event	Message	Time
Message	{ "action": "receiveChat", "data": "hi" }	7:17:48 PM
Message	{ "action": "receiveChat", "data": "hi shalha" }	7:18:16 PM

7:18:16 PM

```
{
"action": "receiveChat",
"data": "hi shalha"
}
```

ws://3.83.179.16:3000/

```
{
"action": "Chat",
"data": "hi rishard"
}
```

Event	Message	Time
Message	{ "action": "Chat", "data": "\"Shalha\";5Z8LqHVGsFMNHyIR3n2ZHQ==" }	7:20:32 PM
Message	{ "action": "receiveChat", "data": "hi rishard" }	7:21:04 PM

Message Received R-10 7:21:04 PM

```
{
"action": "receiveChat",
"data": "hi rishard"
}
```

9. Real-time messaging between players

Embedded System Testing

- Connecting the Remote-Billiard device (pi module) to the mobile app via Bluetooth
Based on the ball movements the device sends SIGNALS to the opponents mobile application.
https://cepdnaclk.github.io/e17-3yp-remote-billiard/assets/videos/bluetooth%20final.mp4

- IoT device to IoT device communication Testing

    - Raspberry pi module is taken as one IoT device and the PC is taken as the other IoT device.
    - Both devices are subscribed to the same topic.
    - Using OpenCV implementations ball movements are detected and once the balls stop movements, the Image is captured and the raw image is processed inside pi.
    - This processed image is then published to the relevant topic. (Test: Raspberry pi to PC)

  https://cepdnaclk.github.io/e17-3yp-remote-billiard/assets/videos/rasptoiot.mp4

References

1. Raspberry Pi 3 Model B+ data sheet
https://docs.rs-online.com/a608/A700000007750677.pdf

2. Raspberry Pi 3 Model B+ product brief
https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf

3. Raspberry pi camera module :
https://cdn.sparkfun.com/datasheets/Dev/RaspberryPi/RPiCamMod2.pdf

4. Node JS Documentation : https://nodejs.dev/learn/introduction-to-nodejs

5. Npm Websocket package documentation: https://www.npmjs.com/package/websocket

6. Flutter Bluetooth Package: https://pub.dev/packages/flutter_bluetooth_serial

7. Flutter WebSocket Package: https://pub.dev/packages/web_socket_channel

8. AWS documentation: https://docs.aws.amazon.com/