DEVELOPMENT OF AN APPLICATION FOR STREAM DECK CONTROLLER USING EMBEDDED SYSTEM

CHAI CHUN WEI

A project report submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering (Hons) Electronic Engineering

Faculty of Engineering and Green Technology
Universiti Tunku Abdul Rahman

May 2023

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature	:	[illegible]
Name	:	CHAI CHUN WEI
ID No.	:	18AGB05194
Date	:	10/5/2022

APPROVAL FOR SUBMISSION

I certify that this project report entitled **“Application for Webserver Stream Deck Controller using Embedded System”** was prepared by **CHAI CHUN WEI** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons) Electronic Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	[illegible]
Supervisor	:	Ts. Dr. TOH PEK LAN
Date	:	15/5/2023

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Ts. Dr. TOH PEK LAN for her invaluable advice, guidance, and her enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parent and friends who had helped and given me encouragement.

DEVELOPMENT OF AN APPLICATION FOR WEBSERVER STREAM DECK CONTROLLER USING EMBEDDED SYSTEM

ABSTRACT

Nowadays, stream deck is a very well-liked application for streamers or editors. It is designed to automate basic actions like switching sceneries and going live, but the same platform is helpful even if you don't play games for an audience. For instance, copying and pasting for editors, opening a file with a single click (recorded the keystroke and macro), and other features. Due to pandemic situation, most of the workers is going to work from home while some workers are lost their jobs, and there is keep borning out new generation, they would like to touch more new technology items. However, stream deck is a good tool to conduct their job easier especially for streamers, editors, teacher, programmers and so on. Therefore, stream deck is being developed by plenty of companies recently. In order to carry out this project, a PCB is necessary. A PCB is built to combine all hardware and become a system. A 3D printer is necessary to build for a casing for stream deck. A webserver is constructed to act as user interface for users to change their configuration. Moreover, ESP32 is act as a “heart” or “brain” that communicate with TFT screen while the program code for the module will be written and compiled by using programming IDE software. The main function of the stream deck is to create a shortcut key into a key to do a specific task. After the configuration or setup is done, the LED touch screen- ILI9488 TFT screen with XPT2046 touch controller will show up the options on the screen. At the end of this project, a DIY stream deck system is to create 6 buttons to let users to choose their option to conduct specific task by settings up the programmable buttons.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

AM-LCD	Active-matrix Liquid-Crystal Display
BMS	Battery Management System
BPM	Ballistic Particle Manufacturing
CAD	Computer Aided Drawing
CdSe	Cadmium Selenide
CPU	Central Processing Unit
DBI	Display Bus Interface
DPI	Display Pixel Interface
DSI	Display Serial Interface
ESP	Epressif
GRAM	Graphics Random Access Memory
IDE	Integrated Development Environment
IoT	Internet of Things
IPO	Initial Public Offering
LCD	Liquid Crystal Display
MDF	Medium Density Fireboard
MIPI	Mobile Industry Processor Interface
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
OBS	Open Broadcaster Software
PCB	Printed Circuit Board
RAM	Random Access Memory
RGB	Red-Green-Blue
RTC	Real Time Clock
RTOS	Real Time Operating System
SD	Secure Digital
SGC	Solid Ground Curing
SLA	Stereolithographic

SPI	Serial Peripheral Interfaces
SRAM	Static Random Access Memory
TFT	Thin-Film Transistor
WIMP	Windows, Icons, Menus and Pointer

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

According to Lim (1997), WIMPs (Windows, Icons, Menus, and Pointer) have been the prevailing design for software applications for the last ten years. Therefore, WIMP user interfaces have effectively overtook command-line user interfaces. The progress serves as a definitive confirmation of the effectiveness of employing a simplistic human factors methodology to allow the production of computer systems that are easily navigable for users. As a result of the replacement of the command-line interface, users are no longer required to commit command syntax to memory. The development of recall-directed interaction, specifically shortcut keys, in conjunction with the WIMP interface presents a potential contradiction. The tool provides users with an opportunity to utilize both the command line and WIMP user interface. In Table 1.1 is shown for the classification of the shortcut keys (Lim, 1997).

Table 1.1: Classification of Shortcut Keys (Lim, 1997).

Number	Classification
1	Separated into simple or complicated shortcut keys.
2	The frequency of using the shortcut keys.
3	The physical effort to use shortcut keys.

Table 1.1 outlines three key considerations that users must consider when deciding whether or not to utilise shortcut keys. In addition, a significant proportion of users attempt to bypass the utilisation of shortcut keys due to their inability to commit them to memory. According to Lim's study in 1997, individuals must exert additional effort in order to commit shortcut keys to memory. Consequently, certain users express a desire to utilize shortcut keys, yet struggle by an inability to commit them to memory. As a result, they resort to compiling a written record of the shortcut keys or affixing adhesive labels to the keyboard as a mnemonic aid (Madan, 2018). In addition, in cases where the shortcut keys are intricate, such as "ctrl + shift + P", users exhibited unwillingness to relinquish their mouse in order to utilize the aforementioned shortcut key combination. The reason for their prompt retrieval of the mouse may be attributed to a perceived sense of discomfort upon releasing it. According to Lim (1997), users rely on a mouse-based method to carry out their tasks. Madan (2018) reports that the implementation of stream deck had been fortunate for this world. The stream deck is a practical device that comprises a limited number of programmable buttons, enabling users to customise their shortcut keys by assigning them to the buttons (Adria willings, 2022). In addition, it is possible to incorporate icons onto the buttons to serve as visual cues for users regarding the respective function of each key (Madan, 2018). Upon activation of the designated button, the stream deck facilitates the execution of predetermined shortcut keys, thereby streamlining user functionality. In summary, Adrian Willings (2022) suggests that the stream deck has practical applications for a variety of users, including streamers, editors, officers, teachers, and lecturers. According to Adrian Willings (2022), streamers frequently alter multiple scenes in order to interact with their viewers. As such, streamers can effortlessly switch between desired scenes by simply pressing the pre-configured buttons on their stream deck, thereby eliminating the need to navigate through various shortcut keys. During the current pandemic, a significant number of employees, particularly educators such as teachers and lecturers, have transitioned to carrying out work from their homes. The educators are required to deliver their instructional sessions to students via various online platforms such as Microsoft Teams, Zoom, Google Meet, among others. According to Adrian Willings (2022), individuals who work remotely may encounter unexpected events, such as an unannounced visitor entering their workspace, which may cause them to forget to mute their audio or hastily search for the mute function.

Given that the stream deck buttons can incorporate icons and shortcut keys, it is advantageous to have the ability to promptly mute by simply clicking on the designated button. In recent times, there has been a surge in the adoption and proliferation of the Internet of Things (IoT). The Internet of Things (IoT) is a technological advancement that enables the utilisation of physical objects equipped with sensors and software capable of processing data, allowing their connection and exchange of information with other devices and systems via the internet (Parihar, 2019). According to Pravalika and Prasad (2019), Internet of Things (IoT) products are typically designed to function as automated devices that facilitate data collection, control, modification, and other related functions using Bluetooth, WiFi, or cloud storage. A single component of hardware used to build IoT products is ESP. It is a less expensive and more energy-efficient system on a chip microprocessor. Built-in features include WiFi and dual-mode Bluetooth (Pravalika, V., and Prasad, C. R., 2019). The ESP serves as the project's heart. It is a microcontroller board that connects many components, such as buttons, displays, sensors, and more. The board is programmed using the project's source code to run the project's activities. The source code is stored in the ESP's on-chip memory (Yogendra Singh Parihar, 2019). This block may be viewed as the communication between the programmer and the user. It is therefore considered as the project's core. Internet of Things (IoT) and wireless are now combined in a novel way. After years of research, wireless communication was first used in the 19th century. Wireless communication is an essential method for transferring data between devices. Wireless communication is necessary for consumers to receive information quickly. Users no longer have to take cables or wires with them everywhere they go, which makes it easier for them to complete jobs quickly and can boost output (Pravalika, V., and Prasad, C. R., 2019).

1.2 Problem Statements

1.2.1 Efficiency

In contemporary times, individuals lead hectic lifestyles, resulting in infrequent recollection of the distinct functions of each keyboard key. The traditional method of using the function, which is by using our own memory or create some stickers on the surface of the keyboard to use the shortcut keys or hotkeys to operate the task. A stream deck would help users to increase the work efficiency and stay in professional by just clicking one of the configured buttons to operate the task. (Adrian willings, 2022)

1.2.2 Price factor

As society adjusts to Internet of Things (IoTs) technology, most items are becoming safer and more convenient to use as technology advances at a rapid rate. However, the price that have related to any technology is increasing and cause users cannot afford to buy the items. Of course, in this modern and gaming world, the stream deck is also an expensive item especially for the stream deck from Elgato, Razer, and Loupedeck companies nowadays. For example, the costs of Elgato stream deck are around RM450 for the 6 buttons version and RM700 for 15 buttons, respectively. Therefore, a cheaper or budget version with a multifunction stream deck for users is needed to reduce the expenses of the users nowadays.

1.2.3 Reviewing projects

In 2022, Freotech studied the stream deck using ATmega328 microcontroller (Freotech, 2022). In Table 1.2, the components and materials used are ATmega328 pro mini microcontroller, press buttons, and Medium Density Fireboard (MDF) board. The aim of the project is to solve the price problem of stream deck. The Medium Density Fireboard (MDF) board was used to design as the casing of the stream deck, as shown in Figure 1.1. From the diagram, the circuits were built inside of the model. In this work, the printed icons were used to stick onto the press buttons to indicate the specific tasks. Unfortunately, there have some parts of the project still can be improved. For example, the circuit used were not tidied up, and the casing was designed using the Medium Density Fireboard (MDF) board, as shown in Figure 1.2. Therefore, to avoid the circuit connection problem, the circuit should tidy up and clean in this FYP project. It will help to check the circuit connection easily when the circuit problem occurs. Next, the casing of the stream deck should design using 3D printer due to 3D printing is easy to precisely manage a design's dimensions and specifications. In this modern era, everyone can learn technology easily by just touching the display on the spots. Moreover, the buttons should change to touch screen in this FYP project. The icon can import into the screen instead printing out the icons.

Table 1.2: Component and Materials Used of Stream Deck (Freotech, 2022).

Components and materials	Amount
Arduino Pro Micro with ATMega32 chip	x1
Press Buttons	x9
MDF Board	x3

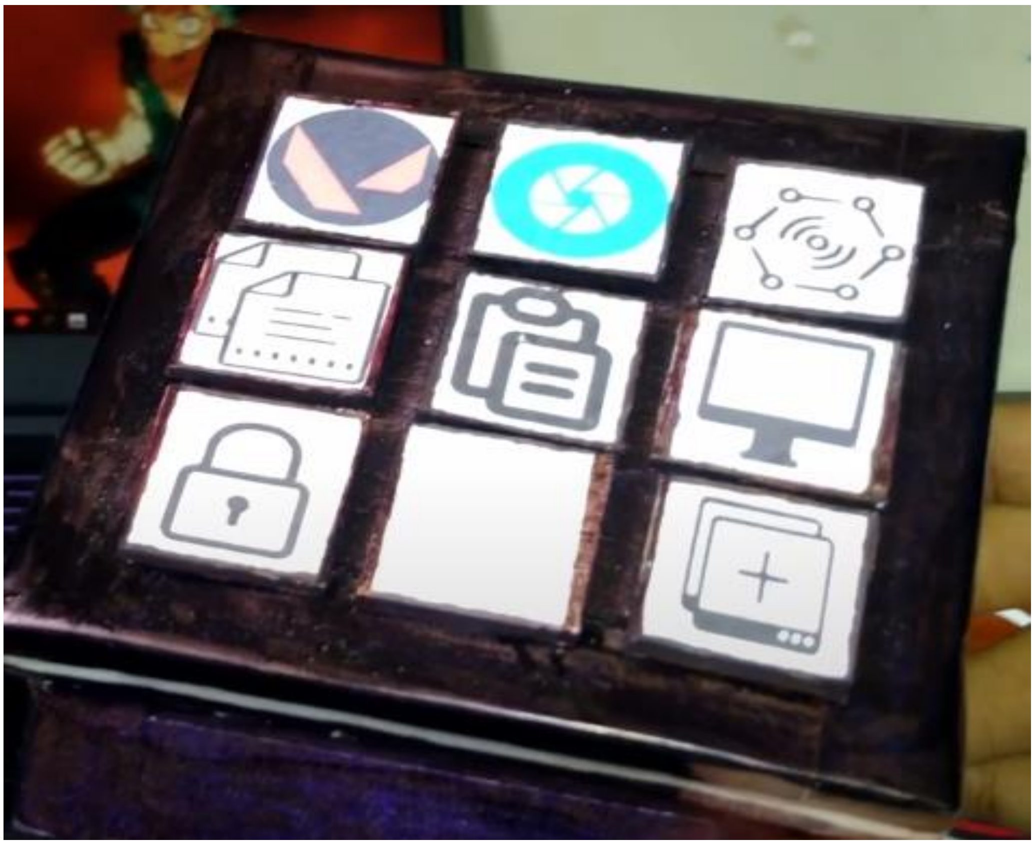


Figure 1.1: The Front View of Stream Deck (Freotech, 2022).

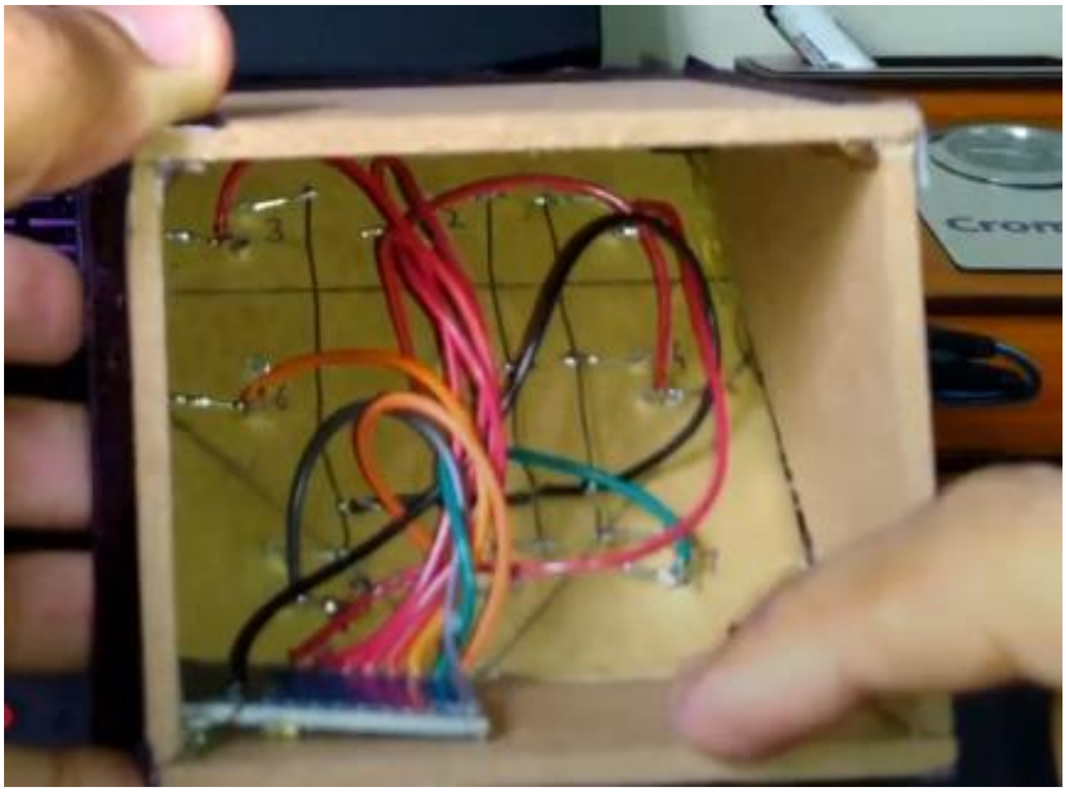


Figure 1.2: The Bottom View of Stream Deck (Freotech, 2022).

In 2019, NL1_CS also studied the stream deck using Arduino (NL1_CS, 2019). In Table 1.3, the components and materials used are Arduino Pro Micro, press buttons, PCB and resistors. Again, the purpose of this project is to solve the price of the stream deck maximize the amount of buttons for users to configure the settings. This project does not use any method to create for casing of the stream deck. The PCB had been used to connect all the component together on the board. Therefore, the DIY stream deck will be group together to one place and will be look like cleaner and tidier as the Figure 1.3 shown. From the Figure 1.4, the circuits were built behind of the PCB. The resistors had been used to connect the buttons in order to make sure the circuits are stable by reducing current flow, adjusts signal levels, divide voltages and terminates transmission lines. Sad to say, there have some parts of the project still can be improved. For example, the circuit used were not tidied up, and there are no labels for every button, as Figure 1.3 is shown. However, the circuit should be tidied up and clean in this FYP project to avoid circuit connection problem. This will help to check the circuits connection easily. Moreover, the labels should be indicated to every button, and this will let users to know the function and no need to memorize to every button.

Table 1.3: Component and Materials Used of Stream Deck (NL1_CS, 2019).

Components and materials	Amount
Arduino Pro Micro with ATMega32 chip	x1
Press Buttons	x12
PCB	x1
10k ohm resistors	x12

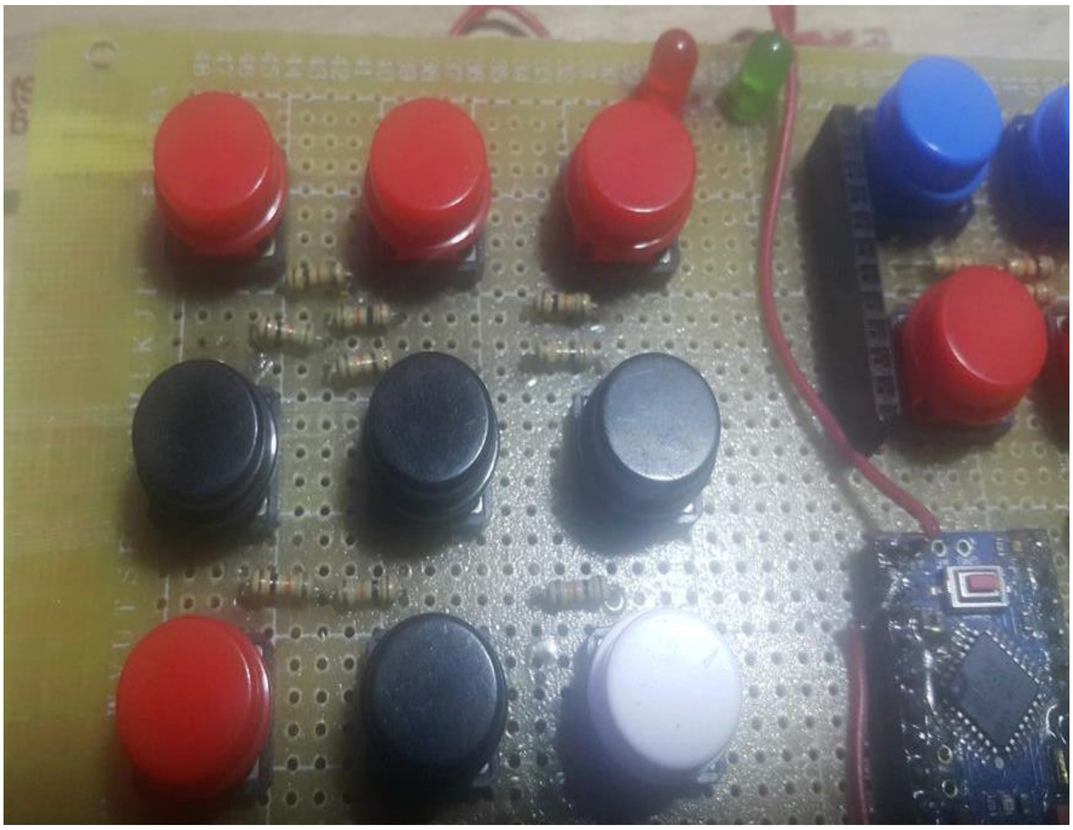


Figure 1.3: The Front View of Stream Deck (NL1_CS, 2019).

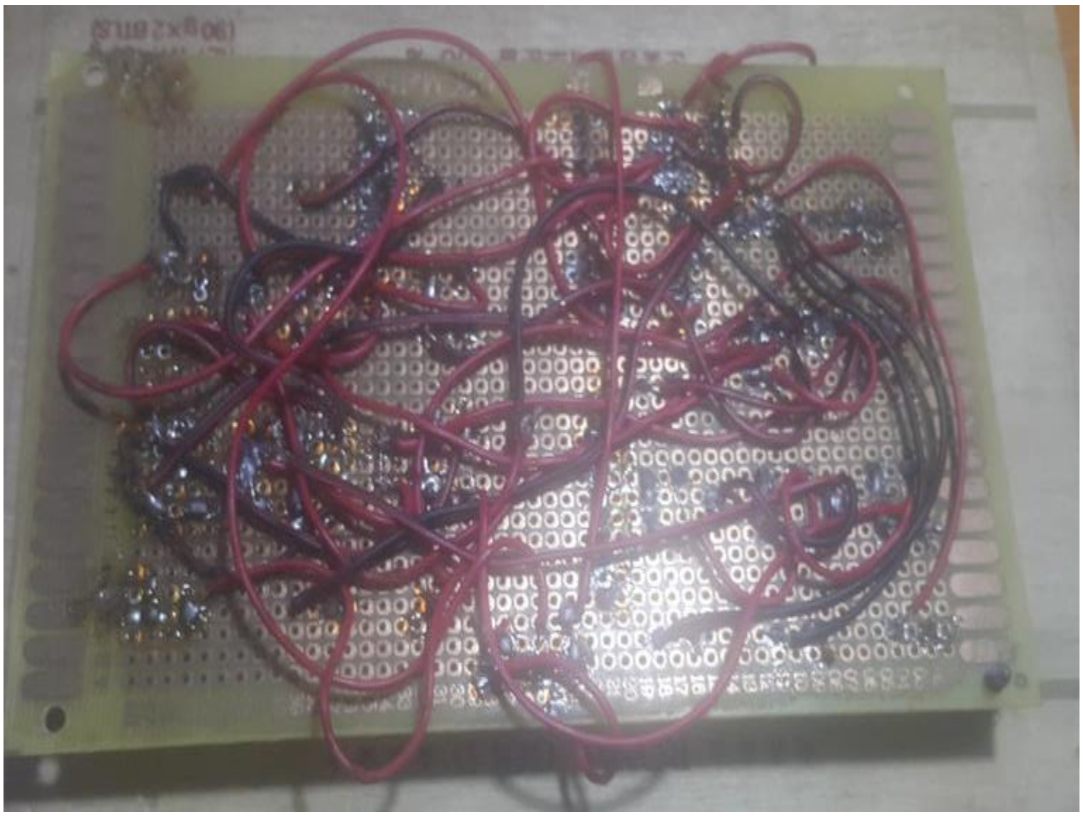


Figure 1.4: The Back View of Stream Deck (NL1_CS, 2019).

In 2021, Arduino Team had published one project related to stream deck (Arduino Team, 2021). They named the product as Crumble Deck to act as an alternative of stream deck. From the Table 1.4, the components and materials used are Arduino Due, press buttons, TFT screen, and 3D printer. The aim of the project is to solve the price problem of stream deck. It also creates a beautify casing for the stream deck. In Figure 1.5, the product contains 20 buttons which is using 4 X 5 row and column to manually navigate the product. Furthermore, the product had been designed by using TFT screen to obtain the touch screen feature. Besides, the Arduino Team had done a prototype for the case of the Crumble Deck before using 3D printer. After that, they used 3D printer to create a casing model for their product to look nicer. As Figure 1.6 is shown, their wiring is good example for my fyp project because they have used different colour of the wires to connect the 4 x 5 buttons connection. Therefore, it is easily to modify or check to avoid circuits connection problems. Unfortunately, they mentioned that their product is not stable. Their TFT screen has no respond when the touch on it and cause them to build a manual system (buttons system) to navigate the product. When the TFT screen is not respond, it is just acting a screen to show the labels. In addition, the crumble deck had been designed to have 20 buttons. In this design, 20 navigation control is squeeze into one small screen and it will lead to messy to users' vision.

Table 1.4: Component and Materials Used of Stream Deck (Arduino Team, 2021).

Components and materials	Amount
Arduino Due with AT91SAM3X8E chip	x1
Press Buttons	x20
TFT screen	x1
3D printed case	x1

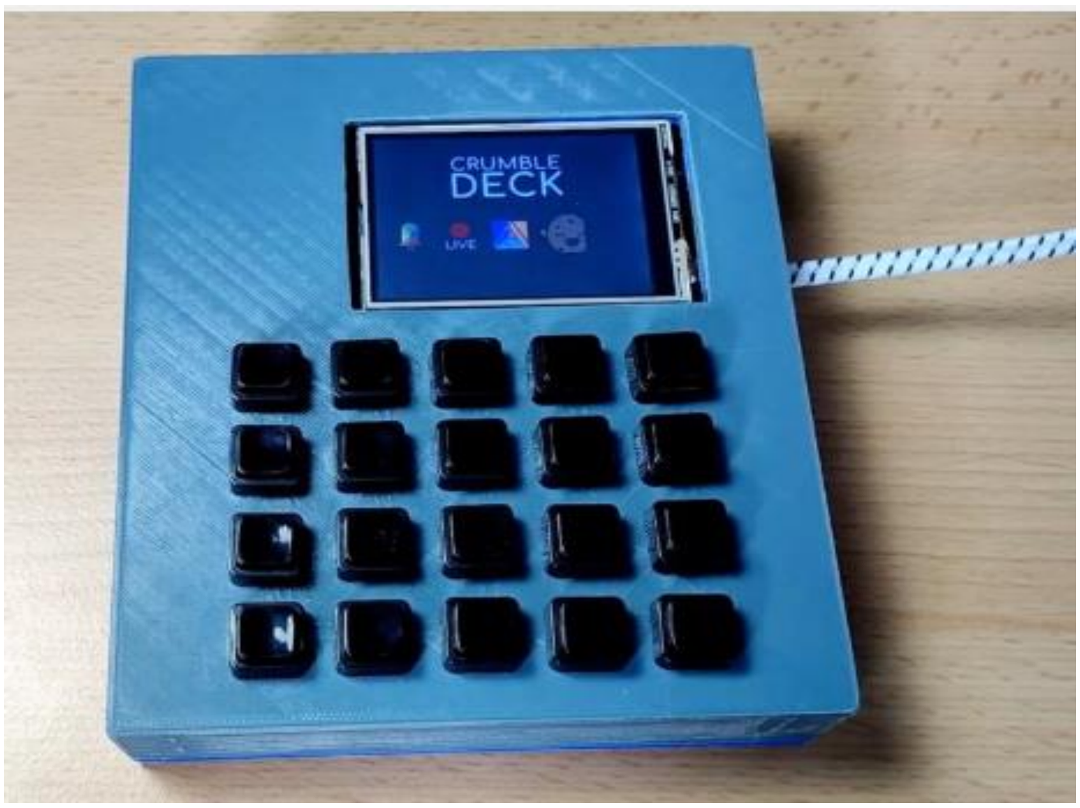


Figure 1.5: The Front View of Crumble Deck (Arduino Team, 2021).

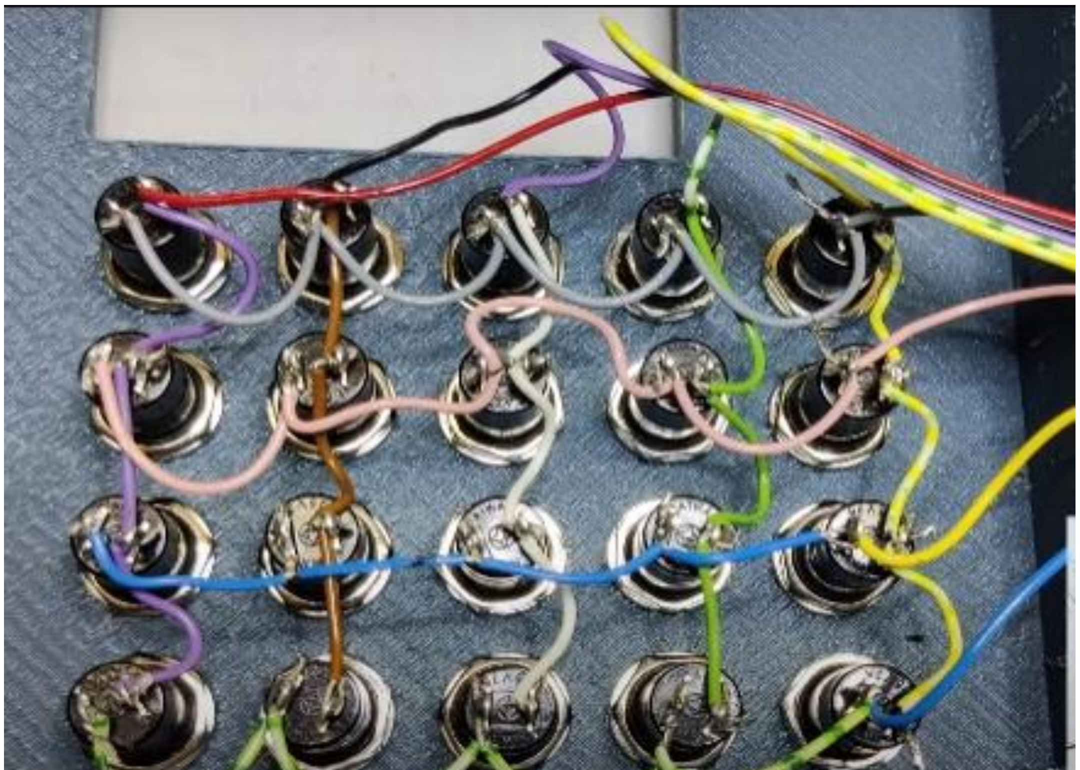


Figure 1.6: The Wiring View of Crumble Deck's Buttons (Arduino Team, 2021).

1.3 Objectives

The objective of this project is to develop a device that facilitates the execution of the task for its users. The study aims to achieve specific objectives:

i) The configuration of the user interface of the stream deck can be achieved through either the access point or station point.
ii) To develop a customizable control that can perform various functions.
iii) To configure the touch screen feature.
iv) To enable wireless connectivity and establish a Bluetooth connection.
v) To develop a low-cost stream deck.

CHAPTER 2

LITERATURE REVIEW

2.1 Stream Deck

The first stream deck has been invented by Elgato systems (Elgato,2012), and the stream deck was launched in 2017. From literature studies, Elgato systems was introduced by Markus Fest in 2010 (Hargreaves, Eddie, 2007). The headquarter of the company was in Munich, Germany. In 2018, Elgato Systems was rebranded as Eve System and Elagato gaming was renamed as Elgato (Elgato, 2012). As we known that Elgato stream deck is a convenient control pad, which can be used as programmable keyboard (Madan, 2018), as shown in Figure 2.1.

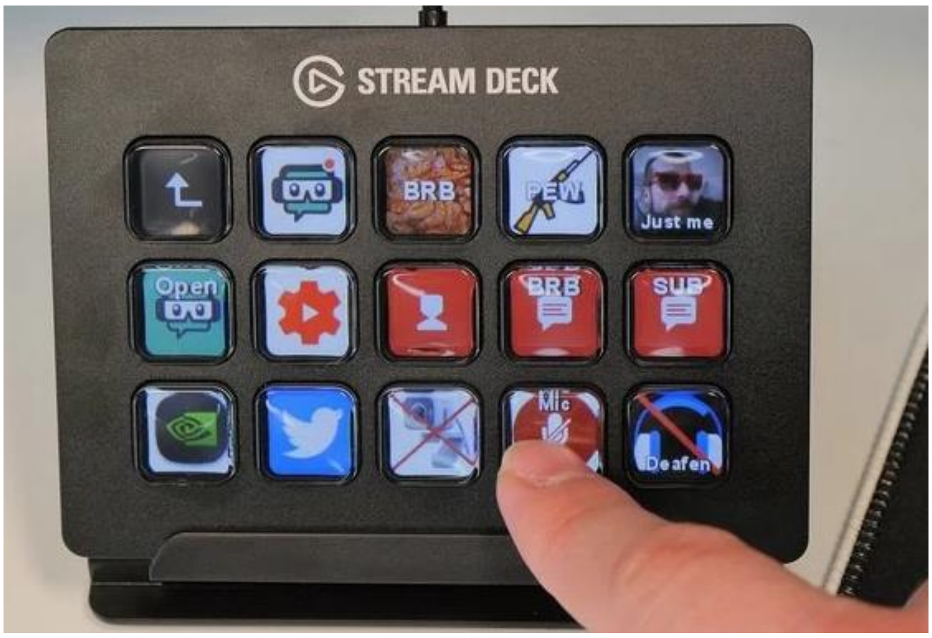


Figure 2.1: Overview Programmable Button (Adrian willings, 2022).

In addition, it can also let users to create their own macros. The macro is a piece of code that causes variety of things to happen when hitting a specific key on the stream deck. If a copy feature, “ctrl + c” is set into one of buttons in the stream deck, then the users can just press the button to conduct the function. Furthermore, this stream deck also to assists streamers to remain professional. If streamer need to eat or converse with a family member who has stepped into the room, that button might be an easy-access feature that rapidly turns off the webcam or mutes the microphone. Discord also has a mute and deafen option that may be used to temporarily prevent friends from being heard and let streamer chat to their audiences uninterrupted. Since this stream deck can be configured, it is good for editors to use it. Therefore, the controls that editors can add include those for splitting video tracks, trimming the beginning and finish of clips, enabling multi-camera editing, eliminating undesirable clips, saving, and many other things. All of this may replace keyboard shortcuts, allowing editors to use the stream deck to work more quickly and effectively when editing videos. Nowadays, there have several sizes of Elgato stream deck, such as mini size (6 buttons), standard version (15 buttons), and XL size (32 buttons) (Elgato, 2012). As we known that the market price of Elgato stream deck increases when the size increase. For example, the price of XL size is about RM1109, while the mini size is around RM489. Razer gaming is a company that expended into other gaming related products such as keyboards, monitors, controllers, phone, laptops and so on. Initially, the Razer brand started as a subsidiary of Karna LLC in 1998 (E.Ferraz and G.Fernandez, 2020). The idea at that time was to create and promote high end computer mice. However, Karna was closed in 2000. The Razer brand name was then in hiatus for around 5 years. In 2005, Min-Liang Tan co-founded Razer with Robert Krakoff in California (E.Ferraz and G.Fernandez, 2020). In 2022, the company Razer, partnership with Loupedeck company had launched a stream deck and name it as “Razer Stream Controller” (Michael Gariffo, 2022). The Razer Stream Controller has twelve configurable buttons, menu buttons and adjustable lube on the surface of it. It cost RM 1210 (Tom, Warren, 2022). In 2016, Loupedeck has been founded by Mikko Kesti in Finland (Loupedeck, 2021). Loupedeck is a leading brand of custom editing consoles within professional and hobbyist workstations. For sure, they had launched their stream deck and called Loupedeck+, Loupedeck Live and Loupedeck CT and the prices are RM 1119, RM 1209 and RM2243.

Table 2.1: Products and Prices from the Companies.

Company	Products	Price
Elgato	1 6 Customizable LCD keys 2 Multi-platform Compatible 3 Ideal for Live Streaming STREAM DECK Stream Deck Mini (Elgato, 2012)	RM489
	1 15 Customizable LCD keys 2 Multi-platform Compatible 3 Ideal for Live Streaming STREAM DECK Stream Deck Standard version (Elgato, 2012)	RM719
	1 32 Customizable LCD keys 2 Multi-platform Compatible 3 Ideal for Live Streaming STREAM DECK Stream Deck XL version (Elgato, 2012)	RM 1109
Razer	Razer Stream Controller (Razer, 2012)	RM1210.14

Loupedeck	Loupedeck CT (Loupedeck, 2021)	RM 2243.30
	Loupedeck Live (Loupedeck, 2021)	RM 1209.32
	Loupedeck + (Loupedeck, 2021)	RM 1119.40

2.2 Thin-Film-Transistor Screen

A unique kind of field-effect transistor was called as a thin-film transistor (Hiro Kawamoto,2011). The transistor is thin in comparison to that of device's plane. Moreover, a thin-film transistor (TFT) was introduced. It possesses supportive, but non-conducting substrate. Due to liquid-crystal display (LCD) is one of important applications for the TFTs, glass was chosen as the substrate in the study. TFT-LCD was presented as the thin-film transistor technology to enhance the visual properties, such as addressability and contrast. In 1957, Wallmark reported a thin film MOSFET application (Hiro Kawamoto,2011). In 1962, Weimer applied the Wallmark's concepts, and created the thin-film transistor (Hiro Kawamoto,2011). In the work, the characteristics of MOSFET are different with that of bulk MOSFET. The thin coatings of cadmium selenide and cadmium sulphide were used in the study. In addition, Lechner, in 1968, also studied the TFT-based LCD (Hiro Kawamoto,2011). Then, the dynamic scattering mode of LCD was investigated by Lechner *et al.*, 1971 (Hiro Kawamoto,2011). In 1973, Brody *et al.* studied CdSe thin-film transistor TFT-LCD (Yue Kuo,2013). Luo, in 1975 demonstrated the flat AM-LCD using CdSe TFTs (Hiro Kawamoto,2011). In 2013, TFT-based active-matrix displays were used in all contemporary high-resolution and high-quality electronic visual display systems (Yue Kuo,2013).

Table 2.2: Characteristics of different module TFT Screen.

No	Characteristics	Module of TFT SCREEN		
		ILI9488	OTM8009A	ST7735
1.	Type	TFT	TFT	TFT
2.	Screen Size	3.5 inch	3.97inch	1.8inch
3.	Touch Screen	Yes and No	Yes and No	No
4.	Active area(mm)	48.96x73.44	51.84x86.40	28.03*35.04
5.	Weight (gram)	No touch: 45g With touch: 57g	No touch: 50g With touch:62g	25g
6.	Driver IC	ILI9488	OTM8009A	ST7735
7.	Operating Temperature	-20℃~60℃	-20℃~60℃	-20℃~60℃
8.	Display Colour	RGB 65K	RGB 65K	RGB 65K colour
9.	Resolution(pixel)	480*320	800*480	128*160
10.	Operating Voltage	5V/3.3V	5V/3.3V	5V/3.3V
11.	Price	RM42	RM75	RM15

Figure 2.2: ILI9488 Module TFT Screen (LCwiki,2021).

Figure 2.3: OTM8009A Module TFT Screen (LCwiki,2019).

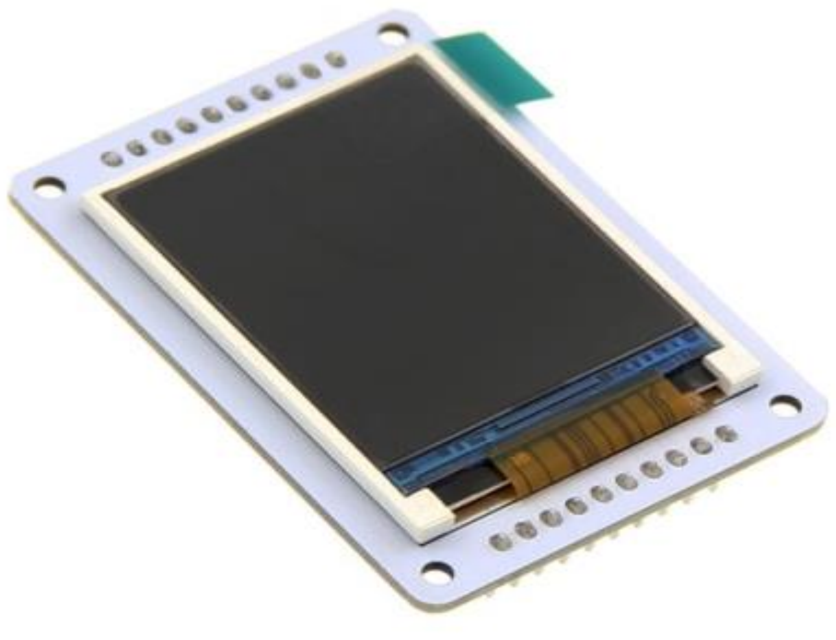


Figure 2.4: ST7735 Module TFT Screen (LCwiki,2022).

2.3 Espressif System

The ESP32 is referred to as the enhanced version of ESP8266, a line of microcontroller chips made by Espressif Systems in Shanghai. Teo Swee Ann founded this Espressif System in 2008, and it is based and headquartered in Shanghai. The ESP32 is often referred to as the enhanced version of ESP8266, a line of microcontroller chips produced by Espressif Systems. Espressif System was established in 2008 by Teo Swee Ann, with its headquarters in Shanghai (DroneBotWorkshop,2020). There are 4 locations, including China, Singapore, and India. Brazil and the Czech Republic (Espressif,2015). The primary objective of this organisation is to provide cutting-edge WiFi and Bluetooth, low-power IoT solutions. The well-known ESP8266, ESP32, and ESP32-S families of chips, modules, and development boards are produced by them (Espressif,2015). They offered eco-friendly, adaptable, and affordable chipsets by utilising wireless computing. Their objective and ambition are for developers to leverage Espressif's technology globally and create intelligent, linked products in the modern day. Espressif's IPO took place on the Sci-Tech Innovation Board (STAR) of the Shanghai Stock Exchange in July 2019 (Espressif,2015). Espressif Systems also provides a related module called ESP-WROOM-32 along with the launch of ESP32. The module is incredibly simple to use despite its small size (25.5 x 18.0 x 2.8mm), due to integrated components such an antenna, oscillator, and flash. Similar modules for various microcontrollers are frequently used by amateurs or for testing and prototyping.

2.3.1 Comparison between microcontrollers

Table 2.3a: Comparison between different microcontrollers (Maier, A., Sharp, A. and Vagapov. Y, 2017).

Chip (Module)	ESP32 (ESP-WROOM-32)	ESP8266 (ESP8266-12E)
Details:		
CPU	TensilicaXtensaLX6 32 bit Dual-Core at 160/240 MHz	Tensilica LX106 32 bit at 80 MHz (up to 160 MHz)
SRAM	520 KB	36 KB available
FLASH	2MB (max. 64MB)	4MB (max. 16MB)
Voltage	2.2V to 3.6V	3.0V to 3.6V
Operating Current	80 mA average	80 mA average
Programmable	Free (C, C++, Lua, etc.)	Free (C, C++, Lua, etc.)
Open source	Yes	Yes 802.11 b/g/n
Connectivity:		
WiFi	802.11 b/g/n	802.11 b/g/n
Bluetooth®	4.2 BR/EDR + BLE	-
UART	3	2
I/O:		
GPIO	32	17
SPI	4	2
I2C	2	1
PWM	8	-
ADC	18 (12-bit)	1 (10-bit)
DAC	2 (8-bit)	-
Size (mm)	25.5 x 18.0 x 2.8	24.0 x 16.0 x 3.0
Price	RM 36.42	RM 22.76

Table 2.4b: Comparison between different microcontrollers (Maier, A., Sharp, A. and Vagapov. Y, 2017).

Chip (Module)	CC32 (CC3220MODSF)	Xbee (XB2B-WFPS-001)
Details:		
CPU	ARM Cortex-M4 at 80 MHz	N/A
SRAM	256 KB	N/A
FLASH	1MB (max. 32MB)	N/A
Voltage	2.3V to 3.6V	3.14V to 3.46V
Operating Current	N/A	N/A
Programmable	C (SimpleLink SDK)	AT and API commands
Open source	No	No
Connectivity:		
WiFi	802.11 b/g/n	802.11 b/g/n
Bluetooth®	-	-
UART	2	1
I/O:		
GPIO	21	10
SPI	1	1
I2C	1	-
PWM	6	-
ADC	4 (12-bit)	4 (12-bit)
DAC	-	-
Size (mm)	20.5 x 17.5 x 2.5	24.0 x 22.0 x 3.0
Price	RM 36.42	RM 22.76

The Table 2.3a and Table 2.3b provides information on the four modules that are utilised in the design of Internet of Things (IoT) devices (Maier, A., Sharp, A. and Vagapov. Y, 2017). The Internet of Things (IoT) offers a wide range of modules and microcontrollers that exhibit significant diversity. However, most of these components encounter similar challenges in terms of their physical dimensions, operational efficiency, and financial implications. In contrast to the Xbee, RTLDuino boards possess open-source capabilities and exhibit distinct expertise in executing complex operations. Nevertheless, they exhibit a significant size differential in comparison. However, the ESP32 QFN48 is a much smaller component than other microcontrollers, measuring only 5mm by 5mm. The published circuit of the ESP-WROOM-32 module facilitates the integration of ESP32 onto a customized printed circuit board, thereby enabling the development of a compact device. The ESP32-DevKitC board is a readily available and convenient option that is conducive to testing and instructional applications and is also compatible with breadboards. Although ESP32 is a superior option that can be used in more complicated projects, ESP8266, ESP32's predecessor, was very well-liked for the design in many IoT-related projects (Maier, A., Sharp, A. and Vagapov. Y, 2017). The ESP32 system has two Harvard Architecture Xtensa LX6 CPUs and is dual-core (Babiuch,M., Foltynek,P., Smutny,P. ,2019). All of these CPUs' internal and external memory as well as peripherals are connected to the data bus and/or instruction bus. The two cores in the microcontroller, PRO CPU for protocol and APP CPU for application, can be used for a variety of things. The address space for the data bus and instruction bus combined is 4GB, whereas the address space for peripherals is 512KB. The embedded memories also include two 8KB RTC memories, 520KB SRAM, and 448KB ROM. The external memory may accommodate up to four 16MB Flash cards (Maier, A., Sharp, A. and Vagapov. Y, 2017). Since C is the most popular programming language for ESP32, most API libraries are also available in C. But C++ can also be used to easily programme the microcontroller (Babiuch,M., Foltynek,P., Smutny,P. ,2019). The C++ programming option allows for the usage of various Arduino libraries, albeit modifications may be needed. A Texas engineer named Neil Kolban offers a tonne of C++ libraries for the ESP32 APIs in his GitHub repository (Babiuch,M., Foltynek,P., Smutny,P. ,2019). There are also online tools to programme the ESP32 in LUA, JavaScript, etc. because this chip is open source, and anyone may create a "operating system" for it. From the table, ESP8266, predecessor of ESP32 and CC32 also can be programmed in C language while Xbee microcontroller only can be programmed in AT and API commands. This type of commands from Xbee is not famous as C language and most of the users need to have training and then can program it (Maier, A., Sharp, A. and Vagapov. Y, 2017). So, the ESP32 can also be programmed in C++ language and Lua language. The ESP32 is widely use in different languages and led to users easily to drive the microcontroller. Therefore, most of the users will choose for ESP32. It has been demonstrated that ESP32 is a great choice for IoT devices because of its performance characteristics and affordability (Babiuch,M., Foltynek,P., Smutny,P. ,2019). The microcontroller comes in a few different form factors. The ESP-WROOM-32 module offers a small solder friendly footage, whereas the ESP32 QFN48 is the alternative for industrial manufactures and small sized solutions. The bread board friendly version of the ESP32-DevKitC is the ideal answer for hobbyist and educational applications. The widely used ESP32 performs far better than its predecessor, the ESP8266, in a wide range of IoT applications. The dual core architecture of the microcontroller and a major expansion of the operational features are responsible for its great performance. FreeRTOS, an open-source operating system for microcontrollers, offers excellent support for real-time applications. ESP32 is therefore anticipated to be a key component in the design of upcoming embedded systems and Internet of Things (IoT) systems (Maier, A., Sharp, A. and Vagapov. Y, 2017).

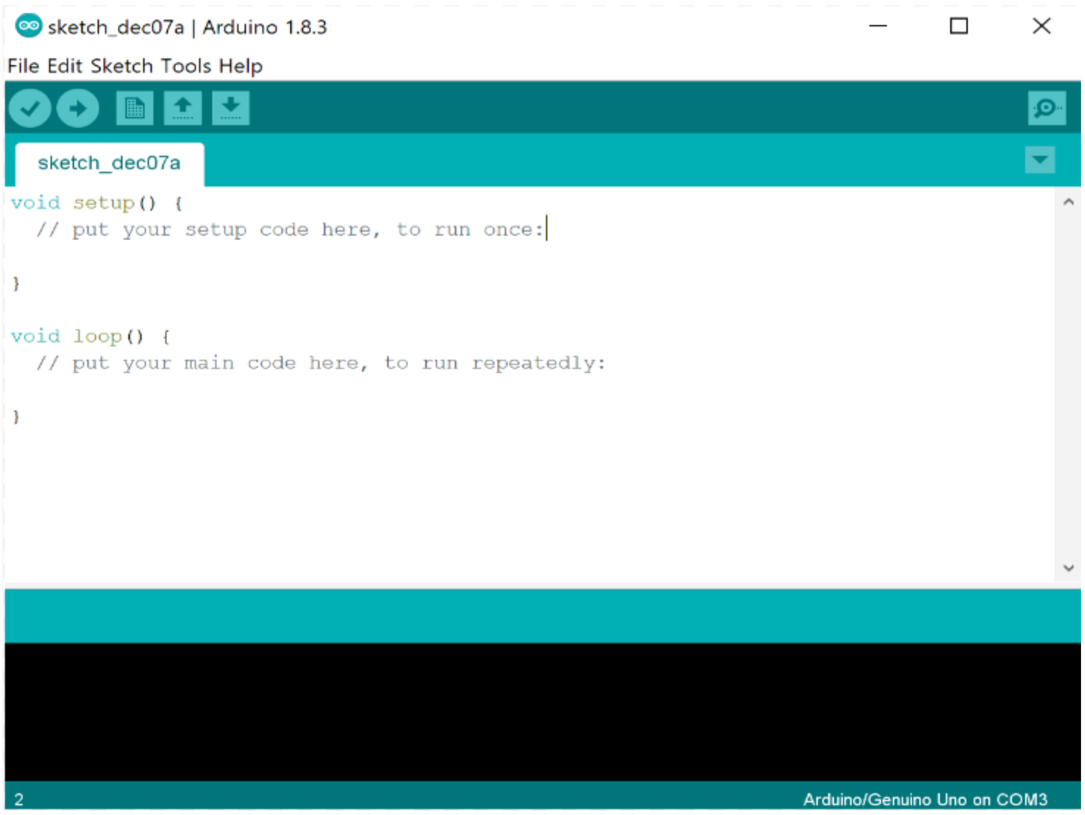
2.4 Software IDE

2.4.1 Arduino IDE

The Arduino Integrated Development Environment is free software created by Arduino.cc exclusively for Arduino Modules. This program's main feature is a text editor that is comparable to notepad but has additional functionality (Nikola Zaltanov, 2015). To write code, compile it for error checking, and upload it to Arduino Modules that are compatible with the programme, utilise the text editor portion (Mohamed FEZARI and Ali Al Dahoud,2018). As an example, the integrated development environment (IDE) offers a text editor with syntax highlighting and autocomplete tools in addition to a serial monitor that lets you inspect and debug the output of your code. Moreover, it contains an integrated library manager that makes adding libraries to your projects simple. The two main components of this software environment are the editor, which is used to write the code and is depicted in the accompanying image, and the compiler, which is used to compile and upload the code to an Arduino module. Operating System such as Windows, Linux, MacOS are accessible to use this software. It is designed to be user-friendly and simple to use, even for novice programmers with no prior experience. This software is compatible with every Arduino board currently on the market, including the Arduino UNO, Arduino Mega, Arduino Leonardo, Arduino Ethernet, and others that use the C/C++ programming language like ESP32. For instance, to use Arduino IDE, an Arduino board, a USB cable to connect it to computer, and the necessary drivers set up on computer in order to utilise the Arduino IDE. The Arduino IDE can be downloaded and installed and begin writing and uploading code to Arduino board as soon as have all the required parts. The Arduino file is referred to as a Sketch because the user writes the code in it and saves it in the .ino file extension (Mohamed FEZARI and Ali Al Dahoud,2018). The sketch made on the IDE platform generates a Hex File that needs to be transferred and uploaded into the controller. Given that the software is open-source, users are free to include their own modules and functions into it for any project they like (Mohamed FEZARI and Ali Al Dahoud,2018).

2.4.2 Microsoft Visual Code

Microsoft created the popular code editor known as Microsoft Visual Code (VS Code). It is a free and open-source code editor for Windows, macOS, and Linux that supports many different programming languages. This essay will give a summary of the benefits, features, and uses of VS Code in academic contexts. The goal of VS Code was to provide a quick, lightweight code editor that could be tailored to the needs of the user. It includes a cutting-edge user interface with adjustable options, syntax highlighting, code completion, debugging, and version control integration. VS Code is the best option for developers that need to write, debug, and test code fast and effectively because of these capabilities. The adaptability of VS Code is one of its key benefits. Numerous programming languages, including well-known ones like JavaScript, Python, C++, and Java, are supported. As a result, developers that work on a range of projects and want a code editor that can handle different languages frequently choose it. The extensibility of VS Code is an additional benefit. It contains a sizable ecosystem of extensions that can be added to give the editor more features. The addition of support for certain programming languages, frameworks, and tools via these extensions can increase developer productivity and efficiency. Students and researchers that need to develop, debug, and test code for their work in academic contexts can utilize VS Code. Numerous uses are possible for it, such as data analysis, machine learning, and scientific computing. Its versatility and support for several programming languages make it a useful tool for academics who must work with various sorts of data and code. Along with its benefits and features, VS Code has a sizable and vibrant user and developer community that contributes to its growth and support. Users may seek assistance from this community's resources, including tutorials, documentation, and forums, and they can also discuss their experiences using the editor. In conclusion, developers and academics alike frequently utilize Microsoft Visual Code, a strong and flexible code editor. It is the best option for individuals that need to create, debug, and test code rapidly and effectively due to its capabilities, adaptability, and extensibility. Data analysis, machine learning, and scientific computing are only a few examples of its uses in academic contexts. The editor's vibrant user and developer community offers helpful tools and assistance to users.


sketch_dec07a | Arduino 1.8.3
File Edit Sketch Tools Help
sketch_dec07a
2
Arduino/Genuino Uno on COM3

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 2.5: Main Section of Arduino IDE (Mohamed FEZARI and Ali Al Dahoud,2018).

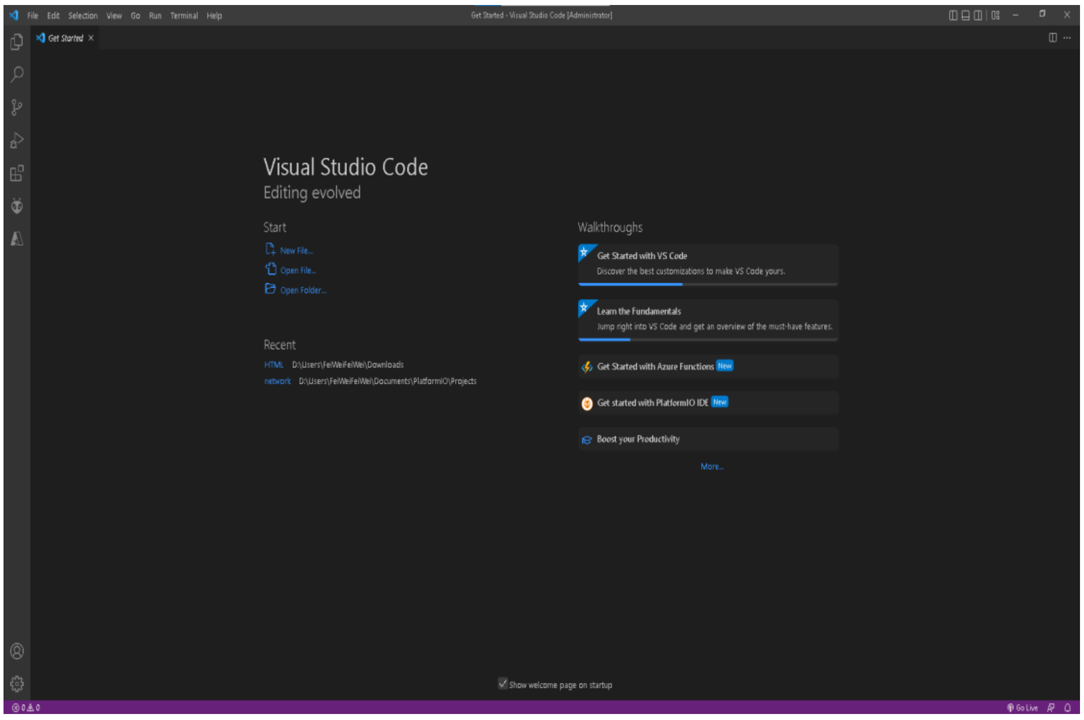

File Edit Selection View Go Run Terminal Help
Get Started - Visual Studio Code [Administrator]
Get Started
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Recent
HTML D:\Users\FeiWeiFeiWei\Downloads
network D:\Users\FeiWeiFeiWei\Documents\PlatformIO\Projects
Walkthroughs
Get Started with VS Code
Discover the best customizations to make VS Code yours.
Learn the Fundamentals
Jump right into VS Code and get an overview of the must-have features.
Get Started with Azure Functions New
Get started with PlatformIO IDE New
Boost your Productivity
More...
Show welcome page on startup
Go Live

Figure 2.6: Main Section of Microsoft Visual Code.

2.5 3D Printer

A 3D printer uses an additive manufacturing process to create 3D components and objects by building them up from many layers of material. Another name for it is rapid prototyping (Ativya Gupta,Garima, and Harshit Srivastava, *et al.*,2021). In other words, 3D printing is adding layers consecutively, while traditional machining processes rely on the removal of material through procedures like cutting or drilling. As a result, an object is manufactured layer by layer until it is complete using a layering approach. This is how 3D printing transitions us from a mass production line to a custom, one-off manufacture. There were three main 3D printing methods introduced in the 1980s (Ativya Gupta,Garima, and Harshit Srivastava, *et al.*,2021). Dr. Kodana was the first to demonstrate the layer-by-layer method of assembly, as well as the first to develop a quick prototyping strategy. Additionally, he created the ancestor of SLA. He attempted to use UV light to polymerize a photosensitive gum but was unsuccessful. Dr. Kodana was shocked to discover that he had not documented the entire patent detail prior to the application's one-year due date. You can trace the origins of 3D printing invention back to 1983 (Ativya Gupta,Garima, and Harshit Srivastava, *et al.*,2021). The other 3D printing innovations and technologies came into being in the 1990s (Ativya Gupta,Garima, and Harshit Srivastava, *et al.*,2021). as well as the debut of fresh 3D printer producers and CAD tools. The SLA system is sold for the first time on a commercial basis by 3D systems. Additionally, Itzchak Pomerantz *et al.* and William Masters patents for BPM and SGC were among the other growing procedures. A CAD tool has to be used to designed or model the object using a digital camera and a very special photogrammetry programme, or a 3D scanner. The fact that these 3D printed models were made using CAD technology reduces the number of flaws that were discovered and may be fixed before printing. The manual modelling method is related to sculpting in the sense that it prepares geometric data for 3D computer graphics. This information can be used to create three-dimensional representations of the scanned object. After being created in a CAD application, the model is frequently saved in .skp, .dae, .3ds, or another format. To enable software to read it, the model must then be converted to a .STL or .OBJ format (Konstantinos Kitsakis,Nikos Petrou,Ilias Tanos, and John Kechagias,2016). Therefore, there are some software can be used like tinkercad , fusion360 , AUTODESK and so on (Konstantinos Kitsakis,Nikos Petrou,Ilias Tanos, and John Kechagias,2016).

2.6 Rechargeable system

The rechargeable system needed 2 components to complete the system which are batteries and charging board. First, a disposable or primary battery is supplied fully charged and is thrown away after use (Melissa Morris and Sabri Tosunoglu,2012). A rechargeable battery, storage battery, or secondary cell (formally a type of energy accumulator) is an electrical battery that can be charged, discharged into a load, and recharged numerous times (Melissa Morris and Sabri Tosunoglu,2012). It is one or more electrochemical cells make up its structure. It stores and accumulates energy by a reversible electrochemical reaction; hence the term is called "accumulator" (Melissa Morris and Sabri Tosunoglu,2012). From button cells to megawatt systems connected to stabilise an electrical distribution network, rechargeable batteries are created in a wide variety of sizes and configurations. Lead-acid, zinc-air, nickel-cadmium (NiCd), nickel-metal hydride (NiMH), lithium-ion (Li-ion), lithium iron phosphate (LiFePO4), and lithium-ion polymer (Li-ion polymer) are just a few examples of the various electrode materials and electrolytes that are combined. In Table 2.4, the table is shown the characteristic for every battery. From late 1800s, the lead-acid battery is in use (Liang Y, Zhao C-Z, Yuan H, *et al*.,2019). Start from 1950, Ni-Cd battery is in used while Ni-MH battery is in used since 1990 (Liang Y, Zhao C-Z, Yuan H, *et al*.,2019). After the research, the Li-ion battery has been used from 1991 until today. The latest battery is called Li-ion polymer (Liang Y, Zhao C-Z, Yuan H, *et al*.,2019).

Table 2.5: Characteristics for Every Battery (Liang Y, Zhao C-Z, Yuan H, *et al.*,2019).

Characteristics	Lead-acid battery (A)	Ni-Cd battery (B)	Ni-MH battery (C)	Li-ion battery (D)
Gravimetric energy density (Wh/kg)	30-50	40-60	60-120	170-250
Volumetric energy density (Wh/L)	60-110	150-190	140-300	350-700
Battery voltage (V)	2	1.2	1.2	3.7
Cycle life (to 80% of the initial capacity)	300	1500	1000	500-2000
Self-discharge per month (%)	5	20	30	<10
Fast charging time (h)	8-16	1	1-4	1 or less
In use since	late 1800s	1950	1990	1991
Toxicity	high	high	low	low
Overcharge tolerance	high	moderate	low	low
Operating temperature range (Celsius)	-20 to 60	-40 to 60	-20 to 60	-20 to 60

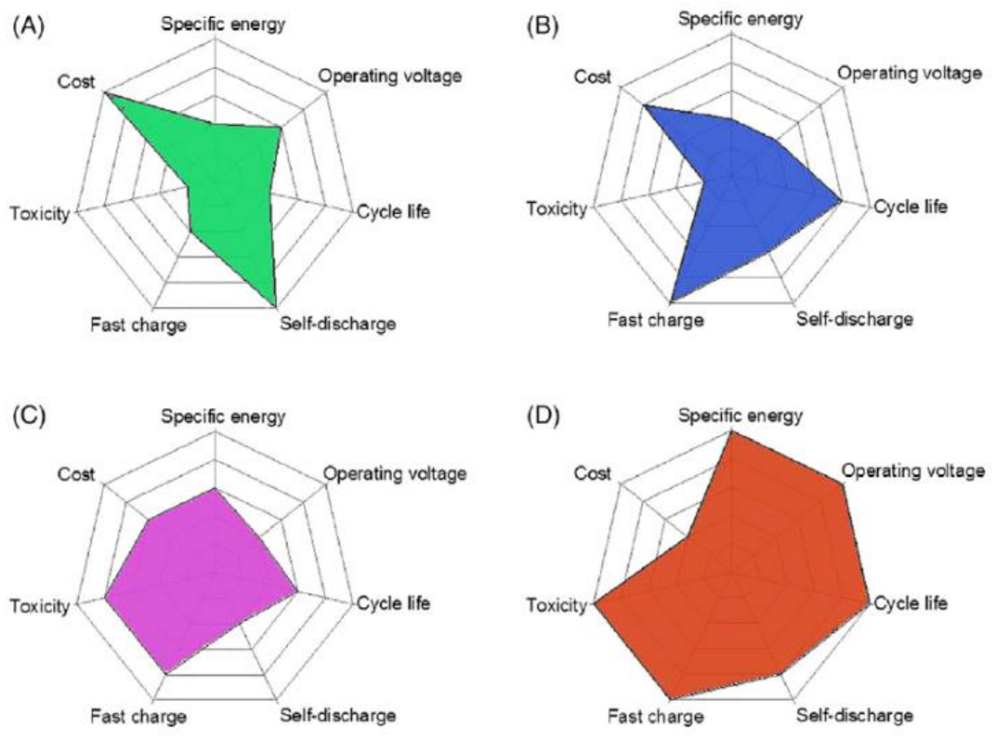


Figure 2.7: Overall Characteristic (Liang Y, Zhao C-Z, Yuan H, *et al.*,2019).

From the Table 2.4, the cycle life for Li-ion battery is varied and start from 500 until to 2000 because its need to depend on the environment factor. For sure, the Table 2.4 is stated that the Li-ion battery has the highest cycle life compared to other three batteries. For Ni-MH battery, it has highest self-discharge per month while lead- acid battery and Li-ion battery have the least (Liang Y, Zhao C-Z, Yuan H, *et al.*,2019). Moreover, Li-ion batteries are less in weight, offer a higher energy density, no bothersome memory effects, and improved safety compared to Ni-MH batteries. Lastly, Li-ion battery has the cheapest cost compared to other batteries and this will be a good choice to the project. As a conclusion, Li-ion battery has totally replaced successfully to the lead acid battery, Ni-Cd battery and Ni-MH battery after comparison as Figure 2.7 is shown. Therefore, the Li-ion battery had been used widely by different electronic company like Apple, Lenovo, Huawei and so on (Liang Y, Zhao C-Z,Yuan H, *et al.*,2019). After a few years, there is one developed Li-ion battery that called Li-ion polymer batterie since the polymer electrolytes replace liquid electrolytes. As the polymer electrolyte, a high-conductivity gel containing lithium salts is frequently employed. It should be noted that Li-ion polymer batteries, particularly for ultra-slim computers, mobile phones, tablets, and wearable electronic gadgets, are very appealing for portable electronics devices due to their small and adaptable structure (Liang Y, Zhao C-Z,Yuan H, *et al.*,2019).

Any rechargeable battery system's charger circuit layout and related control strategy determine how well it performs (Alvaro Aguilar,2019). Initiate charging, rate optimization, and charge termination are the three main jobs of a battery charger (Nourallah Ghaeminezhad and Mohammad Monfared,2021). After measuring the voltage across the battery, the charging procedure starts and continues until a certain voltage is reached, and then it stops at the point. By doing this, each charging system has a BMS that manages all charging activities. In other words, communication between the battery, charger, and load occurs via the BMS. There are few methods to represent of different charger circuits which are linear charger, pulse charger and switch mode charger.

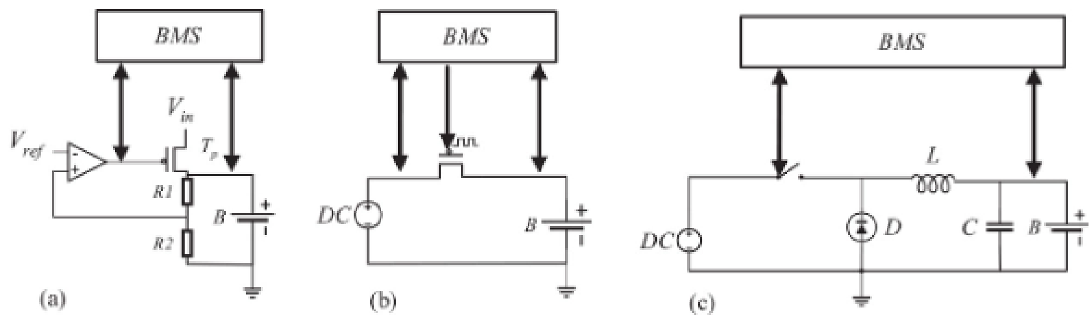


Figure 2.8: Simplified Representation of Different Battery Charger Circuits.

In Figure 2.8a, a linear charger operates in the same straightforward manner as a linear regulator. Through a resistor or a transistor, the linear regulating element in linear regulators lowers the input voltage to a predetermined output voltage. The charger features additional circuitry intended to control and protect battery charge, which distinguishes it from the linear regulator. Linear chargers are popular due to their simplicity and affordability, but the constant current that constantly runs through the regulating element causes heat loss and an inefficient charger (Nourallah Ghaeminezhad and Mohammad Monfared,2021). The current is pulsed into the battery with pulse chargers by toggling transistor, as shown in Figure 2.8b. In order to increase efficiency and speed up charging, pulse chargers can adjust the pulse width and period with an additional circuit. A pulse charger is more efficient than a linear charger and easier to use than a switch-mode charger and causing the costs are increasing. For pulse chargers, the input voltage needs to be carefully regulated (Nourallah Ghaeminezhad and Mohammad Monfared,2021). The only difference between switch-mode chargers and switch-mode power supply is that switch-mode chargers use a sophisticated circuit design to control charging and safeguard the battery. Switch-mode chargers use less electricity to function and produce less heat because the switches are not always active. However, compared to linear chargers, switch-mode chargers are far more complex and expensive. The switch-mode charger is represented simply in Figure 2.8c (Nourallah Ghaeminezhad and Mohammad Monfared,2021).

CHAPTER 3

METHODOLOGY

3.1 Design Architecture in Detailed

In my project, the characteristics of Stream Deck that I would like to build are small size, lighter weight, convenient to carry, and wireless with rechargeable. Microcontroller used in this project is ESP32 – WROOM – 32D. There are 6 buttons slots are set in my project. In this stream deck, the signal will be generated from the TFT-LCD screen and then the ESP32 – WROOM – 32D is used to operate the task what have set into the configuration. The ESP32 – WROOM – 32D has a built-in WiFi and Bluetooth features itself which is an electronic device that allows us to connect to between devices and devices through internet or Bluetooth. In this project, the ESP32 – WROOM – 32D is required to connect to the specific internet protocol address to configure the tasks for each button. Moreover, a device come with Bluetooth like laptop or smart phone is required to connect with the ESP32 – WROOM – 32D. This is because the signal will be sent to the laptop or smart phone and the task will be operated together also. First, a lithium battery with built in smart protective PCB charging module is connected with charging board to create a wireless feature. This lithium battery will help to power up the stream deck no matter how the far the distance. The stream deck can be charged by using type-c cable and universal serial bus (USB) cable. After powering up the stream deck, TFT-LCD screen is turned on, there are 6 configured buttons on the screen. User can just press on the screen and the specific task will be operated on the connected Bluetooth device.

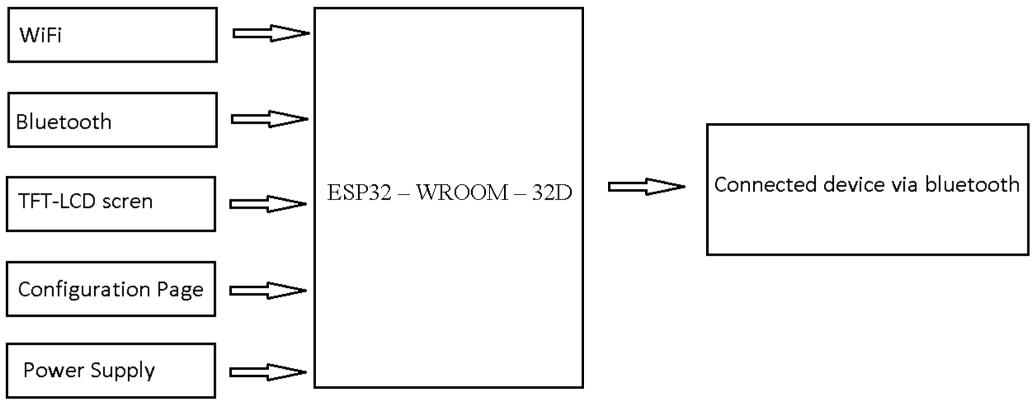


Figure 3.1: Block Diagram of a Stream Deck.

3.2 Project Management

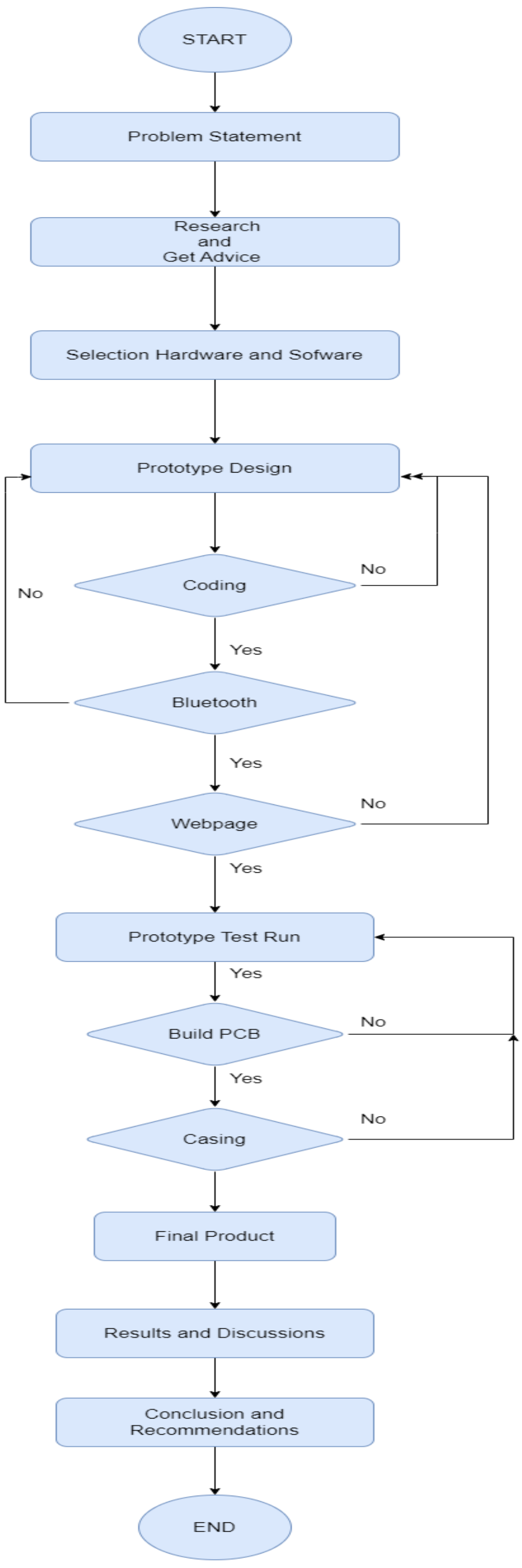
The project schedules are shown in Table 3.1 and 3.2, respectively.

Table 3.1: Gantt Chart for FYP1.

Task/Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Discussion with supervisor	🟩													
Project title selection/ Proposed a title	🟩	🟩												
Research		🟩	🟩	🟩										
Build Prototype				🟩	🟩	🟩	🟩	🟩						
FYP1														
Introduction					🟩	🟩								
Problem statement					🟩	🟩								
Objective					🟩	🟩								
Literature review						🟩	🟩	🟩	🟩					
Methodology		🟩	🟩						🟩	🟩	🟩	🟩		
Presentation														
FYP1 presentation													🟩	

Table 3.2: Gantt Chart for FYP2.

Task/Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
FYP2														
Build Prototype	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩			
Results								🟩	🟩	🟩	🟩			
Conclusion											🟩	🟩		
Presentation														
Poster Presentation												🟩	🟩	
FYP2 Presentation														🟩



Flowchart 1:Work Plane of Stream Deck.

3.3 Hardware used in a Stream Deck

3.3.1 ESP32-WROOM- 32D Module Board

The main reason of choosing ESP32-WROOM-32D board in this project is due to its open source, Arduino-like hardware, lower cost, lighter weight, small size, 3.3V operated, can be powered by USB, this board has built-in WiFi, Bluetooth, and contains GPIO pins. Arduino module is a great choice to the project, but these modules do not contain built-in WiFi feature. Therefore, additional fees have to be paid for the additional features. The more features are added on to the system, the more fees you have to pay. However, ESP32-WROOM-32D board as shown in Figure 3.2, incorporates a built-in WiFi support and Bluetooth support, giving an easy pathway to design IoT applications. The features as stated above make the ESP32-WROOM-32D board enormously powerful tool for WiFi networking. ESP32-WROOM-32D board can be used to host a web server, used as access point, or link to internet to upload or fetch data. It is an open-source firmware that is extremely important in designing IoT product. The pin layout of ESP32-WROOM-32D board is shown in Figure 3.3.

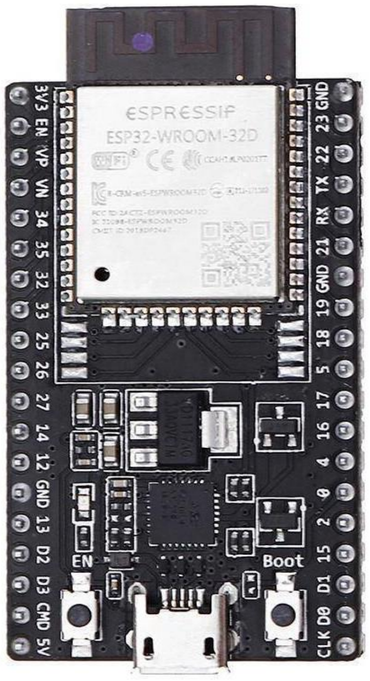

ESPRESSIF
ESP32-WROOM-32D
3V3 EN VP VN 34 35 32 33 25 26 27 14 12 GND 13 D2 D3 CMD 5V
GND 23 22 TX RX 21 GND 19 18 5 17 16 4 0 2 15 D1 D0 CLK
EN
Boot

Figure 3.2: ESP32-WROOM-32D Board (Hallroad,2022).

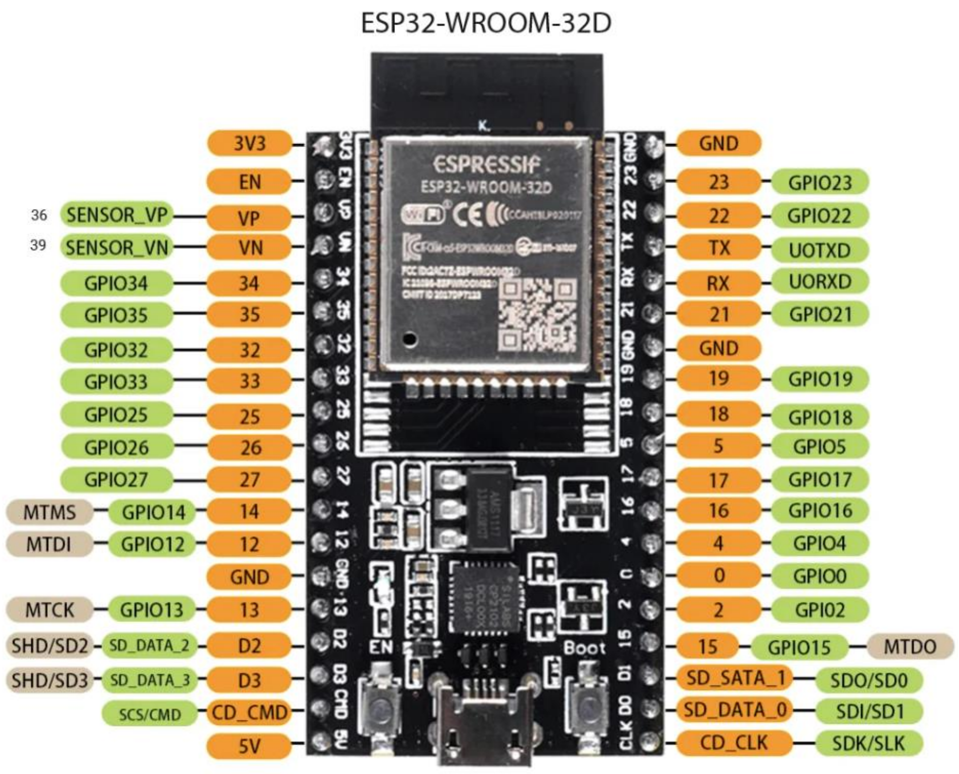


Figure 3.3: Pinouts of ESP32 – WROOM – 32D (ebay,2018).

Table 3.3: Features of Each Pin Label.

Pins category	Pin Name	Descriptions
Power	Micro-USB, 3.3V,5V,GND	Micro-USB: ESP32 can be powered through USB port 5V: Regulated 5v can be supplied to this pin which is we be again regulated to 3.3V by on board regulator, to power the board. 3.3V: Regulated 3.3V can be supplied to this pin to power the board. GND: Ground pins.
Enable	EN	To reset the microcontroller.
Analog Pins	ADC1_0 to ADC1_5 and ADC2_0 to ADC2_9	Used to measure analog voltage in the range of 0-3.3V. 12-bit 18channel ADC.
DAC pins	DAC1 and DAC2	Used for Digital to analog Conversion.
Input/output Pins	GPIO0 to GPIO39	Used as input or outputs pins. 0v (low) and 3.3V (high). But pins 34 to 39 can be used as input only.
Capacitive Touch Pins	T0 to T9	Used a touch pin normally used for capacitive pads.
RTC GPIO Pins	RTCIO0 to RTCIO17	Used to wake up the ESP32 from deep sleep mode.
Serial	Rx,Tx	Used to receive and transmit TTL serial data.
External Interripts	All GPIO	Used to trigger and interrupt
PWM	All GPIO	16 independent channel is available for PWM any GPIO can be made to work as PWM though software
SVSPI	GPIO23 (MOSI),GPIO19(MISO),GPIO18 (CLK) and GPIO5 (CS)	Used for SPI-1 communication
HSPI	GPIO13 (MOSI),GPIO12(MISO),GPIO14 (CLK) and GPIO15 (CS)	Used for SPI-2 communication
IIC	GPIO21 (SDA), GPIO22(SCL)	Used for I2C communication
AREF	AREF	To provide reference voltage for input voltage

3.3.2 Display Screen

From the Figure 3.4, there is a TFT screen is selected and called SPI MSP 3520 module ILI 9488 TFT screen. The ILI9488 is a 16.7M single-chip SoC driver for 320(RGB) x 480 dot a-Si TFT liquid crystal display panels. The ILI9488 consists of a power supply circuit, a 960-channel source driver, a 480-channel gate driver, and 345,600 bytes of GRAM for 320 (RGB) x 480-dot graphics. Parallel DBI Type B 8/9/16/18/24-bit data bus interfaces and DBI Type C 3-/4-line SPI are both supported by the ILI9488 for command input. Shift registers, sensors, and SD cards are just a few examples of the small peripherals that are frequently connected to microcontrollers via the SPI interface bus. The device users want to talk to is selected using a choose line, which also uses separate clock and data lines. For the presentation of video images, the ILI9488 offers a DPI (16-/18-/24-bit) data bus. The ILI9488 also offers one data lane and one clock lane that can enable up to 500Mbps on MIPI-DSI link for MIPI*-DSI* high-speed interface mode. The ILI9488 supports a variety of analogue power supply and can run at 1.65V I/O interface voltage. The ILI9488 is perfect for portable products where battery power conservation is desired, such as digital cellular phones, smart phones, MP3 players, personal media players, and similar devices with colour graphics displays. It supports an 8-colour display and sleep mode power management functions ILI 9488 TFT screen has been separated into two model which are MSP 3520 and MSP 3521. MSP 3520 has touch screen feature while MSP 3521 has no touch screen feature. Of course, the price will be different in RM 20 ringgit to get the touch feature. Furthermore, the pin label and the features had been listed into Table 3.4.

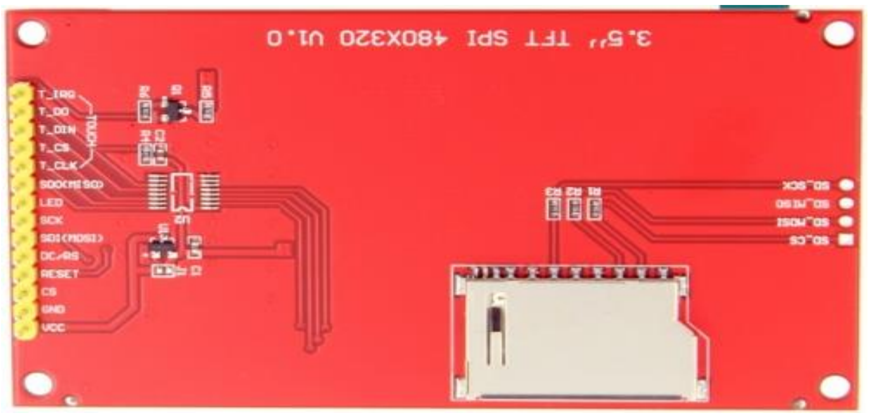


Figure 3.4: Pinouts of SPI MSP 3520 module ILI 9488 TFT screen(LCwiki,2021.)

Table 3.4: Features of Each Pin Label.

Number	Pin Label	Description
1	VCC	5V/3.3V power input.
2	GND	Ground.
3	CS	LCD chips select signal, low label enables.
4	RESET	LCD reset signal low level reset.
5	DC/RS	LCD register/ data selection signal High level: register, low level: data.
6	SDI(MOSI)	SPI bus writes data signal.
7	SCK	SPI bus clock signal.
8	LED	LED backlight control, high level lighting, if not controlled, connect 3.3V always bright.
9	SDO(MISO)	SPI bus read data signal if you do not need to the read function. You cannot connect it.
THE FOLLOWING IS THE TOUCH SCREEN SIGNAL LINE WIRING, if you do not need to touch function or the module itself does not have touch function, you cannot connect them.		
10	T_CLK	Touch SPI bus clock signal.
11	T_CS	Touch screen chip select signal, low level enable.
12	T_DIN	Touch SPI bus input.
13	T_DO	Touch SPI bus output.
14	T_IRQ	Touch screen interrupt signal, low level when touch is detected.

3.3.3 Wireless and Rechargeable System

A type-C USB 5V 2A step-up boost converter with USB charger is selected, as shown in Figure 3.5. Moreover, a lithium battery capacity indicator has been chosen and show in Figure 3.6. In order to complete the wireless system, a power supply 3.7V LiPo battery (Lithium Polymer) had been selected as Figure 3.7 is shown.
The main reason to choose type-C USB 5V 2A step-up boost converter with USB charger is because of having a great step-up module with 5V input, output is approximately to 5V for many digital devices, so this is really a great module for designing a portable charger. Useful for your USB charger projects and onboard USB device supply solutions. This charger will cause the output turn off when the load current is less than 50mA continuously. When the charging current drops to 100mA after reaching the final floating charging voltage, the charging cycle will be automatically terminated. However, when the battery voltage drops below 4.1V, the charging cycle starts again. When the battery voltage is lower than 2.8V, the battery will be precharged with a current of 180mA. As the title of the product, the charger will boost the output to approximately 2A and with 92.5% of conversion efficiency from 3.6V input and 5V2A output. Since the Stream Deck is ready to build in a small size, this charger has small dimension of size, lighter weight, and indicator LED for charging. Therefore, this charger is suitable for this project since it has great specification and characteristics. Since there are many products of the batteries with approximately same specifications and price but the main reason of choosing LiPo battery is due to it has built-in BMS protecting circuit to prevent overcharging damage issues. Moreover, the self-discharge per month of this LiPo battery is less than 10% as stated in Chapter 2. Moreover, the capacity of the of this battery is 2000mah but it has small size and lighter weight. In addition, it can last longer and fulfil the requirement of the characteristic. For rechargeable system, a capacity indicator of battery is selected to display the rest of the capacity of the battery.

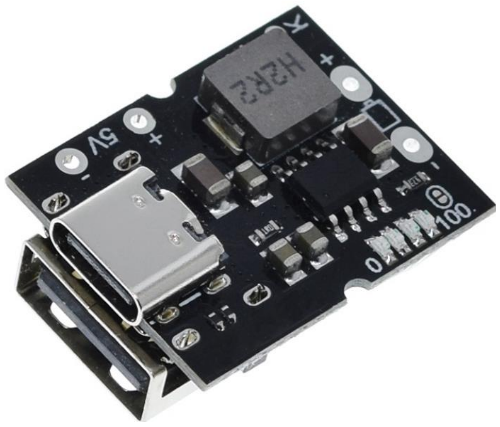


Figure 3.5: Type-C USB 5V 2A Step-up Boost Converter with USB Charger (ShharviElectronic,2022).

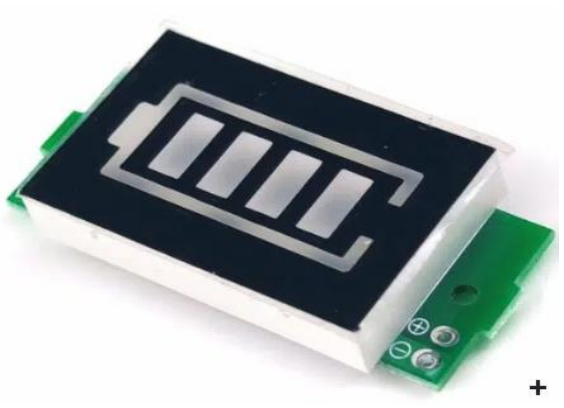


Figure 3.6: A Lithium Battery Capacity Indicator (Universal-solder,2020).

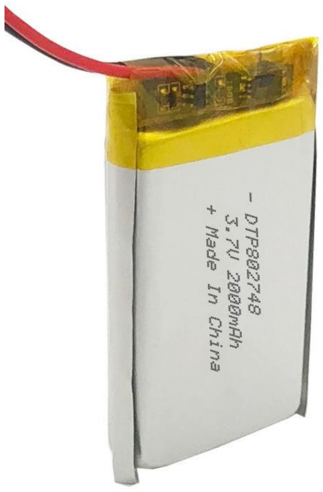


Figure 3.7: A Power Supply 3.7V LiPo Battery (Lithium Polymer) (Shopee,2023.).

3.4 Circuit Connection of Hardware

The Figure 3.8 is shown for Stream Deck circuit connection by using the LiPo battery, type-C USB 5V 2A step-up boost converter with USB charger, slicer, lithium battery capacity indicator, ESP32-Wroom-32D development board and SPI MSP 3520 module ILI9488 TFT screen with touch screen mode.

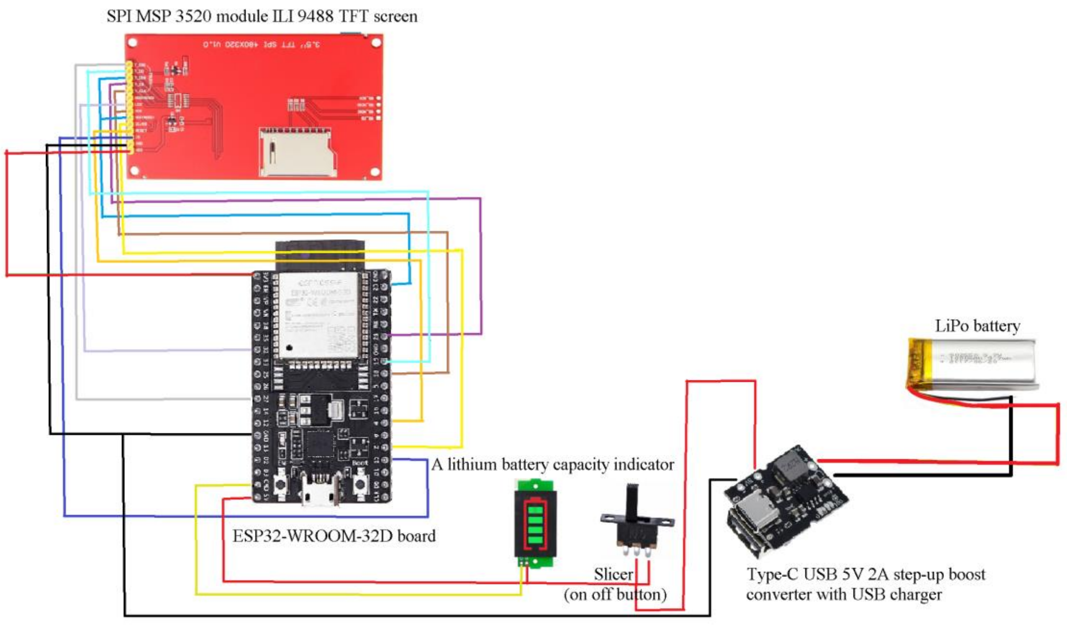

SPI MSP 3520 module ILI 9488 TFT screen
LiPo battery
A lithium battery capacity indicator
ESP32-WROOM-32D board
Slicer
(on off button)
Type-C USB 5V 2A step-up boost converter with USB charger

Figure 3.8:Stream Deck Circuit Connection.

Table 3.5: Hardware of the Stream Deck.

Components and materials	Amount
ESP32-Wroom-32D	x1
SPI MSP 3520 module ILI9488 TFT Screen	x1
Lithium Battery Capacity Indicator	x1
Slicer (On/Off Button)	x1
Type-C USB 5V 2A step-up boost converter with USB charger	x1
3D printed case	x1

3.4.1 Circuit Connection between the ILI9488 TFT Screen and the ESP32-Wroom-32D

The Table 3.6 shows two electronic devices and its’ pins connection. The ILI9488 TFT screen contains T_IRQ, T_DO, T_DIN, T_CS, SDO(MISO), LED, SCK, SDI(MOSI), DC/RS, RESET, CS, GND, and VCC as well as connected to pins respectively from ESP32-Wroom-32D which are GPIO27, GPIO19, GPIO23, GPIO21, GPIO18, no connected, GPIO32, GPIO18, GPIO23, GPIO2, GPIO4, GPIO15,GND, and 3V3.

Table 3.6: Connection between the ILI9488 TFT Screen with ESP32-Wroom-32D.

ILI9488 TFT Screen Pins	ESP32-Wroom-32D Pins
T_IRQ	GPIO27
T_DO	GPIO19
T_DIN	GPIO23
T_CS	GPIO21
T_CLK	GPIO18
SDO (MISO)	N.C
LED	GPIO32
SCK	GPIO18
SDI (MOSI)	GPIO23
DC/RS	GPIO2
RESET	GPIO4
CS	GPIO15
GND	GND
VCC	3V3

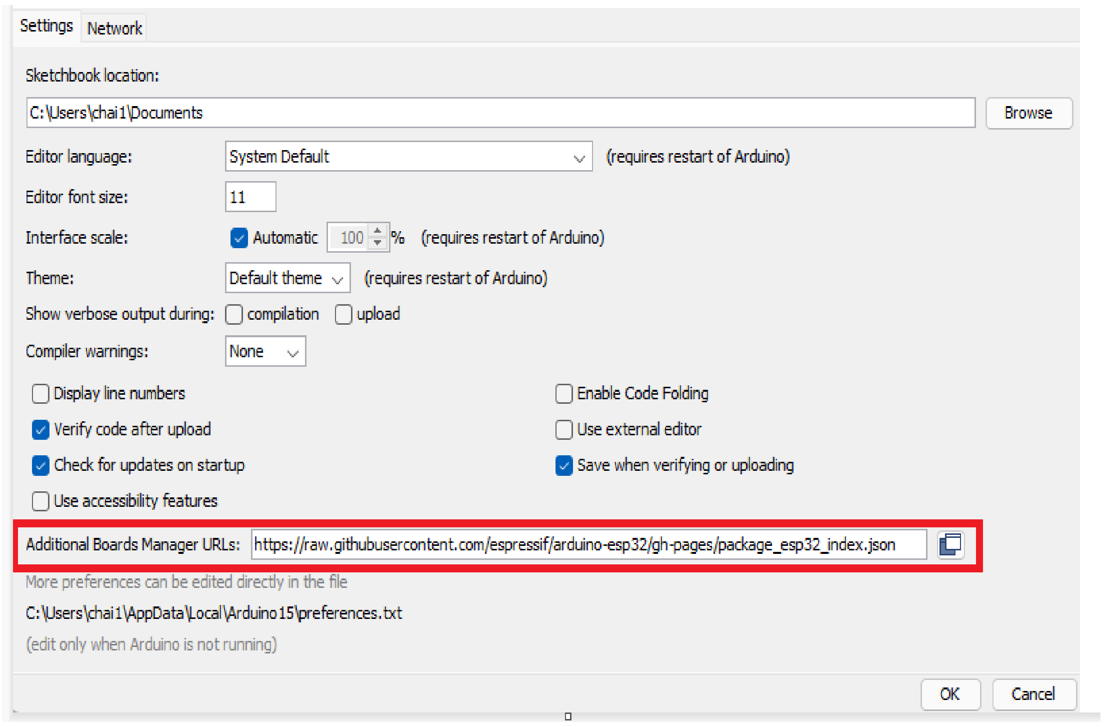
3.5 Arduino IDE (Integrated Development Environment)

The selection of the Arduino IDE software for coding purposes is illustrated in Figure 3.9. The Arduino Integrated Development Environment (IDE) version 1.8.19, which have been obtained from the official website "https://www.arduino.cc/en/software", is considered a legacy version.

Figure 3.9: Arduino IDE Software.

3.5.1 Arduino IDE Setup

Figure 3.10 illustrated the procedure for integrating the Arduino IDE ESP32 core with the Arduino IDE application. Upon opening the Arduino IDE software, the "File" and "Preferences" options is proceeded to select. In Figure 3.9 illustrated the process of copying and pasting the link "https://raw.githubusercontent.com/espressif/arduinoesp32/ghpages/package_esp32_index.json" into the "Additional Boards Manager URLs" section.



Settings Network

Sketchbook location:
C:\Users\chai1\Documents
Browse

Editor language: System Default (requires restart of Arduino)
Editor font size: 11
Interface scale: [✓ Automatic] 100 % (requires restart of Arduino)
Theme: Default theme (requires restart of Arduino)
Show verbose output during: ☐ compilation ☐ upload
Compiler warnings: None

☐ Display line numbers
[✓ Verify code after upload]
[✓ Check for updates on startup]
☐ Use accessibility features
☐ Enable Code Folding
☐ Use external editor
[✓ Save when verifying or uploading]

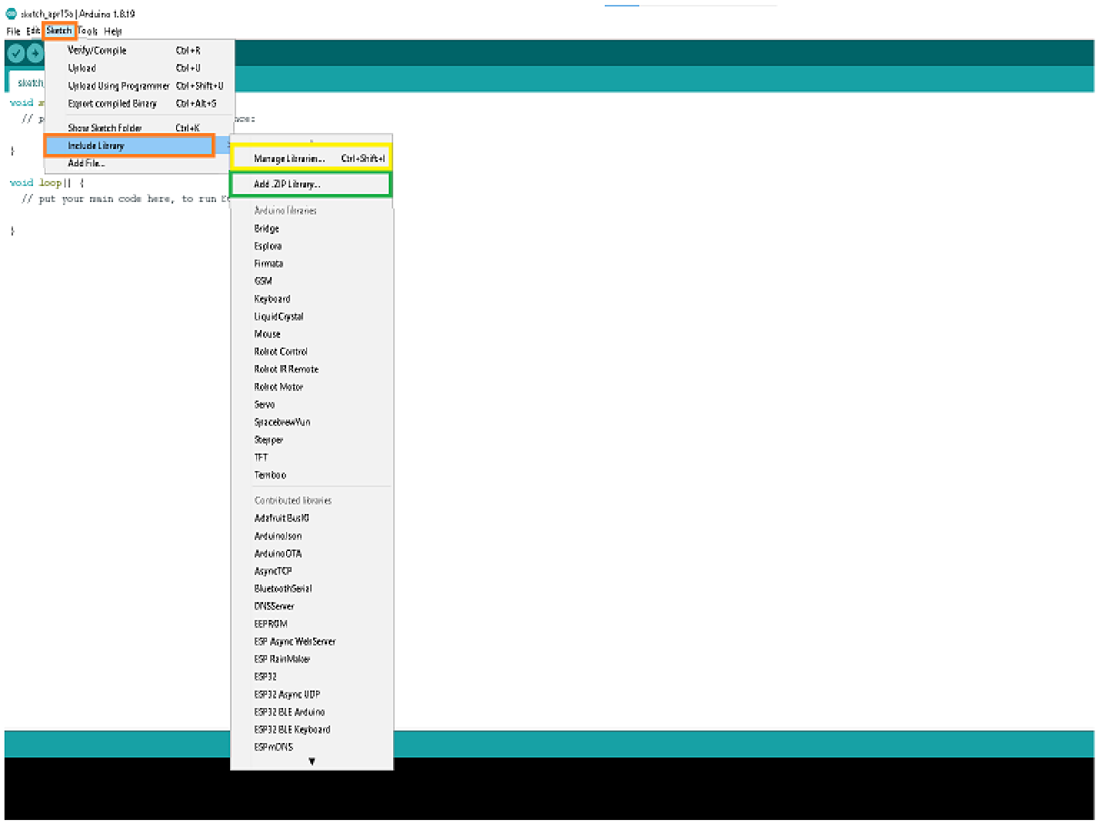
Additional Boards Manager URLs: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

More preferences can be edited directly in the file
C:\Users\chai1\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

OK Cancel

Figure 3.10: Installing the Arduino IDE ESP32 core.

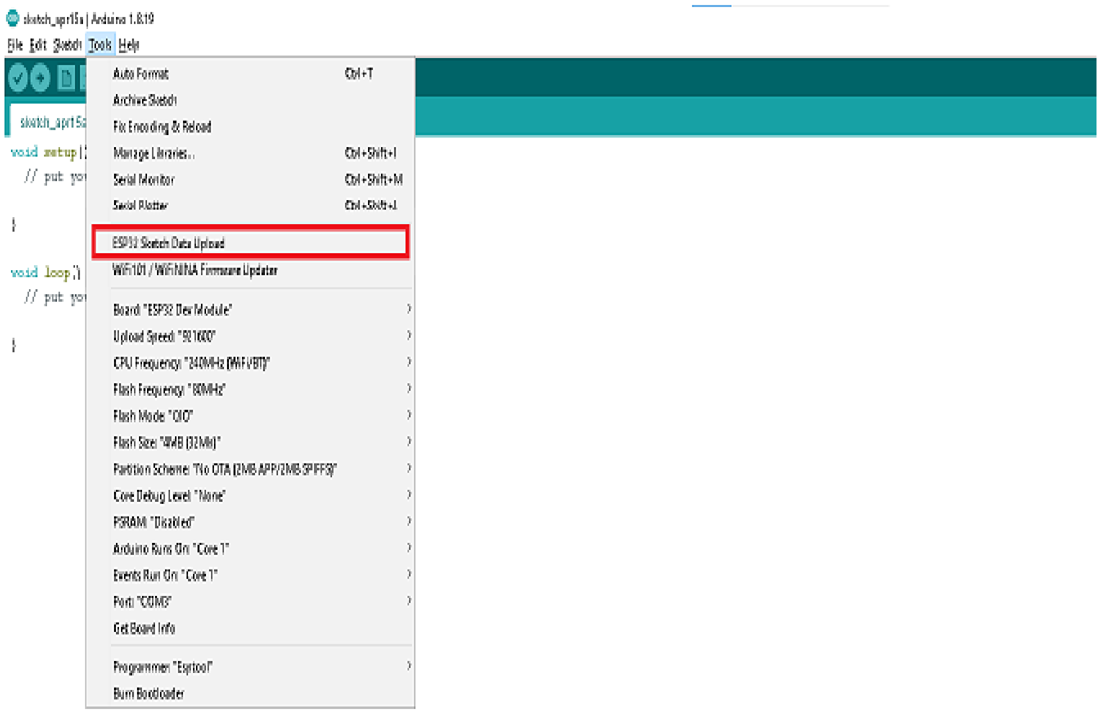
Figure 3.11 illustrated the process of installing libraries into the Arduino Integrated Development Environment (IDE) software. To facilitate the installation of libraries, the "Sketch" button is pressed and subsequently selected "Include Library" after navigating to the "Manage Libraries" option denoted by the yellow box. Subsequently, the green box labelled "Add ZIP Library" is selected to facilitate the installation of the source library obtained from the internet or GitHub. Consequently, a limited number of libraries have been installed, including Adafruit-GFX-Library, TFT-eSPI, ArduinoJson, ESPAsyncWebServer, AsyncTCP, and ESP32-BLE-Keyboard.



```
void s
  // p
}

void loop() {
  // put your main code here, to run r
}
```

Figure 3.11: Installing the Necessary Libraries.

The stream deck utilized SPIFFS, a flash memory technology utilized by ESP32, to retain configuration and image data. It is necessary to transmit these to the ESP32 before uploading the functional design. The utilization of the ESP32 Sketch Data Upload application is necessary. The resource in question is accessible for download on the GitHub platform, specifically at the following URL: https://github.com/me-no-dev/arduino-esp32fs-plugin". Figure 3.13 depicted the extraction of the tools file to the Arduino sketchbook directory. After this, the ESP32 Sketch Upload Tool will become available upon the relaunch of the Arduino IDE software, as depicted in Figure 3.12.



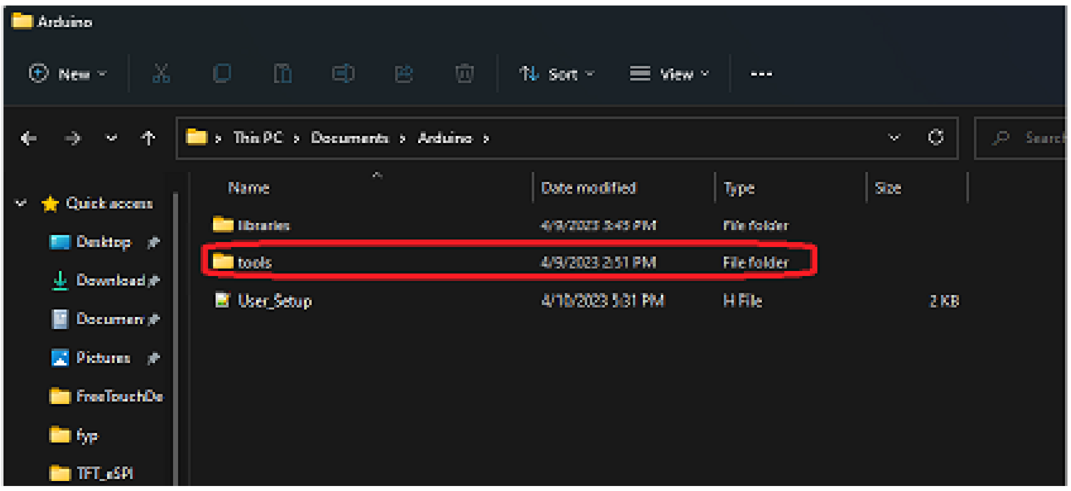
sketch_apr15a | Arduino 1.8.19

File Edit Sketch Tools Help

```
void setup() {
  // put yo

}

void loop() {
  // put yo

}
```

Auto Format	Ctrl+T
Archive Sketch	
Fix Encoding & Reload	
Manage Libraries...	Ctrl+Shift+I
Serial Monitor	Ctrl+Shift+M
Serial Plotter	Ctrl+Shift+L
ESP32 Sketch Data Upload	
WiFi101 / WiFiNINA Firmware Updater	
Board: "ESP32 Dev Module"	
Upload Speed: "921600"	
CPU Frequency: "240MHz (WiFi/BT)"	
Flash Frequency: "80MHz"	
Flash Mode: "QIO"	
Flash Size: "4MB (32Mb)"	
Partition Scheme: "No OTA (2MB APP/2MB SPIFFS)"	
Core Debug Level: "None"	
PSRAM: "Disabled"	
Arduino Runs On: "Core 1"	
Events Run On: "Core 1"	
Port: "COM3"	
Get Board Info	
Programmer: "Esptool"	
Burn Bootloader	

Figure 3.12: Installing ESP32 Sketch Data Upload tool.

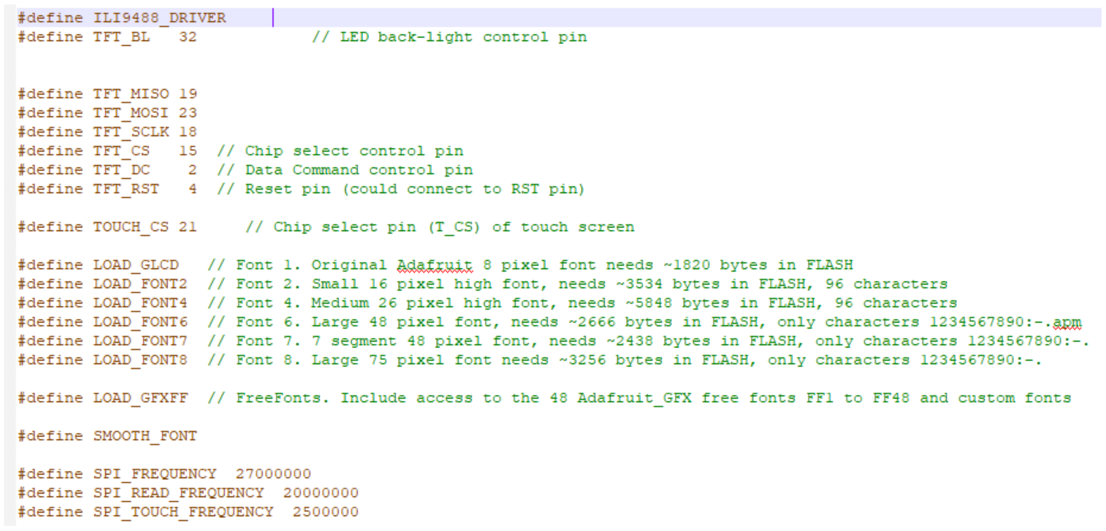


Name	Date modified	Type	Size
libraries	4/9/2023 3:45 PM	File folder	
tools	4/9/2023 2:51 PM	File folder	
User_Setup	4/10/2023 5:31 PM	H File	2 KB

Figure 3.13: Installing ESP32 Sketch Upload Tool.

3.5.2 Setup of SPI MSP 3520 module ILI9488 TFT Screen

Figure 3.14 presented a code segment that defines several constants for a program designed to operate a TFT display with touch screen capabilities on an ESP32-Wroom-32D device. The initial lines of code seem to be defining the pin numbers for the various connections of the display, such as the SPI communication pins (MISO, MOSI, SCLK), chip select pins (CS), and data/command pin (DC). The code additionally designated a pin for the purpose of controlling the backlight of the display and specifies the requisite level, either HIGH or LOW, to activate it. Furthermore, the source code incorporates various font and free typeface libraries that can be integrated into the program's flash memory. The code additionally ascertains the SPI read frequency, which is utilized for the purpose of retrieving data from the display, and the touch frequency, which is employed for communication with the touch screen.

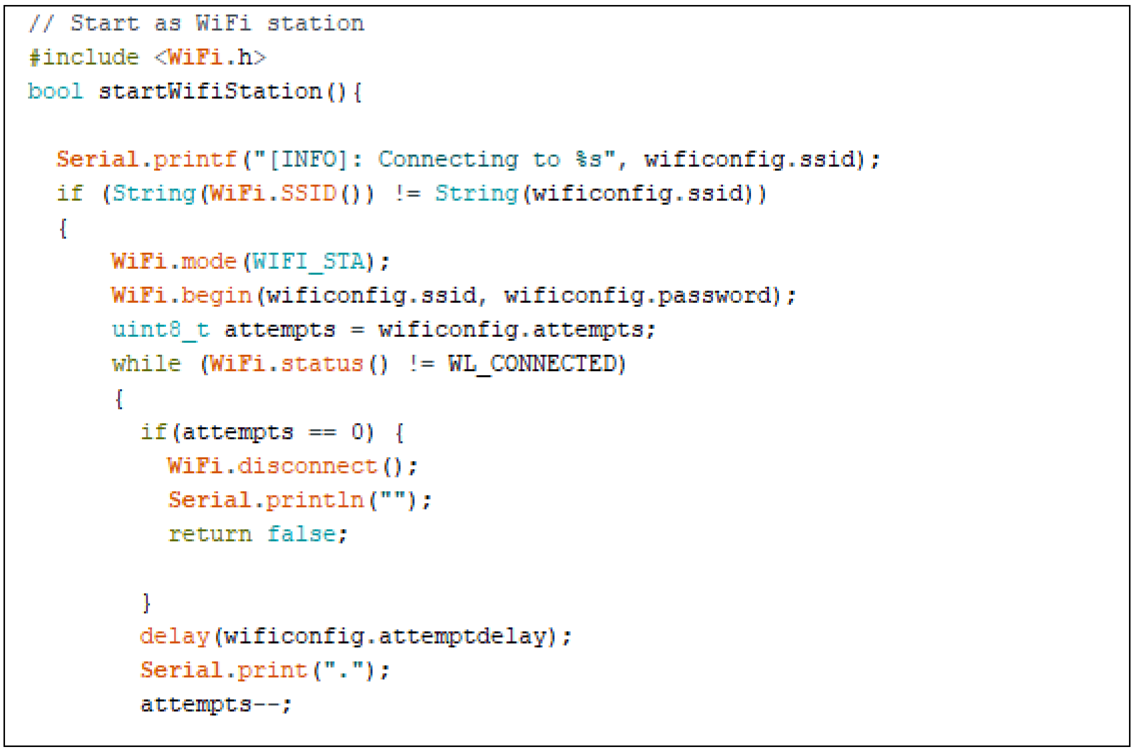


```
#define ILI9488_DRIVER
#define TFT_BL   32            // LED back-light control pin


#define TFT_MISO 19
#define TFT_MOSI 23
#define TFT_SCLK 18
#define TFT_CS   15  // Chip select control pin
#define TFT_DC    2  // Data Command control pin
#define TFT_RST   4  // Reset pin (could connect to RST pin)

#define TOUCH_CS 21     // Chip select pin (T_CS) of touch screen

#define LOAD_GLCD   // Font 1. Original Adafruit 8 pixel font needs ~1820 bytes in FLASH
#define LOAD_FONT2  // Font 2. Small 16 pixel high font, needs ~3534 bytes in FLASH, 96 characters
#define LOAD_FONT4  // Font 4. Medium 26 pixel high font, needs ~5848 bytes in FLASH, 96 characters
#define LOAD_FONT6  // Font 6. Large 48 pixel font, needs ~2666 bytes in FLASH, only characters 1234567890:-.apm
#define LOAD_FONT7  // Font 7. 7 segment 48 pixel font, needs ~2438 bytes in FLASH, only characters 1234567890:-.
#define LOAD_FONT8  // Font 8. Large 75 pixel font needs ~3256 bytes in FLASH, only characters 1234567890:-.

#define LOAD_GFXFF  // FreeFonts. Include access to the 48 Adafruit_GFX free fonts FF1 to FF48 and custom fonts

#define SMOOTH_FONT

#define SPI_FREQUENCY  27000000
#define SPI_READ_FREQUENCY  20000000
#define SPI_TOUCH_FREQUENCY  2500000
```

Figure 3.14: Defining for ILI9488 TFT Screen.

3.5.3 WiFi Setup Function

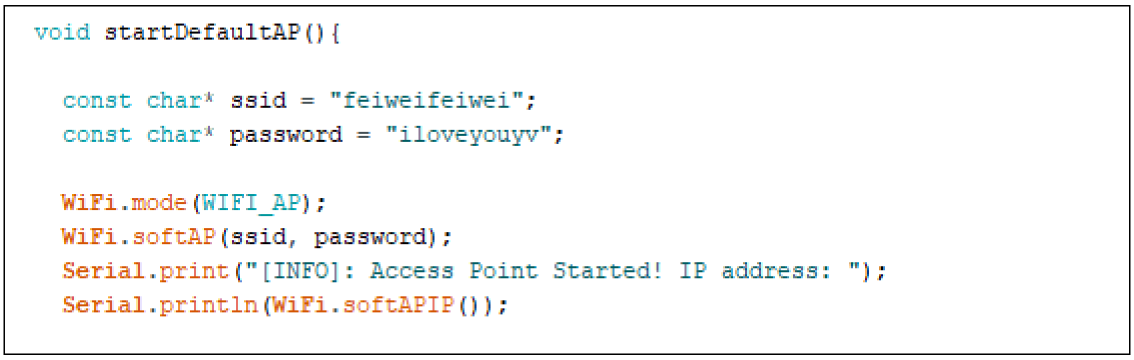
The execution of the webserver function is based upon the establishment of a connection to either the Station Point (STA) or Access Point (AP) within the WiFi setup. In Figure 3.15, a C++ function called "startWifiStation" makes up the offered snippet of code. To connect to a WiFi network in station (client) mode, use this function. The function produces a boolean result that indicates whether the connection is successful or not and requires no parameters. The program initially prints an informative message to the console using the Serial.printf() function, stating which WiFi network it is attempting to connect to. By gaining access to the "wificonfig" global variable, it is expected to have the SSID (network name) and password of the WiFi network to connect to. The function next determines whether the ESP32 WiFi module is currently associated with the chosen WiFi network. If not, it switches the module to station mode (WiFi.mode(WIFI_STA)) and uses the WiFi.begin() method to try to connect to the network. After that, a loop in the method checks the WiFi connection's state. If the connection has not yet been established, the function writes a "." to the console to show that it is still trying to connect after waiting for a certain length of time (wificonfig.attemptdelay). Until the connection is made or until the maximum number of attempts (wificonfig.attempts) is reached, this procedure keeps going. The function disconnects from the network and returns the result "false" if the connection cannot be made within the allotted number of tries. In all other cases, it returns a value of "true" to denote a successful connection. Overall, this code sample shows how to use the Arduino IDE to connect an ESP32 WiFi module quickly and easily to a WiFi network in client mode.



```
// Start as WiFi station
#include <WiFi.h>
bool startWifiStation(){

  Serial.printf("[INFO]: Connecting to %s", wificonfig.ssid);
  if (String(WiFi.SSID()) != String(wificonfig.ssid))
  {
      WiFi.mode(WIFI_STA);
      WiFi.begin(wificonfig.ssid, wificonfig.password);
      uint8_t attempts = wificonfig.attempts;
      while (WiFi.status() != WL_CONNECTED)
      {
        if(attempts == 0) {
          WiFi.disconnect();
          Serial.println("");
          return false;

        }
        delay(wificonfig.attemptdelay);
        Serial.print(".");
        attempts--;
```

Figure 3.15: Attempting WiFi Setup.

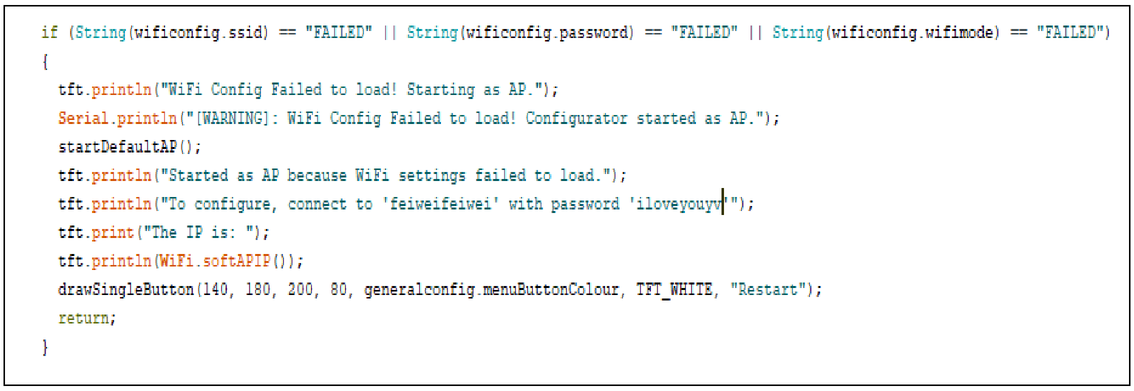
In Figure 3.16, the code then compares these values to those in the "wificonfig" variable to determine if the WiFi configuration is set to default (i.e., the SSID and password are "YOUR WIFI SSID" and "YOUR WIFI PASSWORD", respectively). If the default configuration is still active, the code initiates the device as an access point (AP) with a predefined SSID and password ("feiweifeiwei" and "iloveyouyv", respectively) and displays the AP's IP address on the TFT screen.



```
void startDefaultAP(){

  const char* ssid = "feiweifeiwei";
  const char* password = "iloveyouyv";

  WiFi.mode(WIFI_AP);
  WiFi.softAP(ssid, password);
  Serial.print("[INFO]: Access Point Started! IP address: ");
  Serial.println(WiFi.softAPIP());
```

Figure 3.16: SSID and Password for Access Point.

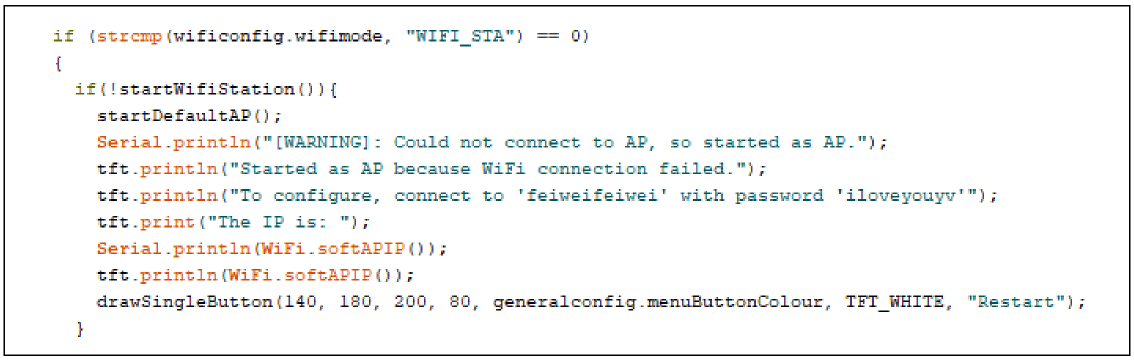
Figure 3.17 showed if the WiFi configuration failed to load or is wifinvalid, the code restarts the device as an access point (AP) with a predefined SSID and password and displays the AP's IP address on the TFT screen.



```
if (String(wificonfig.ssid) == "FAILED" || String(wificonfig.password) == "FAILED" || String(wificonfig.wifimode) == "FAILED")
{
  tft.println("WiFi Config Failed to load! Starting as AP.");
  Serial.println("[WARNING]: WiFi Config Failed to load! Configurator started as AP.");
  startDefaultAP();
  tft.println("Started as AP because WiFi settings failed to load.");
  tft.println("To configure, connect to 'feiweifeiwei' with password 'iloveyouyv'");
  tft.print("The IP is: ");
  tft.println(WiFi.softAPIP());
  drawSingleButton(140, 180, 200, 80, generalconfig.menuButtonColour, TFT_WHITE, "Restart");
  return;
}
```

Figure 3.17: WiFi and Access Point Setup.

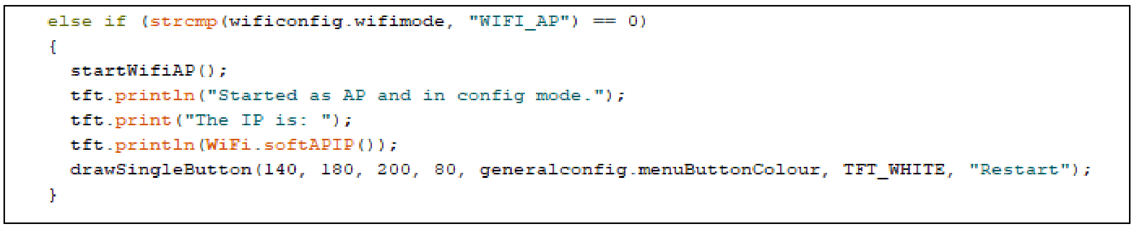
Figure 3.18, if the WiFi configuration is valid and set to "WIFI STA" (station) mode, the code attempts to connect to an SSID and password-protected WiFi access point. If the connection fails, the code restarts the device as an access point (AP) with a predefined SSID and password and displays the AP's IP address on the TFT screen. If the connection is successful, the code displays the IP address of the device on the TFT display.



```
if (strcmp(wificonfig.wifimode, "WIFI_STA") == 0)
{
  if(!startWifiStation()){
    startDefaultAP();
    Serial.println("[WARNING]: Could not connect to AP, so started as AP.");
    tft.println("Started as AP because WiFi connection failed.");
    tft.println("To configure, connect to 'feiweifeiwei' with password 'iloveyouyv'");
    tft.print("The IP is: ");
    Serial.println(WiFi.softAPIP());
    tft.println(WiFi.softAPIP());
    drawSingleButton(140, 180, 200, 80, generalconfig.menuButtonColour, TFT_WHITE, "Restart");
  }
```

Figure 3.18: WiFi and Access Point Setup.

Figure 3.19 illustrated the code starts the device as an AP with the specified SSID and password and displays the AP's IP address on the TFT screen if the WiFi configuration is valid and set to "WIFI AP" (access point) mode.



```
else if (strcmp(wificonfig.wifimode, "WIFI_AP") == 0)
{
  startWifiAP();
  tft.println("Started as AP and in config mode.");
  tft.print("The IP is: ");
  tft.println(WiFi.softAPIP());
  drawSingleButton(140, 180, 200, 80, generalconfig.menuButtonColour, TFT_WHITE, "Restart");
}
```

Figure 3.19: Access Point Setup.

3.6 PCB Board

The selection of Autodesk Eagle software for the purpose of creating a PCB is depicted in Figure 3.20. The software is Autodesk Eagle, specifically version 9.6.2, which is available via download from the official website at "https://www.autodesk.com/products/eagle/free-download".

Figure 3.20: Autodesk Eagle software.

3.6.1 Autodesk Eagle Setup for Schematic File

The creation of a new schematic file can be initiated by selecting the "File" option followed by "New" and subsequently "Schematic", as depicted in Figure 3.21. Figure 3.22 depicted the process of installing libraries for the purpose of drawing schematic and board. Initially, the "Library" can be located on the toolbars and subsequently, the option to "Open library manager" is chosen. The schematic diagram depicted in Figure 3.23 is successfully designed utilizing a primary microcontroller, specifically the ESP32-Wroom-32D, in conjunction with an SPI MSP 3520 module ILI9488 TFT screen featuring touch screen functionality. The schematic diagram is produced using version 9.6.2 of the Eagle software.

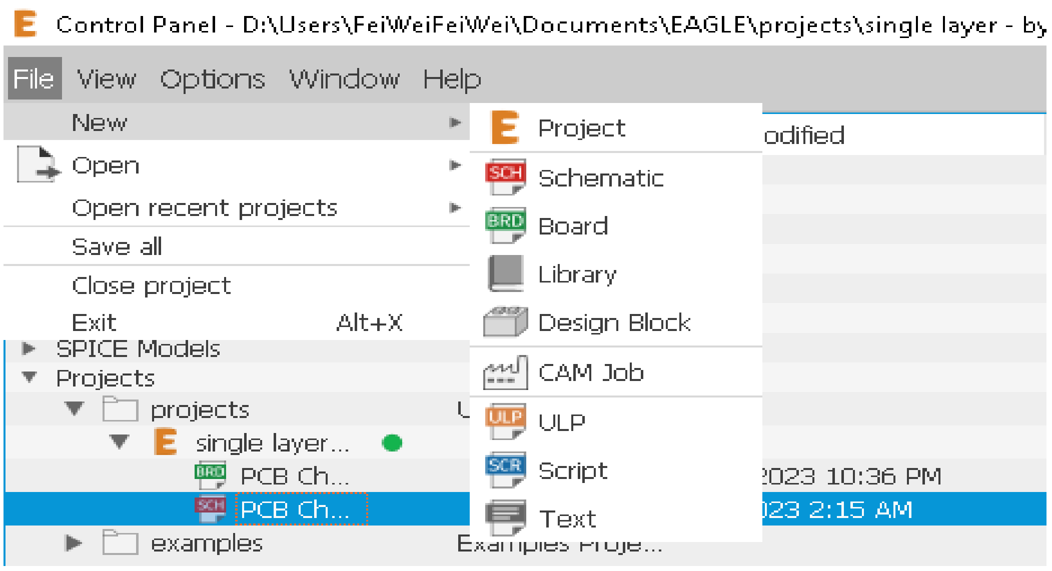


Figure 3.21: New Project had been created.

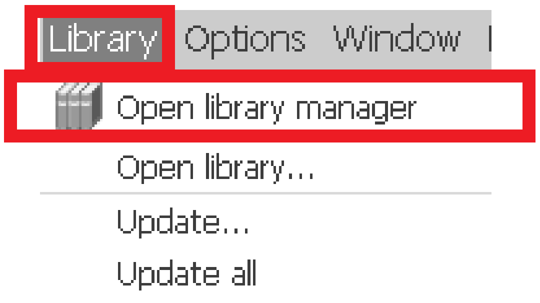


Figure 3.22: Installing Libraries.

Figure 3.24 showed three separate tabs on the "Library Manager" window in green-, blue-, and yellow-coloured boxes. While the blue box is for the libraries that are previously available in the software, the green box refers to processing design libraries. For the program, the yellow box is the place where searching or download libraries and install external source libraries. The library containing the ESP32-Wroom-32D development board's footprint and symbol for this project is downloaded at “https://www.snapeda.com/parts/ESP32-DEVKITC-32D/Espressif%20Systems/view-part/?ref=search&t=ESP32-WROOM 32D%20development%20board&fbclid=IwAR2F1HJVm3cLbTo_smcmajDdxmg2b LSAbNKtEfB9HqOFlAnlFw8FfIhtII”. Once the.zip folder had been downloaded, pick the relevant file by clicking the "Browse" option as shown in Figure 3.24.

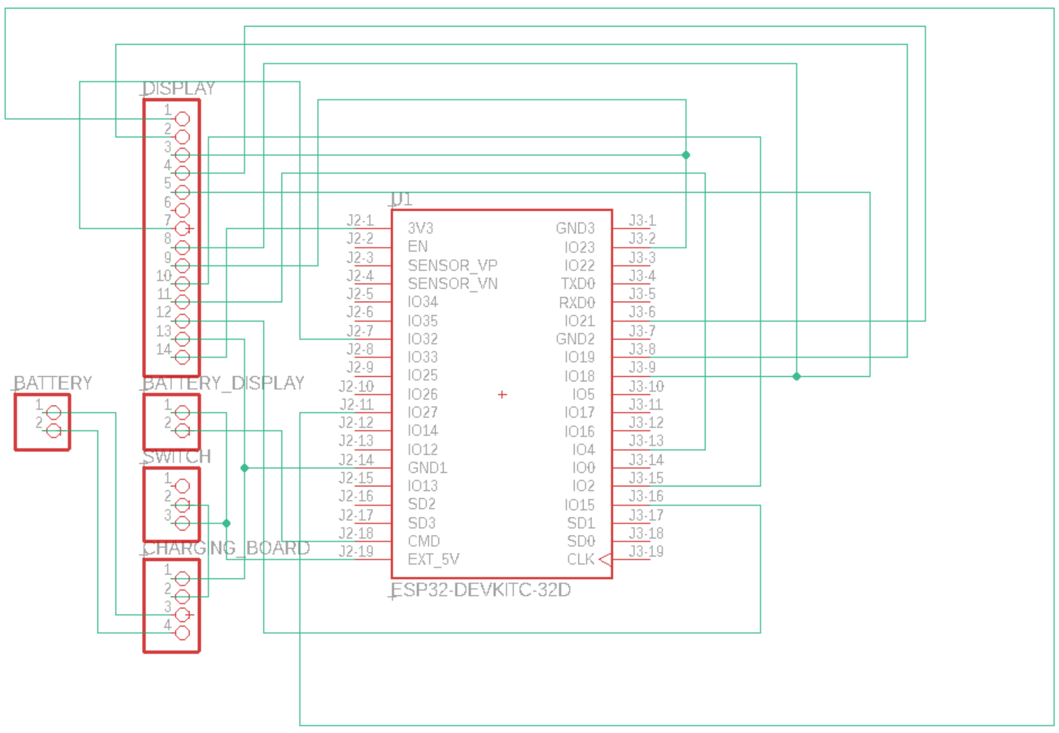
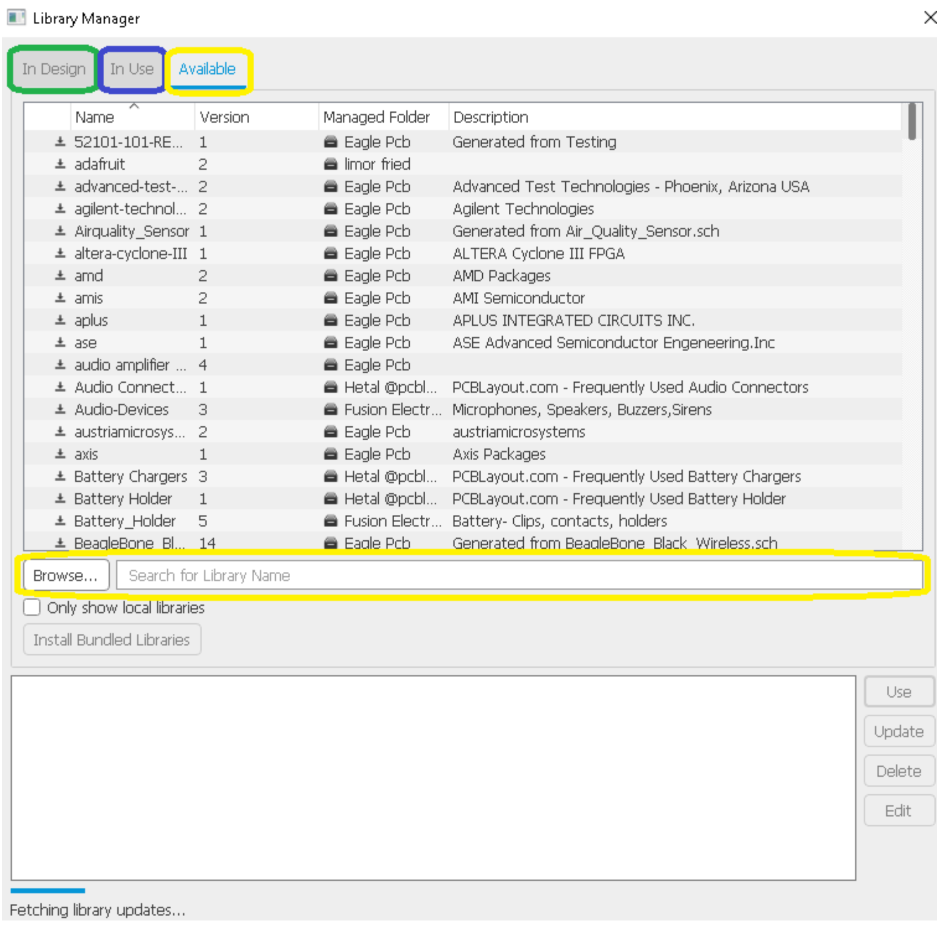


Figure 3.23: Schematic Diagram.



Name	Version	Managed Folder	Description
52101-101-RE...	1	Eagle Pcb	Generated from Testing
adafruit	2	limor fried	
advanced-test-...	2	Eagle Pcb	Advanced Test Technologies - Phoenix, Arizona USA
agilent-technol...	2	Eagle Pcb	Agilent Technologies
Airquality_Sensor	1	Eagle Pcb	Generated from Air_Quality_Sensor.sch
altera-cyclone-III	1	Eagle Pcb	ALTERA Cyclone III FPGA
amd	2	Eagle Pcb	AMD Packages
amis	2	Eagle Pcb	AMI Semiconductor
aplus	1	Eagle Pcb	APLUS INTEGRATED CIRCUITS INC.
ase	1	Eagle Pcb	ASE Advanced Semiconductor Engeneering.Inc
audio amplifier ...	4	Eagle Pcb	
Audio Connect...	1	Hetal @pcbl...	PCBLayout.com - Frequently Used Audio Connectors
Audio-Devices	3	Fusion Electr...	Microphones, Speakers, Buzzers,Sirens
austriamicrosys...	2	Eagle Pcb	austriamicrosystems
axis	1	Eagle Pcb	Axis Packages
Battery Chargers	3	Hetal @pcbl...	PCBLayout.com - Frequently Used Battery Chargers
Battery Holder	1	Hetal @pcbl...	PCBLayout.com - Frequently Used Battery Holder
Battery_Holder	5	Fusion Electr...	Battery- Clips, contacts, holders
BeagleBone_Bl...	14	Eagle Pcb	Generated from BeagleBone_Black_Wireless.sch

Fetching library updates...

Figure 3.24: Library Manager Window.

3.6.2 Autodesk Eagle Setup for Schematic File for Board file

The process of setting up a board file is shown in Figure 3.25. "Design rules..." is selected after clicking the "Edit" button. After that, a window will be revealed. The values of the design criteria for printing boards are shown in Figures 3.26 through 3.29, namely clearance, distance, sizes, annular ring, and the rest are left the same or unaltered.

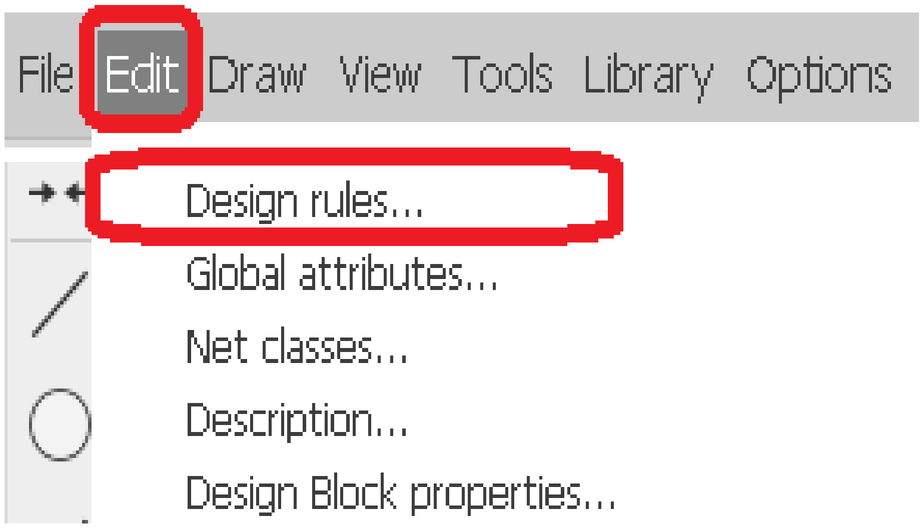
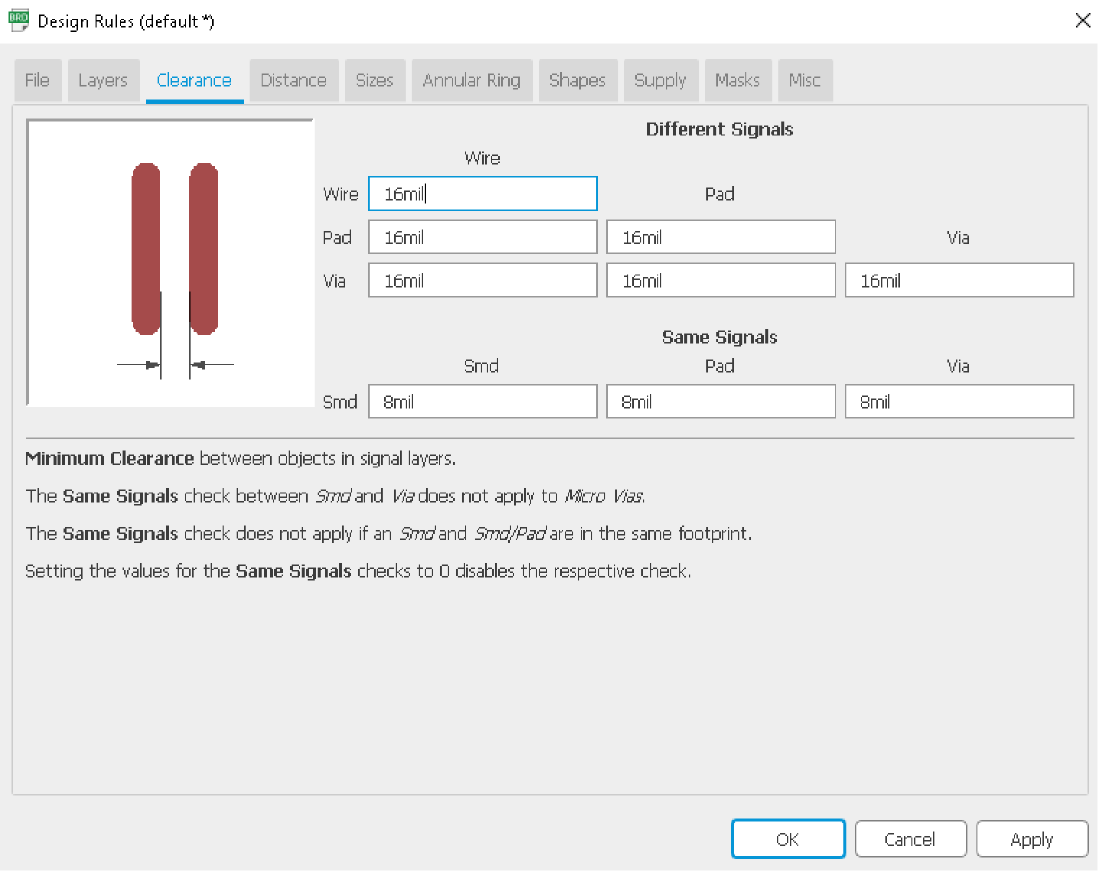


Figure 3.25: Setup for Board file.



Different Signals

	Wire	Pad	Via
Wire	16mil		
Pad	16mil	16mil	
Via	16mil	16mil	16mil

Same Signals

	Smd	Pad	Via
Smd	8mil	8mil	8mil

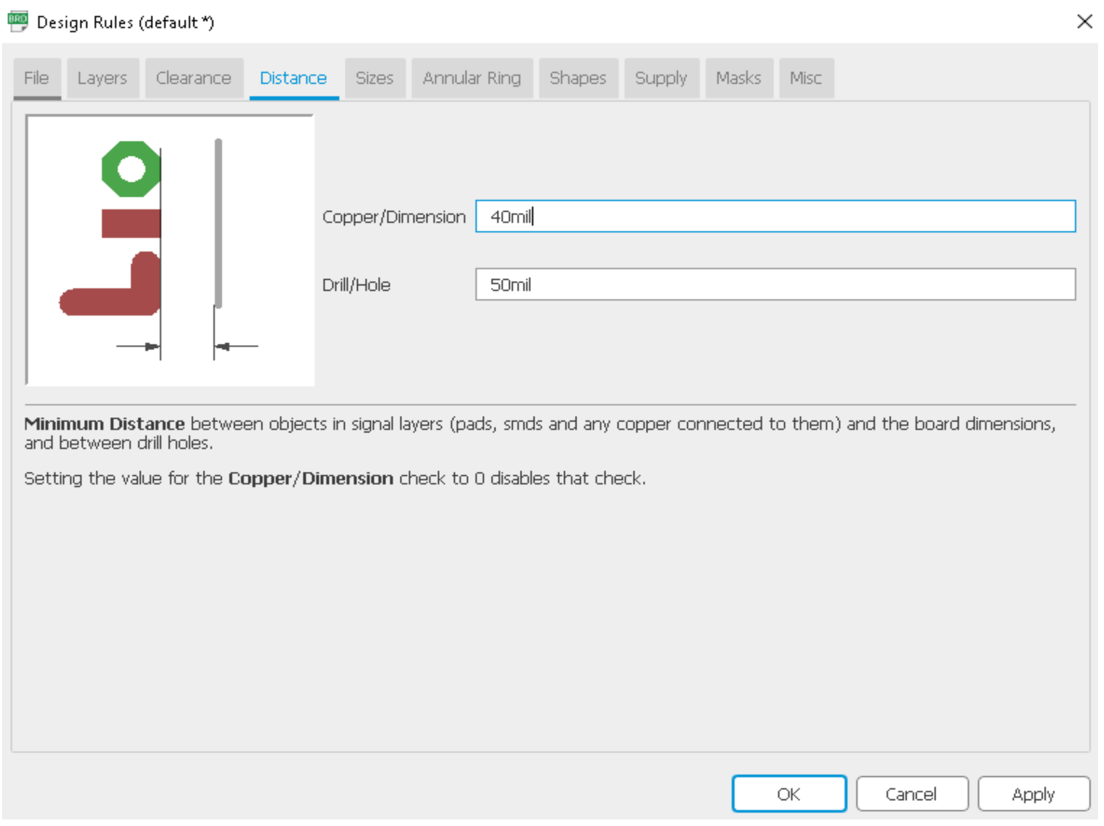
Minimum Clearance between objects in signal layers.

The **Same Signals** check between *Smd* and *Via* does not apply to *Micro Vias*.

The **Same Signals** check does not apply if an *Smd* and *Smd/Pad* are in the same footprint.

Setting the values for the **Same Signals** checks to 0 disables the respective check.

Figure 3.26: Settings for Clearance Tab



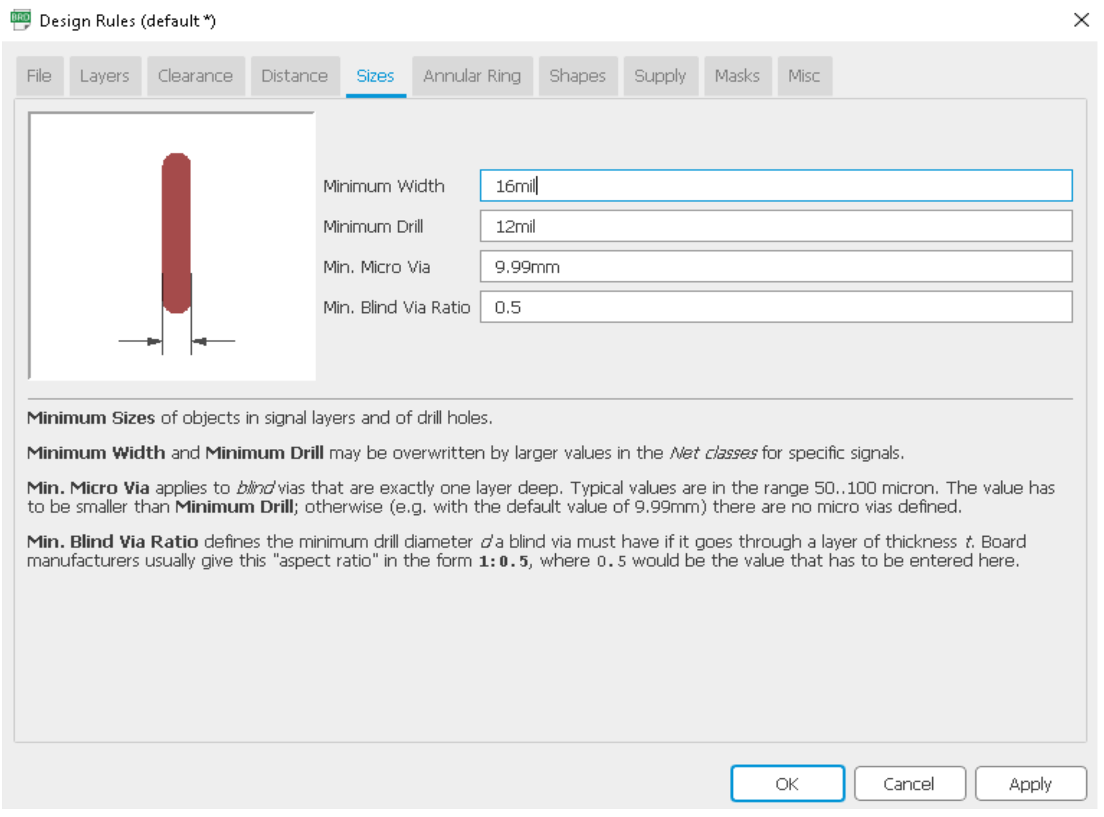
Copper/Dimension 40mil

Drill/Hole 50mil

Minimum Distance between objects in signal layers (pads, smds and any copper connected to them) and the board dimensions, and between drill holes.

Setting the value for the **Copper/Dimension** check to 0 disables that check.

Figure 3.27: Settings for Distance.



Minimum Width 16mil

Minimum Drill 12mil

Min. Micro Via 9.99mm

Min. Blind Via Ratio 0.5

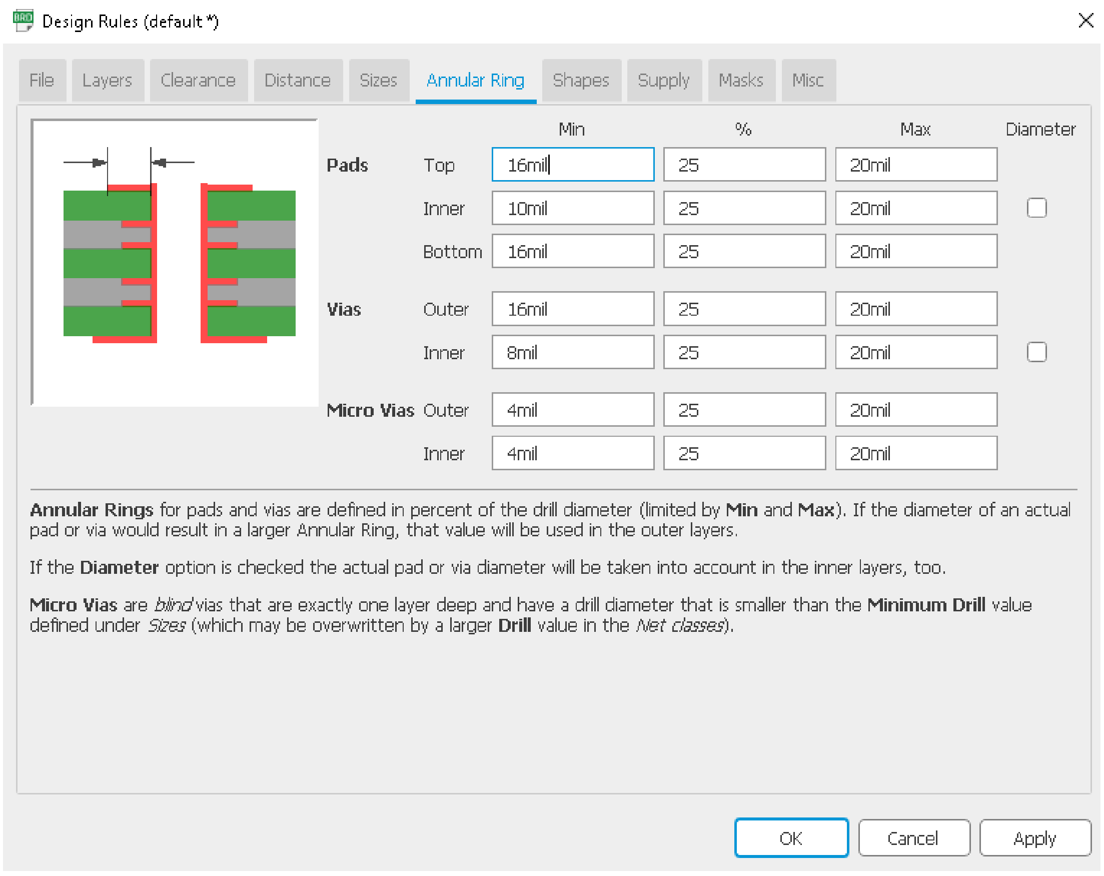
Minimum Sizes of objects in signal layers and of drill holes.

Minimum Width and **Minimum Drill** may be overwritten by larger values in the *Net classes* for specific signals.

Min. Micro Via applies to *blind* vias that are exactly one layer deep. Typical values are in the range 50..100 micron. The value has to be smaller than **Minimum Drill**; otherwise (e.g. with the default value of 9.99mm) there are no micro vias defined.

Min. Blind Via Ratio defines the minimum drill diameter *d* a blind via must have if it goes through a layer of thickness *t*. Board manufacturers usually give this "aspect ratio" in the form **1:0.5**, where 0.5 would be the value that has to be entered here.

Figure 3.28: Settings for Sizes.



		Min	%	Max	Diameter
Pads	Top	16mil	25	20mil	
	Inner	10mil	25	20mil	☐
	Bottom	16mil	25	20mil	
Vias	Outer	16mil	25	20mil	
	Inner	8mil	25	20mil	☐
Micro Vias	Outer	4mil	25	20mil	
	Inner	4mil	25	20mil	

Annular Rings for pads and vias are defined in percent of the drill diameter (limited by **Min** and **Max**). If the diameter of an actual pad or via would result in a larger Annular Ring, that value will be used in the outer layers.

If the **Diameter** option is checked the actual pad or via diameter will be taken into account in the inner layers, too.

Micro Vias are *blind* vias that are exactly one layer deep and have a drill diameter that is smaller than the **Minimum Drill** value defined under *Sizes* (which may be overwritten by a larger **Drill** value in the *Net classes*).

Figure 3.29: Settings for Annular Ring.

The final board layout, shown in Figure 3.30, is successfully created with a single main microcontroller, the ESP32-Wroom-32D, and a main display, an ILI9488 TFT screen with SPI MSP 3520 module. The board layout is developed by Eagle software 9.6.2. The yellow line from the SPI MSP 3520 module ILI9488 TFT screen with touch screen mode to the microcontroller ESP32-Wroom-32D is not connected, as illustrated in Figure 3.30, because the copper's diameter is greater and cannot fit through the pin. Finally, the issue is resolved by altering the copper's diameter and how the pins flow through them, as seen in Figure 3.31.

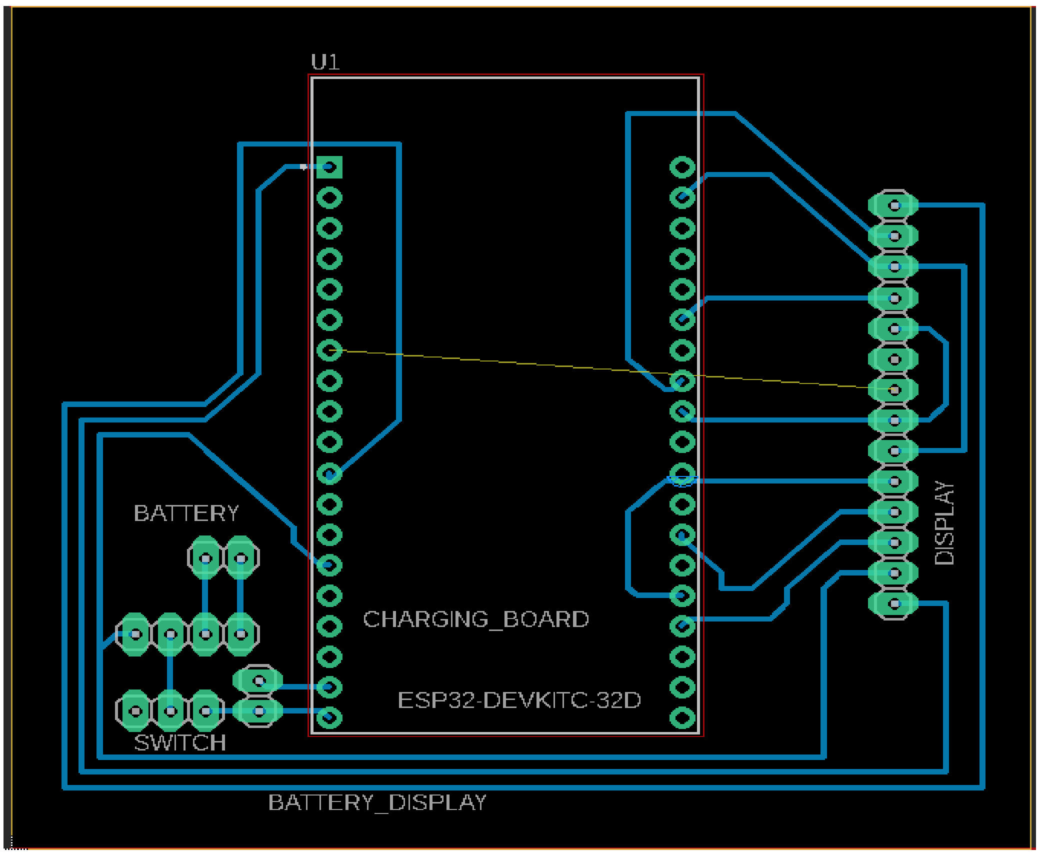


Figure 3.30: Board Diagram.

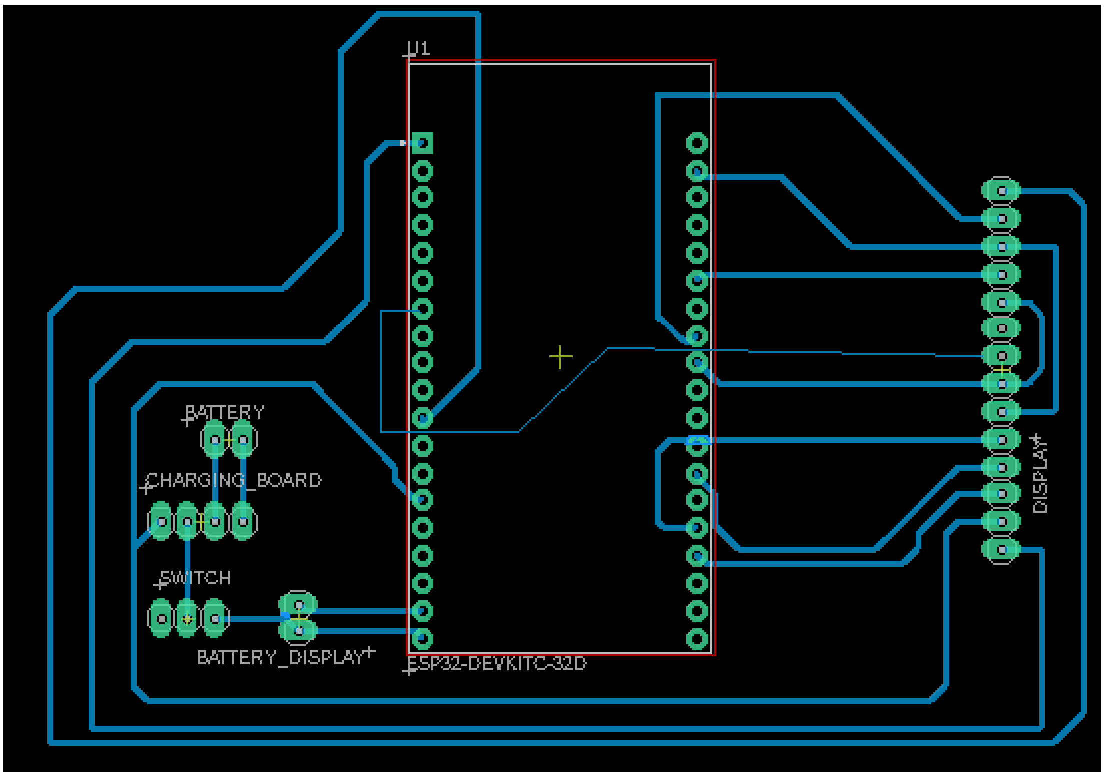


Figure 3.31: Pass through 2 Pins.

3.7 3D Printing

As seen in Figure 3.32, a casing for the project had been created using Tinker CAD and the link "https://www.tinkercad.com". The front casing (the blue form) in Figure 3.33 measures 120mm, 96mm, and 90mm in length, width, and height. The inner bottom casing measures 112mm in length, 82mm in width, and 4mm in height. The outer bottom casing measures 120mm in length, 96mm in width, and 5mm in height. The.stl file had been downloaded and delivered to the 3D printer for printing.

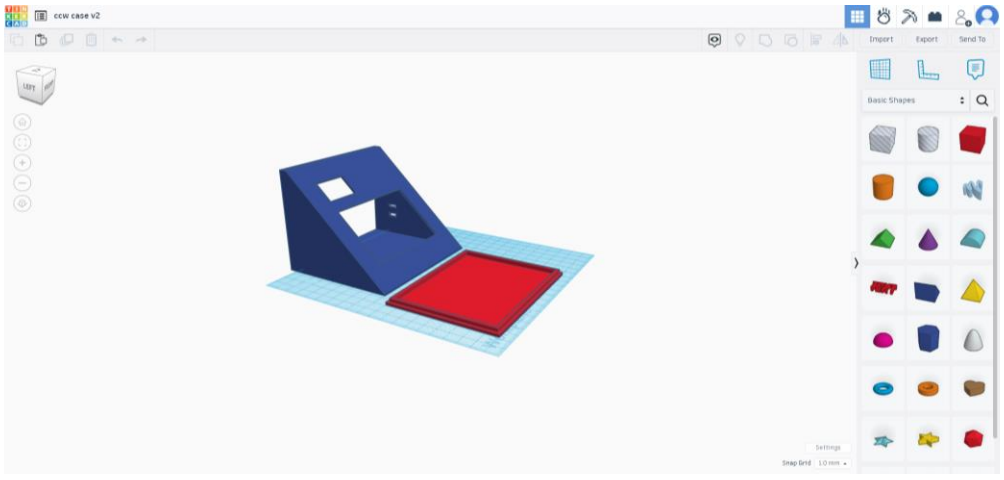


Figure 3.32: Designed Case in Tinker CAD.

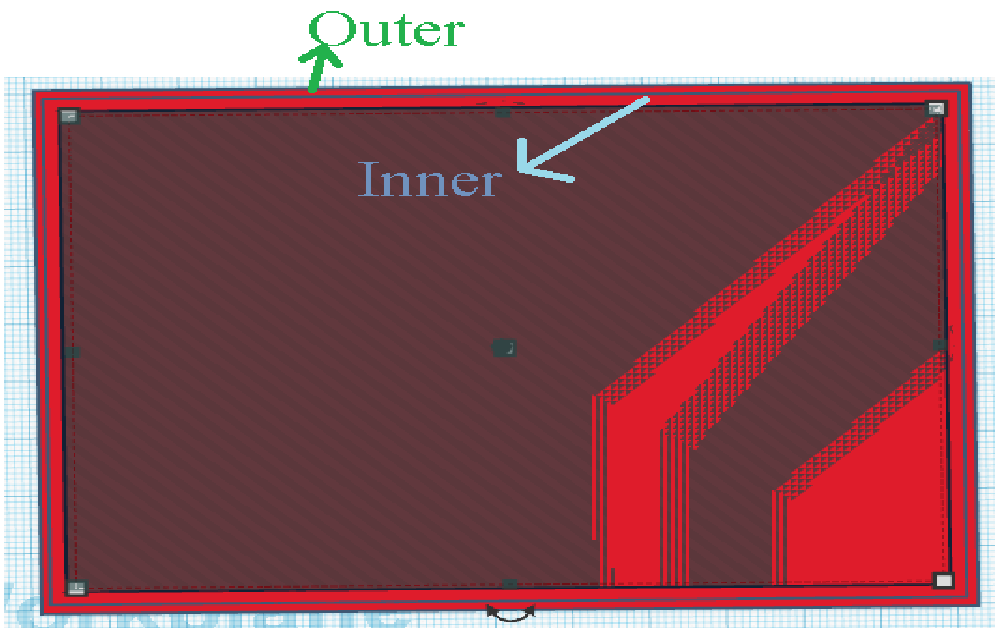
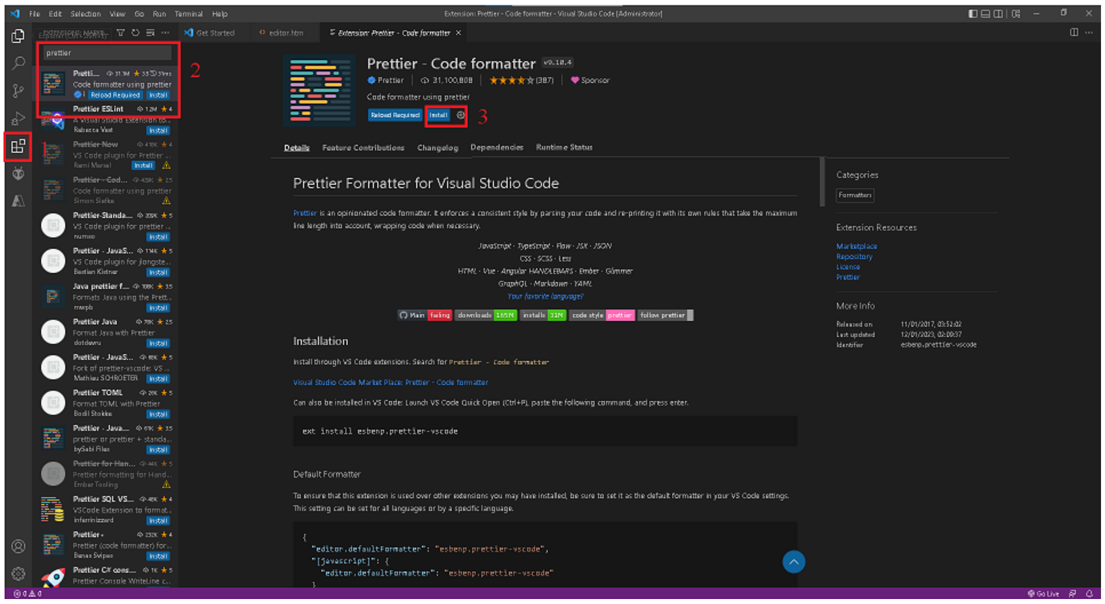


Figure 3.33: Dimension of the Inner and Outer Bottom Case.

3.8 Webpage Interface

3.8.1 Microsoft Visual Code

In Microsoft Visual Code, HTML programming language is used. This Microsoft Visual Code is a combination of the powerful developer tooling with a source code editor. Hypertext Markup Language (HTML) is the standard markup language for generating web applications and web pages. Microsoft Visual Code (VS Code) supports the widely used code formatter Prettier, which automatically formats code in accordance with a pre-set set of criteria. It supports a wide range of programming languages, including HTML, CSS, JavaScript, and TypeScript. Once installed, Prettier may be configured to format code automatically on save or manually using a keyboard shortcut or menu option. It can be installed as a Code plugin and showed as Figure 3.34. Furthermore, LiveServer plugin also need to be installed into the Microsoft Visual Code. In Figure 3.35, the HTML code on Microsoft Visual Code is shown.


File Edit Selection View Go Run Terminal Help
Extension: Prettier - Code formatter - Visual Studio Code [Administrator]
Get Started
editor.htm
Extension: Prettier - Code formatter
prettier
1
2
3
Prettier - Code formatter v9.10.4
Prettier | 31,100,808 | (387) | Sponsor
Code formatter using prettier
Reload Required
Install
Details
Feature Contributions
Changelog
Dependencies
Runtime Status
Categories
Formatters
Extension Resources
Marketplace
Repository
License
Prettier
More Info

Prettier Formatter for Visual Studio Code

Prettier is an opinionated code formatter. It enforces a consistent style by parsing your code and re-printing it with its own rules that take the maximum line length into account, wrapping code when necessary.

JavaScript · TypeScript · Flow · JSX · JSON
CSS · SCSS · Less
HTML · Vue · Angular HANDLEBARS · Ember · Glimmer
GraphQL · Markdown · YAML
Your favorite language?

Main failing | downloads 165M | installs 31M | code style prettier | follow prettier

Installation

Install through VS Code extensions. Search for `Prettier - Code formatter`

Visual Studio Code Market Place: Prettier - Code formatter

Can also be installed in VS Code: Launch VS Code Quick Open (Ctrl+P), paste the following command, and press enter.

```
ext install esbenp.prettier-vscode
```

Default Formatter

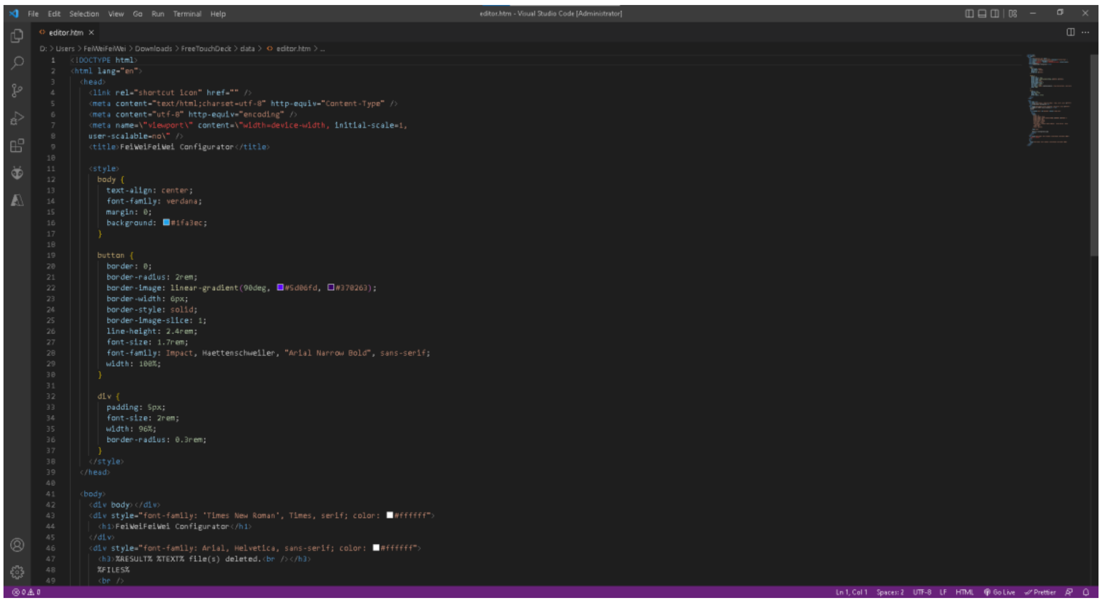
To ensure that this extension is used over other extensions you may have installed, be sure to set it as the default formatter in your VS Code settings. This setting can be set for all languages or by a specific language.

```
{
  "editor.defaultFormatter": "esbenp.prettier-vscode",
  "[javascript]": {
    "editor.defaultFormatter": "esbenp.prettier-vscode"
  }
```

Released on	11/01/2017, 03:52:02
Last updated	12/01/2023, 02:09:37
Identifier	esbenp.prettier-vscode

Figure 3.34: Installing Extension.

In Figure 3.34, the Prettier plugin is installed to the Microsoft Visual Code following the steps of red box. Step1, the icon “Extension” at the first red box is clicked. Step2, “Prettier” is type into the searching box and then the plugin is clicked. Step3, “Install” button is pressed. Then, the plugin is installed successfully. In order to install LiveServer plugin, all the steps is remain unchanged but just change the “Prettier” to “LiveServer” at the step2.


File Edit Selection View Go Run Terminal Help
editor.htm - Visual Studio Code [Administrator]
editor.htm
D: > Users > FeiWeiFeiWei > Downloads > FreeTouchDeck > data > editor.htm > ...

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <link rel="shortcut icon" href="" />
    <meta content="text/html;charset=utf-8" http-equiv="Content-Type" />
    <meta content="utf-8" http-equiv="encoding" />
    <meta name=\"viewport\" content=\"width=device-width, initial-scale=1,
    user-scalable=no\" />
    <title>FeiWeiFeiWei Configurator</title>

    <style>
      body {
        text-align: center;
        font-family: verdana;
        margin: 0;
        background: #1fa3ec;
      }

      button {
        border: 0;
        border-radius: 2rem;
        border-image: linear-gradient(90deg, #5d06fd, #370263);
        border-width: 6px;
        border-style: solid;
        border-image-slice: 1;
        line-height: 2.4rem;
        font-size: 1.7rem;
        font-family: Impact, Haettenschweiler, "Arial Narrow Bold", sans-serif;
        width: 100%;
      }

      div {
        padding: 5px;
        font-size: 2rem;
        width: 96%;
        border-radius: 0.3rem;
      }
    </style>
  </head>

  <body>
    <div body></div>
    <div style="font-family: 'Times New Roman', Times, serif; color: #ffffff">
      <h1>FeiWeiFeiWei Configurator</h1>
    </div>
    <div style="font-family: Arial, Helvetica, sans-serif; color: #ffffff">
      <h3>%RESULT% %TEXT% file(s) deleted.<br /></h3>
      %FILES%
      <br />
```

Figure 3.35: HTML Code on Microsoft Visual Code.

Figure 3.36: Visual Studio Code.

3.8.2 Arduino JSON

A well-known C++ library called Arduino JSON makes it simple to parse and generate JSON (JavaScript Object Notation) data on ESP32 boards. JSON is a simple data format that is frequently used for data transmission between client-side web applications and online services. In IoT (Internet of Things) devices, it is also utilised to transfer data online. Users can read and write JSON data from and to ESP32 boards using the capabilities offered by the Arduino JSON library. It is appropriate for usage on resource-constrained gadgets like ESP32 boards since it is simple to use and doesn't require any additional dependencies. Users must define a "DynamicJsonDocument" object, which is used to store the JSON data, and include the library header file in order to utilise the Arduino JSON library. Users can simply explore JSON structures, conduct operations like merging and filtering data, and easily serialise and deserialize data due to the library's straightforward and user-friendly API for working with JSON data. It supports a range of JSON data formats, including data that is encoded in UTF-8, UTF-16, and UTF-32. The library offers tools for adding to, changing, and extracting data from JSON documents.

3.9 Equipment and Cost Analysis

Table 3.7 shows the components list with price. The total cost for this project is RM 107.38.

Table 3.7: Components List with Price.

No.	Components	Unit Price (RM)	Unit	Total Price (RM)	Remarks
1.	ESP32-Wroom-32D Development Board	16.00	1	16.00	Aliexpress
2.	Jumper Wire	2.50	2	5.00	Cytron
3.	3.5-inch TFT Screen	38.00	1	38.00	Aliexpress
4.	Filament for Casing	RM48/KG	200g	9.60	Shopee
5.	Printed Circuit Board	10.00	1	10.00	Aliexpress
6.	Slicer	0.20	10	1.00	Shopee
7.	Lithium Battery Charging Protection Board	3.88	1	3.88	Shopee
8.	8S Lithium Battery Capacity Indicator	3.30	1	3.30	Shopee
9.	3.7V LiPo Battery 2000mAh	28.00	1	28.00	Shopee
10.	Bread Board	3.90	2	7.80	Cytron
11.	Other Fees			10.00	Soldering, Hot Glue, Electric, Shipping fees
			Total	107.38	

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Schematic Diagram of a Stream Deck

The stream deck consists of one main microcontroller, which are ESP32-Wroom-32D. A schematic diagram of the stream deck is generated using the Eagle software 9.6.2 version as shown in Figure 4.1.

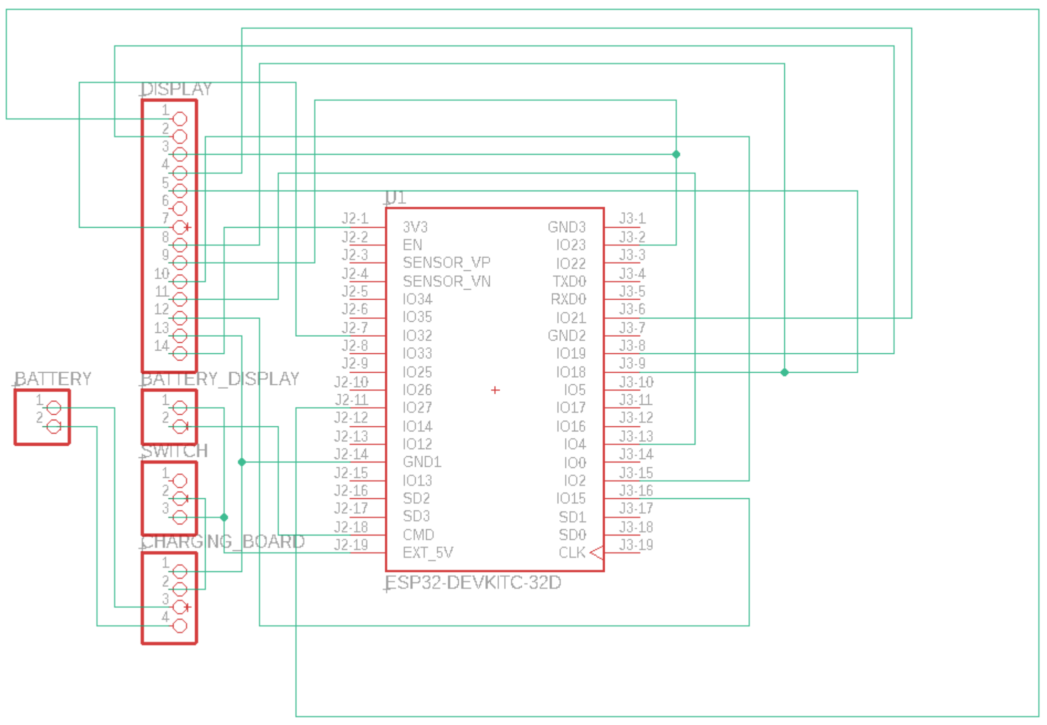


Figure 4.1: Schematic of Stream Deck.

4.2 Overview of the framework of the Stream Deck

The SPI MSP 3520 module ILI 9488 TFT screen, ESP32-Wroom-32D microcontroller, battery capacity indicator, slicer, type-C charging protection board, and 2000mAh LiPo battery are the electronic components used to construct the stream deck. Figure 4.2 showed the stream deck prototype before the project is finished. Figure 4.3 depicted the stream deck's overall project. Figure 4.3 shows the soldering of the components into the PCB and placement of them in the precise locations that TinkerCAD had previously created. The case is created using a 3D printer and produced in a particular colour of grey. Eagle software 9.6.2 version is used to create the printed circuit board in yellow colour. Figure 4.4 depicted the stream deck's front perspective. Figure 4.5 depicts the stream deck's backside. Figure 4.6 depicts the stream deck's side perspective.

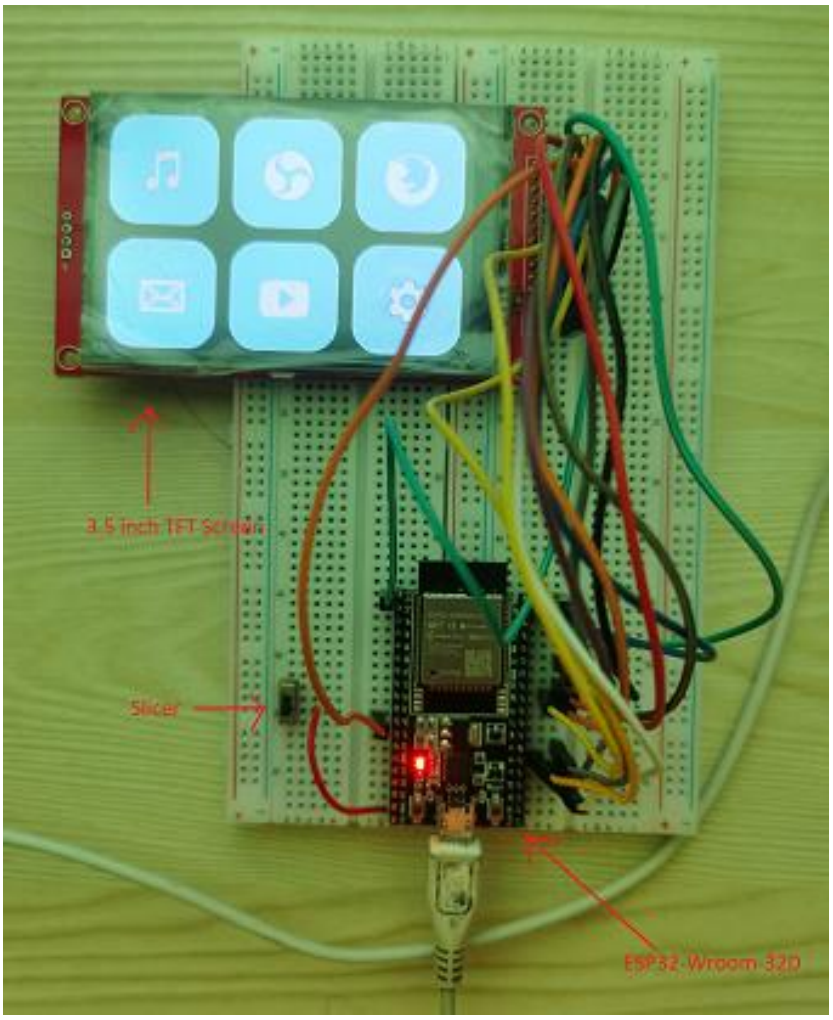


Figure 4.2: Prototype of the Stream Deck.

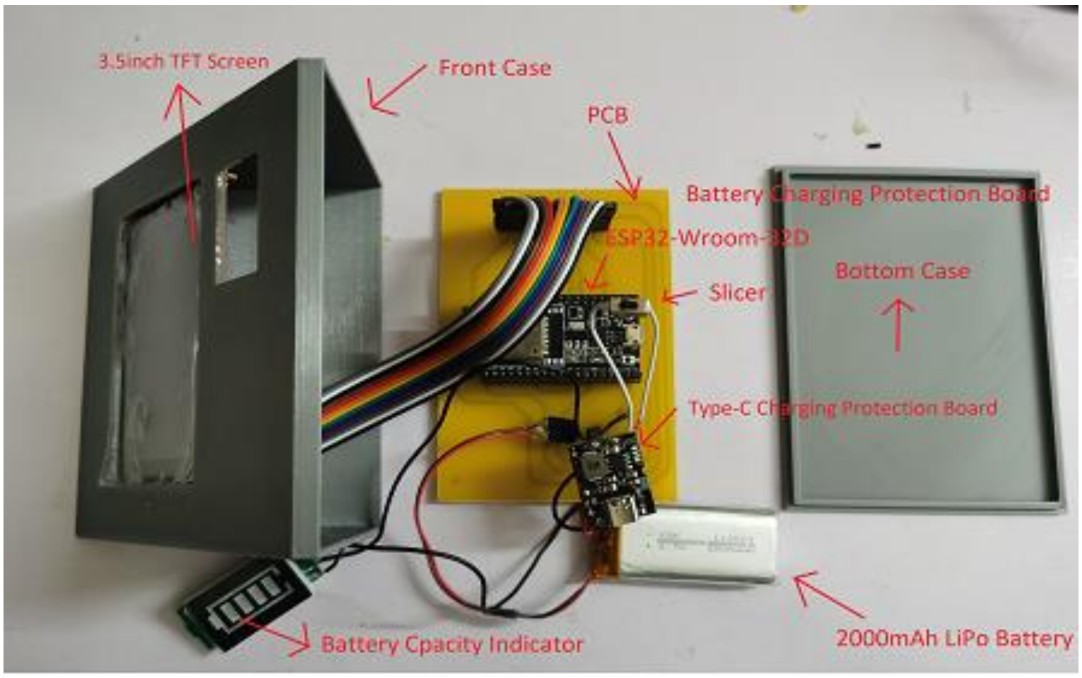

3.5inch TFT Screen
Front Case
PCB
Battery Charging Protection Board
ESP32-Wroom-32D
Slicer
Bottom Case
Type-C Charging Protection Board
Battery Cpacity Indicator
2000mAh LiPo Battery

Figure 4.3: Overview Project of the Stream Deck.

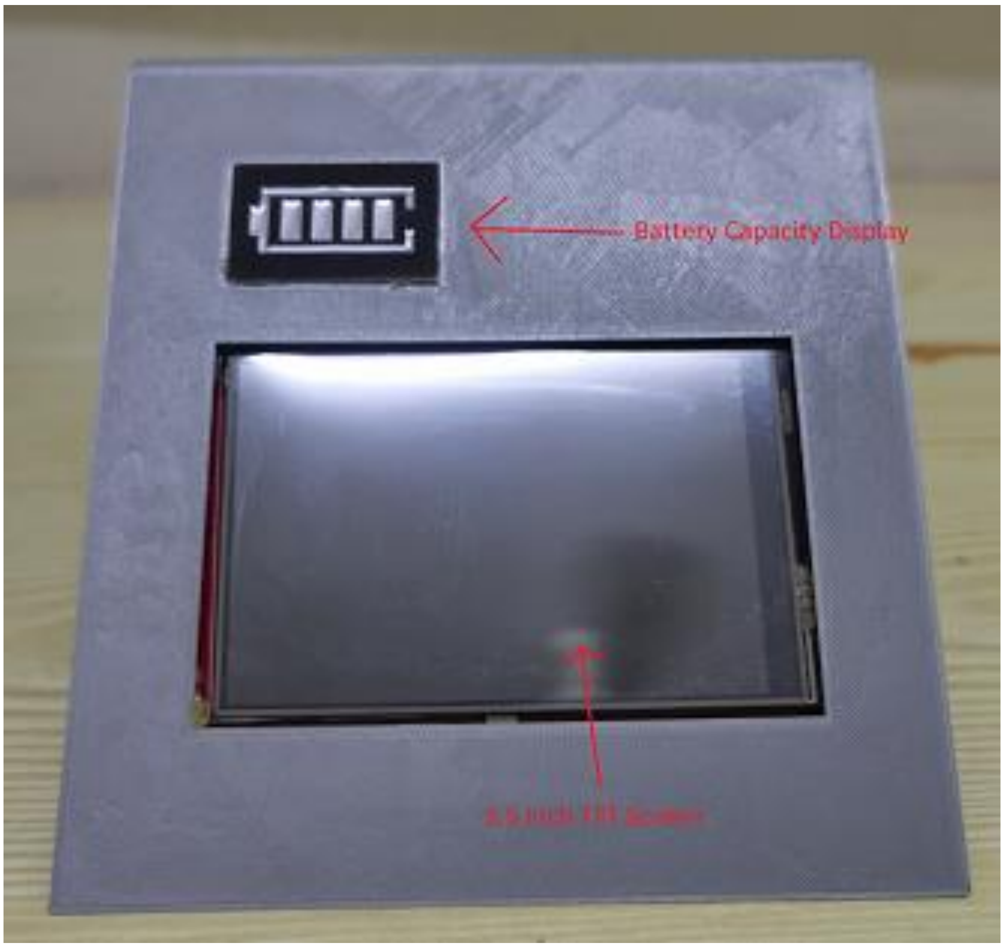


Figure 4.4: Front View of the Stream Deck.

Figure 4.5: Back View of the Stream Deck.

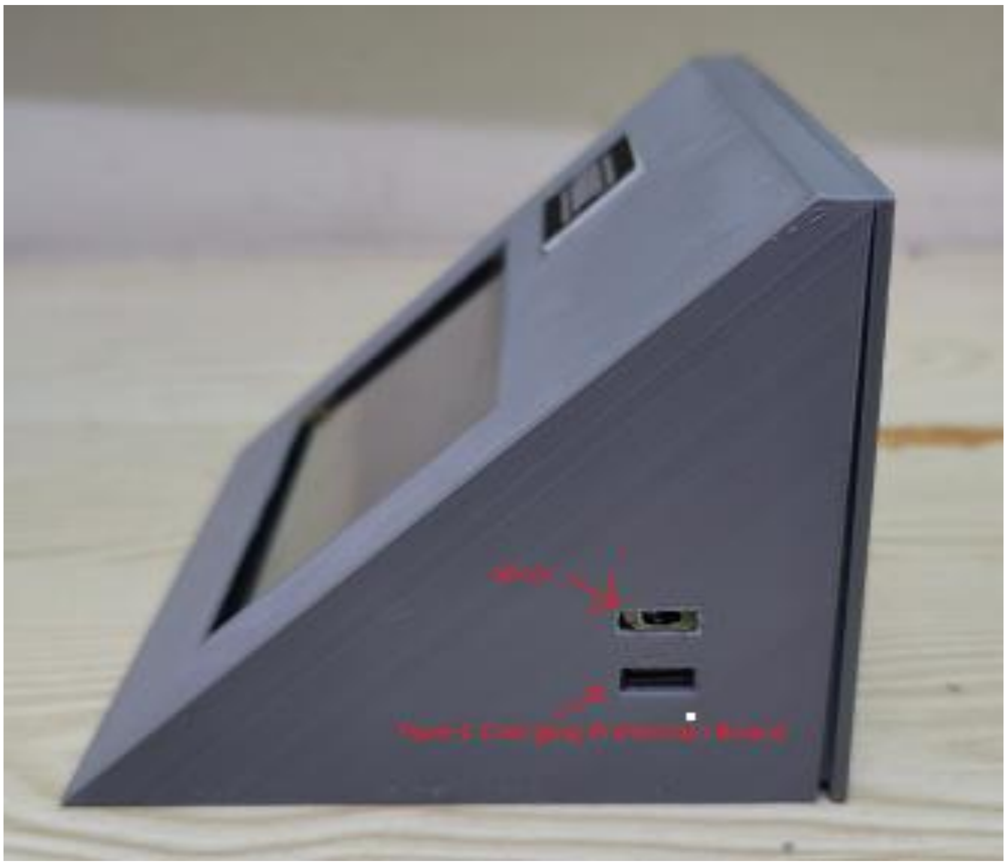


Figure 4.6: Side View of the Stream Deck.

4.3 Default User Interface of Stream Deck

In this project, a single TFT screen and a single battery capacity indication are utilized. According to Figure 4.7, the TFT screen is positioned at the bottom of the front casing, and the battery capacity indication is positioned at the top. The TFT screen displays six default buttons with blue colour and black background on the main screen along with six distinct logos that use.bmp format icons since the configuration default. The volume control part is for the first button. From the second button to the fifth button, there is only the default logo and no current indication of a function. The stream deck setting itself is included in the last button. Additionally, as illustrated in Figure 4.7, the battery capacity indicator displays the capacity of the battery block by block that is still in the battery. 25% of the battery is represented by each block.

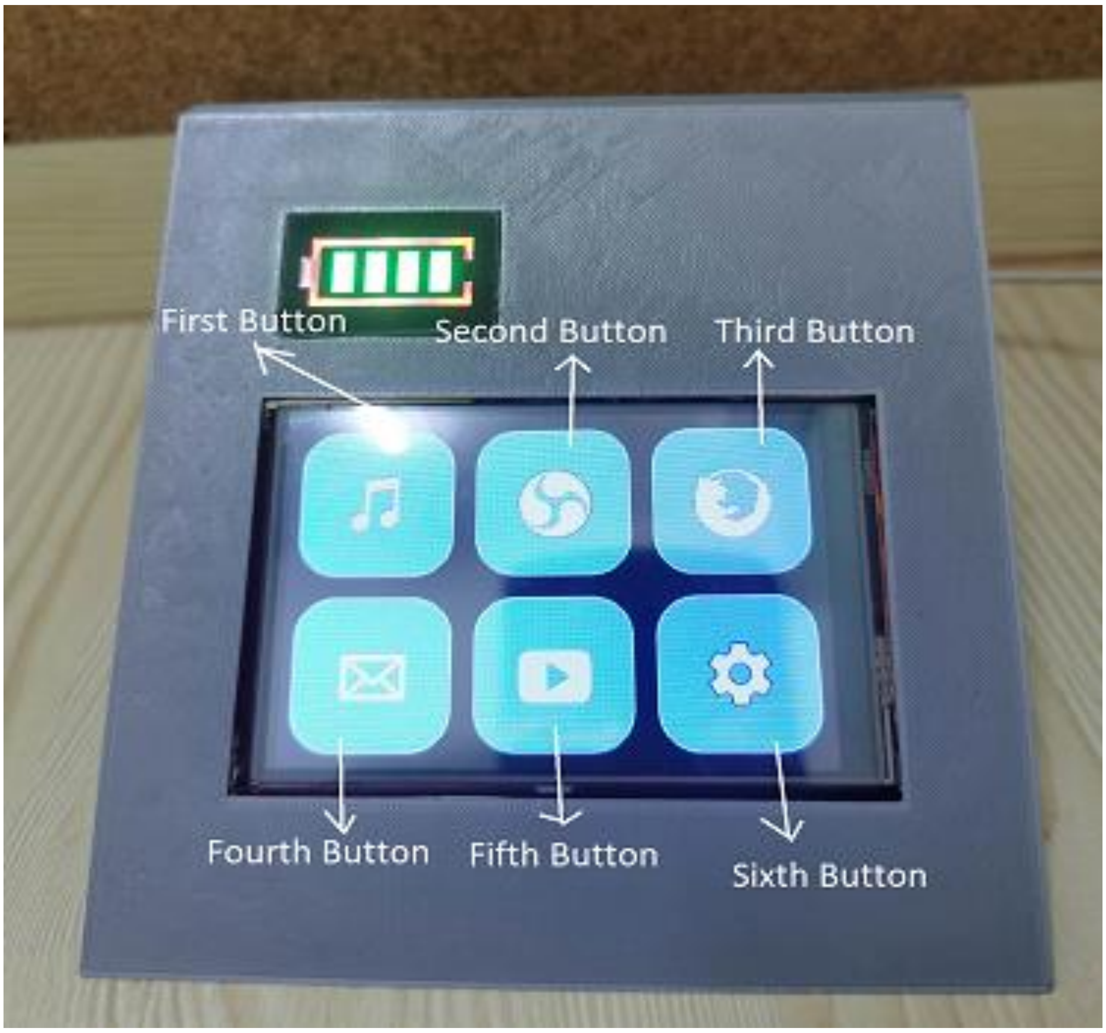


Figure 4.7: Main Menu at TFT Screen of Stream Deck.

By hitting the first button on the main menu, the stream deck is shown in Figure 4.8 with the first section of the function buttons. After pressing the first button, there are six more buttons with additional six functions. The device's mute or unmute feature is controlled by the first button. The second button is a function that controls volume down, while the third button is a function that controls volume up. The fourth button's purpose is to control the play and pause of a video. The fifth button's purpose is to halt the playback of videos. The home button on the sixth button returns to the main menu. After pressing the second, third, fourth, and fifth buttons on the main menu, respectively, there are also six buttons shown from Figure 4.9 to Figure 4.12. The six pushed buttons have no use and are denoted by a "?" icon. The sixth button also serves as the home button. After pressing the sixth button on the main menu, Figure 4.13 displays additional six function buttons. WiFi configuration is the first button. The TFT screen's brightness is able to adjust using the second and third buttons, which decrease and enhance brightness, respectively. The fourth button controls whether the stream deck is in sleep mode during the designated time. The fifth button only provides some stream deck information. The home button is the final one. Every button has a home button because, once the intended functionality is used, main menu must be returned.

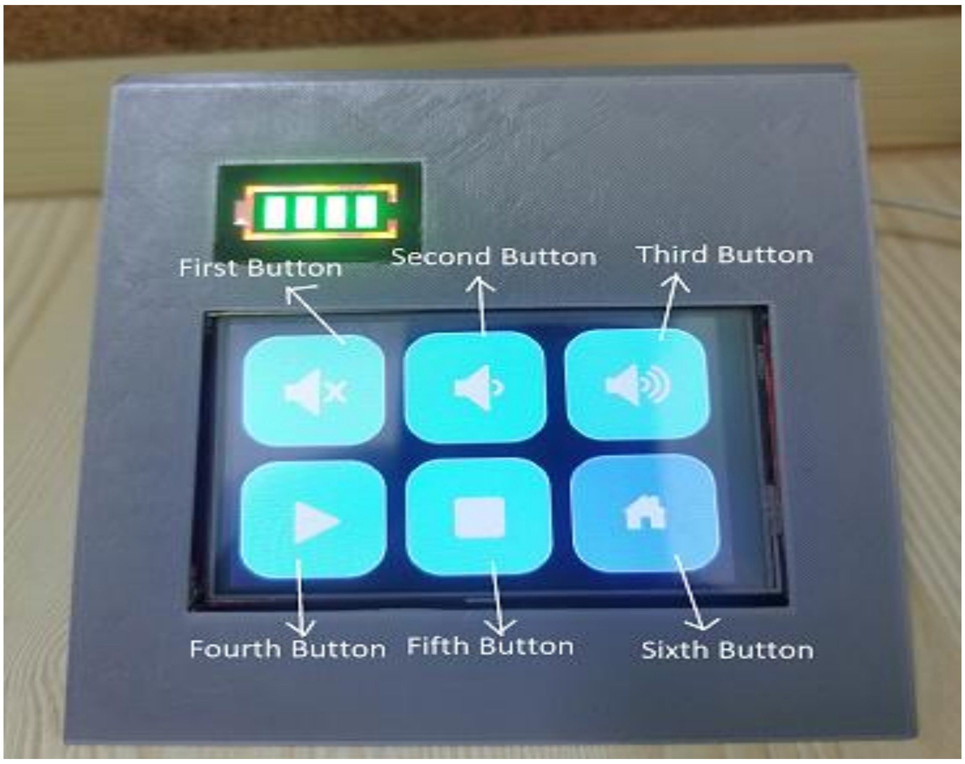


Figure 4.8: The Default Contains of The First Section.

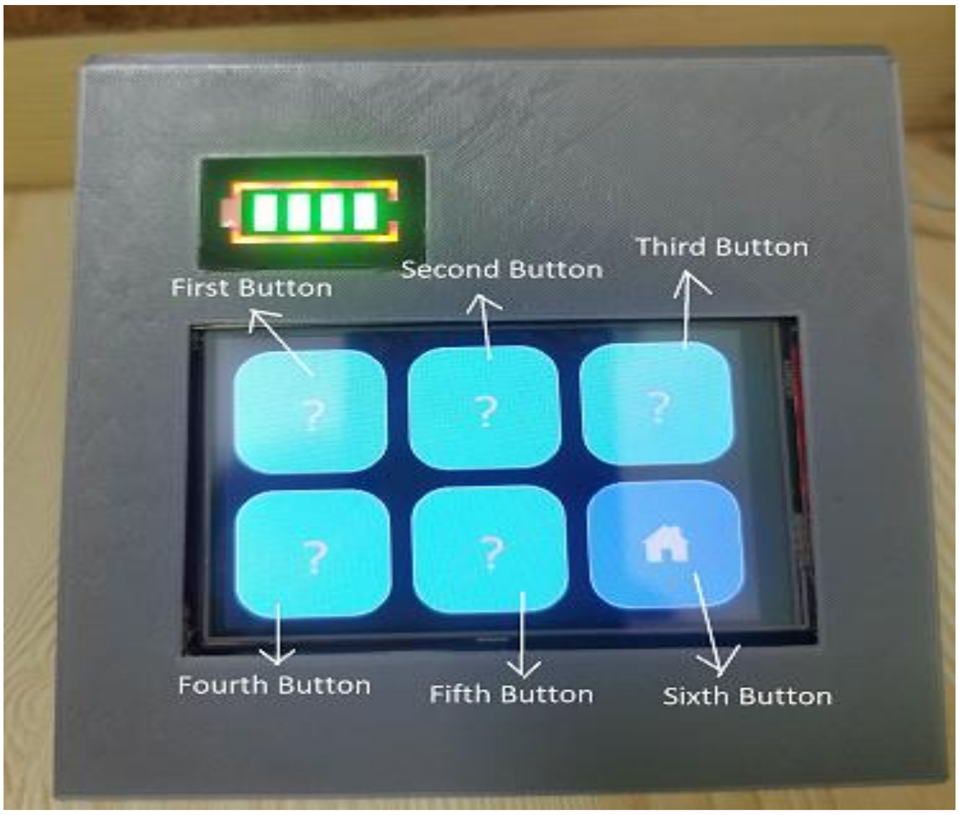


Figure 4.9: The Default Contains of The Second Section.

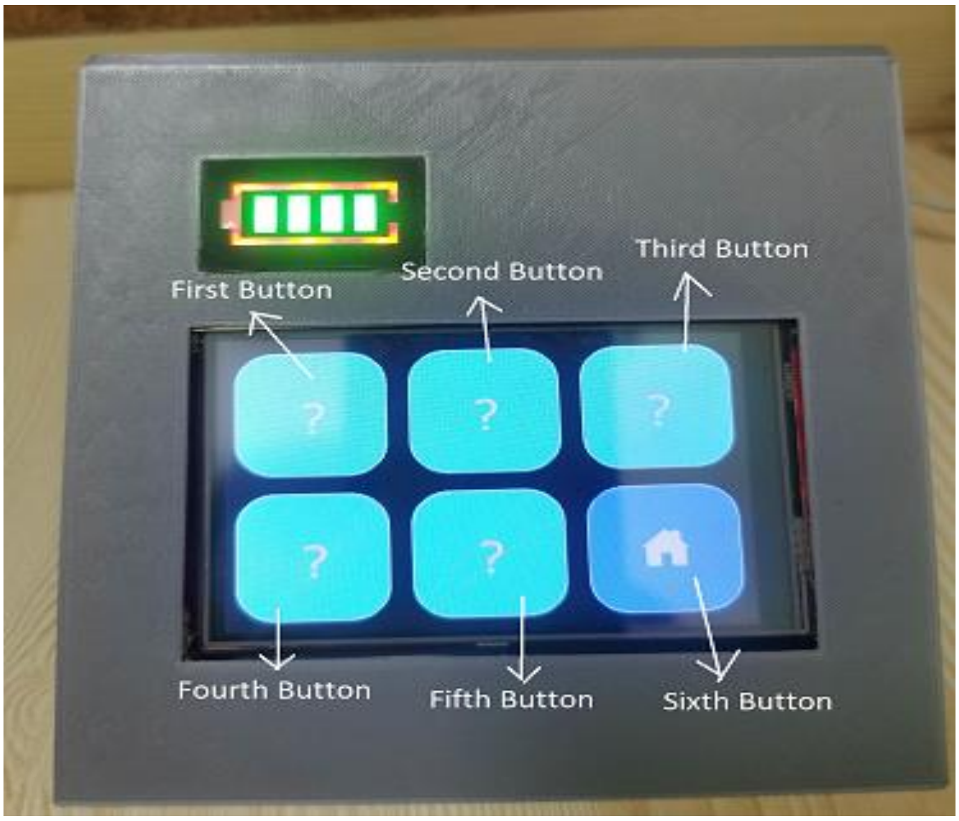


Figure 4.10: The Default Contains of The Third Section.

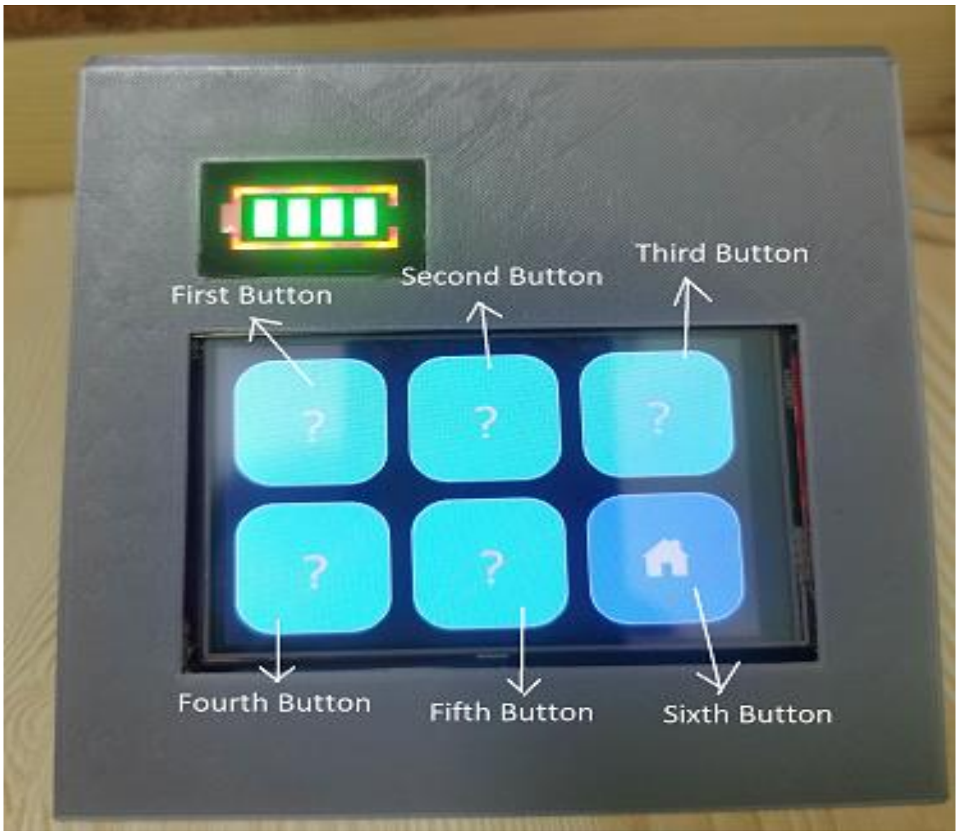


Figure 4.11: The Default Contains of The Forth Section.

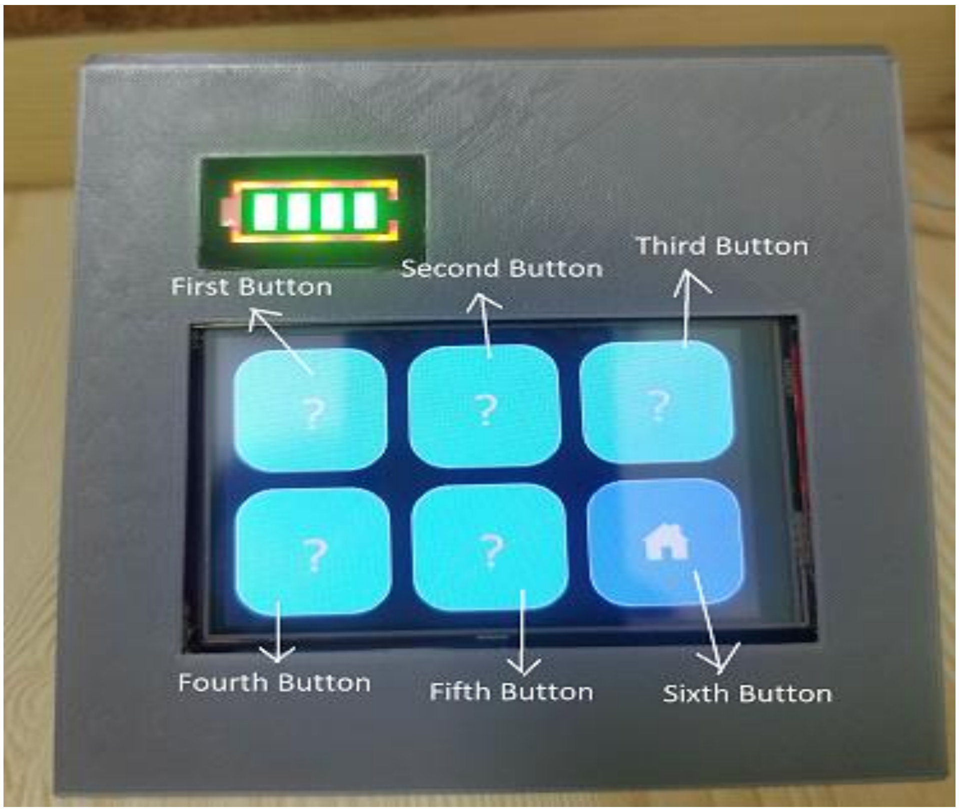


Figure 4.12: The Default Contains of The Fifth Section.

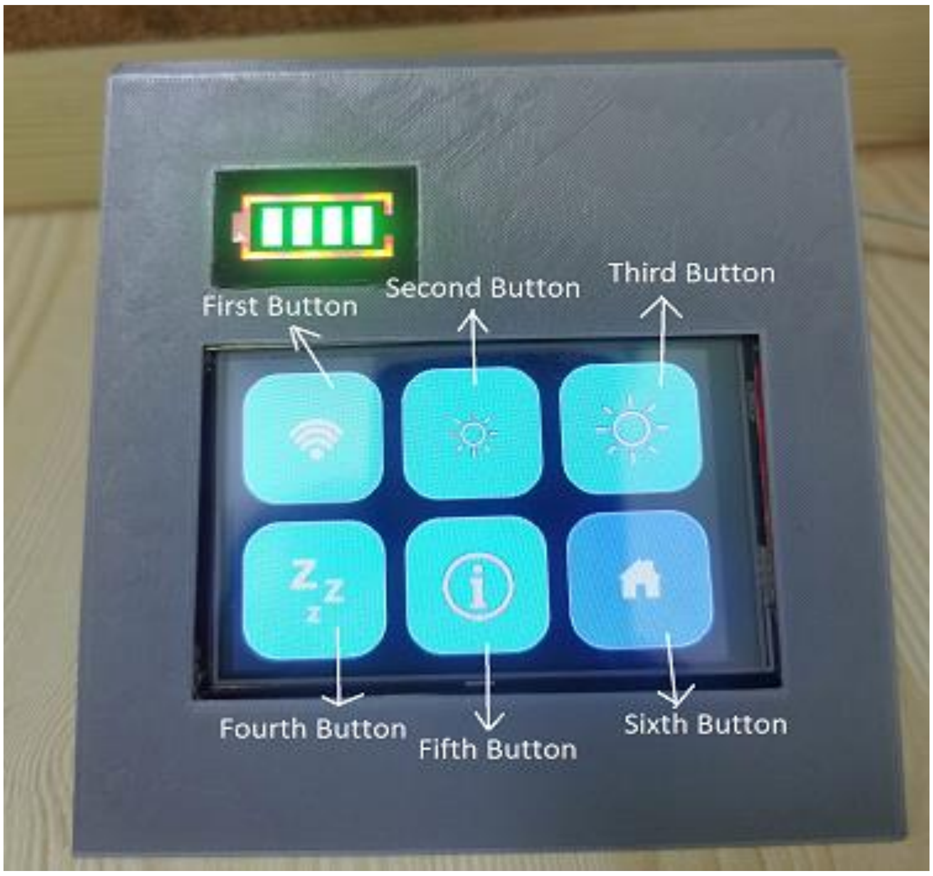


Figure 4.13: The Default Contains of the Sixth Section.

4.4 Access Point (AP) connection

A laptop is utilized in this project to establish a connection between the device and the stream deck. Pressed the sixth button located on the main menu, which bears a setting icon, followed by pressing the first button that displays a WiFi icon. Figure 4.14 depicted the stream deck's trying to establish a connection with either the station point (STA) or access point (AP), as indicated by the message "Connecting to WiFi". Figure 4.15 depicted a scenario where the stream deck is unable to establish a connection between the station point and access point. The default access point's service set identifier (SSID) and password are displayed on the screen of the stream deck. The wireless network identifier, commonly referred to as SSID, for the network is "feiweifeiwei" and the corresponding access credential is "iloveyouyv". The default IP address is also displayed as "192.168.4.1". According to Figure 4.16, the laptop is required to activate its WiFi functionality and initiate a search for the "feiweifeiwei" SSID in order to establish a connection with the access point. Thus, the password "iloveyouyv" is required to be entered into the security network box, as depicted in Figure 4.17. To verify successful connectivity, a web browser is opened and enters "192.168.4.1" into the address bar. Figure 4.18 depicts the successful connection of the webserver.

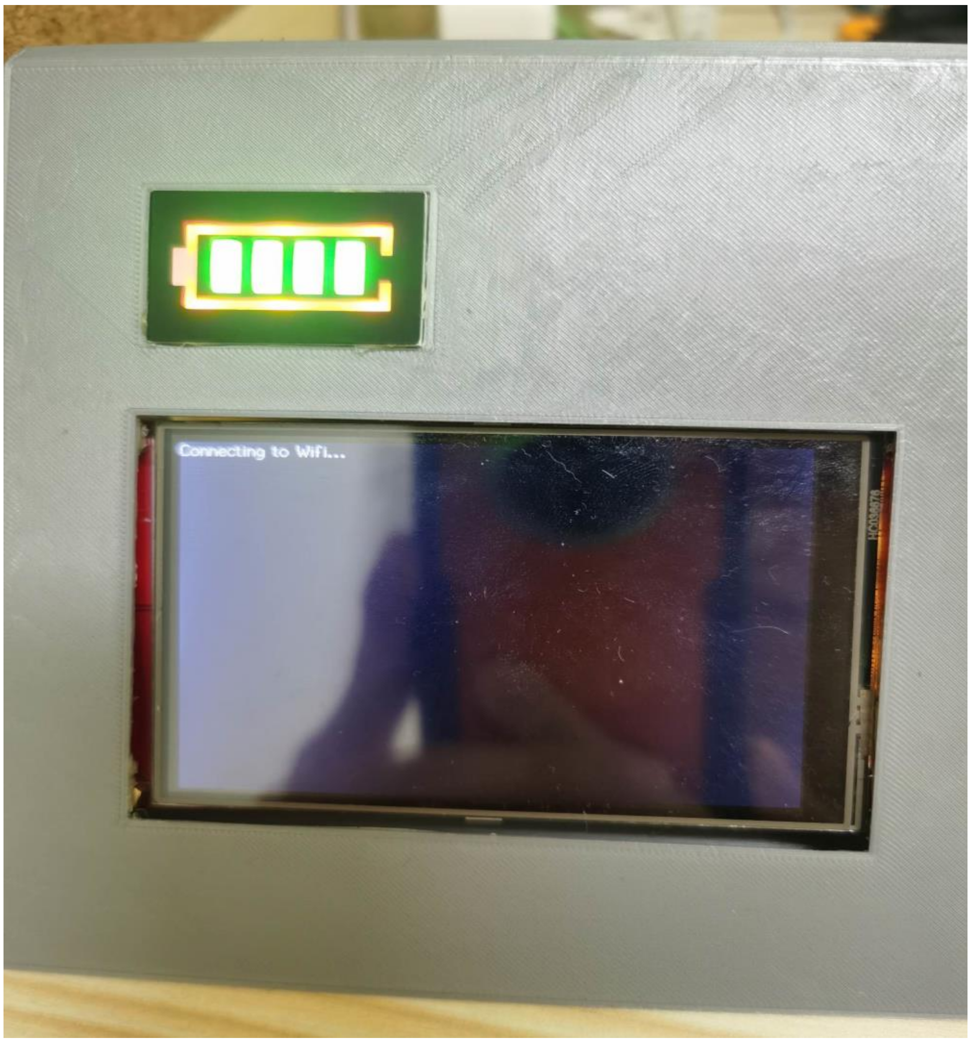
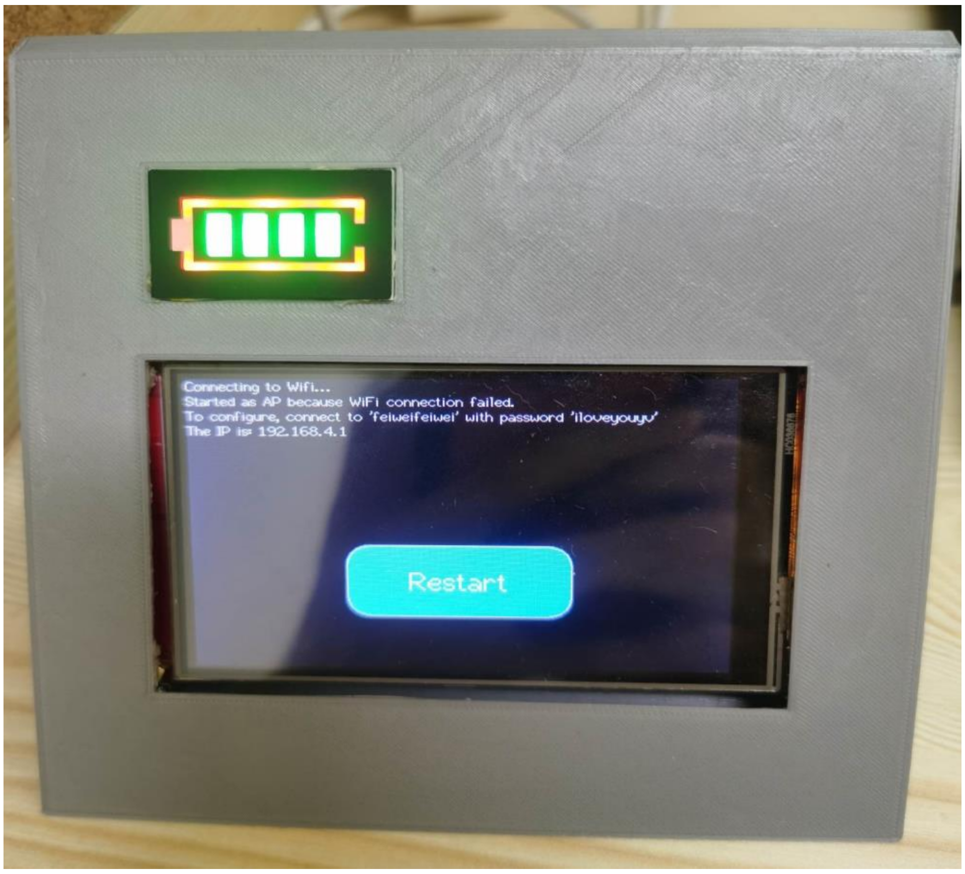


Figure 4.14: Attempting to Connect Access Point or Station Point (WiFi).



```
Connecting to Wifi...
Started as AP because WiFi connection failed.
To configure, connect to 'feiweifeiwei' with password 'iloveyouyu'
The IP is: 192.168.4.1
```

Figure 4.15: Connection fail.

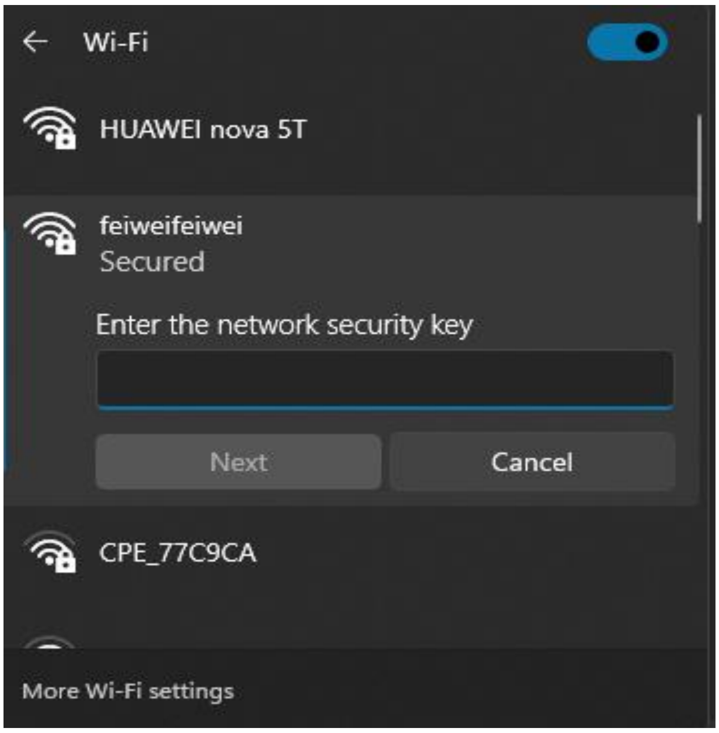


Figure 4.16: SSID of Access Point.

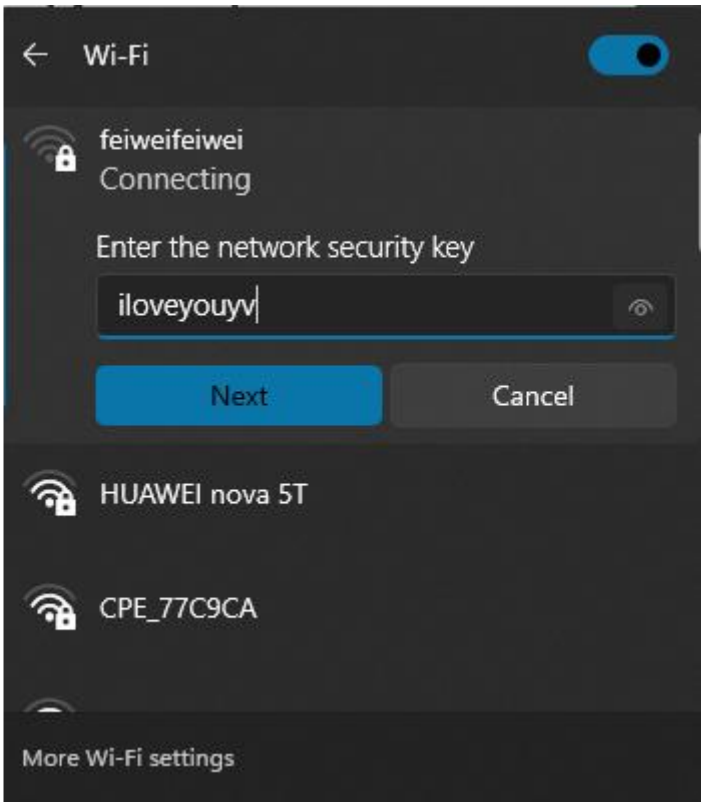


Figure 4.17: Password of Access Point.

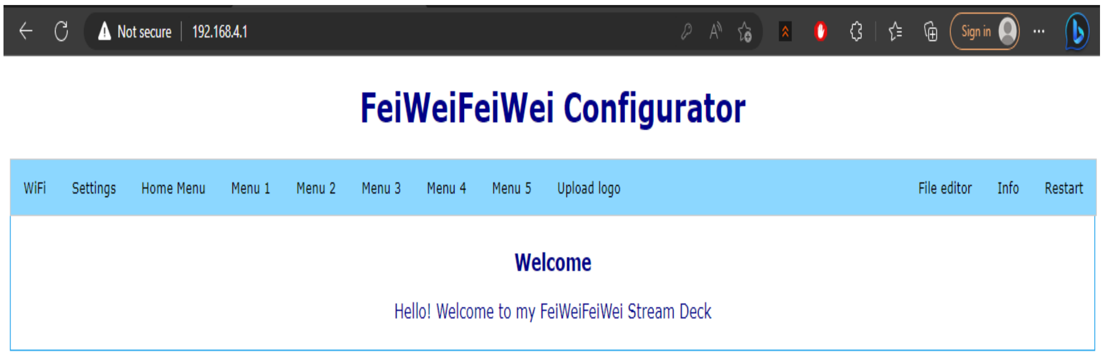
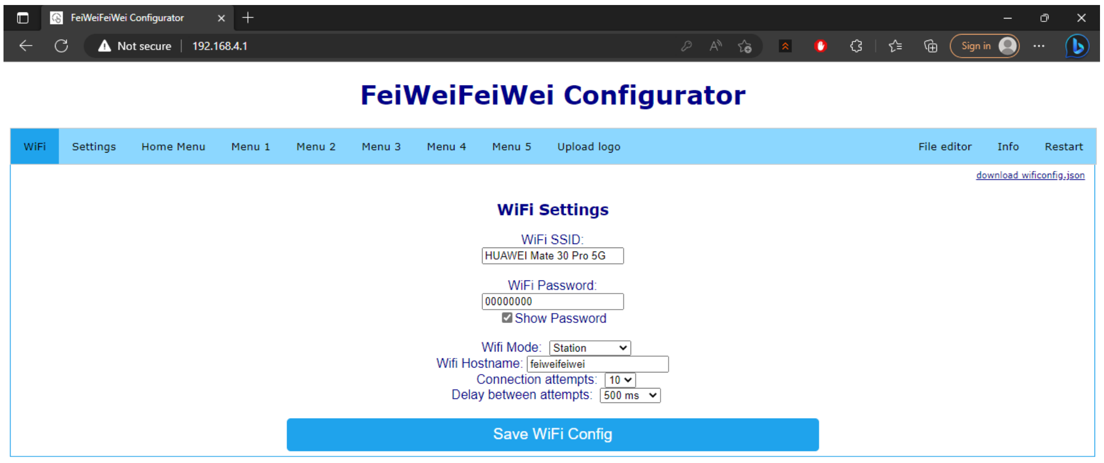


Figure 4.18: Webserver of The Stream Deck.

4.5 Station Point (STA) Connection

Figure 4.19 displayed the WiFi settings page after the completion of the AP connection. The WiFi or hotspot information, such as the Service Set Identifier (SSID) and password, is entered into the WiFi settings page. Upon pressing the "Save WiFi Config" button, two options become available to the user: to either return to the configuration page or to restart the stream deck, as depicted in Figure 4.20. Upon activation of the restart function, the preceding procedures are reiterated and the display exhibits information regarding the establishment of a connection with the station point, along with the display of WiFi or hotspot of IP address. Figure 4.21 depicted the utilization of a phone hotspot, whereby the IP address of the hotspot is displayed on the screen.



WiFi Settings

WiFi SSID: HUAWEI Mate 30 Pro 5G

WiFi Password: 00000000

☑ Show Password

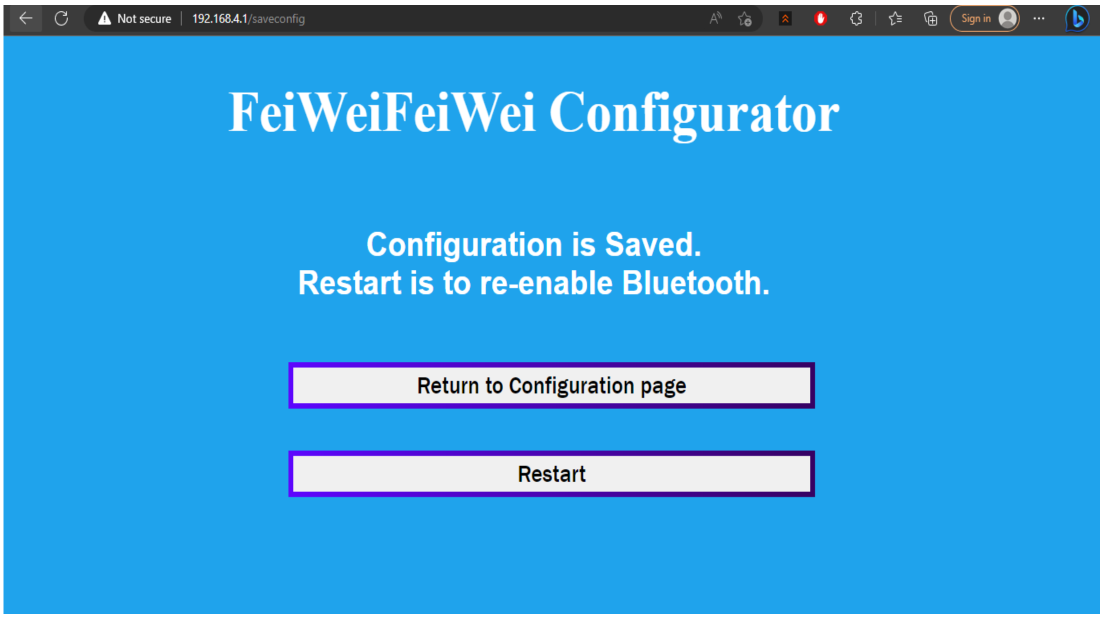
Wifi Mode: Station

Wifi Hostname: feiweifeiwei

Connection attempts: 10

Delay between attempts: 500 ms

Figure 4.19: WiFi Settings Page.


Not secure | 192.168.4.1/saveconfig

FeiWeiFeiWei Configurator

Configuration is Saved.
Restart is to re-enable Bluetooth.

Figure 4.20: Two Option to Use.

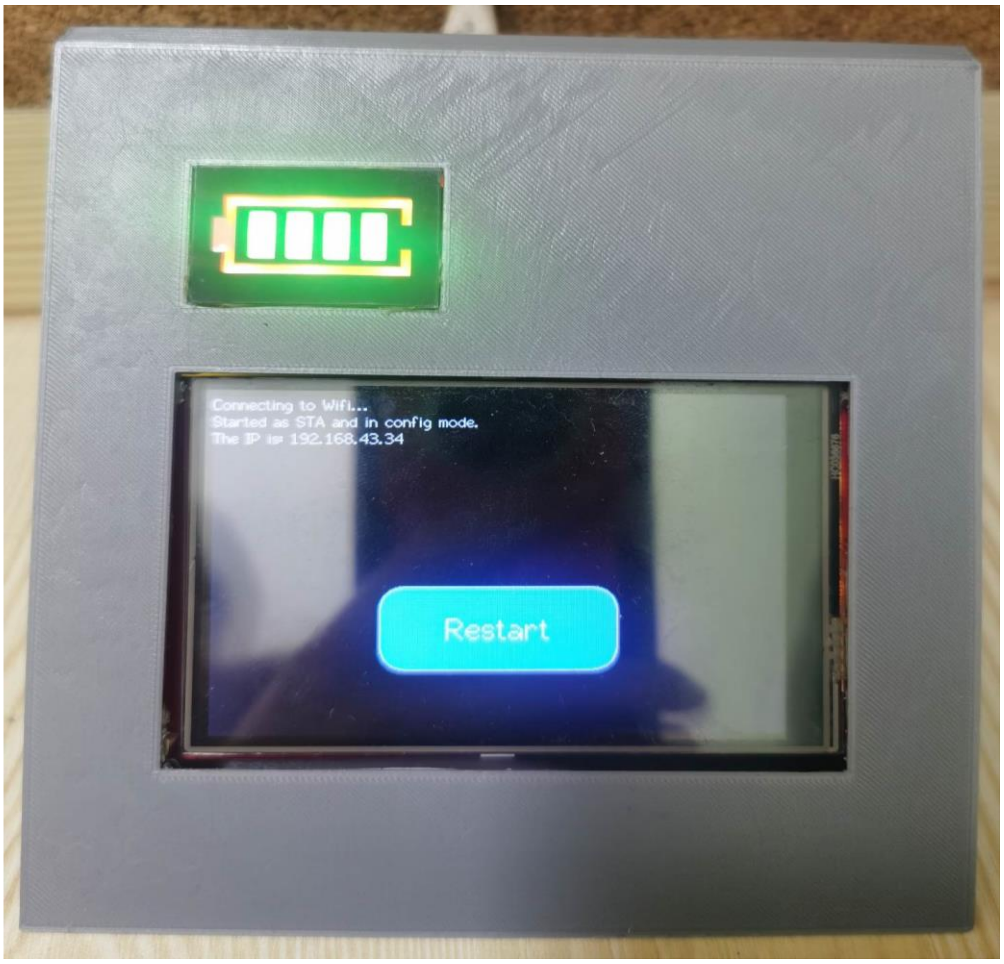
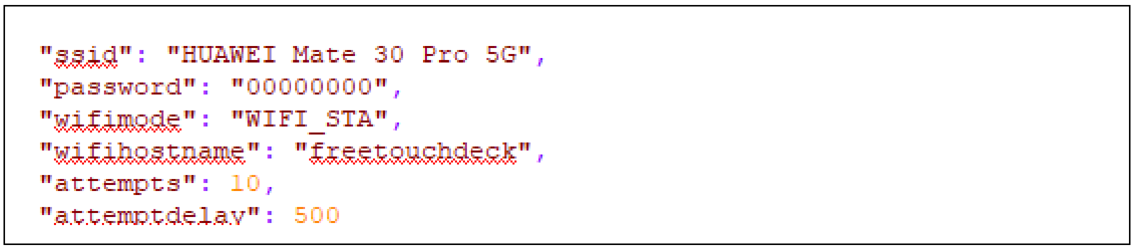

Connecting to WiFi...
Started as STA and in config mode.
The IP is: 192.168.43.34

Figure 4.21: Station Point is Connected.

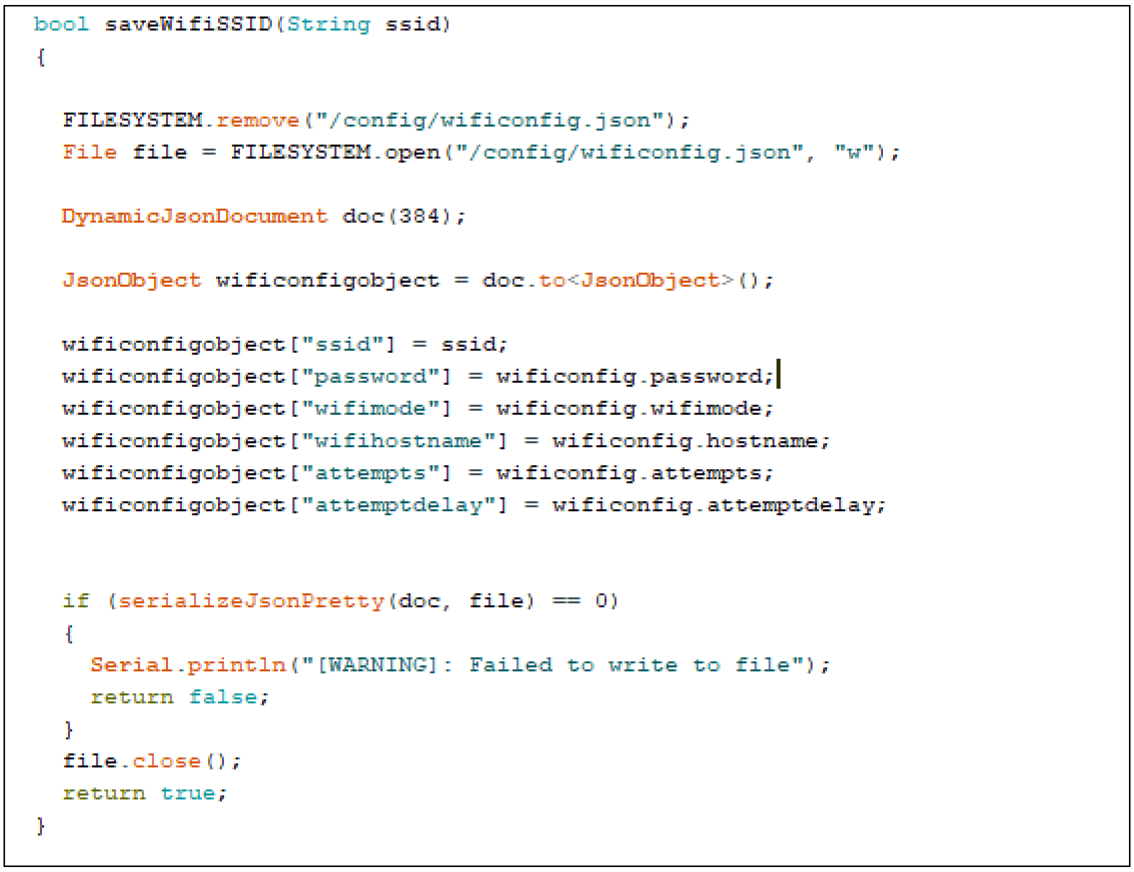
4.6 Second Way to connect the Station Point (STA) or Access Point (AP)

In Figure 4.22, there is another way to connect STA or AP manually. There is a JSON file. The specific information had to key in exactly to the related field. Thus, the JSON file will be worked fine with the code that is shown the Figure 4.23. In Figure 4.23, the code uses the ArduinoJSON library to generate a JSON (JavaScript Object Notation) document. JSON is a lightweight data interchange format that is simple for both humans and machines to read, write, parse, and generate. From the first and second line of the code, the presented code pertains to the management of files on the ESP32. To be precise, the process entails the removal of a pre-existing file and the subsequent creation of a new file. The initial line of code, FILESYSTEM.remove("/config/wificonfig.json"), serves to eliminate the file located at the specified path of "/config/wificonfig.json" from the file system. The function mentioned above is a constituent element of the SPIFFS (SPI Flash File System) library, which facilitates the reading and writing of files to the flash memory of the microcontroller. The second line of code declares a variable named "file" of type "File" and assigns it the value of "FILESYSTEM".The function call open("/config/wificonfig.json", "w") instantiates a new file with identical name and path ("/config/wificonfig.json") and initiates it in write mode ("w"). The implication of this is that all prior contents of the file will be deleted, rendering the file in a state of readiness to accept fresh data. The code generated a new JSON file with a primary size of 384 bytes. The statement implies that the document had the capacity to retain a maximum of 384 bytes of information, with the possibility of expanding its size if necessary. The fourth line of code instantiates a new JSON entity denominated wificonfigobject and designates it as the principal entity of the JSON document. A JSON object is a composite data structure consisting of a set of unordered key-value pairs, where the key is a string, and the value can be any valid JSON data type. The subsequent lines of code append key-value pairs to the wificonfigobject. For example, the addition of a new key-value pair to the wificonfigobject is achieved through the implementation of the statement wificonfigobject["ssid"] = ssid;. The key "ssid" is assigned to the newly added pair, while the value is represented by the contents of the variable SSID. The code snippet wificonfigobject["password"] = wificonfig.password; creates a new key-value pair in which the key is denoted as "password" and the value is equivalent to the contents of the password attribute within the wificonfig construct. Through the inclusion of these key-value pairs into the wificonfigobject, a JSON object is generated that comprehensively encompasses the requisite details pertaining to the WiFi configuration, including but not limited to the SSID and password. Upon the creation of the JSON object, it can be utilised for the purpose of transmitting the aforementioned information to alternative devices or alternatively, it can be stored within a file.



```
"ssid": "HUAWEI Mate 30 Pro 5G",
"password": "00000000",
"wifimode": "WIFI_STA",
"wifihostname": "freetouchdeck",
"attempts": 10,
"attemptdelay": 500
```

Figure 4.22: WiFi Details.

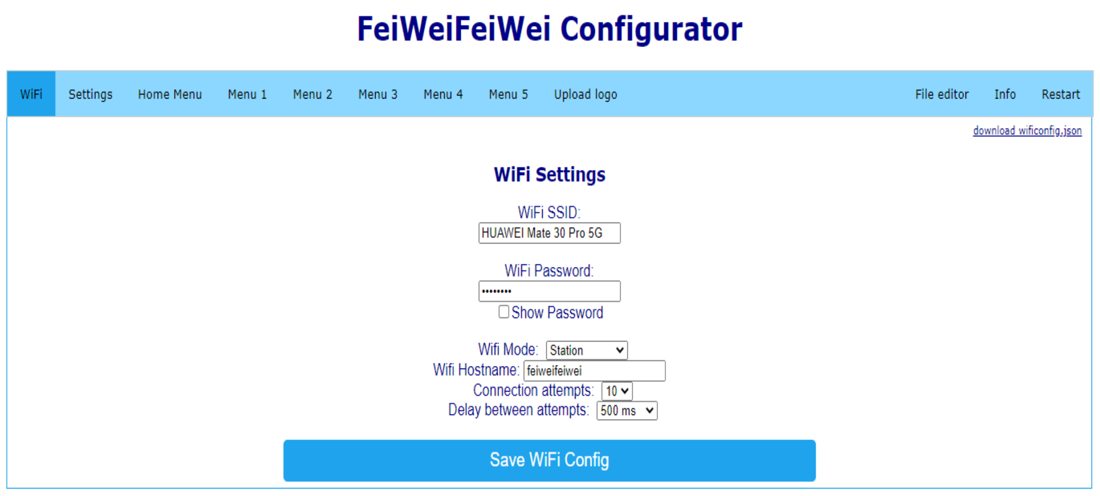


```
bool saveWifiSSID(String ssid)
{

  FILESYSTEM.remove("/config/wificonfig.json");
  File file = FILESYSTEM.open("/config/wificonfig.json", "w");

  DynamicJsonDocument doc(384);

  JsonObject wificonfigobject = doc.to<JsonObject>();

  wificonfigobject["ssid"] = ssid;
  wificonfigobject["password"] = wificonfig.password;
  wificonfigobject["wifimode"] = wificonfig.wifimode;
  wificonfigobject["wifihostname"] = wificonfig.hostname;
  wificonfigobject["attempts"] = wificonfig.attempts;
  wificonfigobject["attemptdelay"] = wificonfig.attemptdelay;


  if (serializeJsonPretty(doc, file) == 0)
  {
    Serial.println("[WARNING]: Failed to write to file");
    return false;
  }
  file.close();
  return true;
}
```

Figure 4.23: Get information from JSON.

4.7 Web Server for Stream Deck Configurator

Section 4.7 presented a section that displayed all the figures pertaining to the web server configuration for the stream deck. The initial tab of the web server features a WiFi configuration page designated for the stream deck, as depicted in Figure 4.24. Additionally, there exists a universal configuration for the stream deck that can be adjusted through a web server. The objective of this page is to modify the hue of the primary menu buttons, the hue of the functional buttons, the hue of the latch, and the background hue of the stream deck. The new configuration had been completed, as depicted in Figure 4.25. As depicted in Figures 4.27 through 4.30, the shortcut keys have been reconfigured within menus 1 through 5 to enhance their functionality for specific purposes. Logos are employed to facilitate user recognition of button functionality. Hence, the web server had incorporated the "Upload a New Logo" page as depicted in Figure 4.32. Additionally, it is suggested to generate personalized logos for the stream deck by utilizing the provided link or URL. The specifications for logo creation necessitate the utilization of images with recommended dimensions of 75x75 pixels and saved in the .bmp file format. Figure 4.33 displayed a compilation of images in the .bmp format. The removal of images is at the discretion of the user.



WiFi Settings

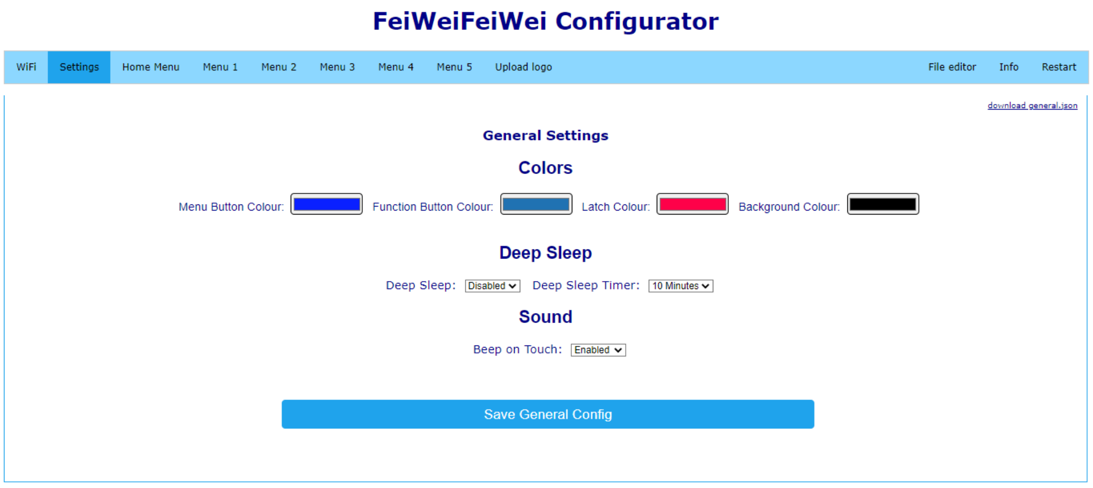
WiFi SSID: HUAWEI Mate 30 Pro 5G

WiFi Password: ••••••••

☐ Show Password

Wifi Mode: Station
Wifi Hostname: feiweifeiwei
Connection attempts: 10
Delay between attempts: 500 ms

Figure 4.24: WiFi Settings from Webserver.



General Settings

Colors

Menu Button Colour: 🟦 Function Button Colour: 🟦 Latch Colour: 🟥 Background Colour: ⬛

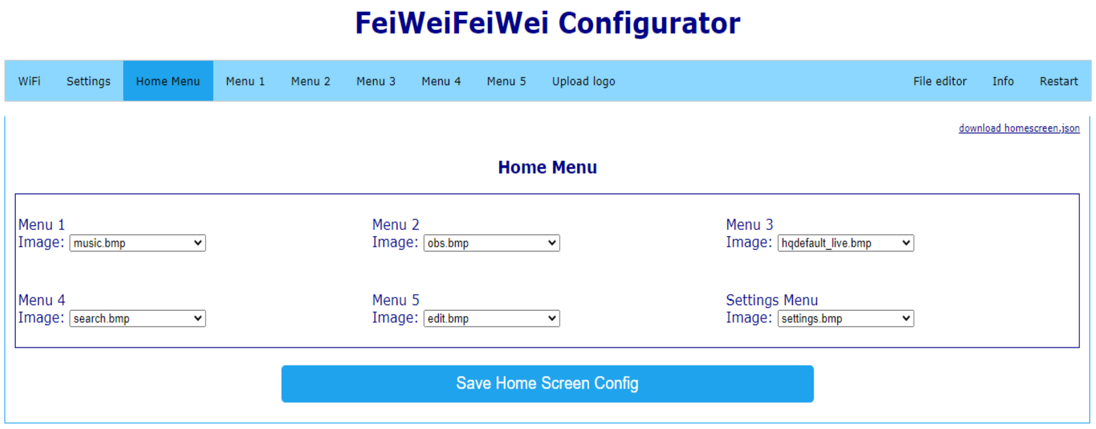
Deep Sleep

Deep Sleep: Disabled Deep Sleep Timer: 10 Minutes

Sound

Beep on Touch: Enabled

Figure 4.25: General Settings from Webserver.



Home Menu

Menu 1
Image: music.bmp

Menu 2
Image: obs.bmp

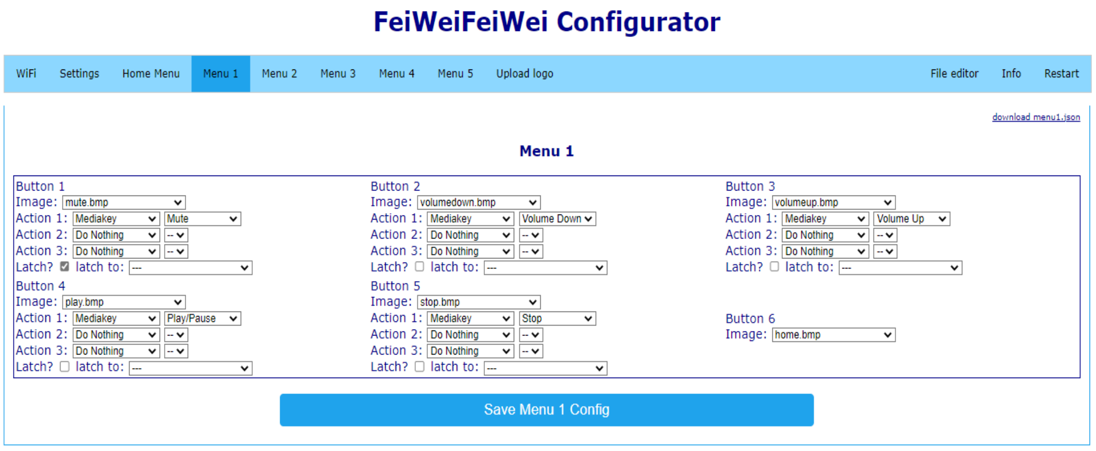
Menu 3
Image: hqdefault_live.bmp

Menu 4
Image: search.bmp

Menu 5
Image: edit.bmp

Settings Menu
Image: settings.bmp

Figure 4.26: Main Menu from Webserver.



Menu 1

Button 1
Image: mute.bmp
Action 1: Mediakey Mute
Action 2: Do Nothing --
Action 3: Do Nothing --
Latch? ✓ latch to: ---

Button 2
Image: volumedown.bmp
Action 1: Mediakey Volume Down
Action 2: Do Nothing --
Action 3: Do Nothing --
Latch? ☐ latch to: ---

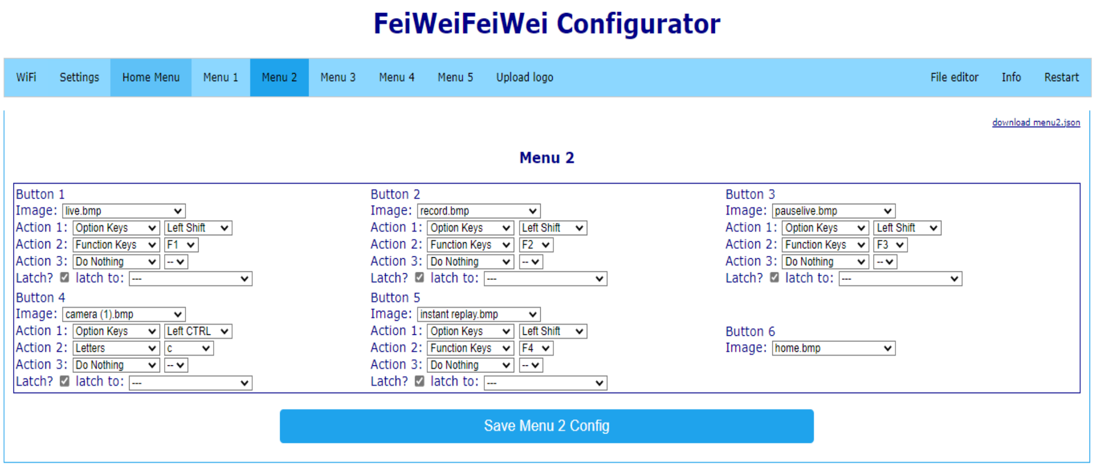
Button 3
Image: volumeup.bmp
Action 1: Mediakey Volume Up
Action 2: Do Nothing --
Action 3: Do Nothing --
Latch? ☐ latch to: ---

Button 4
Image: play.bmp
Action 1: Mediakey Play/Pause
Action 2: Do Nothing --
Action 3: Do Nothing --
Latch? ☐ latch to: ---

Button 5
Image: stop.bmp
Action 1: Mediakey Stop
Action 2: Do Nothing --
Action 3: Do Nothing --
Latch? ☐ latch to: ---

Button 6
Image: home.bmp

Figure 4.27: Menu 1 from Webserver.



Menu 2

Button 1
Image: live.bmp
Action 1: Option Keys | Left Shift
Action 2: Function Keys | F1
Action 3: Do Nothing | --
Latch? ✓ latch to: ---

Button 2
Image: record.bmp
Action 1: Option Keys | Left Shift
Action 2: Function Keys | F2
Action 3: Do Nothing | --
Latch? ✓ latch to: ---

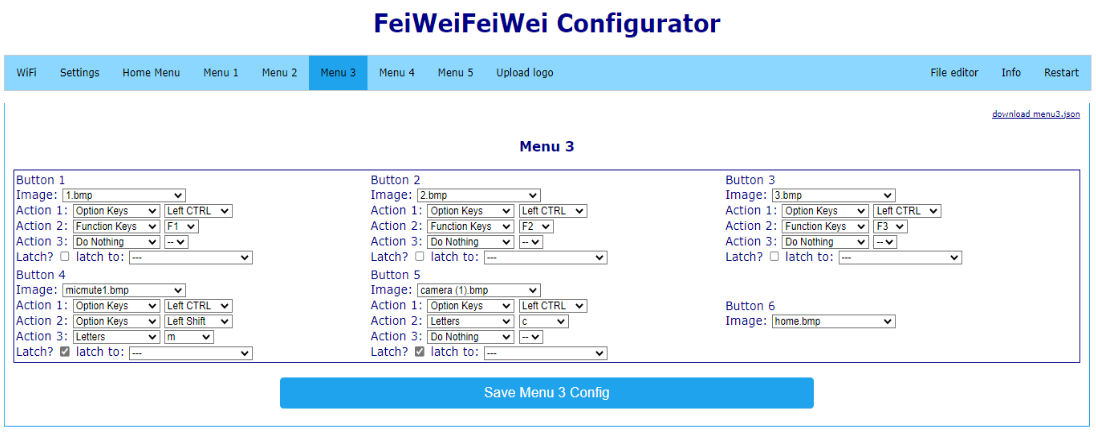
Button 3
Image: pauselive.bmp
Action 1: Option Keys | Left Shift
Action 2: Function Keys | F3
Action 3: Do Nothing | --
Latch? ✓ latch to: ---

Button 4
Image: camera (1).bmp
Action 1: Option Keys | Left CTRL
Action 2: Letters | c
Action 3: Do Nothing | --
Latch? ✓ latch to: ---

Button 5
Image: instant replay.bmp
Action 1: Option Keys | Left Shift
Action 2: Function Keys | F4
Action 3: Do Nothing | --
Latch? ✓ latch to: ---

Button 6
Image: home.bmp

Figure 4.28: Menu 2 from Webserver.



Menu 3

Button 1
Image: 1.bmp
Action 1: Option Keys | Left CTRL
Action 2: Function Keys | F1
Action 3: Do Nothing | --
Latch? ☐ latch to: ---

Button 2
Image: 2.bmp
Action 1: Option Keys | Left CTRL
Action 2: Function Keys | F2
Action 3: Do Nothing | --
Latch? ☐ latch to: ---

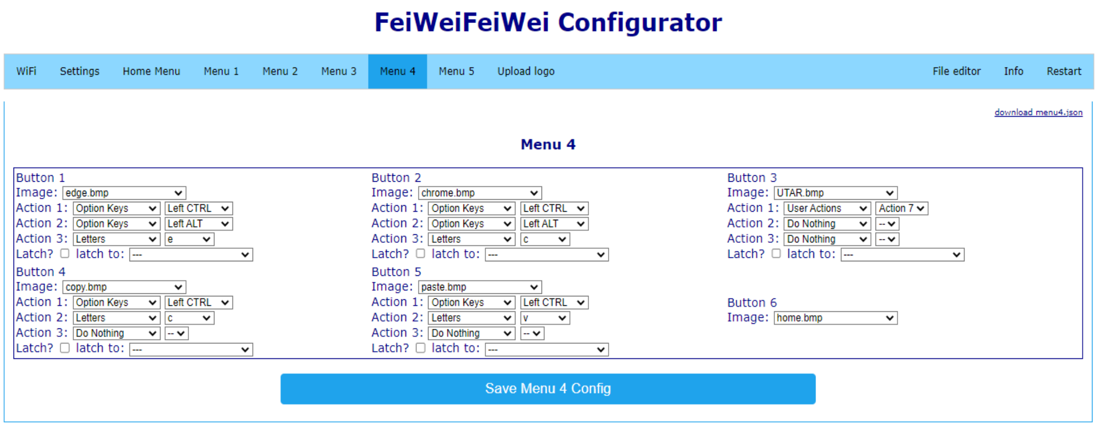
Button 3
Image: 3.bmp
Action 1: Option Keys | Left CTRL
Action 2: Function Keys | F3
Action 3: Do Nothing | --
Latch? ☐ latch to: ---

Button 4
Image: micmute1.bmp
Action 1: Option Keys | Left CTRL
Action 2: Option Keys | Left Shift
Action 3: Letters | m
Latch? ✓ latch to: ---

Button 5
Image: camera (1).bmp
Action 1: Option Keys | Left CTRL
Action 2: Letters | c
Action 3: Do Nothing | --
Latch? ✓ latch to: ---

Button 6
Image: home.bmp

Figure 4.29: Menu 3 from Webserver.



Menu 4

Button 1
Image: edge.bmp
Action 1: Option Keys | Left CTRL
Action 2: Option Keys | Left ALT
Action 3: Letters | e
Latch? ☐ latch to: ---

Button 2
Image: chrome.bmp
Action 1: Option Keys | Left CTRL
Action 2: Option Keys | Left ALT
Action 3: Letters | c
Latch? ☐ latch to: ---

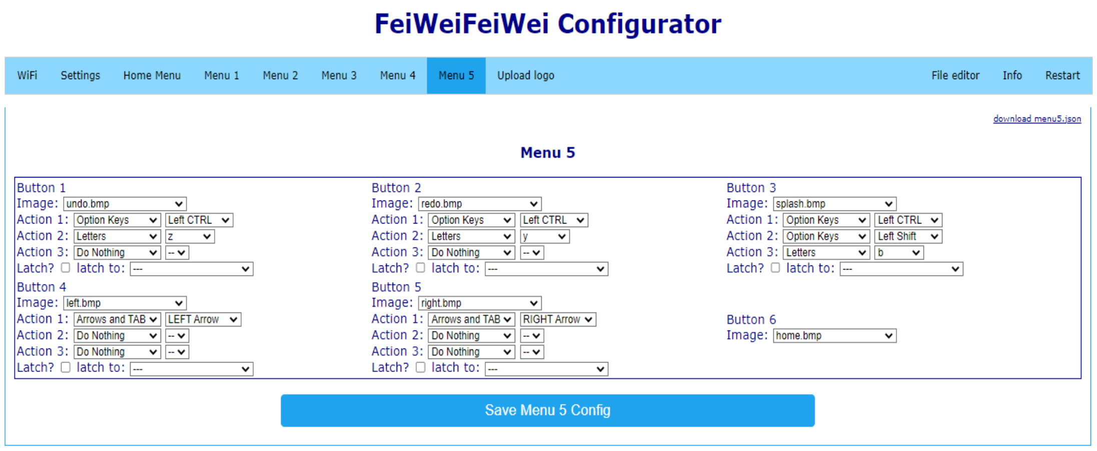
Button 3
Image: UTAR.bmp
Action 1: User Actions | Action 7
Action 2: Do Nothing | --
Action 3: Do Nothing | --
Latch? ☐ latch to: ---

Button 4
Image: copy.bmp
Action 1: Option Keys | Left CTRL
Action 2: Letters | c
Action 3: Do Nothing | --
Latch? ☐ latch to: ---

Button 5
Image: paste.bmp
Action 1: Option Keys | Left CTRL
Action 2: Letters | v
Action 3: Do Nothing | --
Latch? ☐ latch to: ---

Button 6
Image: home.bmp

Figure 4.30: Menu 4 from Webserver.



Menu 5

Button 1
Image: undo.bmp
Action 1: Option Keys, Left CTRL
Action 2: Letters, z
Action 3: Do Nothing, --
Latch? ☐ latch to: ---

Button 2
Image: redo.bmp
Action 1: Option Keys, Left CTRL
Action 2: Letters, y
Action 3: Do Nothing, --
Latch? ☐ latch to: ---

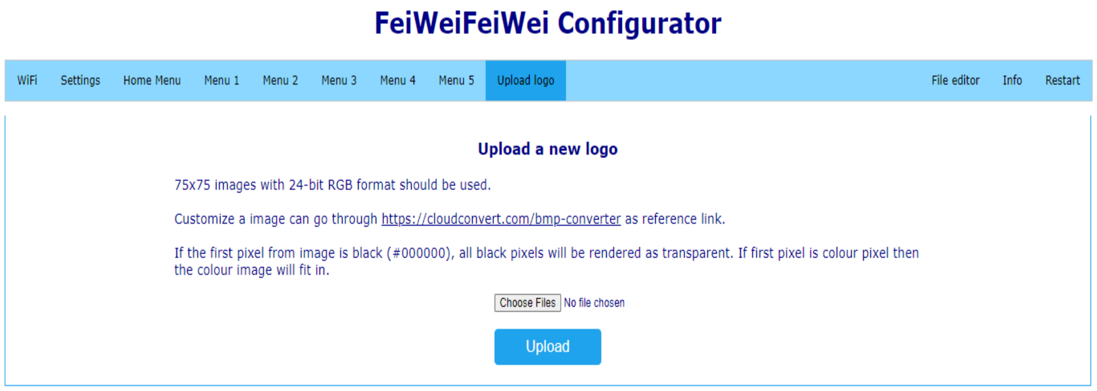
Button 3
Image: splash.bmp
Action 1: Option Keys, Left CTRL
Action 2: Option Keys, Left Shift
Action 3: Letters, b
Latch? ☐ latch to: ---

Button 4
Image: left.bmp
Action 1: Arrows and TAB, LEFT Arrow
Action 2: Do Nothing, --
Action 3: Do Nothing, --
Latch? ☐ latch to: ---

Button 5
Image: right.bmp
Action 1: Arrows and TAB, RIGHT Arrow
Action 2: Do Nothing, --
Action 3: Do Nothing, --
Latch? ☐ latch to: ---

Button 6
Image: home.bmp

Figure 4.31: Menu 5 from Webserver.



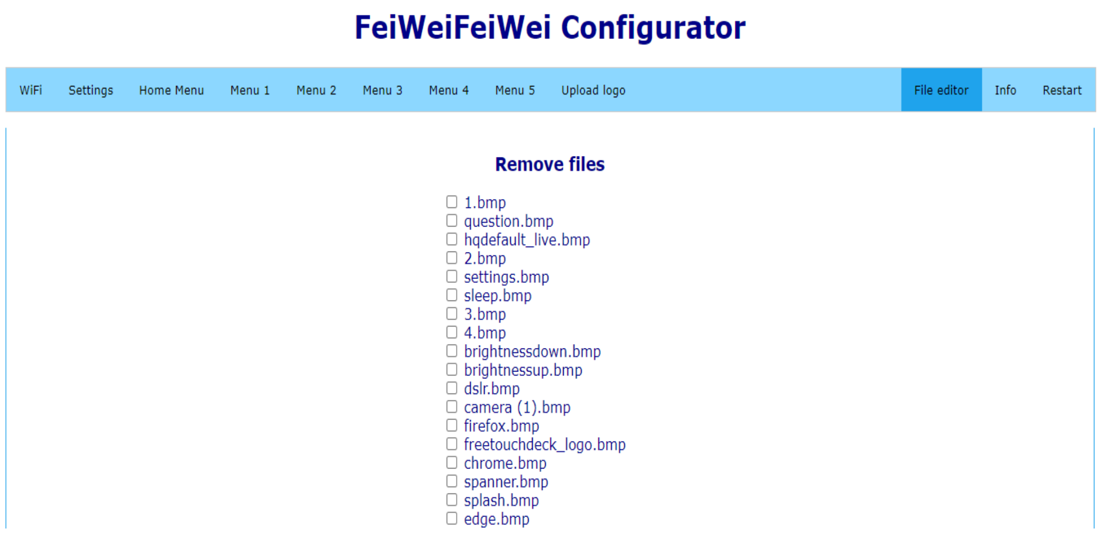
Upload a new logo

75x75 images with 24-bit RGB format should be used.

Customize a image can go through https://cloudconvert.com/bmp-converter as reference link.

If the first pixel from image is black (#000000), all black pixels will be rendered as transparent. If first pixel is colour pixel then the colour image will fit in.

Figure 4.32: Upload a New Logo from Webserver.



Remove files

- ☐ 1.bmp
- ☐ question.bmp
- ☐ hqdefault_live.bmp
- ☐ 2.bmp
- ☐ settings.bmp
- ☐ sleep.bmp
- ☐ 3.bmp
- ☐ 4.bmp
- ☐ brightnessdown.bmp
- ☐ brightnessup.bmp
- ☐ dslr.bmp
- ☐ camera (1).bmp
- ☐ firefox.bmp
- ☐ freetouchdeck_logo.bmp
- ☐ chrome.bmp
- ☐ spanner.bmp
- ☐ splash.bmp
- ☐ edge.bmp

Figure 4.33: Remove Files from Webserver.

4.8 New User Interface of Stream Deck

Figure 4.34 displayed the primary menu of the stream deck on the screen after the implementation of the updated configuration. The stream deck features six distinct buttons, each of which corresponds to a unique menu. These menus include the media control menu, the Open Broadcaster Software Studio (OBS Studio) control menu, the live control menu, the browser menu control, the edit control menu, and the settings of the stream deck. The colour scheme employed in Figure 4.25 is noteworthy. Specifically, the menu buttons are assigned a blue colour code of #0a20ff, the function buttons are assigned a light blue colour code of #2072b2, the latch is assigned a red colour code of #fe0149, and the background is assigned a black colour code of #000000. The outcome of this colour scheme can be observed in Figure 4.34.

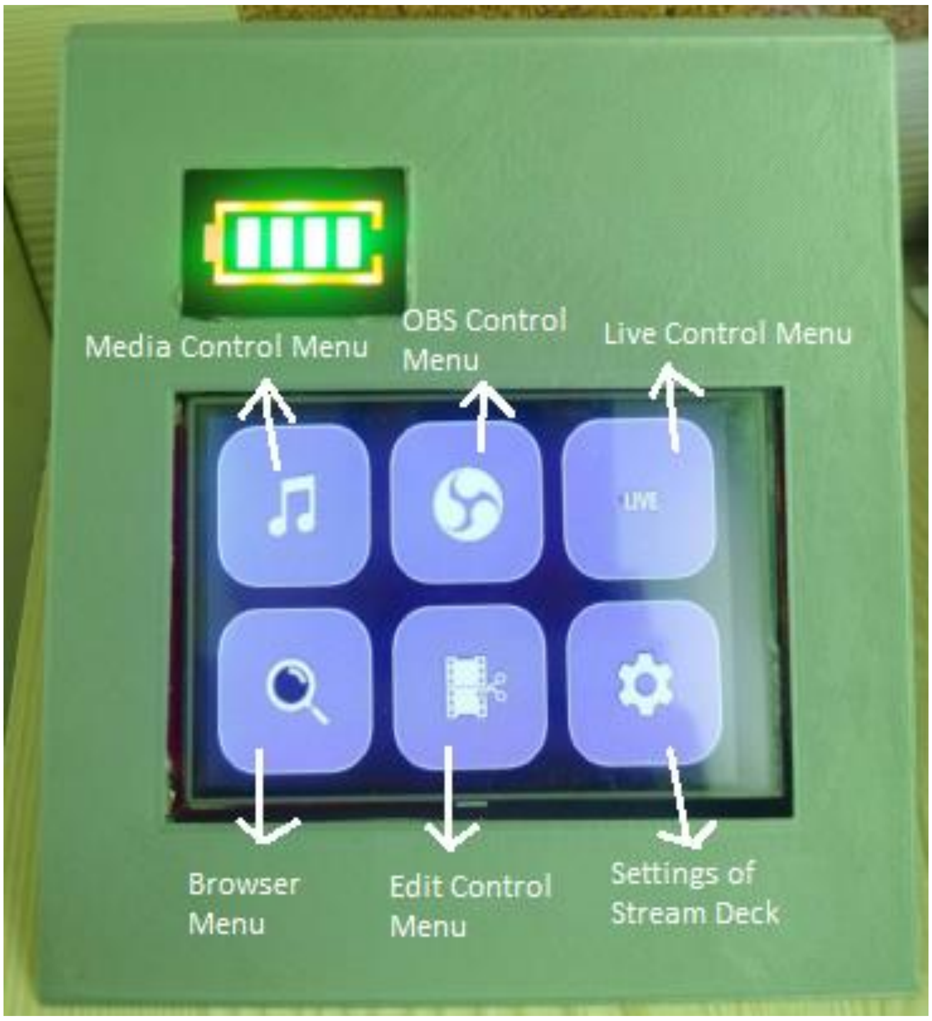


Figure 4.34: Main Menu of Stream Deck.

To utilize the stream deck, the Bluetooth protocol is employed to establish a connection between the stream deck and the corresponding device. The present study involved the utilization of a laptop device that will be connected to a stream deck through the employment of Bluetooth technology. Figure 4.35 illustrated the media control menu, which comprises six distinct functions, namely the mute button, decrease volume button, increase volume button, play or pause button, stop button, and home button. Additionally, it should be noted that the function buttons possess a light blue (#2072b2) background colour. Figure 4.36 depicted that the default volume on the laptop is set to 30% of the total volume. The buttons responsible for adjusting the volume are activated, either to increase or decrease the audio output. The alteration in volume is reflected in the outcomes presented in Figure 4.37 and Figure 4.38. The data presented in Figure 4.37 indicated a reduction in volume from 30% to 28%. The data presented in Figure 4.38 indicates an increase in volume from 30% to 32%. The outcomes obtained through the utilization of a stream deck are presented in Figures 4.39 to 4.41. The pause, play, and stop buttons are utilized to demonstrate these results.

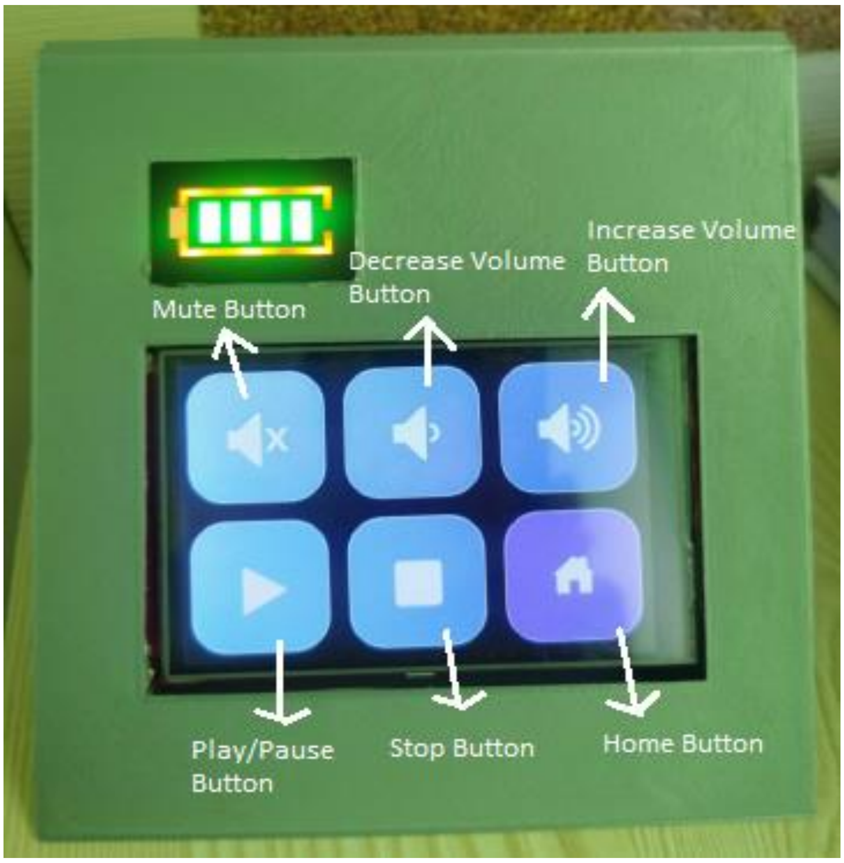


Figure 4.35: Media Control User Interface.

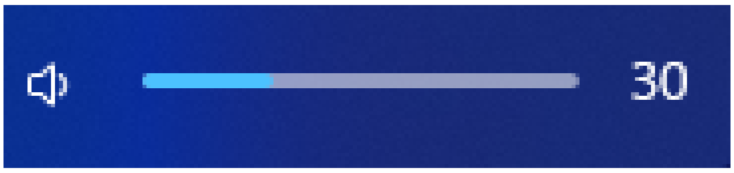


Figure 4.36: Default Volume in Laptop.

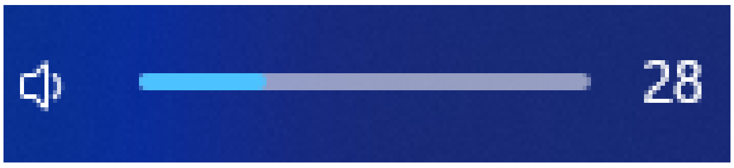


Figure 4.37: Decrease Volume by Stream Deck.

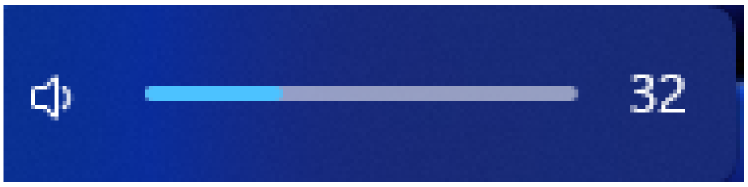


Figure 4.38: Increase Volume by Stream Deck.

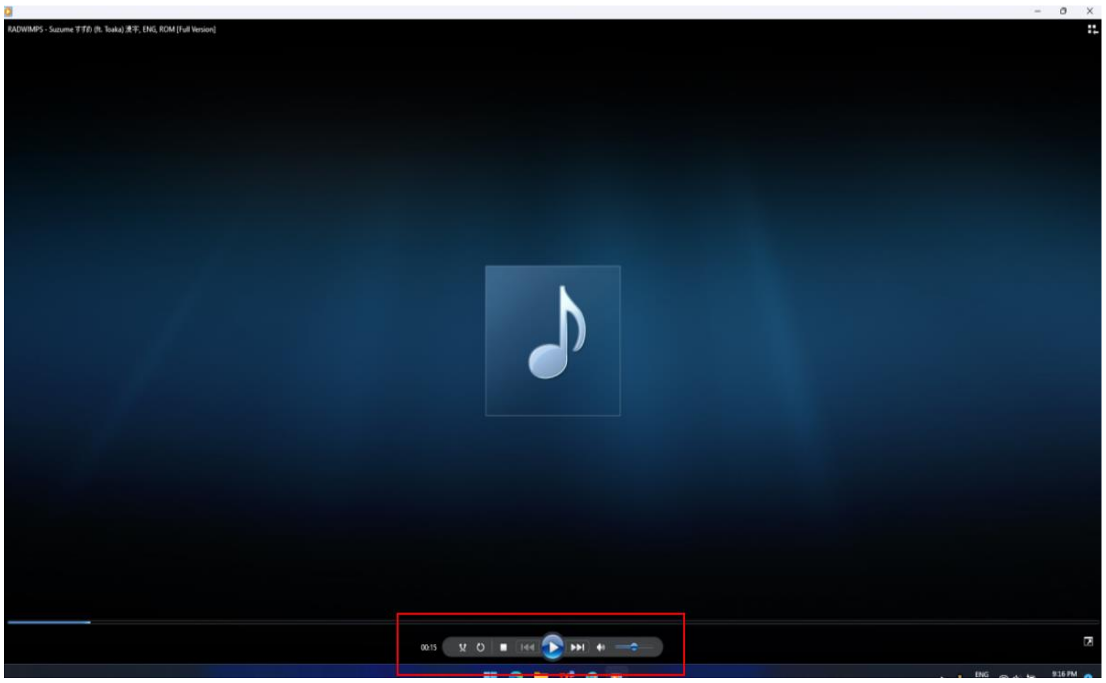


Figure 4.39: Music is Paused.

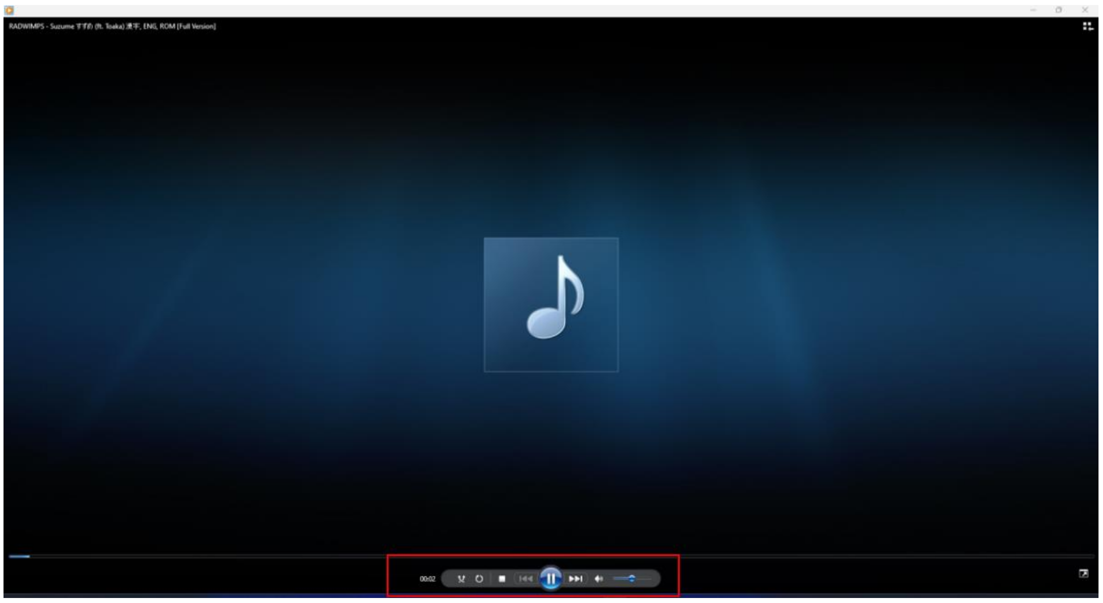

RADWIMPS - Suzume すずめ (ft. Toaka) 漢字, ENG, ROM [Full Version]
00:02

Figure 4.40: Music is Played.

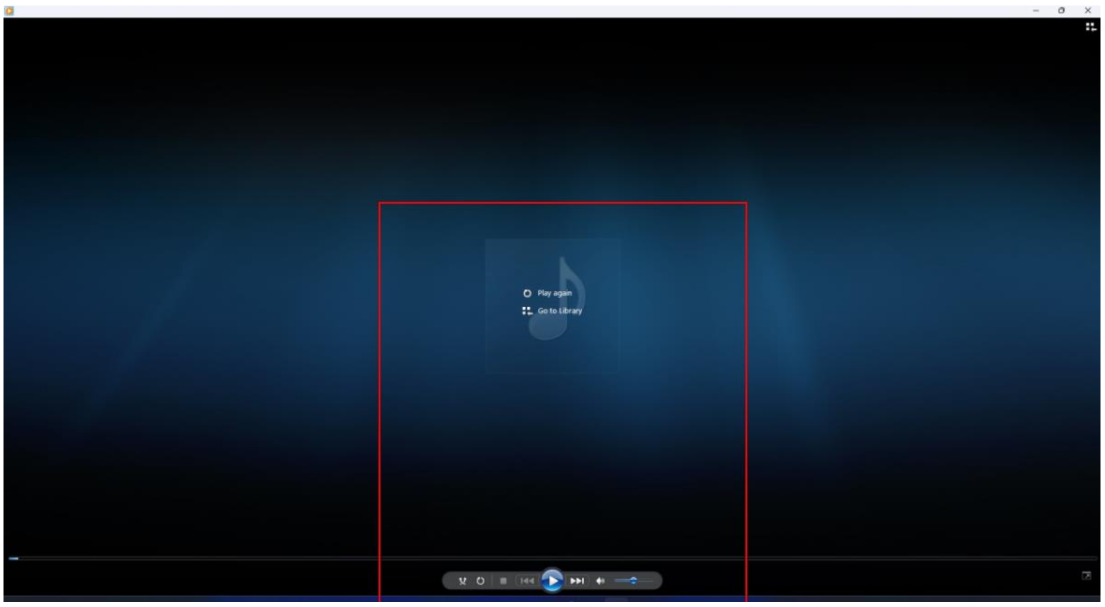


Figure 4.41: Music is Stopped.

Given its primary target audience of streamers and editors, the stream deck is designed to cater to their specific needs. Thus, as depicted in Figure 4.42, the utilization of a broadcasting software, namely Open Broadcaster Software Studio (OBS Studio), is observed. The OBS control menu comprises six buttons, namely the live streaming button, record button, pause live button, on or off webcam, instant replay button, and home button, as illustrated in Figure 4.43. The live control menu comprises six buttons, namely the scene 1 button, scene 2 button, scene 3 button, microphone on/off button, webcam on/off button, and home button, as depicted in Figure 4.44. The buttons had been configured to regulate the OBS studio, as depicted in Figure 4.45 and Figure 4.46. The reason for the white colour of the function buttons is due to the original .bmp image having a white background rather than a transparent one.

Figure 4.42: Open Broadcaster Software Studio (OBS Studio).

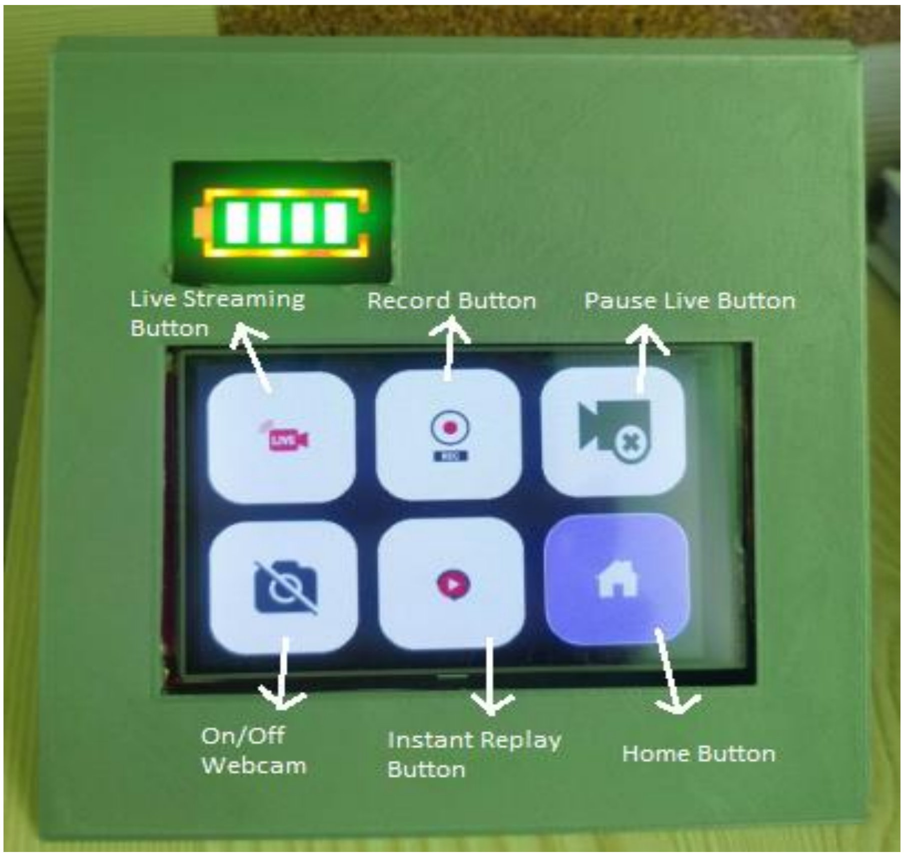


Figure 4.43: OBS Control User Interface.

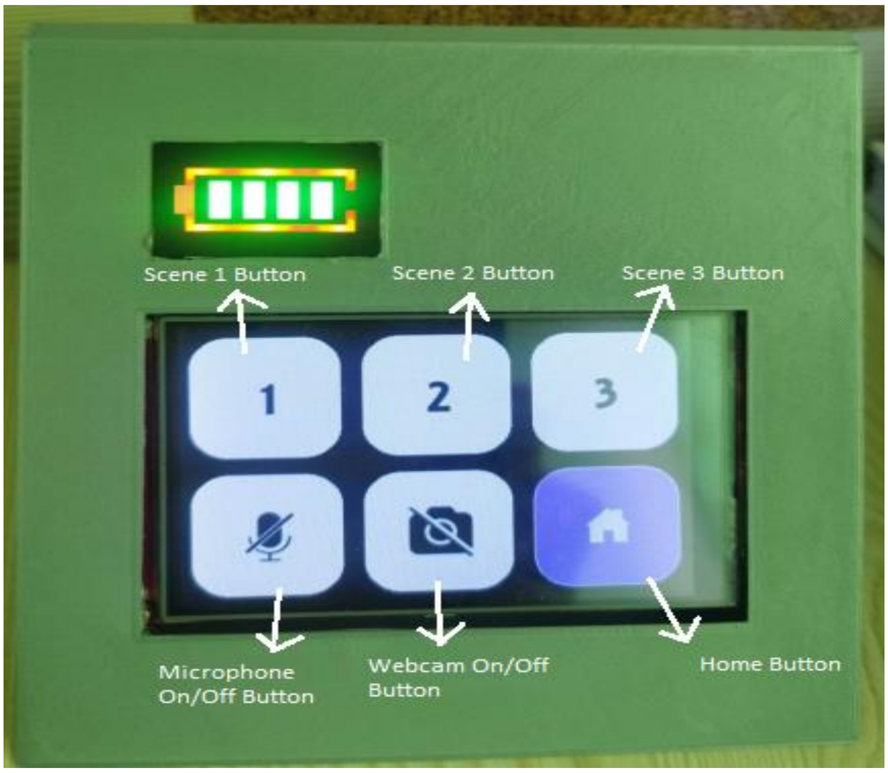
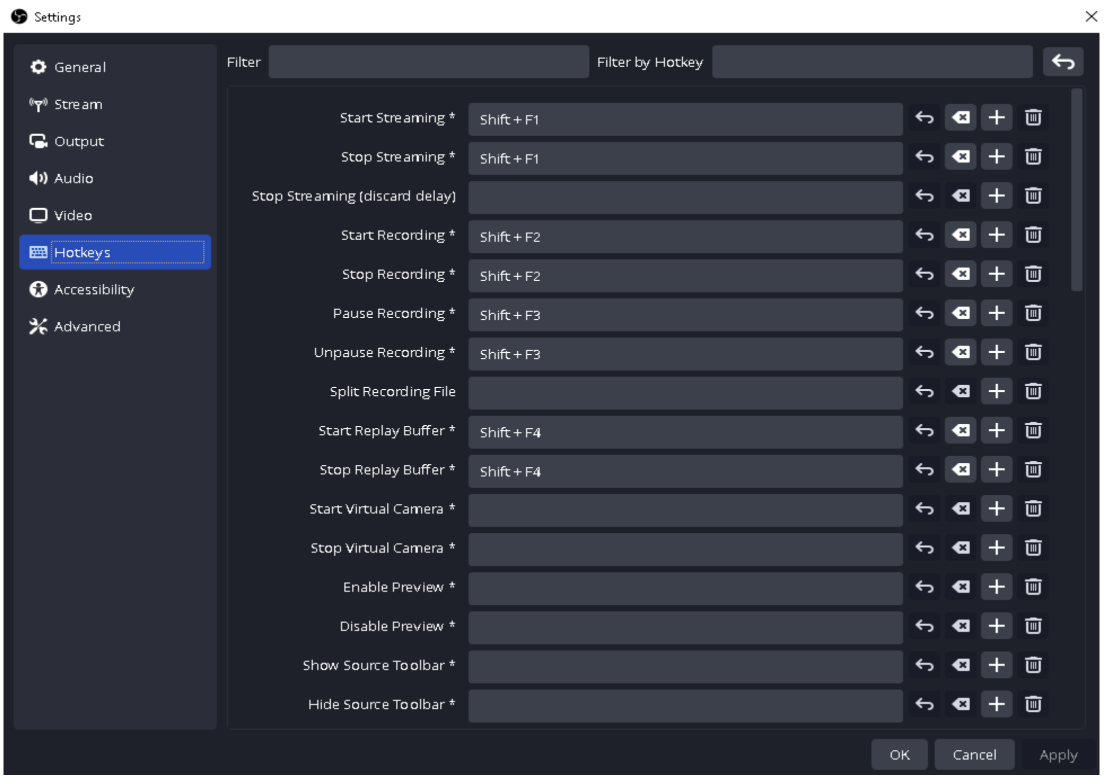
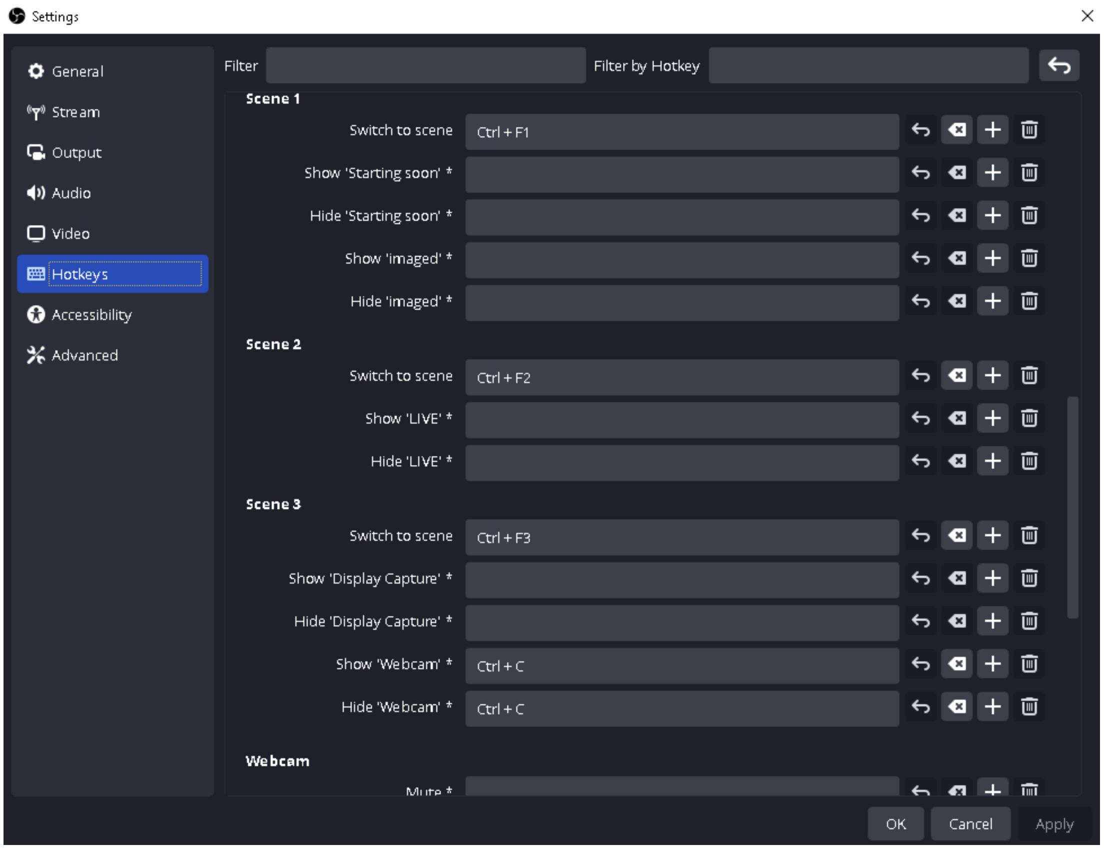


Figure 4.44: Live Control User Interface.



Start Streaming *	Shift + F1
Stop Streaming *	Shift + F1
Stop Streaming (discard delay)	
Start Recording *	Shift + F2
Stop Recording *	Shift + F2
Pause Recording *	Shift + F3
Unpause Recording *	Shift + F3
Split Recording File	
Start Replay Buffer *	Shift + F4
Stop Replay Buffer *	Shift + F4
Start Virtual Camera *	
Stop Virtual Camera *	
Enable Preview *	
Disable Preview *	
Show Source Toolbar *	
Hide Source Toolbar *	

Figure 4.45: Hotkeys of OBS Studio.



Scene 1

Switch to scene	Ctrl + F1
Show 'Starting soon' *	
Hide 'Starting soon' *	
Show 'imaged' *	
Hide 'imaged' *	

Scene 2

Switch to scene	Ctrl + F2
Show 'LIVE' *	
Hide 'LIVE' *	

Scene 3

Switch to scene	Ctrl + F3
Show 'Display Capture' *	
Hide 'Display Capture' *	
Show 'Webcam' *	Ctrl + C
Hide 'Webcam' *	Ctrl + C

Webcam

Mute *	

Figure 4.46: Hotkeys of OBS Studio.

Figure 4.47 displayed the browser menu which included buttons for Microsoft Edge, Google Chrome, UTAR Portal auto login, copy, and paste. The stream deck is capable of launching Microsoft Edge and Google Chrome browsers through the utilization of pre-configured shortcut keys, as depicted in Figures 4.48 and 4.49. The UTAR Portal Login Button is a feature that facilitates automatic access to the UTAR Portal upon activation, by launching a web browser and initiating the login process by automatically writing for personal account information. The copy and paste functions are executed on a connected device through the operation of the respective copy and paste buttons. The colour of the function buttons is white because the .bmp image utilized possesses a white background as opposed to a transparent one.

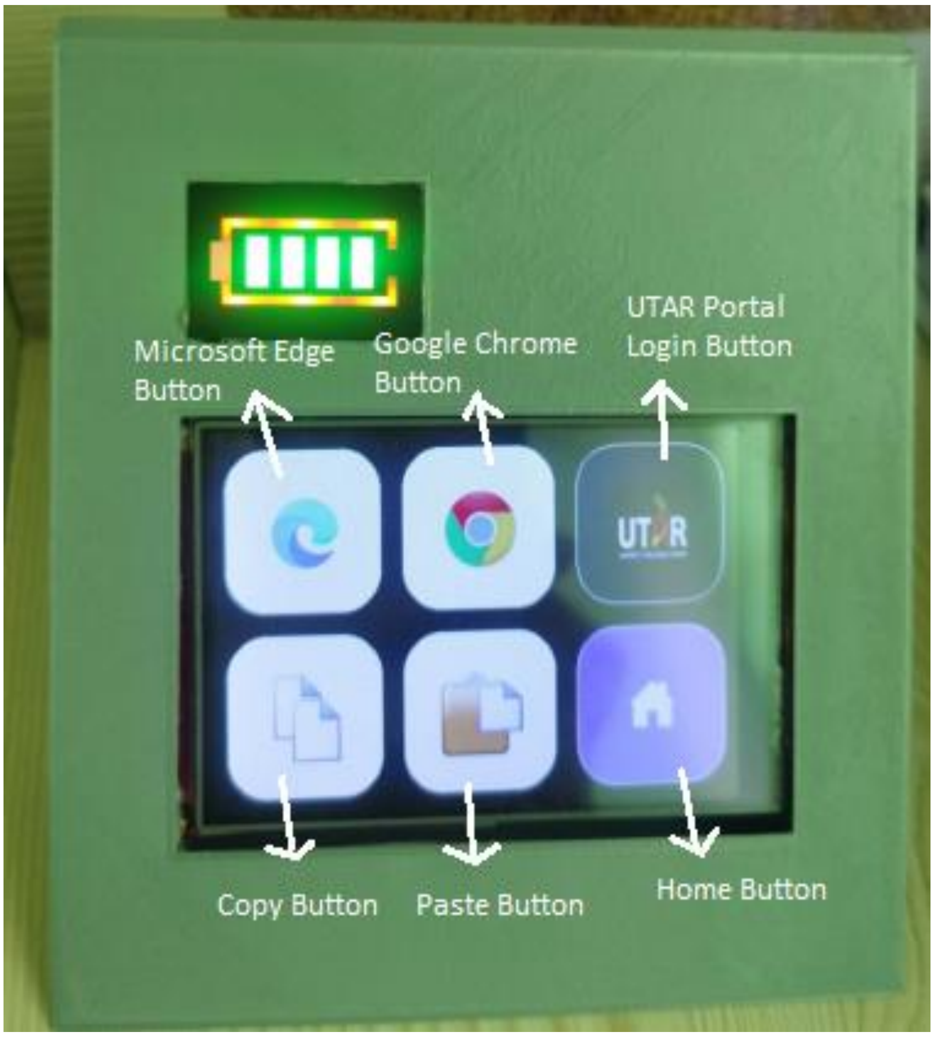
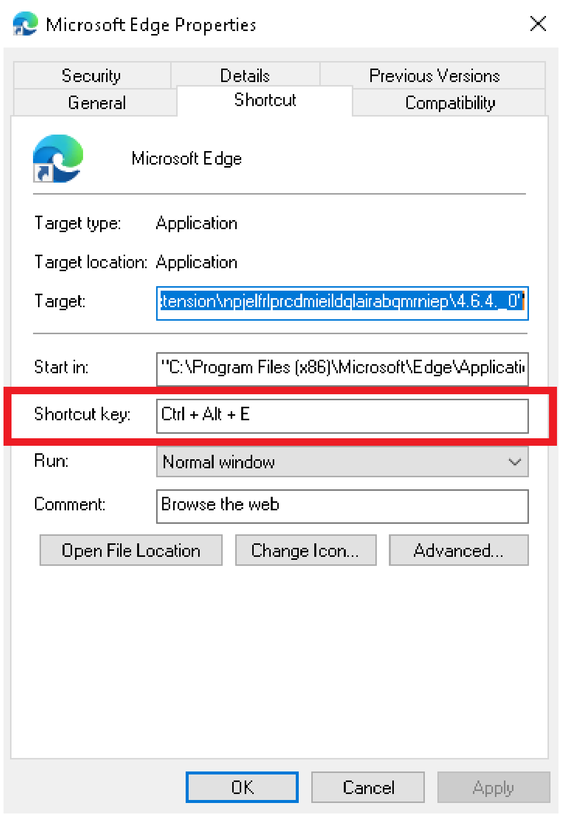


Figure 4.47: Browser User Interface.


Microsoft Edge Properties

Microsoft Edge

Target type: Application

Target location: Application

Target: tension\npjelfrlprcdmieildqlairabqmmiep\4.6.4._0"

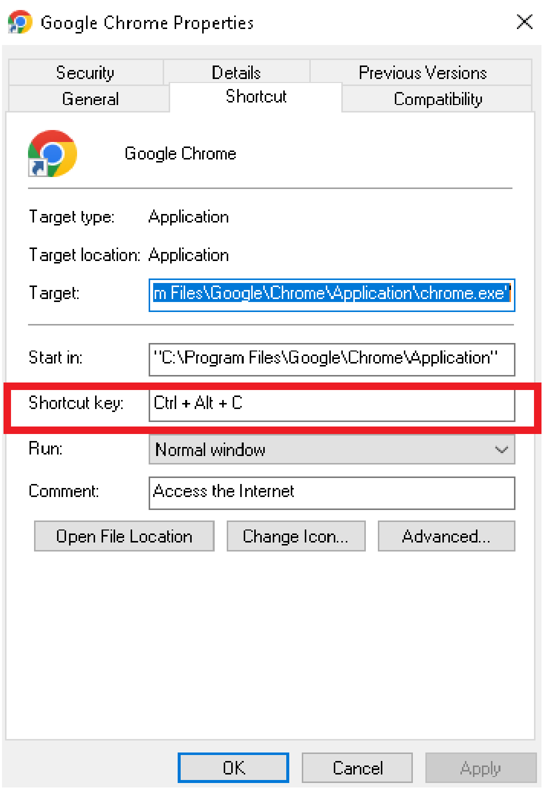
Start in: "C:\Program Files (x86)\Microsoft\Edge\Applicati

Shortcut key: Ctrl + Alt + E

Run: Normal window

Comment: Browse the web

Figure 4.48: Shortcut Key of Microsoft Edge.


Google Chrome Properties

Google Chrome

Target type: Application

Target location: Application

Target: m Files\Google\Chrome\Application\chrome.exe"

Start in: "C:\Program Files\Google\Chrome\Application"

Shortcut key: Ctrl + Alt + C

Run: Normal window

Comment: Access the Internet

Figure 4.49: Shortcut Key of Google Chrome.

Figure 4.50 depicted an edit control menu that comprises commonly used keys such as the undo button, redo button, split button, left arrow button, right arrow button, and home button. The split button referred to a button that with capability to divide a single video or audio file into two distinct sections, as depicted in Figure 4.51.

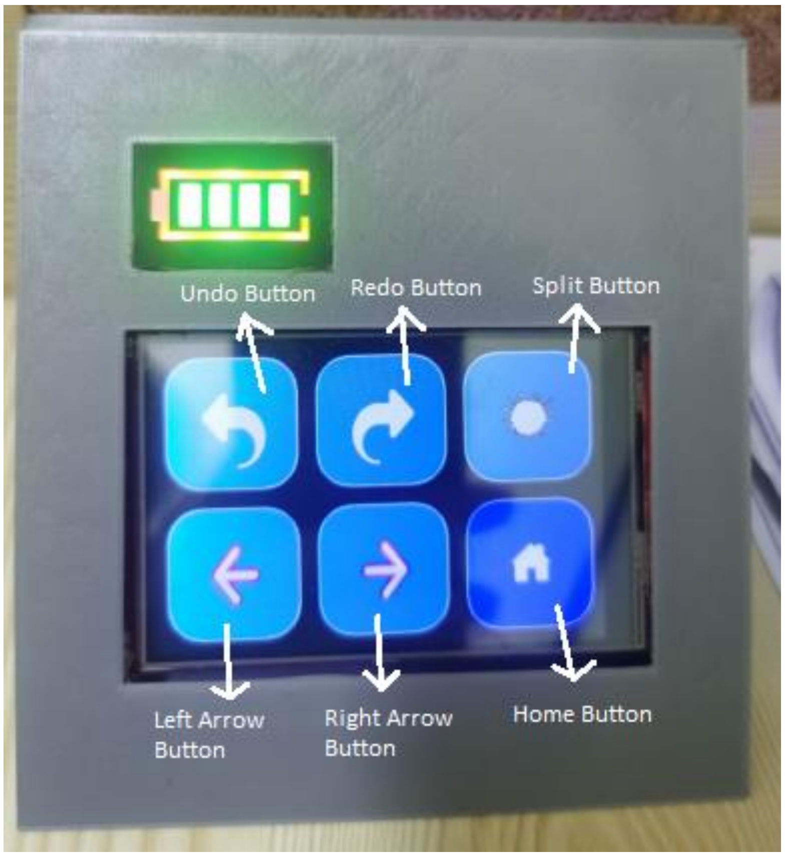


Figure 4.50: Editor Control User Interface.

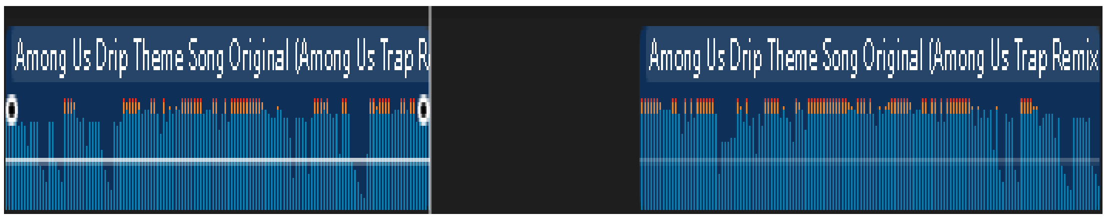

Among Us Drip Theme Song Original (Among Us Trap R
Among Us Drip Theme Song Original (Among Us Trap Remix

Figure 4.51: Splited Music.

Figure 4.52 displayed the settings menu, which included a WiFi button for network connectivity, a decrease brightness button to reduce the brightness of the stream deck, an increase brightness button to enhance the brightness of the stream deck, an on/off sleep mode button, an information button about the stream deck, and a home button. The operational state of the stream deck depicted in Figure 4.53 is sleep mode. To demonstrate the brightness of the stream deck, it had been set within a dimly lit environment to capture a more distinct image. Figures 4.54 and 4.55 illustrated the minimum and maximum brightness levels of the stream deck. Each button had a maximum of 9 possible presses. The brightness can be augmented or diminished by a magnitude of 25 units on each time. Thus, it can be observed that the minimum brightness value is 25, while the maximum brightness value is 255.

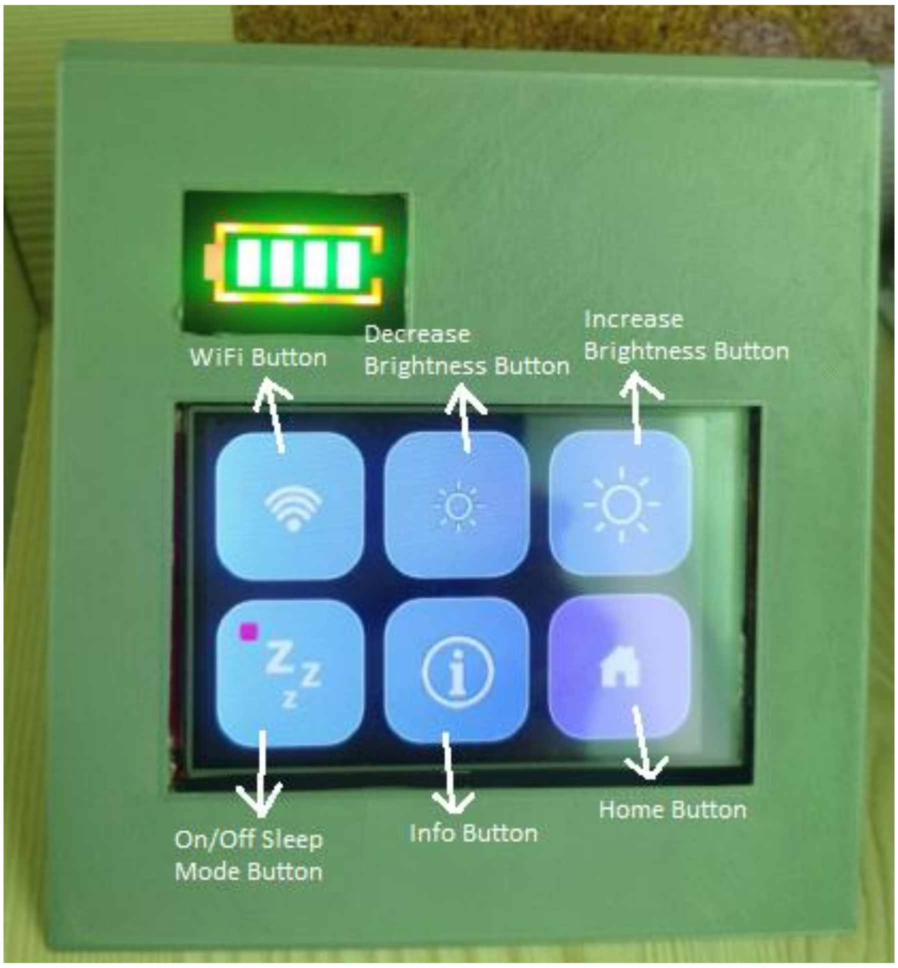


Figure 4.52: Settings User Interface.

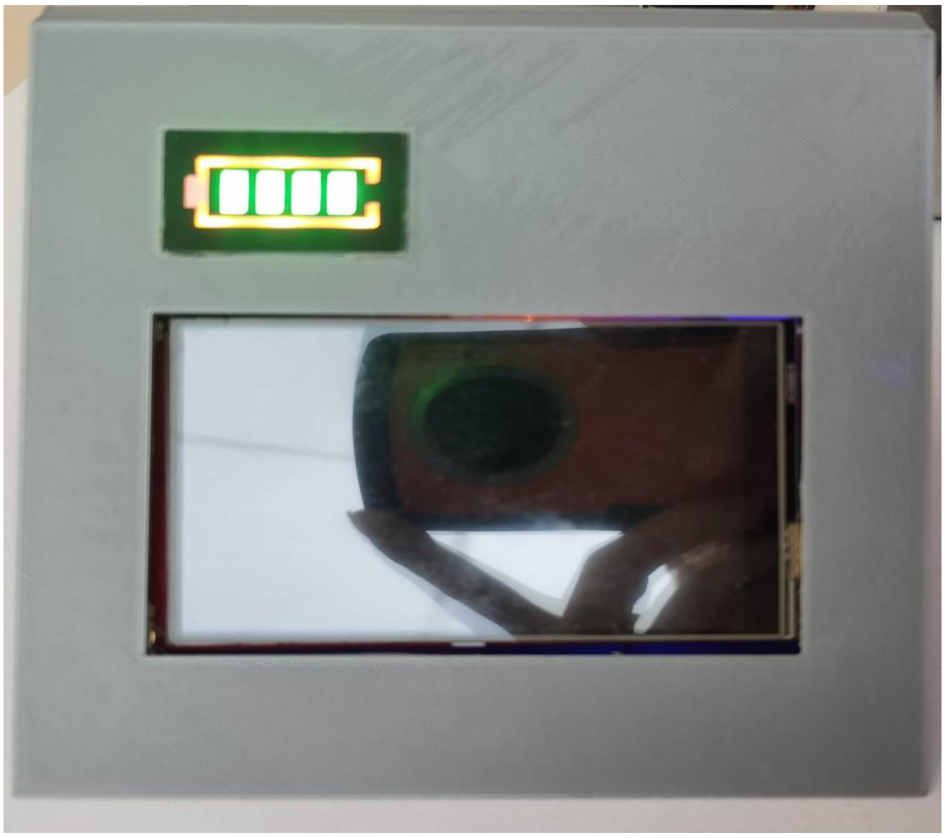


Figure 4.53: Sleep Mode is Enabled.

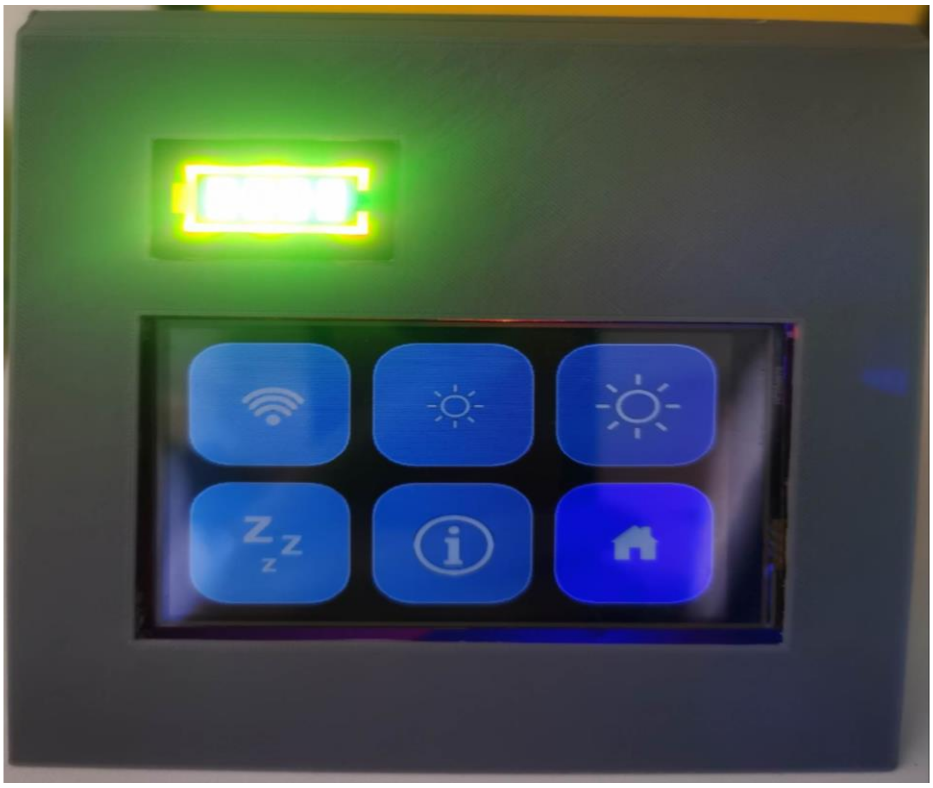


Figure 4.54: Lowest Brightness of Stream Deck.

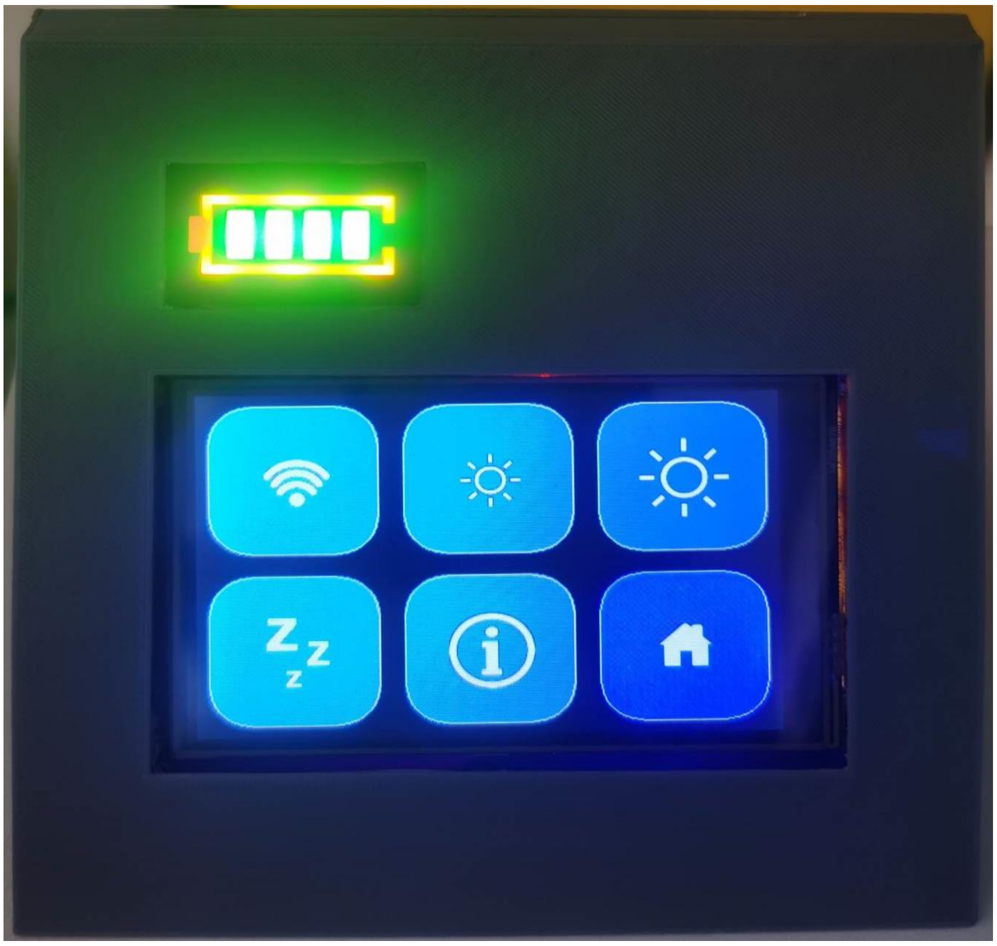


Figure 4.55: Highest Brightness of Stream Deck.

4.9 Discussion on Actions to be Avoided

To achieve a transparent logo, it is necessary to alter the background of the .bmp image to black. Hence, if the stream deck intends to incorporate a black colour background for the function buttons, it is not possible to employ the black colour (#000000) .bmp images. Nonetheless, it is possible to modify them through the webserver. The .bmp file format may occasionally lack transparency, resulting in loading issues when attempting to upload the image into the stream deck. Additionally, even if the background is black, the logo may still appear distorted. It is advisable to select an alternative background colour from the .bmp image as a recommendation. Moreover, the utilization of a type-C charging board is observed in the setup of the stream deck. The stream deck ran into an issue whereby it cannot be reopened after its closure. A period of approximately 30 seconds is required to allow for the discharge of the board from the charging apparatus. Subsequently, the stream deck will resume normal functionality and can be reactivated following depletion of its power. Moreover, if the user desires to activate the stream deck promptly. The type-C charging board required the slicer from the stream deck to be toggled twice and subsequently turned off and on again in order to effectuate discharging. Consequently, the stream deck will function properly once more. Moreover, the modification of the configuration in the webserver necessitated a section-by-section alteration in the menu, as opposed to a comprehensive adjustment in a single step.

4.10 Discussion on Webserver

Webpages comprise various types of data, such as images, text files, hyperlinks, and database files, which are stored on a computer, commonly referred to as server space, that is connected to the Internet. A webserver is a specialized software application that operates on the server-side. Upon a user's request for a webpage, the webserver compiles the necessary data materials into a structured webpage and transmits it to the user's web browser via the Internet. The fundamental aim of a webserver is to gather, handle, and deliver webpages to its users. In this project, this webserver functions as a static webserver as opposed to a dynamic web server. Static webservers exclusively serve static content, which is unchanging and displayed as-is. Static web server is servers that exclusively serve static content, meaning that the content is unchanging and displayed as is. For dynamic webservers is servers that allow for the modification and updating of webpage content. The exchange of information between a webserver and a web browser is facilitated through the utilization of a protocol known as HTTP (Hypertext Transfer Protocol). The webpages that are stored primarily employ static content, which includes HTML documents, images, style sheets, and text files, among others (Douglas.K, Sipiwe.C, and *et al,* 2017).

4.11 Discussion on the Working of a Webserver

Initially, individuals input the Uniform Resource Locator (URL) or Internet Protocol (IP) address of the intended webpage into the designated address bar. Subsequently, the IP address of a domain name is obtained by the web browser using a URL. This is achieved either by converting the URL through the Domain Name System (DNS) or by searching for the IP address in cache memory. The Internet Protocol (IP) address is accountable for routing the user's web browser to the designated webserver. Thirdly, once a connection has been established, the web browser will initiate a HTTP request to solicit the webpage from the webserver in Figure 4.56. In fourth place, once the request is received, the webserver expeditiously transmits the requested page or file to the web browser utilizing the HTTP protocol in Figure 4.56. Finally, if the requested webpage is not present, or an error arises during the procedure, the web server will generate an error message. Provided that there are no errors present, the browser will successfully render the webpage (Douglas.K, Sipiwe.C, and *et al*, 2017).

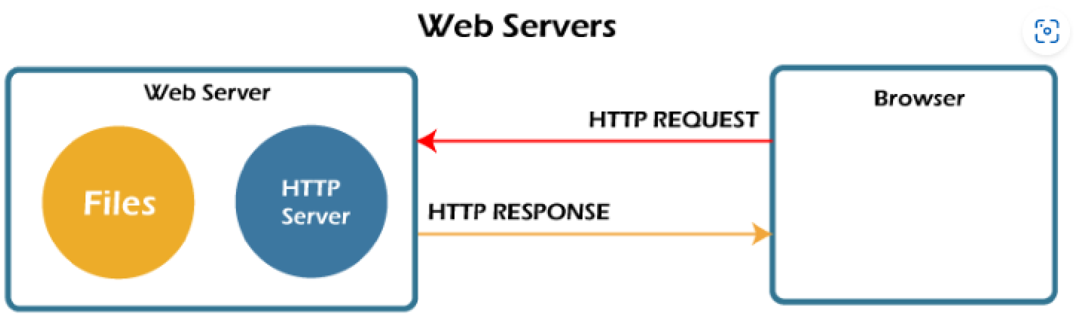


Figure 4.56: Working of Webserver.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

A stream deck using embedded system is developed successfully using one main microcontroller ESP32-Wroom-32D, one main display ILI9488 TFT screen, internal protection battery, battery indicator and type-C charging protection board. The stream deck is functioning well since the TFT screen can precisely display the buttons' various colours while in touch-screen mode. The web server can also modify the shortcut keys as needed. The webserver may also be used to link networks via STA points or APs. Of course, the project's outcomes demonstrate that every electronic circuit on the PCB functions correctly and in line with the intended design. However, the project also properly builds the case for the stream deck. In addition, the type-C protection board may effectively be used to charge the battery. Finally, by physically touching the buttons on the screen, the stream deck may successfully operate the connected device.

In general, the stream deck possesses the capability to be configured to the user interface either through the access point or station point. Additionally, the customizable control panel of the stream deck is designed to execute various tasks. Moreover, the stream deck possesses the capability to function through touch screen and wireless technology. Finally, it is important to note that the stream deck had been developed at a relatively low cost, specifically RM107.38.

5.2 Recommendations

The stream deck may be enhanced in the future, and the web server may be created with a more visually appealing layout. Additionally, due to the stream deck is intended for broadcasters and editors, an internal webcam may be attached to it. The size and amount of buttons on the stream deck may also be changed at the web server to suit the needs of the user. Therefore, a larger TFT panel that can display more than two rows and three columns might be used in place of the current TFT screen.

REFERENCES

Adrian willings,2022. *Elgato Stream Deck: Why this gaming control panel is a must-have for streamers.* [online]. Available at: <https://www.pocket-lint.com/gadgets/news/151906-elagato-stream-deck-best-features >[Accessed 15 Sep 2022].

Alvaro Aguilar,2019. A comparison of battery-charger topologies for portable applications. *Analog Design Journal.*[online]. Available at: <https://www.ti.com/lit/an/slyt769/slyt769.pdf > [Accessed 11 Sep 2022].

Arduino Team, 2021. *The Crumble Deck is a Stream Deck alternative based on an Arduino Due.* [online]. Available at : < https://blog.arduino.cc/2021/06/15/the-crumble-deck-is-a-stream-deck-alternative-based-on-an-arduino-due/ > [Accessed 8 Sep 2022].

Ativya Gupta,Garima, and Harshit Srivastava, *et al*, 2021. DESIGN AND FABRICATION OF 3D PRINTER. *International Journal of Engineering Applied Sciences and Technology, 2021*.6(3), ISSN No. 2455-2143, pp. 328-334.

BABIUCH, M., FOLTÝNEK, P., SMUTNÝ, P. Using the ESP32 microcontroller for data processing. *In Proceedings of 20thInternational Carpathian Control Conference* ICCC´2019. Krakow - Wieliczka; Poland; May 26-29, 2019, pp. 88-93. ISBN:978-172810701-1, DOI: 10.1109/CarpathianCC.2019.8765944.

Douglas.K, Sipiwe.C, and *et al*, 2017. Computer Engineering and Intelligent Systems. *Web Server Performance of Apache and Nginx: A Systematic Literature Review.* 8(2), ISSN No.2222-2863.

DroneBotWorkshop,2020. *Getting started with ESP32*. [online]. Avaiable at: < https://dronebotworkshop.com/esp32-intro/ > [Accessed 11 Sep 2022].

E.Ferraz and G.Fernandez, 2020. *Asian Founders at Work: Stories from the Region's Top Technopreneurs.* [e-book] Makati City, Phhilippines: Apress Publishers. Available at : Google Books < https://books.google.com.my/books?id=4BHGDwAAQBAJ&printsec=frontcover#v=onepage&q&f=false > [Accessed 8 Sep 2022].

Electronics Hub,2018. *TP4056 Lithium Ion Battery Charger*. [online]. Available at: < https://www.electronicshub.org/tp4056-lithium-ion-battery-charger/ > [Accessed 15 Sep 2022].

Elgato,2012.. *History of Elgato*.[online]. Avaiable at :< https://www.elgato.com/en/10-years > [Accessed 15 Sep 2022].

Espressif,2015. *About Espressif.* [online]. Avaiable at: < https://www.espressif.com/en/company/about-us/who-we-are > [Accessed 11 Sep 2022].

Freotech, 2022. *DIY Stream/Hotkey Deck(Using An Arduino!!).* [online]. Available at: <https://www.arduino.coach/diy-stream-hotkey-deckusing-an-arduino.html#forward > [Accessed 8 Sep 2022].

G.E. Blomgren,2000. Current status of lithium ion and lithium polymer secondary batteries. *Fifteenth Annual Battery Conference on Applications and Advances (Cat. No.00TH8490).* pp97-100. https://doi.org/10.1109/BCAA.2000.838386 .

Hargreaves, Eddie, 2007. Elgato ends collaboration with Miglia. [online]. Available at: < https://old.gigaom.com/2007/03/12/elgato-ends-collaboration-with-miglia/ > [Accessed 15 Sep 2022].

Hiro Kawamoto,2011. The Inventors of TFT Active-Matrix LCD Receive the 2011 IEEE Nishizawa Medal. *Journal of Display Technology*. 8(1), pp3-4. https://doi.org/10.1109/JDT.2011.2177740 .

Jennifer Manfrin, 2021. *BEST RECHARGEABLE BATTERIES* .[online]. Available at: <https://bestreviews.com/electronics/chargers/best-rechargeable-batteries > [Accessed 11 Sep 2022].

Konstantinos Kitsakis,Nikos Petrou,Ilias Tanos, and John Kechagias,2016. Design and 3d Printing of a Robotic Arm. *Book of Abstracts/3rd International Conference on Cryptography, Cyber Security and Information Warfare*. At: Athens

LCwiki, 2019. *3.97inch 16BIT Module OTM8009A SKU:MRB3971.* [online]. Available at: <http://www.lcdwiki.com/3.97inch_16BIT_Module_OTM8009A_SKU:MRB3971 >[Accessed 10 Sep 2022].

LCwiki, 2021. *3.5inch SPI Module ILI9488 SKU:MSP3520.* [online]. Avaiable at: < http://www.lcdwiki.com/3.5inch_SPI_Module_ILI9488_SKU:MSP3520 > [Accessed 10 Sep 2022].

LCwiki,2022. *1.8inch Esplora TFT LCD.* [online]. Avaiable at: < http://www.lcdwiki.com/1.8inch_Esplora_TFT_LCD > [Accessed 10 Sep 2022].

Liang Y, Zhao C-Z,Yuan H, *et al*.,2019 . A review of rechargeable batteries forportable electronic devices. *InfoMat.* 2019;1–27. https://doi.org/10.1002/inf2.12000

Lim, K.Y., 1997. Command/Shortcut Keys in WIMP User Interfaces: A Lost Cause? *Human-Computer Interaction*,97, pp.301-306.

Lin,C.C, 2005. Effects of screen luminance combination and text color on visual performance with TFT-LCD. *International Journal of Industrial Ergonomics 35*.pp229- 235. https://doi.org/10.1016/j.ergon.2004.09.002 .

Liu, C.T, 2007. Revolution of the TFT LCD Technology. *JOURNAL OF DISPLAY TECHNOLOGY*. 3(4), pp342-350. https://doi.org/10.1109/JDT.2007.908348.

Loupedeck, 2021. *Meet Loupedeck.* [online]. Avaiable at: < https://loupedeck.com/about-loupedeck/ > [Accessed 10 Sep 2022].

Madan, 2018. ElGateau: A Library for Using the Elgato Stream Deck for Experimental Psychology Research. *Journal of Open Source Software*, 3(31), 1070, https://doi.org/10.21105/joss.01070

Maggie, Tillman, 2022. *Razer made an Elgato Stream Deck-like Stream Controller with Loupedeck.* [online]. Avaiable at: < https://www.pocket-lint.com/gadgets/news/razer/161898-razer-made-an-elgato-stream-deck-like-stream-controller-with-loupedeck > [Accessed 10 Sep 2022].

Maier, A., Sharp, A. and Vagapov. Y, 2017. Comparative analysis and practical implementation of the ESP32 microcontroller module for the Internet of Things. In: *Proc. 7th IEEE Int. Conference on Internet Technologies and Applications ITA-17, Wrexham, UK.*

Melissa Morris and Sabri Tosunoglu,2012. COMPARISON OF RECHARGEABLE BATTERY TECHNOLOGIES. [online]. Available at: < https://www.researchgate.net/publication/281931837_COMPARISON_OF_RECHARGEABLE_BATTERY_TECHNOLOGIES > [Accessed 11 Sep 2022].

Michael Gariffo, 2022. *Razer Teams with Loupedeck for new Stream Controller.* [online]. Avaiable at: < https://www.zdnet.com/home-and-office/home-entertainment/razer-teams-with-loupedeck-for-new-stream-controller/ > [Accessed 10 Sep 2022].

Mohamed FEZARI and Ali Al Dahoud,2018. Introduction to Arduino IDE. *Integrated Development Environment "IDE" For Arduino.* [online]. Available at: <https://www.researchgate.net/profile/Mohamed-Fezari-2/publication/328615543_Integrated_Development_Environment_IDE_For_Arduino/links/5bd8c6d24585150b2b9206df/Integrated-Development-Environment-IDE-For-Arduino.pdf > [Accessed 11 Sep 2022].

Nikola Zaltanov, 2015. Arduino and Open Source Computer Hardware and Software. *IEEE Computer SocietyGrant number.* https://doi.org/10.13140/RG.2.1.1071.7849

NL1_CS, 2019. *Arduino Stream Deck.* [online]. Available at : <https://www.instructables.com/Arduino-Stream-Deck/ > [Accessed 8 Sep 2022].

Nourallah Ghaeminezhad and Mohammad Monfared,2021. Charging control strategies for lithium-ion battery packs: Review and recent developments. *IET Power Electronics.* 15(15), pp349-367. https://doi.org/10.1049/pel2.12219 .

Pengfei Li and Rizwan Bashirullah,2007. A Wireless Power Interface for Rechargeable Battery Operated Medical Implants. *IEEE Transactions on Circuits and Systems II: Express Briefs.* 54(10), pp912-916. https://doi.org/10.1109/TCSII.2007.901613 .

Pravalika, V. and Prasad, C. R., 2019. Internet of Things Based Home Monitoringand Device Control Using Esp32. *International Journal of Recent Technology and Engineering (IJRTE)*. Volume-8, pp58-62. ISSN: 2277-3878.

Razer,2012. *Razer Stream controller*. [online] Available at: < https://www.razer.com/streaming-accessories/razer-stream-controller/RZ20-04350100-R3U1 > [Accessed 4 Sep 2022].

Shopee,2023. *3.7V LiPo Battery (Lithium Polymer) Batteri for MP3 Bluetooth Recorder MP4 GPS Camcorder*. [online] Available at: < https://shopee.com.my/3.7V-LiPo-Battery-(Lithium-Polymer)-Batteri-for-MP3-Bluetooth-Recorder-MP4-GPS-Camcorder-i.126211897.16259884131?sp_atk=9c697ed2-69a7-4279-a3b9-ec3ce5c04620&xptdk=9c697ed2-69a7-4279-a3b9-ec3ce5c04620> [Accessed 24 Apr. 2022].

Tom, Seymour and Ali, Shaheen, 2011. History of Wireless Communication. *Review of Business Information System- Second Quarter*,15(2), pp37-42.

Tom, Warren, 2022. *Razer's Stream Controller takes on the Stream Deck for $269.99 / Elgato has some fresh competition.* [online]. Avaiable at: < https://www.theverge.com/2022/7/14/23215273/razer-stream-controller-release-date-price-specs > [Accessed 10 Sep 2022].

Universal-Solder,2020. *Green LED Battery Gauge for 1-8 Lithium Cells.* [online] Available at: <https://universal-solder.ca/product/green-led-battery-gauge-1-8-cells/ > [Accessed 4 Sep 2022].

Yogendra Singh Parihar, 2019. Internet of Things and Nodemcu. *A review of use of Nodemcu ESP8266 in IoT products*, 6, pp.1085-1088.

Yue Kuo,2013. Thin Film Transistor Technology—Past, Present, and Future. *The Electrochemical Society Interface. 22(55),pp55-61.*

APPENDICES

APPENDIX A: Coding

```
// The pin where the IRQ from the touch screen is connected uses
ESP-style GPIO_NUM_* instead of just pinnumber
#define touchInterruptPin GPIO_NUM_27



// Define the filesystem to be used. For now just SPIFFS.
#define FILESYSTEM SPIFFS

#include <SPIFFS.h>     // Filesystem support header

const char *versionnumber = "1.0";

  /* Version 0.9.18a.
   *
   * Adding ESP32-S3 support
   * Trying to add LitteFS Support
   * Fix #89
   * Fix #90
  */

#include <pgmspace.h> // PROGMEM support header
#include <FS.h>       // Filesystem support header

#include <Preferences.h> // Used to store states before sleep/reboot

#include <TFT_eSPI.h> // The TFT_eSPI library

#if defined(USEUSBHID)

  #include "USB.h"
  #include "USBHIDKeyboard.h"
  #include "Keydefines.h"
  USBHIDKeyboard bleKeyboard;

#else

  #include <BleKeyboard.h> // BleKeyboard is used to communicate
```

```
over BLE
  BleKeyboard bleKeyboard("FeiWeiFeiWei", "Made by me");

    // Checking for BLE Keyboard version
  #ifndef BLE_KEYBOARD_VERSION
    #warning Old BLE Keyboard version detected. Please update.
    #define BLE_KEYBOARD_VERSION "Outdated"
  #endif // !defined(BLE_KEYBOARD_VERSION)

#endif // if

#if defined(USE_NIMBLE)

  #include "NimBLEDevice.h"   // Additional BLE functionaity using
NimBLE
  #include "NimBLEUtils.h"    // Additional BLE functionaity using
NimBLE
  #include "NimBLEBeacon.h"   // Additional BLE functionaity using
NimBLE

#else

  #include "BLEDevice.h"   // Additional BLE functionaity
  #include "BLEUtils.h"    // Additional BLE functionaity
  #include "BLEBeacon.h"   // Additional BLE functionaity

#endif // defined(USE_NIMBLE)

#include "esp_sleep.h"   // Additional BLE functionaity
#include "esp_bt_main.h"   // Additional BLE functionaity
#include "esp_bt_device.h" // Additional BLE functionaity

#include <ArduinoJson.h> // Using ArduinoJson to read and write
config files

#include <WiFi.h> // Wifi support

#include <AsyncTCP.h>          //Async Webserver support header
#include <ESPAsyncWebServer.h> //Async Webserver support header

#include <ESPmDNS.h> // DNS functionality

#ifdef USECAPTOUCH
  #include <Wire.h>
  #include <FT6236.h>
  FT6236 ts = FT6236();
#endif // defined(USECAPTOUCH)

AsyncWebServer webserver(80);

TFT_eSPI tft = TFT_eSPI();

Preferences savedStates;

// This is the file name used to store the calibration data
// You can change this to create new calibration files.
// The FILESYSTEM file name must start with "/".
#define CALIBRATION_FILE "/TouchCalData"

// Set REPEAT_CAL to true instead of false to run calibration
// again, otherwise it will only be done once.
// Repeat calibration if you change the screen rotation.
#define REPEAT_CAL false

// Set the width and height of your screen here:
#define SCREEN_WIDTH 480
#define SCREEN_HEIGHT 320

// Keypad start position, centre of the first button
#define KEY_X SCREEN_WIDTH / 6
#define KEY_Y SCREEN_HEIGHT / 4

// Gaps between buttons
#define KEY_SPACING_X SCREEN_WIDTH / 24
#define KEY_SPACING_Y SCREEN_HEIGHT / 16

// Width and height of a button
#define KEY_W (SCREEN_WIDTH / 3) - KEY_SPACING_X
#define KEY_H (SCREEN_WIDTH / 3) - KEY_SPACING_Y

// Font size multiplier
#define KEY_TEXTSIZE 1

// Text Button Label Font
#define LABEL_FONT &FreeSansBold12pt7b

// placeholder for the pagenumber we are on (0 indicates home)
int pageNum = 0;

// Initial LED brightness
int ledBrightness = 255;

// Every button has a row associated with it
uint8_t rowArray[6] = {0, 0, 0, 1, 1, 1};
// Every button has a column associated with it
uint8_t colArray[6] = {0, 1, 2, 0, 1, 2};

//path to the directory the logo are in ! including leading AND
trailing / !
char logopath[64] = "/logos/";

// templogopath is used to hold the complete path of an image. It is
empty for now.
char templogopath[64] = "";

// Struct to hold the logos per screen
struct Logos
{
  char logo0[32];
  char logo1[32];
  char logo2[32];
  char logo3[32];
  char logo4[32];
  char logo5[32];
};

// Struct Action: 3 actions and 3 values per button
struct Actions
{
  uint8_t action0;
  uint8_t value0;
  char symbol0[64];
  uint8_t action1;
  uint8_t value1;
  char symbol1[64];
  uint8_t action2;
  uint8_t value2;
  char symbol2[64];
};

// Each button has an action struct in it
struct Button
{
  struct Actions actions;
  bool latch;
  char latchlogo[32];
};

// Each menu has 6 buttons
struct Menu
{
  struct Button button0;
  struct Button button1;
  struct Button button2;
  struct Button button3;
  struct Button button4;
  struct Button button5;
};

// Struct to hold the general logos.
struct Generallogos
{
  char homebutton[64];
  char configurator[64];
};

//Struct to hold the general config like colours.
struct Config
{
  uint16_t menuButtonColour;
  uint16_t functionButtonColour;
  uint16_t backgroundColour;
  uint16_t latchedColour;
  bool sleepenable;
  uint16_t sleeptimer;
  bool beep;
  uint8_t modifier1;
  uint8_t modifier2;
  uint8_t modifier3;
  uint16_t helperdelay;
```

```
};

struct Wificonfig
{
  char ssid[64];
  char password[64];
  char wifimode[9];
  char hostname[64];
  uint8_t attempts;
  uint16_t attemptdelay;
};

// Array to hold all the latching statuses
bool islatched[30] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

// Create instances of the structs
Wificonfig wificonfig;

Config generalconfig;

Generallogos generallogo;

Logos screen0;
Logos screen1;
Logos screen2;
Logos screen3;
Logos screen4;
Logos screen5;
Logos screen6;

Menu menu1;
Menu menu2;
Menu menu3;
Menu menu4;
Menu menu5;
Menu menu6;

unsigned long previousMillis = 0;
unsigned long Interval = 0;
bool displayinginfo;
char* jsonfilefail = "";

// Invoke the TFT_eSPI button class and create all the button
objects
TFT_eSPI_Button key[6];

//--------- Internal references ------------
// (this needs to be below all structs etc..)
#include "ScreenHelper.h"
#include "ConfigLoad.h"
#include "DrawHelper.h"
#include "ConfigHelper.h"
#include "UserActions.h"
#include "Action.h"
#include "Webserver.h"
#include "Touch.h"
```

```
//-------------------------------- SETUP ---------------------------
------------------------------------

void setup()
{

  // Use serial port
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println("");

  Serial.println("[INFO]: Loading saved brightness state");
  savedStates.begin("ftd", false);

  ledBrightness = savedStates.getInt("ledBrightness", 255);

  Serial.println("[INFO]: Reading latch stated back from memory:");
  savedStates.getBytes("latched", islatched, sizeof(islatched));

  for(int i = 0; i < sizeof(islatched); i++){

  Serial.print(islatched[i]);

  }
  Serial.println("");

#ifdef USECAPTOUCH
  #ifdef CUSTOM_TOUCH_SDA
    if (!ts.begin(40, CUSTOM_TOUCH_SDA, CUSTOM_TOUCH_SCL))
  #else
    if (!ts.begin(40))
  #endif // defined(CUSTOM_TOUCH_SDA)
  {
    Serial.println("[WARNING]: Unable to start the capacitive
touchscreen.");
  }
  else
  {
    Serial.println("[INFO]: Capacitive touch started!");
  }
#endif // defined(USECAPTOUCH)

  // Setup PWM channel and attach pin bl_pin
  ledcSetup(0, 5000, 8);

#ifdef TFT_BL
  ledcAttachPin(TFT_BL, 0);
#else
  ledcAttachPin(32, 0);

#endif // defined(TFT_BL)
  ledcWrite(0, ledBrightness); // Start @ initial Brightness

  // --------------- Init Display -------------------------

  // Initialise the TFT screen
  tft.init();

  // Set the rotation before we calibrate
  tft.setRotation(1);

  // Clear the screen
  tft.fillScreen(TFT_BLACK);

  esp_sleep_wakeup_cause_t wakeup_reason;
  wakeup_reason = esp_sleep_get_wakeup_cause();


  // -------------- Start filesystem ----------------------

  if (!FILESYSTEM.begin())
  {
    Serial.println("[ERROR]: FILESYSTEM initialisation failed!");
    drawErrorMessage("Failed to init FILESYSTEM! Did you upload the
data folder?");
    while (1)
      yield(); // We stop here
  }
  Serial.println("[INFO]: FILESYSTEM initialised.");

  // Check for free space

  Serial.print("[INFO]: Free Space: ");
  Serial.println(FILESYSTEM.totalBytes() - FILESYSTEM.usedBytes());

  //------------------ Load Wifi Config ----------------------------
------------------

  Serial.println("[INFO]: Loading Wifi Config");
  if (!loadMainConfig())
  {
    Serial.println("[WARNING]: Failed to load WiFi Credentials!");
  }
  else
  {
    Serial.println("[INFO]: WiFi Credentials Loaded");
  }

  // ----------------- Load webserver ---------------------

  handlerSetup();

  // ------------------- Splash screen ------------------

  // If we are woken up we do not need the splash screen
  if (wakeup_reason > 0)
  {
    // But we do draw something to indicate we are waking up
    tft.setTextFont(2);
    tft.println(" Waking up...");
  }
  else
  {
    // Draw a splash screen
    drawBmp("/logos/freetouchdeck_logo.bmp", 0, 0);
    tft.setCursor(1, 3);
    tft.setTextFont(2);
    tft.setTextSize(1);
    tft.setTextColor(TFT_WHITE, TFT_BLACK);
    tft.printf("Loading version %s\n", versionnumber);
    Serial.printf("[INFO]: Loading version %s\n", versionnumber);
  }

// Calibrate the touch screen and retrieve the scaling factors
#ifndef USECAPTOUCH
  Serial.println("[INFO]: Waiting for touch calibration...");
  touch_calibrate();
  Serial.println("[INFO]: Touch calibration completed!");
#endif // !defined(USECAPTOUCH)

  // Let's first check if all the files we need exist
  if (!checkfile("/config/general.json"))
  {
    Serial.println("[ERROR]: /config/general.json not found!");
    while (1)
      yield(); // Stop!
  }

  if (!checkfile("/config/homescreen.json"))
  {
    Serial.println("[ERROR]: /config/homescreen.json not found!");
    while (1)
      yield(); // Stop!
  }

  if (!checkfile("/config/menu1.json"))
  {
    Serial.println("[ERROR]: /config/menu1.json not found!");
    while (1)
      yield(); // Stop!
  }

  if (!checkfile("/config/menu2.json"))
  {
    Serial.println("[ERROR]: /config/menu2.json not found!");
    while (1)
      yield(); // Stop!
  }

  if (!checkfile("/config/menu3.json"))
  {
    Serial.println("[ERROR]: /config/menu3.json not found!");
    while (1)
      yield(); // Stop!
  }

  if (!checkfile("/config/menu4.json"))
  {
    Serial.println("[ERROR]: /config/menu4.json not found!");
    while (1)
      yield(); // Stop!
  }

  if (!checkfile("/config/menu5.json"))
  {
    Serial.println("[ERROR]: /config/menu5.json not found!");
    while (1)
      yield(); // Stop!
  }

  // After checking the config files exist, actually load them
  if(!loadConfig("general")){
    Serial.println("[WARNING]: general.json seems to be
corrupted!");
    Serial.println("[WARNING]: To reset to default type 'reset
general'.");
    jsonfilefail = "general";
    pageNum = 10;
  }

    // Setup PWM channel for Piezo speaker

#ifdef speakerPin
  ledcSetup(2, 500, 8);

if(generalconfig.beep){
  ledcAttachPin(speakerPin, 2);
  ledcWriteTone(2, 600);
  delay(150);
  ledcDetachPin(speakerPin);
  ledcWrite(2, 0);

  ledcAttachPin(speakerPin, 2);
  ledcWriteTone(2, 800);
  delay(150);
  ledcDetachPin(speakerPin);
  ledcWrite(2, 0);

  ledcAttachPin(speakerPin, 2);
  ledcWriteTone(2, 1200);
  delay(150);
  ledcDetachPin(speakerPin);
  ledcWrite(2, 0);
}

#endif // defined(speakerPin)

  if(!loadConfig("homescreen")){
    Serial.println("[WARNING]: homescreen.json seems to be
corrupted!");
    Serial.println("[WARNING]: To reset to default type 'reset
homescreen'.");
    jsonfilefail = "homescreen";
    pageNum = 10;
  }
  if(!loadConfig("menu1")){
    Serial.println("[WARNING]: menu1.json seems to be corrupted!");
    Serial.println("[WARNING]: To reset to default type 'reset
menu1'.");
    jsonfilefail = "menu1";
    pageNum = 10;
  }
  if(!loadConfig("menu2")){
    Serial.println("[WARNING]: menu2.json seems to be corrupted!");
    Serial.println("[WARNING]: To reset to default type 'reset
menu2'.");
    jsonfilefail = "menu2";
    pageNum = 10;
  }
  if(!loadConfig("menu3")){
    Serial.println("[WARNING]: menu3.json seems to be corrupted!");
    Serial.println("[WARNING]: To reset to default type 'reset
menu3'.");
    jsonfilefail = "menu3";
    pageNum = 10;
  }
  if(!loadConfig("menu4")){
    Serial.println("[WARNING]: menu4.json seems to be corrupted!");
    Serial.println("[WARNING]: To reset to default type 'reset
menu4'.");
    jsonfilefail = "menu4";
    pageNum = 10;
  }
  if(!loadConfig("menu5")){
    Serial.println("[WARNING]: menu5.json seems to be corrupted!");
    Serial.println("[WARNING]: To reset to default type 'reset
menu5'.");
    jsonfilefail = "menu5";
    pageNum = 10;
  }
  Serial.println("[INFO]: All configs loaded");



  strcpy(generallogo.homebutton, "/logos/home.bmp");
  strcpy(generallogo.configurator, "/logos/wifi.bmp");
  Serial.println("[INFO]: General logos loaded.");

  // Setup the Font used for plain text
  tft.setFreeFont(LABEL_FONT);

  //------------------BLE Initialization ---------------------------
---------------------------------------------

#if defined(USEUSBHID)

  // initialize control over the keyboard:
  bleKeyboard.begin();
  USB.begin();

#else

  Serial.println("[INFO]: Starting BLE");
  bleKeyboard.begin();

#endif //if defined(USEUSBHID)

  // ---------------- Printing version numbers ---------------------
--------------------------

#if defined(USEUSBHID)
  Serial.println("[INFO]: Using USB Keyboard");
#else
  Serial.print("[INFO]: BLE Keyboard version: ");
  Serial.println(BLE_KEYBOARD_VERSION);
#endif //if defined(USEUSBHID)

  Serial.print("[INFO]: ArduinoJson version: ");
  Serial.println(ARDUINOJSON_VERSION);
  Serial.print("[INFO]: TFT_eSPI version: ");
  Serial.println(TFT_ESPI_VERSION);

  // ---------------- Start the first keypad -------------

  // Draw background
  tft.fillScreen(generalconfig.backgroundColour);

  // Draw keypad
  Serial.println("[INFO]: Drawing keypad");
  drawKeypad();

#ifdef touchInterruptPin
  if (generalconfig.sleepenable)
  {
    pinMode(touchInterruptPin, INPUT_PULLUP);
    Interval = generalconfig.sleeptimer * 60000;
    Serial.println("[INFO]: Sleep enabled.");
    Serial.print("[INFO]: Sleep timer = ");
    Serial.print(generalconfig.sleeptimer);
    Serial.println(" minutes");
    islatched[28] = 1;
  }
#endif // defined(touchInterruptPin)

  Serial.println("[INFO]: Boot completed and successful!");

}

//--------------------- LOOP ---------------------------------------
------------------------------

void loop(void)
{

  // Check if there is data available on the serial input that needs
to be handled.

  if (Serial.available())
  {
    String command = Serial.readStringUntil(' ');

    if (command == "cal")
    {
      FILESYSTEM.remove(CALIBRATION_FILE);
      ESP.restart();
    }
    else if (command == "setssid")
    {

      String value = Serial.readString();
      if (saveWifiSSID(value))
      {
        Serial.printf("[INFO]: Saved new SSID: %s\n",
value.c_str());
        loadMainConfig();
        Serial.println("[INFO]: New configuration loaded");
      }
    }
    else if (command == "setpassword")
    {
      String value = Serial.readString();
      if (saveWifiPW(value))
      {
        Serial.printf("[INFO]: Saved new Password: %s\n",
value.c_str());
        loadMainConfig();
        Serial.println("[INFO]: New configuration loaded");
      }
    }
    else if (command == "setwifimode")
    {
      String value = Serial.readString();
      if (saveWifiMode(value))
      {
        Serial.printf("[INFO]: Saved new WiFi Mode: %s\n",
value.c_str());
        loadMainConfig();
        Serial.println("[INFO]: New configuration loaded");
      }
    }
    else if (command == "restart")
    {
      Serial.println("[WARNING]: Restarting");
      ESP.restart();
    }

    else if (command == "reset")
    {
      String file = Serial.readString();
      Serial.printf("[INFO]: Resetting %s.json now\n",
file.c_str());
      resetconfig(file);
    }

    else if(command == "menu1" && pageNum !=1 && pageNum != 7)
    {
      pageNum = 1;
      drawKeypad();
      Serial.println("Auto Switched to Menu 1");
    }

    else if(command == "menu2" && pageNum !=2 && pageNum != 7)
    {

      pageNum = 2;
      drawKeypad();
      Serial.println("Auto Switched to Menu 2");
    }

    else if(command == "menu3" && pageNum !=3 && pageNum != 7)
    {

      pageNum = 3;
      drawKeypad();
      Serial.println("Auto Switched to Menu 3");
    }

    else if(command == "menu4" && pageNum !=4 && pageNum != 7)
    {

      pageNum = 4;
      drawKeypad();
      Serial.println("Auto Switched to Menu 4");
    }

    else if(command == "menu5" && pageNum !=5 && pageNum != 7)
    {

      pageNum = 5;
      drawKeypad();
      Serial.println("Auto Switched to Menu 5");
    }
  }

  if (pageNum == 7)
  {
      uint16_t t_x = 0, t_y = 0;
      boolean pressed = false;

    // If pageNum = 7, we are in STA or AP mode.
    // We no check if the button is pressed, and if so restart.
#ifdef USECAPTOUCH
    if (ts.touched())
    {

      // Retrieve a point
      TS_Point p = ts.getPoint();

      //Flip things around so it matches our screen rotation
      p.x = map(p.x, 0, 320, 320, 0);
      t_y = p.x;
      t_x = p.y;
      pressed = true;
    }

#else

    pressed = tft.getTouch(&t_x, &t_y);

#endif // defined(USECAPTOUCH)

    if (pressed)
    {
      // If pressed check if the touch falls within the restart
button
      // drawSingleButton(140, 180, 200, 80,
generalconfig.menuButtonColour, TFT_WHITE, "Restart");
      if (t_x > 140 && t_x < 340){
        if (t_y > 180 && t_y < 260){
          // Touch falls within the boundaries of our button so we
restart
          Serial.println("[WARNING]: Restarting");
          ESP.restart();
        }
      }

    }

  }
  else if (pageNum == 8)
  {

    if (!displayinginfo)
    {
      printinfo();
    }

    uint16_t t_x = 0, t_y = 0;

    //At the beginning of a new loop, make sure we do not use last
loop's touch.
    boolean pressed = false;

#ifdef USECAPTOUCH
    if (ts.touched())
    {

      // Retrieve a point
      TS_Point p = ts.getPoint();

      //Flip things around so it matches our screen rotation
      p.x = map(p.x, 0, 320, 320, 0);
      t_y = p.x;
      t_x = p.y;

      pressed = true;
    }

#else
```

```
    pressed = tft.getTouch(&t_x, &t_y);

#endif // defined(USECAPTOUCH)

    if (pressed)
    {
      displayinginfo = false;
      pageNum = 6;
      tft.fillScreen(generalconfig.backgroundColour);
      drawKeypad();
    }
  }
  else if (pageNum == 9)
  {

    // We were unable to connect to WiFi. Waiting for touch to get
back to the settings menu.
    uint16_t t_x = 0, t_y = 0;

    //At the beginning of a new loop, make sure we do not use last
loop's touch.
    boolean pressed = false;

#ifdef USECAPTOUCH
    if (ts.touched())
    {

      // Retrieve a point
      TS_Point p = ts.getPoint();

      //Flip things around so it matches our screen rotation
      p.x = map(p.x, 0, 320, 320, 0);
      t_y = p.x;
      t_x = p.y;

      pressed = true;
    }

#else

    pressed = tft.getTouch(&t_x, &t_y);

#endif // defined(USECAPTOUCH)

    if (pressed)
    {
      // Return to Settings page
      displayinginfo = false;
      pageNum = 6;
      tft.fillScreen(generalconfig.backgroundColour);
      drawKeypad();
    }
  }
  else if (pageNum == 10)
  {
    // A JSON file failed to load. We are drawing an error message.
And waiting for a touch.
    uint16_t t_x = 0, t_y = 0;

    //At the beginning of a new loop, make sure we do not use last
loop's touch.
    boolean pressed = false;

#ifdef USECAPTOUCH
    if (ts.touched())
    {

      // Retrieve a point
      TS_Point p = ts.getPoint();

      //Flip things around so it matches our screen rotation
      p.x = map(p.x, 0, 320, 320, 0);
      t_y = p.x;
      t_x = p.y;

      pressed = true;
    }

#else

    pressed = tft.getTouch(&t_x, &t_y);

#endif // defined(USECAPTOUCH)

    if (pressed)
    {
      // Load home screen
      displayinginfo = false;
      pageNum = 0;
      tft.fillScreen(generalconfig.backgroundColour);
      drawKeypad();
    }
  }
  else
  {

    // Check if sleep is enabled and if our timer has ended.

#ifdef touchInterruptPin
    if (generalconfig.sleepenable)
    {
      if (millis() > previousMillis + Interval)
      {

        // The timer has ended and we are going to sleep  .
        tft.fillScreen(TFT_BLACK);
        Serial.println("[INFO]: Going to sleep.");
#ifdef speakerPin
        if(generalconfig.beep){
        ledcAttachPin(speakerPin, 2);
        ledcWriteTone(2, 1200);
        delay(150);
        ledcDetachPin(speakerPin);
        ledcWrite(2, 0);

        ledcAttachPin(speakerPin, 2);
        ledcWriteTone(2, 800);
        delay(150);
        ledcDetachPin(speakerPin);
        ledcWrite(2, 0);

        ledcAttachPin(speakerPin, 2);
        ledcWriteTone(2, 600);
        delay(150);
        ledcDetachPin(speakerPin);
        ledcWrite(2, 0);
        }
#endif // defined(speakerPin)
        Serial.println("[INFO]: Saving latched states");

//        You could uncomment this to see the latch stated before
going to sleep
//        for(int i = 0; i < sizeof(islatched); i++){
//
//        Serial.print(islatched[i]);
//
//        }
//        Serial.println("");

        savedStates.putBytes("latched", &islatched,
sizeof(islatched));
        esp_sleep_enable_ext0_wakeup(touchInterruptPin, 0);
        esp_deep_sleep_start();
      }
    }
#endif // defined(touchInterruptPin)

    // Touch coordinates are stored here
    uint16_t t_x = 0, t_y = 0;

    //At the beginning of a new loop, make sure we do not use last
loop's touch.
    boolean pressed = false;

#ifdef USECAPTOUCH
    if (ts.touched())
    {

      // Retrieve a point
      TS_Point p = ts.getPoint();

      //Flip things around so it matches our screen rotation
      p.x = map(p.x, 0, 320, 320, 0);
      t_y = p.x;
      t_x = p.y;

      pressed = true;
    }
```

```
#else

    pressed = tft.getTouch(&t_x, &t_y);

#endif // defined(USECAPTOUCH)

    // Check if the X and Y coordinates of the touch are within one
of our buttons
    for (uint8_t b = 0; b < 6; b++)
    {
      if (pressed && key[b].contains(t_x, t_y))
      {
        key[b].press(true); // tell the button it is pressed

        // After receiving a valid touch reset the sleep timer
        previousMillis = millis();
      }
      else
      {
        key[b].press(false); // tell the button it is NOT pressed
      }
    }

    // Check if any key has changed state
    for (uint8_t b = 0; b < 6; b++)
    {
      if (key[b].justReleased())
      {

        // Draw normal button space (non inverted)

        int col, row;

        if (b == 0)
        {
          col = 0;
          row = 0;
        }
        else if (b == 1)
        {
          col = 1;
          row = 0;
        }
        else if (b == 2)
        {
          col = 2;
          row = 0;
        }
        else if (b == 3)
        {
          col = 0;
          row = 1;
        }
        else if (b == 4)
        {
          col = 1;
          row = 1;
        }
        else if (b == 5)
        {
          col = 2;
          row = 1;
        }

        int index;

        if (pageNum == 2)
        {
          index = b + 5;
        }
        else if (pageNum == 3)
        {
          index = b + 10;
        }
        else if (pageNum == 4)
        {
          index = b + 15;
        }
        else if (pageNum == 5)
        {
          index = b + 20;
        }
        else if (pageNum == 6)
        {
          index = b + 25;
        }
        else
        {
          index = b;
        }

        uint16_t buttonBG;
        bool drawTransparent;

        uint16_t imageBGColor;
        if (islatched[index] && b < 5)
        {
          imageBGColor = getLatchImageBG(b);
        }
        else
        {
          imageBGColor = getImageBG(b);
        }

        if (imageBGColor > 0)
        {
          buttonBG = imageBGColor;
          drawTransparent = false;
        }
        else
        {
          if (pageNum == 0)
          {
            buttonBG = generalconfig.menuButtonColour;
            drawTransparent = true;
          }
          else
          {
            if (pageNum == 6 && b == 5)
            {
              buttonBG = generalconfig.menuButtonColour;
              drawTransparent = true;
            }
            else
            {
              buttonBG = generalconfig.functionButtonColour;
              drawTransparent = true;
            }
          }
        }
        tft.setFreeFont(LABEL_FONT);
        key[b].initButton(&tft, KEY_X + col * (KEY_W +
KEY_SPACING_X),
                          KEY_Y + row * (KEY_H + KEY_SPACING_Y), //
x, y, w, h, outline, fill, text
                          KEY_W, KEY_H, TFT_WHITE, buttonBG,
TFT_WHITE,
                          "", KEY_TEXTSIZE);
        key[b].drawButton();

        // After drawing the button outline we call this to draw a
logo.
        if (islatched[index] && b < 5)
        {
          drawlogo(b, col, row, drawTransparent, true);
        }
        else
        {
          drawlogo(b, col, row, drawTransparent, false);
        }
      }

      if (key[b].justPressed())
      {

        // Beep
        #ifdef speakerPin
        if(generalconfig.beep){
          ledcAttachPin(speakerPin, 2);
          ledcWriteTone(2, 600);
          delay(50);
          ledcDetachPin(speakerPin);
          ledcWrite(2, 0);
        }
        #endif

        int col, row;

        if (b == 0)
        {
          col = 0;
          row = 0;
        }
        else if (b == 1)
        {
          col = 1;
          row = 0;
        }
        else if (b == 2)
        {
          col = 2;
          row = 0;
        }
        else if (b == 3)
        {
          col = 0;
          row = 1;
        }
        else if (b == 4)
        {
          col = 1;
          row = 1;
        }
        else if (b == 5)
        {
          col = 2;
          row = 1;
        }

        tft.setFreeFont(LABEL_FONT);
        key[b].initButton(&tft, KEY_X + col * (KEY_W +
KEY_SPACING_X),
                          KEY_Y + row * (KEY_H + KEY_SPACING_Y), //
x, y, w, h, outline, fill, text
                          KEY_W, KEY_H, TFT_WHITE, TFT_WHITE,
TFT_WHITE,
                          "", KEY_TEXTSIZE);
        key[b].drawButton();

        //---------------------------------------- Button press
handeling --------------------------------------------------

        if (pageNum == 0) //Home menu
        {
          if (b == 0) // Button 0
          {
            pageNum = 1;
            drawKeypad();
          }
          else if (b == 1) // Button 1
          {
            pageNum = 2;
            drawKeypad();
          }
          else if (b == 2) // Button 2
          {
            pageNum = 3;
            drawKeypad();
          }
          else if (b == 3) // Button 3
          {
            pageNum = 4;
            drawKeypad();
          }
          else if (b == 4) // Button 4
          {
            pageNum = 5;
            drawKeypad();
          }
          else if (b == 5) // Button 5
          {
            pageNum = 6;
            drawKeypad();
          }
        }

        else if (pageNum == 1) // Menu 1
        {
          if (b == 0) // Button 0
          {
            bleKeyboardAction(menu1.button0.actions.action0,
menu1.button0.actions.value0, menu1.button0.actions.symbol0);
            bleKeyboardAction(menu1.button0.actions.action1,
menu1.button0.actions.value1, menu1.button0.actions.symbol1);
            bleKeyboardAction(menu1.button0.actions.action2,
menu1.button0.actions.value2, menu1.button0.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu1.button0.latch)
            {
              if (islatched[0])
              {
                islatched[0] = 0;
              }
              else
              {
                islatched[0] = 1;
              }
            }
          }
          else if (b == 1) // Button 1
          {
            bleKeyboardAction(menu1.button1.actions.action0,
menu1.button1.actions.value0, menu1.button1.actions.symbol0);
            bleKeyboardAction(menu1.button1.actions.action1,
menu1.button1.actions.value1, menu1.button1.actions.symbol1);
            bleKeyboardAction(menu1.button1.actions.action2,
menu1.button1.actions.value2, menu1.button1.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu1.button1.latch)
            {
              if (islatched[1])
              {
                islatched[1] = 0;
              }
              else
              {
                islatched[1] = 1;
              }
            }
          }
          else if (b == 2) // Button 2
          {
            bleKeyboardAction(menu1.button2.actions.action0,
menu1.button2.actions.value0, menu1.button2.actions.symbol0);
            bleKeyboardAction(menu1.button2.actions.action1,
menu1.button2.actions.value1, menu1.button2.actions.symbol1);
            bleKeyboardAction(menu1.button2.actions.action2,
menu1.button2.actions.value2, menu1.button2.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu1.button2.latch)
            {
              if (islatched[2])
              {
                islatched[2] = 0;
              }
              else
              {
                islatched[2] = 1;
              }
            }
          }
          else if (b == 3) // Button 3
          {
            bleKeyboardAction(menu1.button3.actions.action0,
menu1.button3.actions.value0, menu1.button3.actions.symbol0);
            bleKeyboardAction(menu1.button3.actions.action1,
menu1.button3.actions.value1, menu1.button3.actions.symbol1);
            bleKeyboardAction(menu1.button3.actions.action2,
menu1.button3.actions.value2, menu1.button3.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu1.button3.latch)
            {
              if (islatched[3])
              {
                islatched[3] = 0;
              }
              else
              {
                islatched[3] = 1;
              }
            }
          }
          else if (b == 4) // Button 4
          {
            bleKeyboardAction(menu1.button4.actions.action0,
menu1.button4.actions.value0, menu1.button4.actions.symbol0);
            bleKeyboardAction(menu1.button4.actions.action1,
menu1.button4.actions.value1, menu1.button4.actions.symbol1);
            bleKeyboardAction(menu1.button4.actions.action2,
menu1.button4.actions.value2, menu1.button4.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu1.button4.latch)
            {
              if (islatched[4])
              {
                islatched[4] = 0;
              }
              else
              {
                islatched[4] = 1;
              }
            }
          }
          else if (b == 5) // Button 5 / Back home
          {
            pageNum = 0;
            drawKeypad();
          }
        }

        else if (pageNum == 2) // Menu 2
        {
          if (b == 0) // Button 0
          {
            bleKeyboardAction(menu2.button0.actions.action0,
menu2.button0.actions.value0, menu2.button0.actions.symbol0);
            bleKeyboardAction(menu2.button0.actions.action1,
menu2.button0.actions.value1, menu2.button0.actions.symbol1);
            bleKeyboardAction(menu2.button0.actions.action2,
menu2.button0.actions.value2, menu2.button0.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu2.button0.latch)
            {
              if (islatched[5])
              {
                islatched[5] = 0;
              }
              else
              {
                islatched[5] = 1;
              }
            }
          }
          else if (b == 1) // Button 1
          {
            bleKeyboardAction(menu2.button1.actions.action0,
menu2.button1.actions.value0, menu2.button1.actions.symbol0);
            bleKeyboardAction(menu2.button1.actions.action1,
menu2.button1.actions.value1, menu2.button1.actions.symbol1);
            bleKeyboardAction(menu2.button1.actions.action2,
menu2.button1.actions.value2, menu2.button1.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu2.button1.latch)
            {
              if (islatched[6])
              {
                islatched[6] = 0;
              }
              else
              {
                islatched[6] = 1;
              }
            }
          }
          else if (b == 2) // Button 2
          {
            bleKeyboardAction(menu2.button2.actions.action0,
menu2.button2.actions.value0, menu2.button2.actions.symbol0);
            bleKeyboardAction(menu2.button2.actions.action1,
menu2.button2.actions.value1, menu2.button2.actions.symbol1);
            bleKeyboardAction(menu2.button2.actions.action2,
menu2.button2.actions.value2, menu2.button2.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu2.button2.latch)
            {
              if (islatched[7])
              {
                islatched[7] = 0;
              }
              else
              {
                islatched[7] = 1;
              }
            }
          }
          else if (b == 3) // Button 3
          {
            bleKeyboardAction(menu2.button3.actions.action0,
menu2.button3.actions.value0, menu2.button3.actions.symbol0);
            bleKeyboardAction(menu2.button3.actions.action1,
menu2.button3.actions.value1, menu2.button3.actions.symbol1);
            bleKeyboardAction(menu2.button3.actions.action2,
menu2.button3.actions.value2, menu2.button3.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu2.button3.latch)
            {
              if (islatched[8])
              {
                islatched[8] = 0;
              }
              else
              {
                islatched[8] = 1;
              }
            }
          }
          else if (b == 4) // Button 4
          {
            bleKeyboardAction(menu2.button4.actions.action0,
menu2.button4.actions.value0, menu2.button4.actions.symbol0);
            bleKeyboardAction(menu2.button4.actions.action1,
menu2.button4.actions.value1, menu2.button4.actions.symbol1);
            bleKeyboardAction(menu2.button4.actions.action2,
menu2.button4.actions.value2, menu2.button4.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu2.button4.latch)
            {
              if (islatched[9])
              {
                islatched[9] = 0;
              }
              else
              {
                islatched[9] = 1;
              }
            }
          }
          else if (b == 5) // Button 5 / Back home
          {
            pageNum = 0;
            drawKeypad();
          }
        }

        else if (pageNum == 3) // Menu 3
        {
          if (b == 0) // Button 0
          {
            bleKeyboardAction(menu3.button0.actions.action0,
menu3.button0.actions.value0, menu3.button0.actions.symbol0);
            bleKeyboardAction(menu3.button0.actions.action1,
menu3.button0.actions.value1, menu3.button0.actions.symbol1);
            bleKeyboardAction(menu3.button0.actions.action2,
menu3.button0.actions.value2, menu3.button0.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu3.button0.latch)
            {
              if (islatched[10])
              {
                islatched[10] = 0;
              }
              else
              {
                islatched[10] = 1;
              }
            }
          }
          else if (b == 1) // Button 1
          {
            bleKeyboardAction(menu3.button1.actions.action0,
menu3.button1.actions.value0, menu3.button1.actions.symbol0);
            bleKeyboardAction(menu3.button1.actions.action1,
menu3.button1.actions.value1, menu3.button1.actions.symbol1);
            bleKeyboardAction(menu3.button1.actions.action2,
menu3.button1.actions.value2, menu3.button1.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu3.button1.latch)
            {
              if (islatched[11])
              {
                islatched[11] = 0;
              }
              else
              {
                islatched[11] = 1;
              }
            }
          }
          else if (b == 2) // Button 2
          {
            bleKeyboardAction(menu3.button2.actions.action0,
menu3.button2.actions.value0, menu3.button2.actions.symbol0);
            bleKeyboardAction(menu3.button2.actions.action1,
menu3.button2.actions.value1, menu3.button2.actions.symbol1);
            bleKeyboardAction(menu3.button2.actions.action2,
menu3.button2.actions.value2, menu3.button2.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu3.button2.latch)
            {
              if (islatched[12])
              {
                islatched[12] = 0;
              }
              else
              {
                islatched[12] = 1;
              }
            }
          }
          else if (b == 3) // Button 3
          {
            bleKeyboardAction(menu3.button3.actions.action0,
menu3.button3.actions.value0, menu3.button3.actions.symbol0);
            bleKeyboardAction(menu3.button3.actions.action1,
menu3.button3.actions.value1, menu3.button3.actions.symbol1);
            bleKeyboardAction(menu3.button3.actions.action2,
menu3.button3.actions.value2, menu3.button3.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu3.button3.latch)
            {
              if (islatched[13])
              {
                islatched[13] = 0;
              }
              else
              {
                islatched[13] = 1;
              }
            }
          }
          else if (b == 4) // Button 4
          {
            bleKeyboardAction(menu3.button4.actions.action0,
menu3.button4.actions.value0, menu3.button4.actions.symbol0);
            bleKeyboardAction(menu3.button4.actions.action1,
menu3.button4.actions.value1, menu3.button4.actions.symbol1);
            bleKeyboardAction(menu3.button4.actions.action2,
menu3.button4.actions.value2, menu3.button4.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu3.button4.latch)
            {
              if (islatched[14])
              {
                islatched[14] = 0;
              }
              else
              {
                islatched[14] = 1;
              }
            }
          }
          else if (b == 5) // Button 5 / Back home
          {
            pageNum = 0;
            drawKeypad();
          }
        }

        else if (pageNum == 4) // Menu 4
        {
          if (b == 0) // Button 0
          {
            bleKeyboardAction(menu4.button0.actions.action0,
menu4.button0.actions.value0, menu4.button0.actions.symbol0);
            bleKeyboardAction(menu4.button0.actions.action1,
menu4.button0.actions.value1, menu4.button0.actions.symbol1);
            bleKeyboardAction(menu4.button0.actions.action2,
menu4.button0.actions.value2, menu4.button0.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu4.button0.latch)
            {
              if (islatched[15])
              {
                islatched[15] = 0;
              }
              else
              {
                islatched[15] = 1;
              }
            }
          }
          else if (b == 1) // Button 1
          {
            bleKeyboardAction(menu4.button1.actions.action0,
menu4.button1.actions.value0, menu4.button1.actions.symbol0);
            bleKeyboardAction(menu4.button1.actions.action1,
menu4.button1.actions.value1, menu4.button1.actions.symbol1);
            bleKeyboardAction(menu4.button1.actions.action2,
menu4.button1.actions.value2, menu4.button1.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu4.button1.latch)
            {
              if (islatched[16])
              {
                islatched[16] = 0;
              }
              else
              {
                islatched[16] = 1;
              }
            }
          }
          else if (b == 2) // Button 2
          {
            bleKeyboardAction(menu4.button2.actions.action0,
menu4.button2.actions.value0, menu4.button2.actions.symbol0);
            bleKeyboardAction(menu4.button2.actions.action1,
menu4.button2.actions.value1, menu4.button2.actions.symbol1);
            bleKeyboardAction(menu4.button2.actions.action2,
menu4.button2.actions.value2, menu4.button2.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu4.button2.latch)
            {
              if (islatched[17])
              {
                islatched[17] = 0;
              }
              else
              {
                islatched[17] = 1;
              }
            }
          }
          else if (b == 3) // Button 3
          {
            bleKeyboardAction(menu4.button3.actions.action0,
menu4.button3.actions.value0, menu4.button3.actions.symbol0);
            bleKeyboardAction(menu4.button3.actions.action1,
menu4.button3.actions.value1, menu4.button3.actions.symbol1);
            bleKeyboardAction(menu4.button3.actions.action2,
menu4.button3.actions.value2, menu4.button3.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu4.button3.latch)
            {
              if (islatched[18])
              {
                islatched[18] = 0;
              }
              else
              {
                islatched[18] = 1;
              }
            }
          }
          else if (b == 4) // Button 4
          {
            bleKeyboardAction(menu4.button4.actions.action0,
menu4.button4.actions.value0, menu4.button4.actions.symbol0);
            bleKeyboardAction(menu4.button4.actions.action1,
menu4.button4.actions.value1, menu4.button4.actions.symbol1);
            bleKeyboardAction(menu4.button4.actions.action2,
menu4.button4.actions.value2, menu4.button4.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu4.button4.latch)
            {
              if (islatched[19])
              {
                islatched[19] = 0;
              }
              else
              {
                islatched[19] = 1;
              }
            }
          }
          else if (b == 5) // Button 5 / Back home
          {
            pageNum = 0;
            drawKeypad();
          }
        }

        else if (pageNum == 5) // Menu 5
        {
          if (b == 0) // Button 0
          {
            bleKeyboardAction(menu5.button0.actions.action0,
menu5.button0.actions.value0, menu5.button0.actions.symbol0);
            bleKeyboardAction(menu5.button0.actions.action1,
menu5.button0.actions.value1, menu5.button0.actions.symbol1);
            bleKeyboardAction(menu5.button0.actions.action2,
menu5.button0.actions.value2, menu5.button0.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu5.button0.latch)
            {
              if (islatched[20])
              {
                islatched[20] = 0;
              }
              else
              {
                islatched[20] = 1;
              }
            }
          }
          else if (b == 1) // Button 1
          {
            bleKeyboardAction(menu5.button1.actions.action0,
menu5.button1.actions.value0, menu5.button1.actions.symbol0);
            bleKeyboardAction(menu5.button1.actions.action1,
menu5.button1.actions.value1, menu5.button1.actions.symbol1);
            bleKeyboardAction(menu5.button1.actions.action2,
menu5.button1.actions.value2, menu5.button1.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu5.button1.latch)
            {
              if (islatched[21])
              {
                islatched[21] = 0;
              }
              else
              {
                islatched[21] = 1;
              }
            }
          }
          else if (b == 2) // Button 2
          {
            bleKeyboardAction(menu5.button2.actions.action0,
menu5.button2.actions.value0, menu5.button2.actions.symbol0);
            bleKeyboardAction(menu5.button2.actions.action1,
menu5.button2.actions.value1, menu5.button2.actions.symbol1);
            bleKeyboardAction(menu5.button2.actions.action2,
menu5.button2.actions.value2, menu5.button2.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu5.button2.latch)
            {
              if (islatched[22])
              {
                islatched[22] = 0;
              }
              else
              {
                islatched[22] = 1;
              }
            }
          }
          else if (b == 3) // Button 3
          {
            bleKeyboardAction(menu5.button3.actions.action0,
menu5.button3.actions.value0, menu5.button3.actions.symbol0);
            bleKeyboardAction(menu5.button3.actions.action1,
menu5.button3.actions.value1, menu5.button3.actions.symbol1);
            bleKeyboardAction(menu5.button3.actions.action2,
menu5.button3.actions.value2, menu5.button3.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu5.button3.latch)
            {
              if (islatched[23])
              {
                islatched[23] = 0;
              }
              else
              {
                islatched[23] = 1;
              }
            }
          }
          else if (b == 4) // Button 4
          {
            bleKeyboardAction(menu5.button4.actions.action0,
menu5.button4.actions.value0, menu5.button4.actions.symbol0);
            bleKeyboardAction(menu5.button4.actions.action1,
menu5.button4.actions.value1, menu5.button4.actions.symbol1);
            bleKeyboardAction(menu5.button4.actions.action2,
menu5.button4.actions.value2, menu5.button4.actions.symbol2);
            bleKeyboard.releaseAll();
            if (menu5.button4.latch)
```

```
        {
          if (islatched[24])
          {
            islatched[24] = 0;
          }
          else
          {
            islatched[24] = 1;
          }
        }
      }
      else if (b == 5) // Button 5 / Back home
      {
        pageNum = 0;
        drawKeypad();
      }
    }

    else if (pageNum == 6) // Settings page
    {
      if (b == 0) // Button 0
      {
        bleKeyboardAction(11, 1, 0);
      }
      else if (b == 1) // Button 1
      {
        bleKeyboardAction(11, 2, 0);
      }
      else if (b == 2) // Button 2
      {
        bleKeyboardAction(11, 3, 0);
      }
      else if (b == 3) // Button 3
      {
        bleKeyboardAction(11, 4, 0);
        if (islatched[28])
        {
          islatched[28] = 0;
        }
        else
        {
          islatched[28] = 1;
        }
      }
      else if (b == 4) // Button 4
      {
        pageNum = 8;
        drawKeypad();
      }
      else if (b == 5)
      {
        pageNum = 0;
        drawKeypad();
      }
    }

    delay(10); // UI debouncing
```

```
            }
        }
    }
}
```