chessMate

Design Manual

28th May 2021

Group 15 :

Isurika Adikari	E/16/012
Damsy De Silva	E/16/069
Chaminie De Silva	E/16/070

Table of Contents

Introduction

chessMATE is an IoT platform that provides the grand usual chess board experience to those that need online chess. It is the greatest solution we provide for many problems chess players face.

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

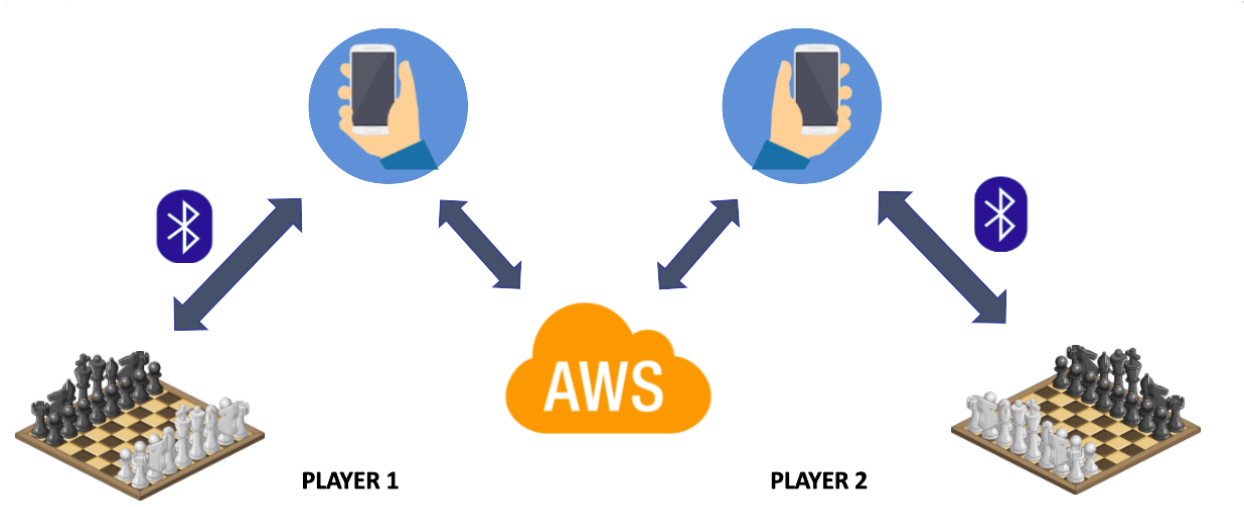


Figure 01 : High Level Architecture Diagram

Hardware Information

Basic Components

ESP32 Development Board

ESP32 is a microcontroller development board that is a product of Espressif, which is developed for the applications of IoT and wearable electronics.

It contains inbuilt Wi-Fi and Bluetooth communications modules with UART, SPI, I2C, I2S and CAN bus communication interfaces.

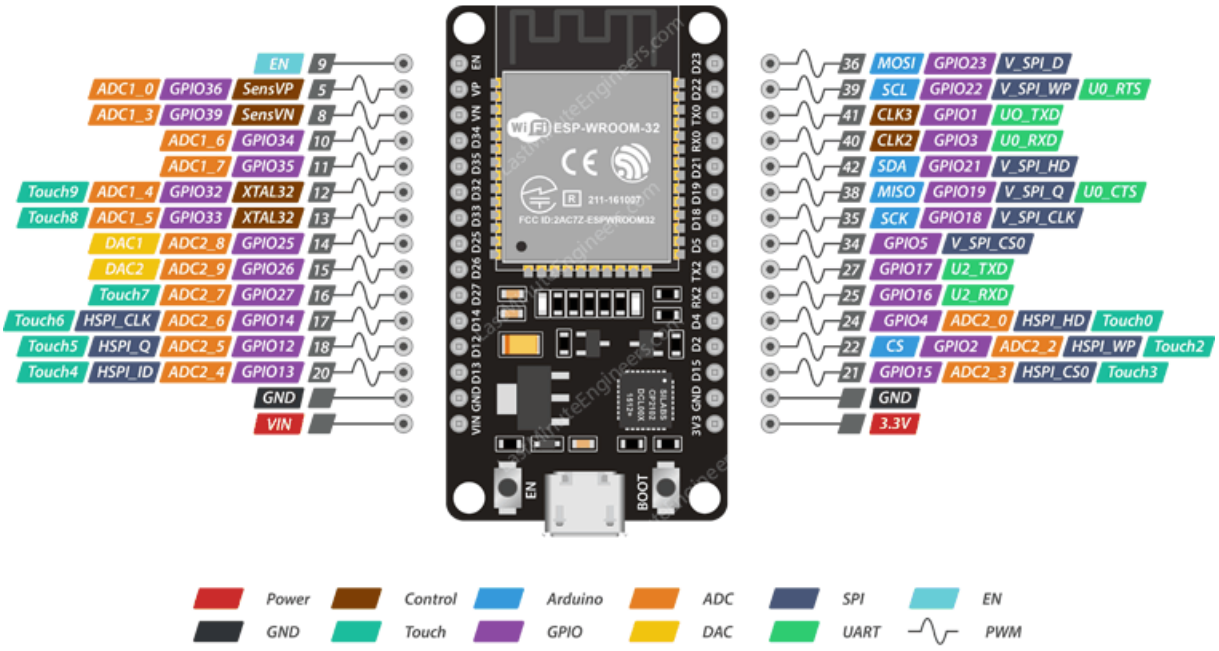


Figure 02 : ESP32 Development Board

Hall effect Sensor (A3144)

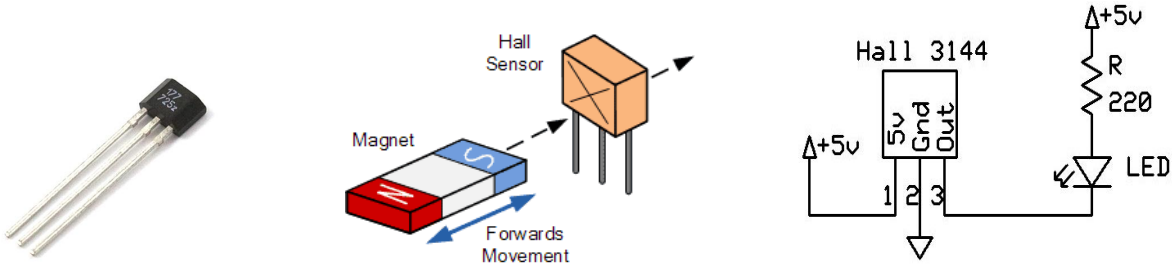


Figure 03 : Hall effect Sensor figures

The A3144 is an integrated Hall effect non-latching sensor. Holding a magnet near the sensor will cause the output pin to toggle. This makes for a robust presence sensor.

The device includes an on-chip Hall voltage generator for magnetic sensing, a comparator that amplifies the Hall voltage, and Schmitt triggers to provide switching hysteresis for noise rejection, and open-collector output.

If a magnetic flux density is larger than the threshold, the output is turned on (low). The output state is held until a magnetic flux density reversal falls below causing the output to be turned off (high).

In our product, Hall-effect Sensors are used to detect the placements and removals of magnetic-based chess pieces. (neodymium disk magnets have used in chess pieces)

Quantity Need: 64

RGB LEDs

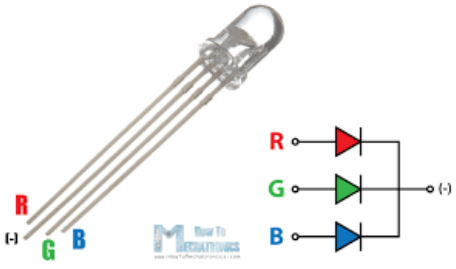


Figure 04 : RGB LED

RGB LEDs have three internal LEDs (Red, Green, and Blue) that can be combined to produce almost any colour output. To produce different kinds of colours, we need to set the intensity of each internal LED and combine the three colour outputs.

In our product, RGB LEDs are used to show the movement paths.

Quantity Need: 64

Shift Registers (74HC595)

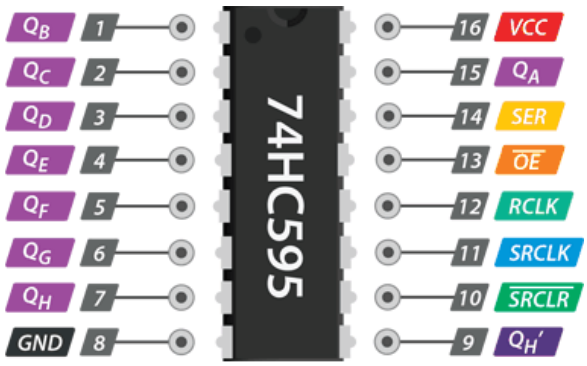


Figure 05: Shift Register

74HC595 as an "8-bit serial-in, a serial or parallel-out shift register with output latches; 3-state." In other words, you can use it to control 8 outputs at a time while only taking up a few pins on your microcontroller. You can link multiple registers together to extend your output even more.

In our product, 74HC595 shift registers are used in Serial-In Parallel-Out mode, To drive 64 RGB LEDs while using fewer pins of the ESP32 development board.
Quantity Need: 4

Data Sheet:
https://www.ti.com/lit/ds/scls041i/scls041i.pdf?ts=1622194231760&ref_url=https%253A%252F%252Fwww.google.com%252F

2n3906 PNP Transistor

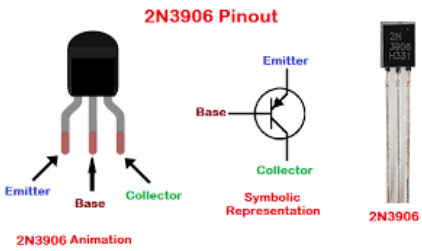


Figure 06: PNP transistor

Quantity Need: 8

74HCT138 Decoder

3-to-8 line decoder.

Data Sheet:

https://www.diodes.com/assets/Datasheets/74HCT138.pdf

Quantity Need: 1

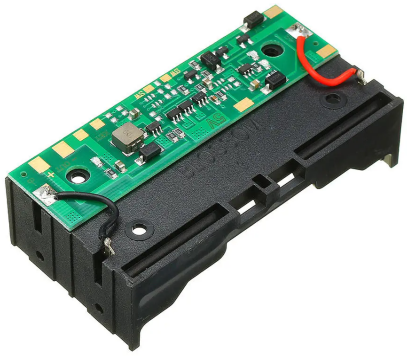
Figure 07: Decorder

Lithium-ion battery

Quantity Need: 2

Figure 08: Lithium ion battery

Lithium-ion Battery Charging Module



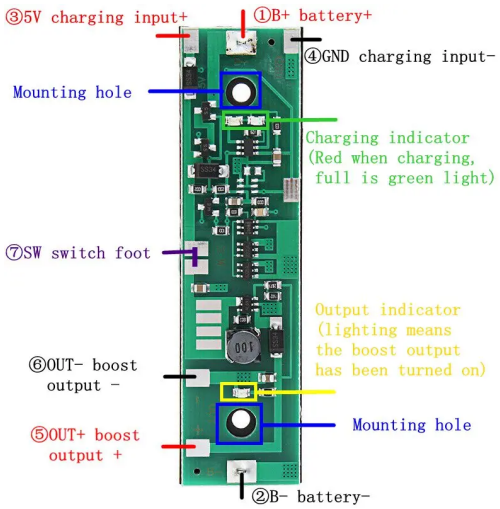
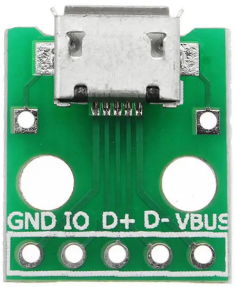


Figure 09: Lithium ion battery charging module

This module comes with Booster, charge control, Lithium Battery Protection and UPS uninterrupted power supply.

Micro USB to DIP Adapter Board



To mount the USB charging port

Figure 10: DIP Adapter

Designs

3D Designs

3D overview of the smart chessboard

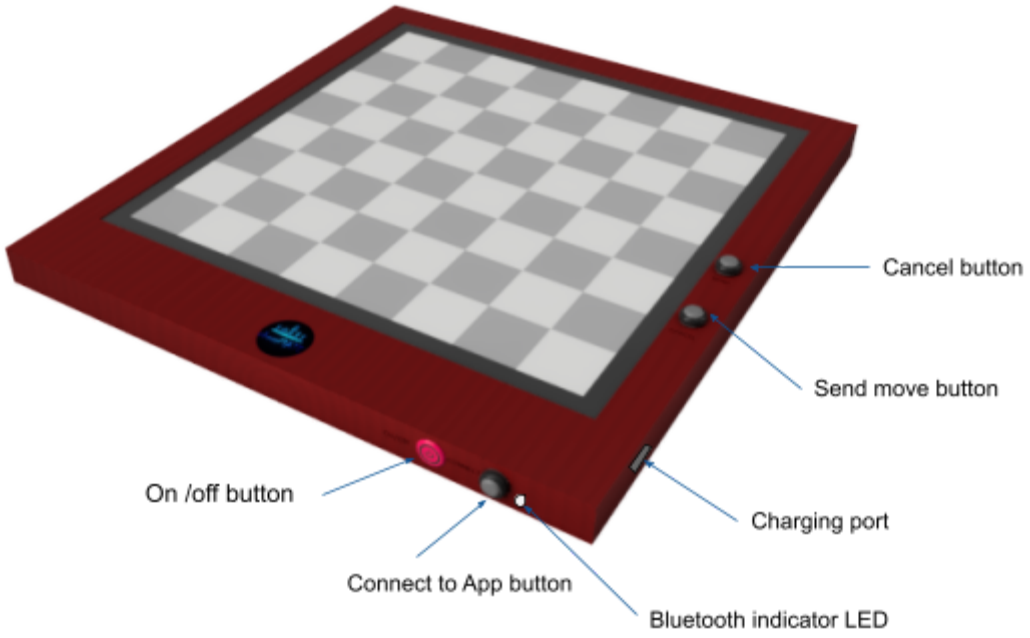


Figure 11: 3D Design overview

The following shows the 3D overview of the inner section of our chessboard. There are 64 compartments where each compartment is used by a square.

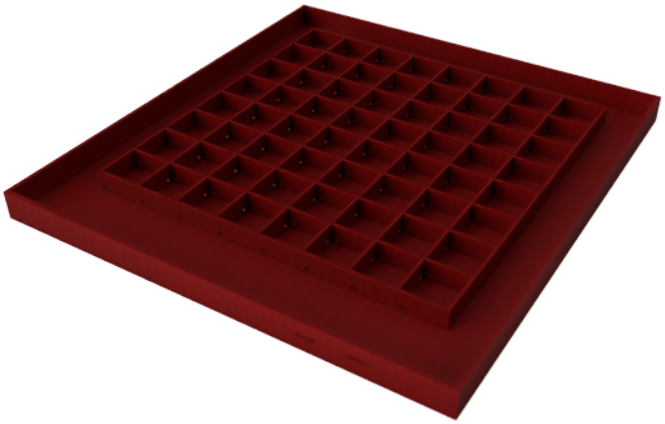


Figure 12: 3D Design overview of inner section of chessboard

Schematic Diagrams

Hall- Effect Sensor Matrix

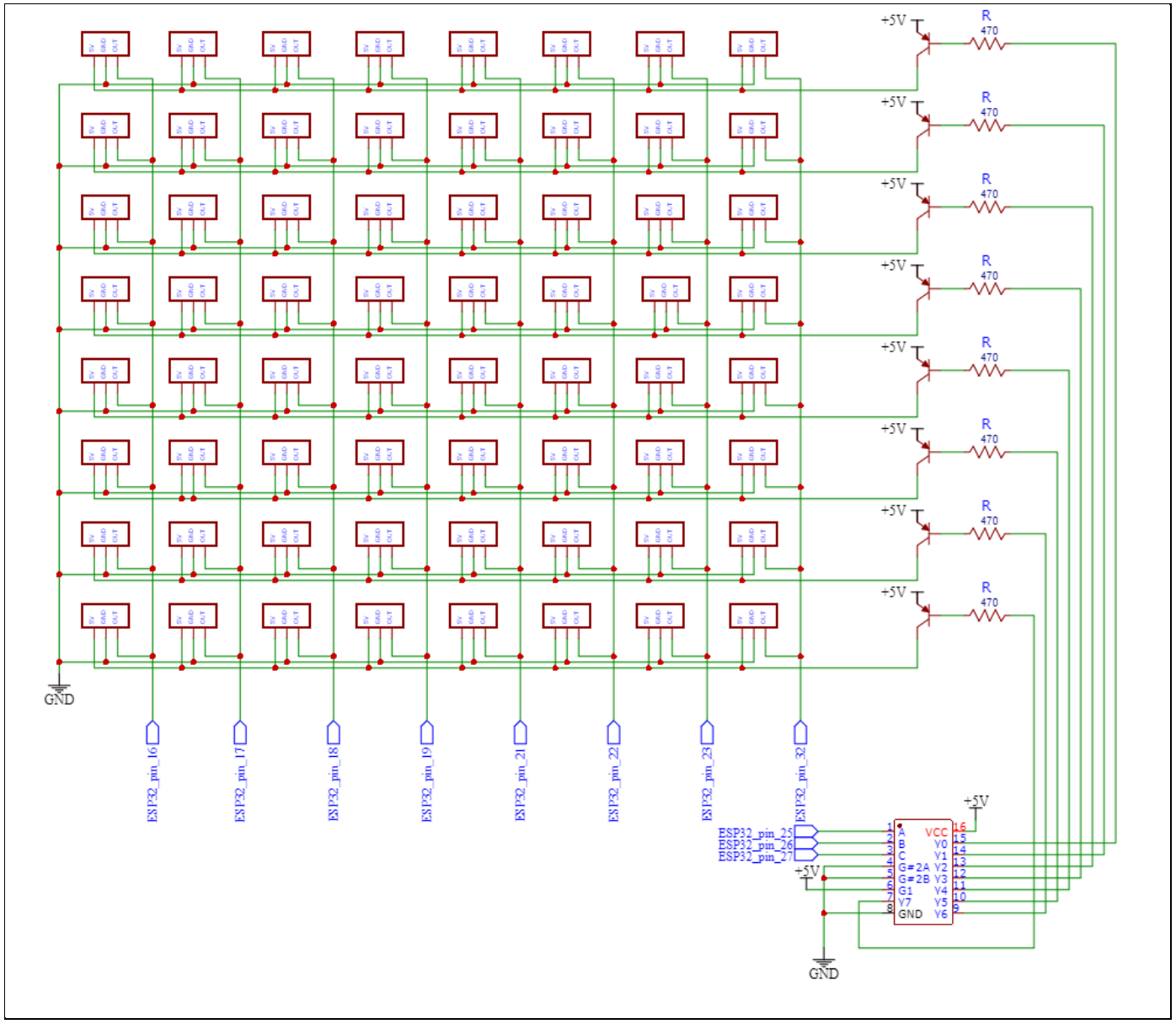


Figure 13: Hall-effect sensor matrix

- This is to detect the placing and removing of chess pieces on the chessboard.
- At each square, there is a hall effect sensor to detect the magnetic chess piece placement and removal. This leads to an 8x8 hall effect sensor matrix.
- To effectively reduce the GPIO pins needed from the ESP32 development board, a 3-to-8 line decoder(74HCT138) has been used.
- PNP transistors have been used to effectively give 5V to the Hall effect sensor matrix.

RGB LED Matrix with Chained Shift Registers

- This is an 8x8 RGB led matrix. (each square of the chessboard has an RGB LED).
- This matrix lights up the path of the opponent's move using different colours.
- Since the GPIO pins are limited, the shift registers (74HC595) are used to handle the LEDs.
- Use four 74HC595 ICs chained together with the first one attached to the 8 common anodes and the remaining 3 connected to the red, green and blue cathodes.
- So it only uses 4 GPIO pins from the development board.
- Also should connect current limiting resistors to protect the LEDs

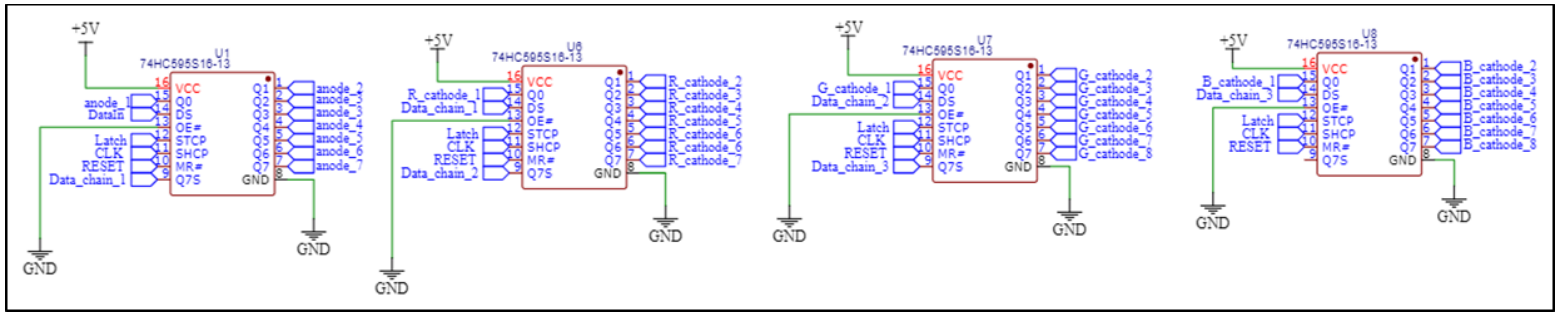


Figure 14: Chained Shift Registers

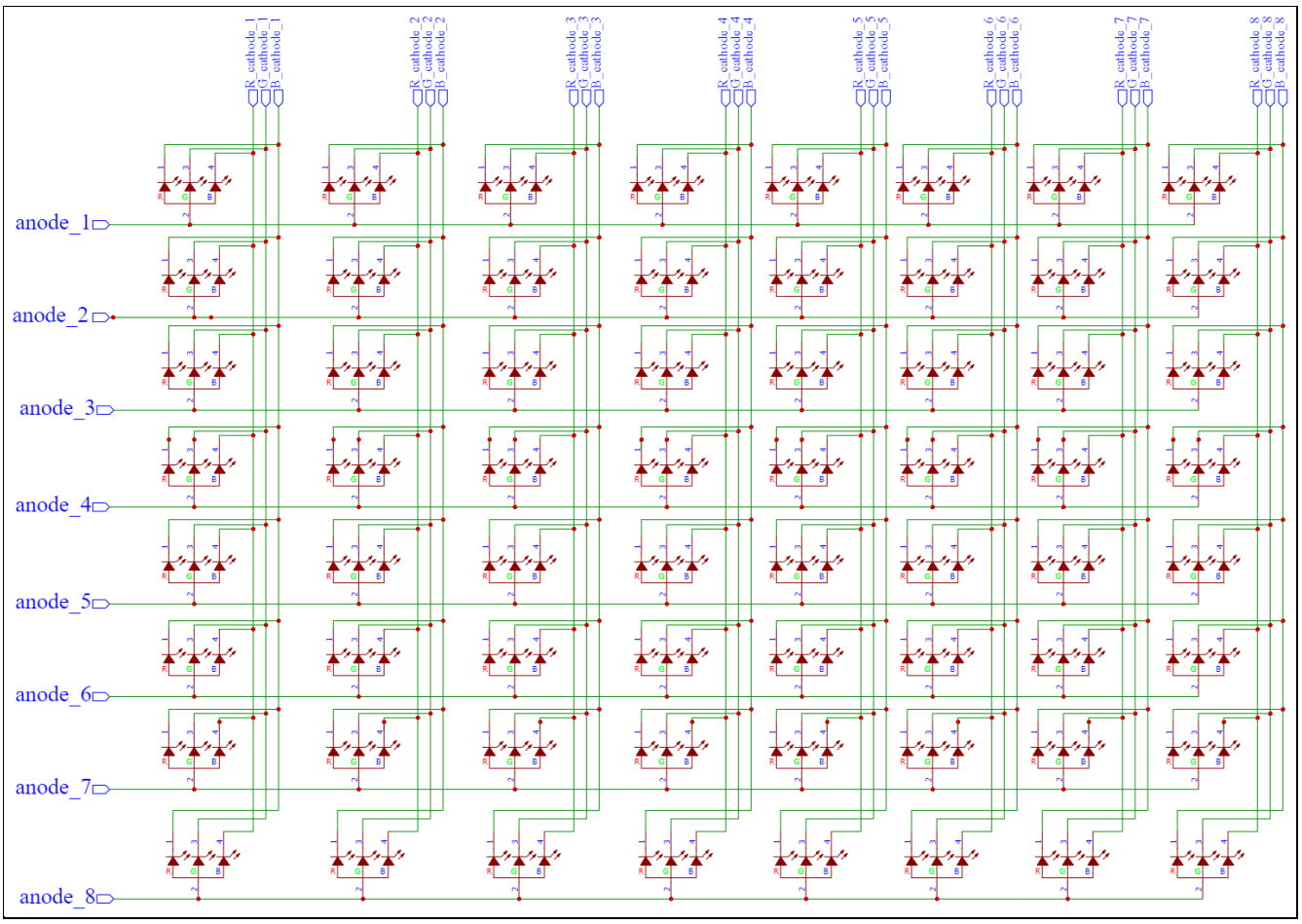
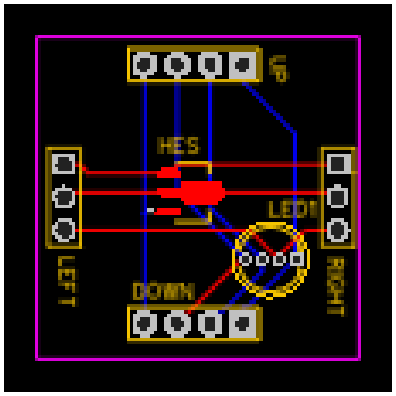


Figure 15: RGB LED Matrix

PCB Design

PCB Design for a Compartment Unit



This PCB module contains the hall effect sensor and the RGB LED corresponding to the particular square.

A module will connect with the modules placed in the adjacent compartments.

Size : 2.5 cm x 2.5 cm

64 PCBs are needed.

Figure 16: PCB for a compartment unit

PCB Design For the Main Unit

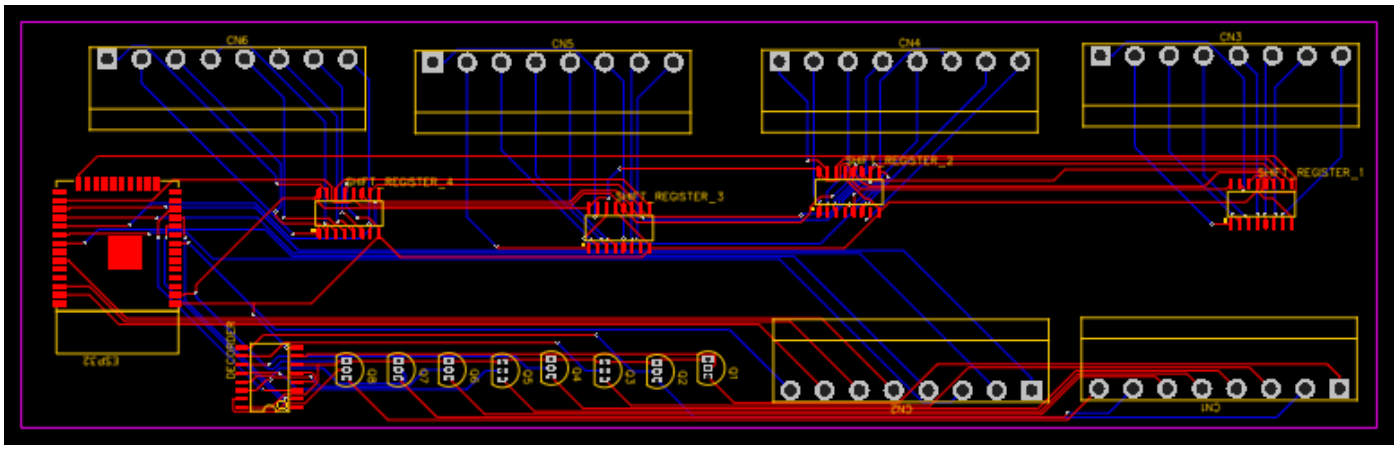


Figure 17: PCB for a main unit

This PCB contains all the components other than hall effect sensors and LEDs.

Algorithms

Movement Detection

To represent the chessboard we have used multiple 8x8 arrays and single 12x12 arrays of bytes. We have used these arrays to build up our algorithm to check if a piece has been moved and identify what piece it was. We are processing the following three possible legal moves for each turn.

- The piece moves to a legal empty space
- The piece captures an opponent's piece
- The piece is picked up and returned back to its original position

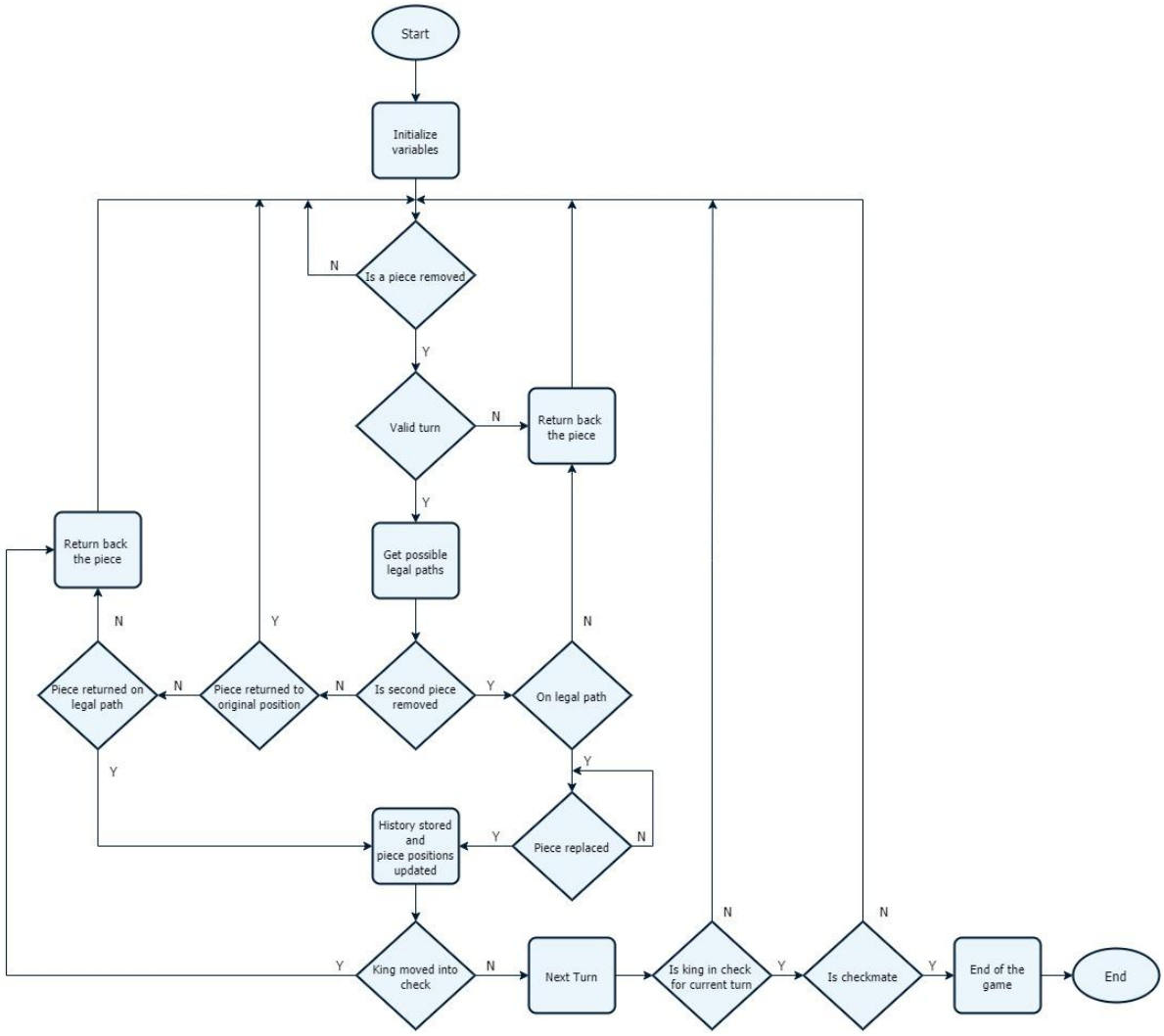


Figure 18: Flowchart of movement detection algorithm

Overall Chessboard Movements Handling

The following flowchart shows movements handling of both the player and the opponent. In the opponent's turn, the path movement done by the opponent is shown on the board using LEDs.

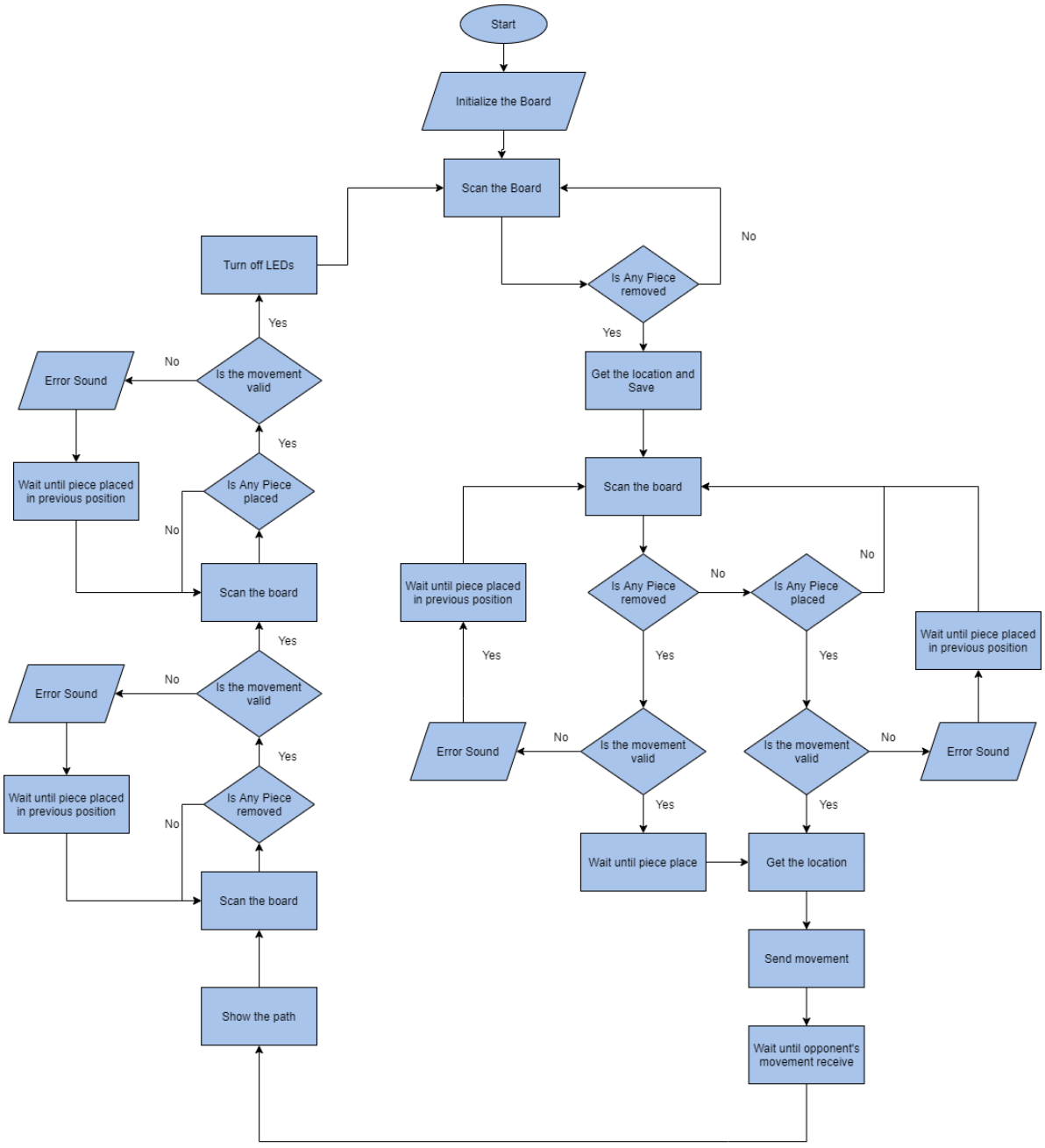


Figure 19: Flowchart of overall algorithm of chessboard

Software Information

Server

The server side was designed using node JS and the database is a simple SQL database. The server is deployed on the AWS EC2 instance and the database is deployed on AWS RDS. 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/e16-3yp-chessMATE/tree/main/Software/Server

Cloud Deployment

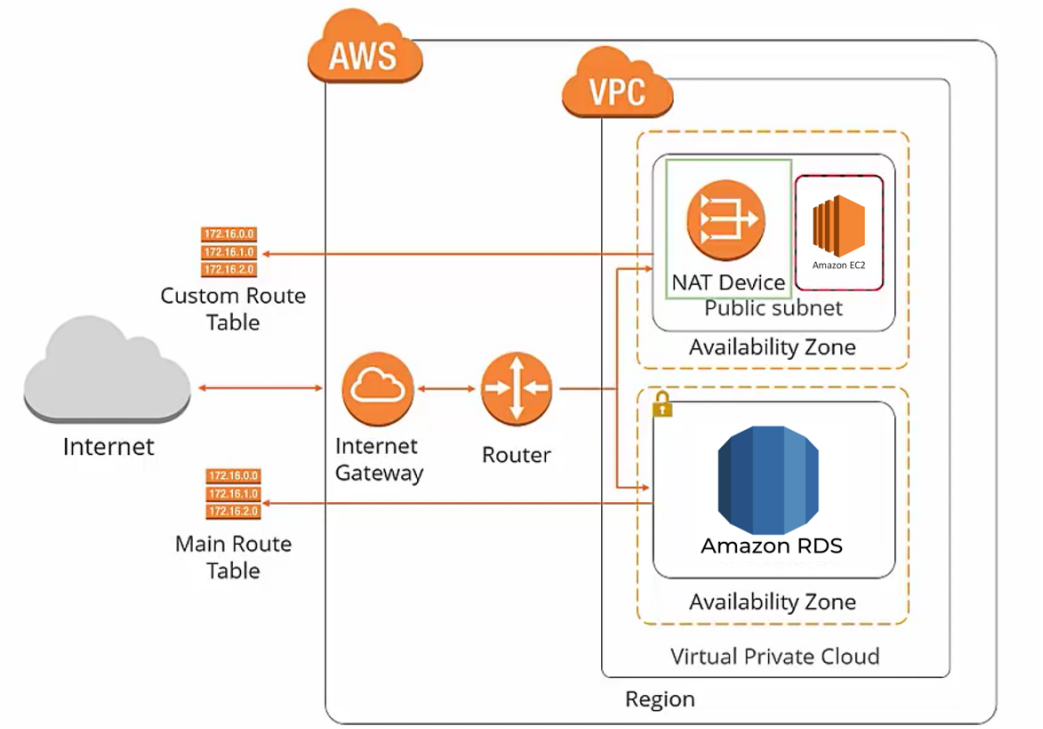


Figure 20: Structure of Cloud Deployment

- The EC2 instance is running in the public subnet and it has auto-scaling property.
- RDS is running in the private subnet and it can scale up to 64TB

Mobile Application

The mobile application was designed using the flutter framework. It plays a huge role in our product. Users sign in, log in, find players and send movements to the opponent and viewers are done with the mobile application.

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 1.0.0
- Flutter_bluetooth_serial package version 0.2.2
- Google_sign_in package version 4.0.0

Code of the mobile App can be obtained from:
https://github.com/cepdnaclk/e16-3yp-chessMATE/tree/main/Software/chessMATE_app

Tests Carried out

WHAT	TEST	WHY	PROCEDURE	TYPE
Server & Database	Client connection Establishments	Each player should be able to successfully connect to the server and get a unique id	Using Firecamp Websocket Client	Integration Test
	Get all available online users	The server should send only currently available users as the friend list		
	Check multiple games between multiple pairs of players.	The server should be able to handle a considerable number of games properly.		
	Check Database access and queries	While a player login and play the game the updates occur in the database and those updates must be valid		
Mobile App	The validity of the Email address	Users may have entered different input values therefore check the validity of user inputs is very important	Built-in Framework (Flutter)	Unit Test
	The validity of the Password			
	Check Login	Login action should be executed only with a valid email and password		
Algorithm Testing	Check chess movements validation algorithm on Esp32 board	Movement validation algorithm should be done on the Esp32 board before it sends to the opponent	Using serial communication Inputs	Algorithm Testing
Embedded Testing	Check the Bluetooth connection	The mobile app and the Esp32 board should have the proper Bluetooth connection to communicate movements	Flutter framework and Bluetooth serial communication	Embedded Testing
	Hall-effect Sensor Calibration	The 64 Hall effect sensors should be able to detect the magnetic chess pieces without disruptions	Manual Testing	
	Check the correct interpretation of movements on the LED Panel	When the opponent's movement received it should correctly interpret using the 8x8 LED Panel	Manual Testing	

Table 01: Details of Tests

References

1. ESP32 datasheet: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf
2. ESP32 Pinout Reference: Which GPIO pins should you use? | Random Nerd Tutorials
3. ESP32 bluetooth : https://www.electronicshub.org/esp32-bluetooth-tutorial/
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/