Final Year Project (2024 – 2025)

ELEC/CPEG Model Final Report*

Smart Light Switch

Project ID: AB04a-20

Supervisor: Professor A

Author (Student ID): CHAN Tai Man (12345678), CHAN Siu Ming (23456789)

Date: 8-Jan-2021

Main Objective for ELSA

This project aims to design and build a small wifi-connected external switching device. We name it the External Light Switch Assistant (ELSA). It is a device that can be fixed to an existing switch in the home and can be operated using a mobile phone. It allows users to turn the switch on and off remotely and manually, whether inside the home or within wifi range. It provides scheduled and timed lighting and other IoT-related features.

Objective Statements

1. To create the switch mechanism that can switch on and off a standard existing electrical switch.
2. To program the microprocessor to link ELSA to a web server and control ELSA through a web browser.
3. Set up a configuration system, implement other features, and create a user interface to operate ELSA.

*Note that this report has been modified from its original version to reflect the current Final Report format, and some of the content has been changed.

Contents

ABSTRACT iv
SECTION 1—INTRODUCTION 1
1.1 Background and Engineering Problem 1
1.2 Objectives 1
1.3 Literature Review of Existing Solutions 1
SECTION 2— METHODOLOGY 3
2.1 Overview of ELSA 3
2.1.1 System Description 3
2.1.2 ELSA Diagram 4
2.1.3 Components List 4
2.1.4 ECE Knowledge 5
2.2 Objective Statement Execution 5
2.2.1 Switching mechanism and external device 5
2.2.2 Set up Hardware and Software 14
2.2.3 Set up a configuration system and implement other features 27
2.3 ELSA Evaluation & Discussion 39
SECTION 3—CONCLUSION 40
REFERENCES 42
APPENDICES 43
Appendix A – Final Project Plan 43
Appendix B – Budget 45
Appendix C – Meeting Minutes 46
Appendix D – Group Members' Contribution 48
Appendix E – Deviation(s) from the proposal and supporting reason(s) 50
Appendix F – Monthly Reports 51
Appendix G – Source code for 2.2.2 Task 2 55
Appendix H – Example file for 2.2.3 Task 2 57

List of Illustrations

List of Figures

Figure 1. Schematic of the whole ELSA system 4
Figure 2a. Toggle switch 5
Figure 2b. European style rocker switch 6
Figure 2c. Australian rocker switch 6
Figure 3a. Electromagnet design sketch 7
Figure 3bi. Motor design sketch 1 7
Figure 3bii. Motor design sketch 2 7
Figure 4a. Pus h-pull electromagnet 8
Figure 4b. Holding electromagnet 8
Figure 5. Model rendered by Fusion 360 9
Figure 6. Extra description of the offset applied to the hole. 10
Figure 7a. Model rendered by Fusion 360 10
Figure 7b. Printed component 11
Figure 8a. Top view of the stuck frame 11
Figure 8b. Side view of the stuck frame 11
Figure 9a. Side view of the new model 13
Figure 9b. Side view from another angle 13
Figure 9c. Top view of the new model 13
Figure 10. Overview of ELSA Hardware System 14
Figure 11. Schematic Diagram of ELSA Hardware System 16
Figure 12. Prototype of ELSA 16
Figure 13. Diagrammatic sketch of GY25z roll value variations 21
Figure 14a.Orientation 1 21
Figure 14b. Orientation 2 21
Figure 15. Flowchart of function calibrate() 23
Figure 16. Flowchart of function on() and off() 24
Figure 17. Arduino IDE Console of ESP32 25
Figure 18. View of Webpage 26
Figure 19. A flowchart showing the rough idea of the backend boot process 28
Figure 20a. The view on a wider screen 30
Figure 20b. The view on a smaller screen 31
Figure 21. Sequence UML Diagram showing the communication process 35
Figure 22. The error message when getting information from different domains 35
Figure 23. Sequence UML Diagram showing how messages are sent and processed 36
Figure 24a. Message on serial monitor of the main device 36
Figure 24b. Message on serial monitor of the client. 36

List of Tables

Table 1. List of Specifications 4
Table 2. Switch actuation mechanism comparison 7
Table 3. Switching times 12
Table 4. Meaning of bits. 18
Table 5. Format of 11-byte data frame. 18
Table 6. ELSA orientation 22
Table 7. Logic of AIN1 and AIN2. 24
Table 8. Result of testing the code. 32
Table 9. ELSA schedule. 43
Table 10. ELSA schedule continued. 44
Table 11. Budget. 45

ABSTRACT

To attract more consumers to apply smart home technology, this project aimed to develop an IoT lighting device that can install easily by attaching to existing switches and control lights remotely and automatically. We named this device the External Light Switch Assistant, ELSA. We applied 3D modeling and printing technology to build the switching mechanism and its frame to allow ELSA to attach to existing switches, C++, JavaScript programming language with some open-source framework to construct ELSA's control system and web interface. As a result, two mechanisms are designed for two standard switch types in Hong Kong. Users can control the device through a webpage created by ELSA. The webpage can configure the ELSA's WiFi settings, remote control the switch, and arrange scheduled tasks. More development is needed to make this model apply in real situations.

Commented [1]: This is an overview of your project report.

SECTION 1—INTRODUCTION

1.1 Background and Engineering Problem

The Internet of things (IoT) is the network between devices and systems; they can exchange data with each other through the internet. It is widely used in different aspects and home automation is one of the examples. It refers to remote and automatic control of electronic appliances via mobile devices. These devices should be programmable and can access a communication network [1]. Smart homes have become increasingly popular because they provide domestic convenience and safety. A few clicks on a smartphone in the user's hand can switch off all home appliances, saving them time to switch them off one by one in different rooms. Even if people forget to turn off an appliance before leaving the home, they can still turn it off outdoors, which not only saves the time needed to return home, saves a lot of wasted energy, and prevents accidents caused by over-operated devices such as irons and heaters. Combining home assistant devices, customized scheduling and sensors improve the functionality of home appliances.

The most common type of home automation is smart lighting. Currently, there are several smart lighting products available on the market. However, these products are quite expensive, usually costing from $20 to $70 us dollars each depending on the functionality. Also, they have limitations in different aspects. Some of them have complicated installation procedures, users have to remove the existing light switch and do wiring connections, which can be dangerous, or users need to pay extra money to find corresponding technicians for help to install. Some of them lack compatibility so they cannot be used with existing lighting hardware. Therefore, they may not be attractive to people who have never used them but want to try smart lighting. This discourages people from utilizing this useful technology. There is a need to develop a smart lighting device that is cheap, easy to install without changing the existing home systems, and fits most lighting systems.

Designing a smart home device to control lightingthat satisfies these requirements is vital. A cheap and basic device that attaches to existing switches and has simple installation procedure that ensures people do not need any knowledge about electronics can solve these problems. This can help more non-technical persons with a small budget to have access to home automation or IoT. A smart lighting device that is cheap, easy to install without changing the existing home systems, and fits most lighting systems will ensure smart lighting is more ubiquitous and that will help save energy and the environment.

Commented [2]: Include an introduction into the broader area of interest (your reasearch area).

Commented [3]: Narrow the focus on the area of interest.

Commented [4]: Identify the problem/s in this area of interest.

Commented [5]: Communicate the impact of solving the problem/s you identified.

1.2 Objectives

This project aims to design and build a small wifi-connected externalsmart switch that has high functionality while improving compatibility, and at the same time makes smart lighting more user-friendly and attractive to users. We name it the External Light Switch Assistant (ELSA). It is a device that can be fixed to an existing switch in the home and can be operated using a mobile phone. It allows users to turn the switch on and off remotely and manually, whether inside the home or within wifi range. It provides scheduled and timed lighting and other IoT-related features.

Objective Statements

1. To create the switch mechanism of ELSA that fits on a standard existing electrical switch in Hong Kong and can activate the existing switch to turn it on and off as required. The switch mechanism should activate in under 1 second.
2. To program the microprocessor to link ELSA to a web server so that ELSA can be accurately controlled through a web browser.
3. To set up a configuration system, implement other features, and create a user interface for ELSA so that ELSA can discover new devices and the user can configure WiFi settings and set automated tasks.

1.3 Literature Review of Existing Solutions

There are three common types of smart lighting products on sale in the market currently: Smart Light Switches, Smart Light Hubs, and Smart Bulbs. These devices will be reviewed in the following.

Smart Switches

While keeping the physical switch method to control lights, smart switches additionally provide remote and automatic control. Same as traditional light switches, smart switches are wall-mounted switches. When the physical switch is clicked, it will send a signal to the microcontroller inside. The microcontroller then connects or breaks the circuit between the light and the power supply to turn the light on or off [2]. With connection and accessibility using wireless communication networks such as Wi-Fi, ZigBee, and Bluetooth, users can perform remote control by using the app on a smartphone or tablet, speaking to a smart home assistant like Google Home and Apple Home Kit, or using the remote controller provided. Smart switches can also do scheduled and timed lighting with instructions set in the app.

Smart switches provide flexibility to users to choose suitable operation methods according to their needs. Users who are not able to use their phones or unfamiliar with using smartphones, especially the elderly, can still control the lights physically. However, installation of smart switches is inconvenient: screwing and wiring work is necessary since they are in walls and hardwired. A skilled electrician is needed to install the system, which causes additional installation costs. Moreover, most smart switches require neutral wire connections to give power supply to the microcontroller [3]. Therefore, houses that do not have a neutral wire network, such as old houses in China and the United States have limited choices.

Smart Plugs

Smart plugs are another alternative to achieve smart lighting. It has a working principle similar to smart switches. It contains a microcontroller to control the power flow between the power supply and the appliance, as well as a radio module to connect and access wireless networks [2]. Likewise, users can operate smart plugs by a smartphone app or digital home assistants. Smart plugs have the same functionality as smart switches.

Compared with smart switches, smart plugs are more convenient to install and have greater flexibility. Smart plugs do not require complicated wiring work, they can be installed by simply plugging in or out from a socket. Because of this, it is convenient to change the entire lighting setup. Users can move the plugs' position or convert a different lighting device whenever they want. However, smart plugs are only suitable for plug-in devices and cannot control hardwired ceiling lights, which are usually the major light source of houses. Also, smart light plugs are often bulky in size and may block adjacent sockets, decreasing the number of available sockets for users.

Smart Bulbs

Smart bulbs consist of a bunch of LEDs, a microprocessor to control LEDs, chips, and modules to perform wireless communications [5]. They can be operated through smartphone apps and smart home assistants to control the bulbs. For other functions such as scheduled lighting automation alone, an extra hub is necessary [4]. The advanced version of smart bulbs can even adjust the color and dimness of the light. However, the more functions smart bulbs have, the more expensive they are. Another thing that requires attention is that the light switch needs to be on at all times, or else the smart bulb has no power to operate.

Installing and removing smart bulbs is as easy as smart plugs. Apart from flexible setup, smart bulbs can be used for both hardwired and plug-in lights, solving the weakness of smart plugs. However, if the original light switch controls multiple light bulbs, using a smart switch is more cost-effective than using several smart bulbs. Most smart bulbs use a standard screw base for mounting, so not all lights can use smart bulbs [2].

All three smart light products demonstrate decent functionality. While some have solved the problem of installation, they still have room for improvement in terms of compatibility. Meanwhile, some smart door lock products solve the flaws in the lighting systems mentioned above, they are “Sesame Lock” [6] and “August Wi-

Fi Smart Lock" [7]. They attach to the original lock and turn the lock in a direct mechanical way. They are powered by a battery and have Bluetooth and Wi-Fi connections. There is no need to replace any part of the lock in the door. The advantage of this product is keeping the original mechanism. Users can use it without any technical knowledge. Additionally, if "Sesame Lock" does not work, the user can remove it easily and use the old mechanism as usual.

This project proposes the design of an external smart switch. It utilizes the existing light switches at home but can still offer the same function as other smart lighting devices. It aims to have high functionality while improving compatibility at the same time to make smart lighting more user-friendly and attractive to users.

SECTION 2— METHODOLOGY

1.1 Overview of ELSA

2.1.1 Product/System Description

In this project, we create a switch mechanism that can switch on and off a standard existing electrical switch and be controlled through wifi using a mobile phone or controlled manually. The External Light Switch Assistant (ELSA) can be adhered to an existing light switch using double-sided adhesive tape, and can function without anyone holding any components. In the best case, it should fit on two or more brands of switch in Hong Kong, because some switch designs are similar. The switch will be activated in under 1 second.

There is a hardware and a software control systems for ELSA. These allow ELSA to connect to the web server for handling user requests, report switch status, and control the mechanical part to complete on/off requests accurately with the aid of sensors.

A simple interface has the function of turning on and off the switch. While the user can change the configuration in the source code, the target users are non-technical users who do not need to learn to edit and compile the code themselves. To make ELSA more user-friendly, it has a user interface, which allows users to easily configure ELSAto perform complex tasks, like task scheduling.

The main working environment of the switch is home or other places with a stable WiFi network. ELSA does not require an authorization system as the "Sesame Lock" [6] reviewed in the literature review, but it needs to be able to communicate with other switches in the LAN. Therefore, WiFi network access and a method to communicate with other switches are needed.

Commented [6]: Describe the intended end product/system. What is it? What does it do? What are the functions? How does it work?

2.1.2 ELSA Diagram

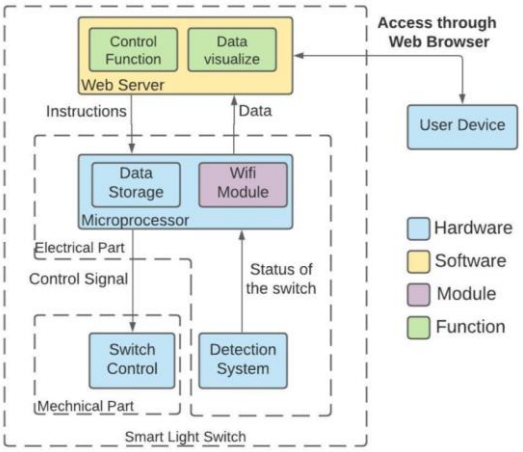

Figure 1. Schematic of the whole ELSA system

Figure 1 shows a basic diagram of ELSA. The system can be divided into three major parts, mechanical part, electrical part, and web server part.

The mechanical part contains all the components that directly toggle the switch and hold other components. It has two main functions. The first function is to apply force to toggle the switch. To achieve this function, motors are used to generate force to push the switch. Also moving parts and a supporting frame is needed, which can be created by using 3D printing technology.

The Electrical part contains the microprocessor and all other ICs to send and receive signals from mechanical parts. Its purpose is to build a control system that is used as an interface to achieve some automation with the webserver. To do so, there will be parts controlling the switch and detecting the status of the switch, and generating signals.

The Web server allows users to configure settings and show data received from the mechanical part. It uses the function from the electrical part to control the switch. This part involves both front and backend programming.

2.1.3 Components List

Table 1. List of Specifications

Items	Specifications/Model
Microprocessor control board	ESP32 Devkit V1
Gear Motor	GA12-N20 (gear reduction rate 1:250)
H-bridge Module	DRV8833
Tilt Module	GY25z
Shaft	2mm hex shaft

Commented [7]: All figures must have a figure number and heading or caption. Figure numbers and headings appear below the figure. Diagrams, photographs, screenshots, code etc. are all classified as figures.

Commented [8]: Explain the figure/diagram.

Commented [9]: All tables must have a table number and heading. Table numbers and headings appear above the table.

Commented [10]: Component

Commented [11]: Specific details of the component.

3.6V Battery	SAFT LS14250

2.1.4 ECE Knowledge

ELEC3300 --- Introduction to Embedded Systems:

Most of the knowledge in this course will be used in this project as we are going to use a microprocessor to hold a web server and control the hardware components. We will utilize the general-purpose input/output (GPIO), universal asynchronous receiver-transmitter (UART), and analog to digital converter (ADC) protocol to achieve automation in the product. Also, use interrupts to achieve some manual control.

ELEC3400 --- Introduction to Integrated Circuits and Systems

This course introduces the characteristics of BJT and MOS transistors and shows the principles of building an amplifier circuit for different applications (ADC, DAC, filtering). In this project, the current from the development board is limited and not enough for the motor. Therefore, we are going to build a circuit to provide a stable power supply for all components.

ELEC3120 -- Computer Communication Networks

The network protocol is used in this project to achieve the connection between switches, giving the command, and making a user interface. For example, UDP, HTTP. Understanding them helps create and debug the part where the device needs to communicate with the browser or other devices.

2.2 Objective Statement Execution

2.2.1 Switching mechanism and external device

To design and fabricate the switch mechanism that can switch on and off a standard existing electrical switch. The switch mechanism should activate in under 1 second.

In this objective, we produce a functional switching mechanism and make the model for the essential parts. The aim is to create a design that can stick on an existing light switch using double-sided adhesive tapes, and function without anyone holding any components. In the best case, it should be working on two or more brands of a switch, because some of their designs are similar.

For Hong Kong, there are three types of switch, toggle switch, European style rocker switch, and Australian rocker switch. Among them, the European-style rocker switch and Australian rocker switch are used in most residences and buildings.

Commented [12]: Communicate your entire project including procedures, results and evaluations of these results (ensure you include supporting diagrams and figures).

Commented [13]: Communicate the measurable objective (testing and results are reported in a task at the end of this objective statement section, which will be used to determine whether this Objective Statement has been successfully achieved).

Figure 2a. Toggle switch

Figure 2b. European style rocker switch

Figure 2c. Australian rocker switch

To simplify the design progress, we selected European-style rocker switches in figure 2b as the basis of the frame design. There are several reasons. First, it uses less force to toggle than the other two types because the moment arm is longer. Second, it has a larger surface area that can apply force, which means it allows a larger tolerance on the design.

Task 1

Aim: Find a way to generate force that pushes the switch on or off.

Expected Outcome: Finding one or more ways that can toggle the button and compare them.

Member in charge: All group members

Work Description: We suggested two different ways to create force to trigger the switch. One using electromagnets, as shown in Figure 3a, and another one using a motor, as shown in Figure 3bi and 3bii. Table 2 presents the design sketch and their advantages and disadvantages that we find after making a rough model with some wood pieces.

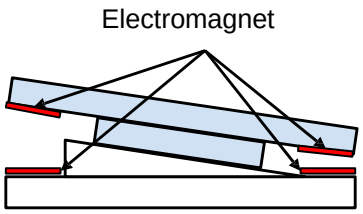

Figure 3a. Electromagnet design sketch

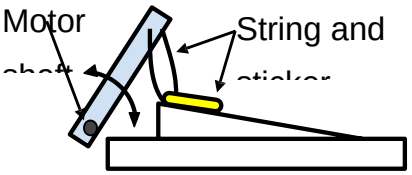

Figure 3bi. Motor design sketch 1

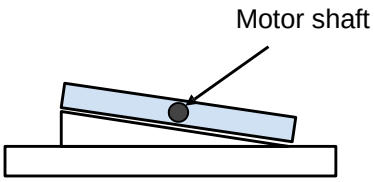

Figure 3bii. Motor design sketch 2

Table 2. Switch actuation mechanism comparison

	Advantages	Disadvantages
Electromagnet	• Can generate the attractive and repulsive force on the corresponding switch side. • Quiet.	• A strong electromagnet may affect the performance of other electronic components.
Motor	• Easy to set up.	• Noisy. • Has fixed torque output (due to fixed gear ratio).

After buying the materials and try building a simple model. We find out that the electromagnet approach is not feasible with the component that we can buy on the market.

Figure 4a. Pus h-pull electromagnet

Figure 4b. Holding electromagnet

The magnets that can be bought on the market have two types, push-pull and holding type. The push-pull type is shown in figure 4a. It creates a linear motion using an electromagnet, which is suitable in some locking mechanism, but not the mechanism for the idea above. The holding type is shown in figure 4b. It can create a strong attraction force in a very short distance, usually for industrial purposes. However, the repulsion force is weak when we test it and the attraction force only works in 2-3 millimeters, which is not the desired electromagnet for the idea.

Because using electromagnet is not feasible after consideration, we have to use motors as the main method for the switching mechanism.

We decided on two motor toggling mechanisms as shown in figure 3bi and 3bii above. The prototypes are built with a small DC Motor N20 with a reduction ratio of 1/100 gearbox and some wooden sticks. Testing shows that the performance of both prototypes is not stable since the torque is not large enough. The performance is guaranteed after using a reduction ratio of 1/250 gearbox.

Although the product using mechanism 1 is smaller in size, it is not energy efficient because of its short lever arm and bad force angle towards the switch surface. A motor with larger torque is needed. Also, it requires the user to install the device and the sticker in a precise position to press and pull the switch well. This may cause inconvenience to the user. Mechanism 2 needs to cover the whole switch which is larger but it is more energy-efficient since the applied force is nearly 90˚ to the switch surface. The installation of the device is simpler. Therefore, we decided to use the mechanism in figure 3bii with N20 250:1 gear motor for ELSA.

Task 2

Aim: Design and fabricate a simple mechanism that uses the same actuation mechanism from task one.

Expected Outcome: Creating a part that is going to apply force to the switch. Hold the component by hand and measure the distance of the switch and the part that works.

Member in charge: CHAN Siu Ming (Designing and printing), CHAN Tai Man (testing, feedback, and measuring)

Work: Before working on the design of parts for this task, we tried a few 3D modeling software, Blender, Freecad, and Fusion 360. In the end, Fusion 360 was chosen for two reasons. First, Fusion 360 is designed for parametric modeling. The models are built based on sketches. Users can change the value on the sketch and change the shape of the object. Also, users can produce models for existing objects with measurement. Second, Fusion 360 can simulate motion. It can simulate collision and movement. This function can help to test the model before printing it.

After choosing the CAD software, A European switch model is created as a reference for design and motion simulation. It is provided as a stepper file under "/model/step&f3d" in the Github repository [8]. One thing to be noticed is that the switch model is greatly simplified. Most curvature is removed because they are hard to measure and have a negligible effect on the simulation for the draft.

Then, two components are created for this task. The one with a cylinder shape is made for connecting the 2mm hexagon shaft and the motor's D-shaft. Another one is made for turning torque into force acting on the switch. The detailed dimensiond can be found in the Github repository [8] under folder "/model/drawing". The expected model is shown in figure 5. The printed part is rendered in yellow in the figure.

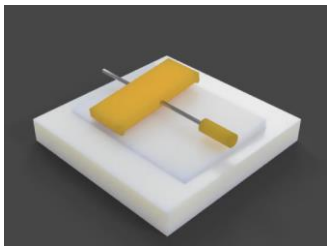

Figure 5. Model rendered by Fusion 360

Commented [14]: Introduce the figure in the text, referring to the figure by its figure number.

We tried two different FDM printers using PLA to print out the two models. They are UPbox+ and Prusa mini+. For the UPbox+, the detail is set to "fine" and printed in 0.1mm layer height with a 0.4 nozzle, others remain default. For Prusa mini+, 0.4 nozzles are used with the default 0.2 layer height profile setting. Both printers have one common problem when printing these parts, the accuracy of the holes.

To combine all the parts, the holes need to be printed with high accuracy. To achieve that, the holes need to be aligned with the vertical axis in the slicer program, so the holes are not affected by layer height setting and gravity. However, it brings out the second problem: the horizontal expansion problem.

Horizontal expansion is a common problem in 3D printing mainly caused by the shrinkage of cooling plastic. It can be solved by applying offset to the face and edge on the model. We tried two different methods on different machines. The easier one is adjusting variables in the slicer program. The variable can have different names in different slicer programs, for example, "horizontal expansion" and "XY Size Compensation". Once the variable is set, the slicer program runs the calculation and adjusts every edge. Another method is adjusting the dimension on the model, but it is less suggested as different machines can have different errors in

Commented [15]: Note that technical challenges and solutions are reported with the description of the work in which they occur.

horizontal expansion. Changing the model is time-consuming and it only applies to one machine. The second method should only be used when there are no variables in the slicer program.

To get the offset for the above method, there is only one method, printing out models with different size holes. After some trials, we suggest printing a cube with five holes, the diameter of the holes are 2mm, 2.1mm, 2.2mm, 2.3mm, 2.4mm and the offset they represent is 0mm, 0.5mm, 0.1mm, 0.15mm, 0.2mm. One thing that needs to be noticed is the sign of the offset, some slicing software uses positive as the shrink in, some uses positive as shrink out.

Therefore, do some test print before printing the actual model.

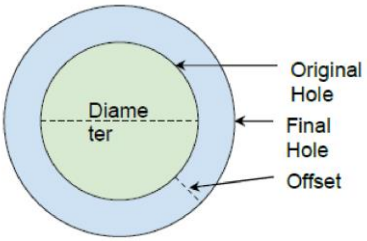

Figure 6. Extra description of the offset applied to the hole.

Task 3

Aim: Build a frame to hold all the parts.

Expected Outcome: The frame should be able to stick on a switch and hold all the components from Task 2 in place.

Member in charge: CHAN Siu Ming (Designing and printing),

Work: Start with measuring the position of the motor and parts. Then, create a cube on top of the switch and then empty the space on the decided position to make a hole for putting the motor and other parts in place.

One thing we did on the model of the frame is cutting it into two. There are two reasons for that. The first reason is to prevent overhang. Overhang is another common 3D printing problem, caused by the model design. This problem can be solved by adding support material or changing the model. We choose to change the model. Cut it into two pieces and combine them using a joint after printing it out. The second reason is we are not sure if the offset still applies. By cutting it into two pieces, the print time can be reduced if we need to modify the dimension of one side.

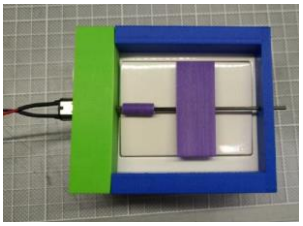

Figure 7a. Model rendered by Fusion 360

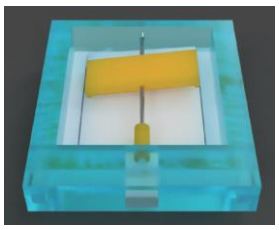

Figure 7b. Printed component

The model and printed part are shown in the figure 7a and 7b. For the rendered model, the new parts are rendered in cyan with transparency. The model dimension and Step file can be found in the Github repository [8] under the folder "/model/drawing" and "/model/step&f3d"

Task 4

Aim: To test the mechanism with frame

Commented [16]: Report the testing and results as a task.

Expected Outcome: The mechanism should be able to work by connecting to the power manually and should activate the existing switch in under 1 second.

Member in charge: CHAN Tai Man

To test the setup, we use the double-side adhesive mounting tape to stick the frame on the switch like Figure 8a and 8b. Then, put all the parts together like in Figure 7b.

Figure 8a. Top view of the stuck frame

Figure 8b. Side view of the stuck frame

The setup is tested by connecting the motor to a power source directly. By reversing the polarity of the power source, we can switch the direction of the motor. The video is on the Github repository [8] under the folder "/video". The final result is as expected, the frame adheres to the existing switch well and the mechanism can switch on and off the standard existing switch.

To test whether there is a delay in actuating the switch, we used a standard stop watch and recorded the time from connecting power to the end ofthe full travel of the existing switch, which turns the light on or off. The results are shown in Table 3. This test was repeated 10 times.

Table 3. Switching times

Test Number	Switching time (seconds)
1	0.26
2	0.37
3	0.81
4	0.18
5	0.25
6	0.49
7	0.55
8	0.27
9	0.38
10	0.35
Average time	0.39

Actually, we notice that it is difficult to start and stop the stopwatch so quickly because It seems the light turns on almost instantly, so there is a lot of varation in the times. However, the figures show that the actuation time is always under 1 second, which was the target actuation time, with the average time being 0.39 seconds.

We notice that there might be a problem with long-term usage. The two holes attached to the 2 mm hexagon shaft may wear down over time and cause problems. A change in the design to replace the shaft may solve this problem.

Commented [17]: The results from this Objective Statement work are compared with the expected results (the Objective Statement) to determine whether this Objective Statement has been successfully achieved.

Commented [18]: Report your technical challenges with the task the challenge occurred in.
In this model report, a new task (Task 5) is carried out to address the technical challenge.

Task 5

Aim: Add support to operate with Australian rocker switch

Expected Outcome: Same as task 3, but with the Australian rocker switch.

Member in charge: CHAN Siu Ming

Work: This task is repeating steps in tasks two and three. Begin with creating a model of a switch. The model stepper file is provided. However, it is not an accurate model, there are a lot of dimension errors. Many curvatures on the real switch are hard to measure and reproduce in the software. The only purpose of this file for this task is for early- stage reference and simulation.

Figure 9a. Side view of the new model

Figure 9b. Side view from another angle

Figure 9c. Top view of the new model

Then, modify the design in task 3 according to the Australian rocker switch. The printed model is shown in figure 9a,b,c. They are printed with the same setting as tasks 2,3 and aligning holes and pillars in the vertical axis. The dimension detail can be found on the Github repository [8] under folder "/model/drawing".

Two important changes in this model are the removal of the shaft and reducing the size of the frame. The reason why removing the shaft is stated in Task 4, the hexagon shaft can lose its function due to wearing down. For the replacement, a cylinder is joined to the moving parts to hold the parts in place. The reducing size of the frame can be done because there are more empty surfaces on the switch.

We tested this model as reported in Task 4. The frame adheres to the switch correctly and the mechanism operates as intended. The demo video is uploaded to the Github repository [8] under folder "/video". The new model is more durable than the old one due to the removal of the shaft. There is one extra discovery while testing. The discovery is that the new moving part can also be used on the European-style rocker switch. The video is also on the Github repository [8] under folder "/video".

Commented [19]: Technical challenge reported in Task 4 is addressed in Task 5.

Commented [20]: In this model report, for this Objective Statement, testing is reported within the task (Task 4).

2.2.1.1 Evaluation of the switching mechanism and external device

Expected outcome:

To create the switch mechanism that can switch on and off a standard existing electrical switch. In the best case, it should be working on two or more brands of a switch, because some switch designs are similar. The switch should be activated in under 1 second.

The actual outcome:

In terms of the type of switch, the mechanism can be applied to two common types of switches that can be found in Hong Kong, the European style rocker switch and the Australian rocker switch. The model could be easily modified to match other similar switch designs.

The model can be printed on two different machines, UPbox+ and Prusa mini+.

The two testing results in tasks 3 and 4 are good, they prove that the mechanism can switch on and off the existing electrical switch. The results of the timing test in Task 5 show that the average time that the switch will be activated is 0.39 seconds, and for all of the ten tests the switching time was under 1 second.

However, there is a problem with durability. The result for task 3 is bad. Parts can be worn down easily causing low durability. The result for task 4 is better. It removed the shaft, so no worn down can only happen for the D-shaft connection. The D-shaft can resist a strong torque before worn down, so the durability problem for result in task 4 is much smaller. The durability problem should be able to be fixed by changing material and production methods.

However, durability is not objective focus of ELSA in this project, and it will be addressed in future work.

2.2.2 Set up Hardware and Software

In this section, we built hardware and software control systems for ELSA. The aim is to allow ELSA to connect to the web server for handling user requests, report switch status, and control the mechanical part to complete on/off requests accurately with the aid of sensors.

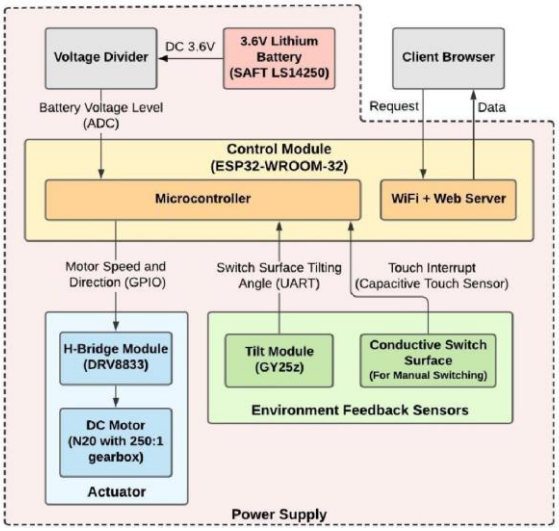

Figure 10. Overview of ELSA Hardware System

Figure 10 shows a detailed design of the hardware part with selected components. It consists of a control module, actuator, environment feedback sensors, and power supply.

We use the ESP32 DEVKIT V1 development board as the control module. It integrates the WiFi module and microcontroller (MCU). While running a web server that handles client requests, it controls the actuator to toggle the switch and receives feedback sensors' signals.

The actuator part holds and rotates the 3D printed moving parts. It includes a DC Motor N20 and H-bridge motor driver DRV8833, which has the following significant features:

- Can power low voltage motors (2.7V - 11.8V) which fits with N20
- Has high energy efficiency
- Provides low power sleep mode so can save power when the switch is in standby mode
- Includes internal shutdown function to protect from short circuit, overcurrent, overheat
- Small and no need heat sink

There are two kinds of environment feedback sensors. The first one is the tilt module, which reports the tilt angle of 3D printed moving parts to the MCU so it can control the actuator rotating the moving parts to the targeted on/off position and know its state when needed. Originally using a motor encoder is also considered to achieve this purpose, but it is shelved after discovering the stripped shaft hole problem mentioned in the last objective. It is because a stripped shaft hole will cause shaft shifting, the motor rotary position cannot reflect a real tilt angle. The tilt module GY25z combines the Inertial Measurement Unit (IMU) MPU6050 and the MCU STM32F030F4P6, IMU provides its acceleration and angular velocity and MCU uses them to calculate the tilt angle (roll, pitch, yaw). Using the GY25z instead of the MPU6050 can skip the calculation part, the board size is even smaller.

The second sensor is the capacitive touch sensor in the ESP32 board. When connecting a conductive material such as copper pads to a touch pin of the ESP32 board, the touch sensor will read the capacitance and obtain an analog reading, touching the material will change its reading. Utilizing this technique users can turn the switch on and off manually without using the web browser, when they touch the copper surface on the 3D printed moving part, a drop of capacitance can form a touch interrupt to inform the MCU corresponding on/off request.

For the power supply, we use the SAFT LS14250 3.6V lithium battery. It is small in size (½ AA-size) but has a large capacity (1.2Ah), it occupies less space but can operate for a long time. The battery voltage level is monitored by reading the analog value in a voltage divider using ADC (analog to digital convertor).

Task 1

Aim: Build an experimental hardware setup of figure 10 for program testing

Expected Outcome: A complete prototype including electronic circuit and mechanical parts

Member in charge: CHAN Tai Man

Work Done: We first checked the components' user manual [9][10] to understand their working principle and pinouts. Then planned the wire connections between the components, figure 11 is the schematic diagram.

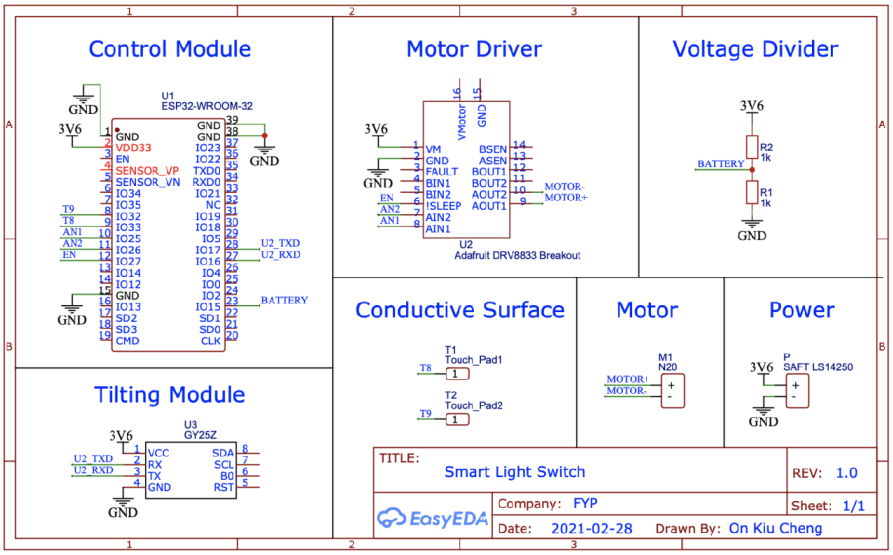

Figure 11. Schematic Diagram of ELSA Hardware System

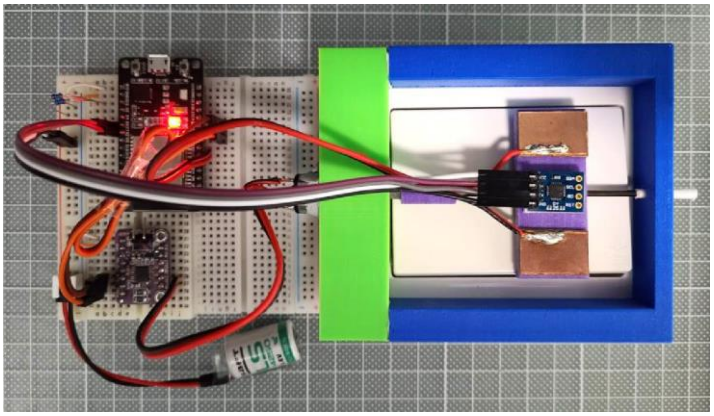

Figure 12. Prototype of ELSA

Following the schematic, we used breadboards and Dupont wires to construct a prototype circuit. As shown in figure 12, the GY25z module is stuck on the centre of the 3D printed moving part, two copper pads are stuck on its upper and lower sides by double-side adhesive tape. The pads were soldered with wires, the upper copper plate connects to the T9 pin and the lower plate connects to the T8 pin. This prototype has a big improvement in wiring and appearance which can be done in further development.

Task 2

Before defining the main functions for on/off requests, we need to define sensor functions to let the MCU obtain environment feedback first. In this task, we defined get roll angle functions. Since the gy25z rotates

around the x-axis only, we only need to obtain the roll angle. We also defined touch read and touch interrupt functions.

Aim: Define GY25z and touch sensor functions and variables

Expected Outcome: Able to let ESP32 board obtain the roll angle of GY25z when needed, able to apply touch interruptions

Member in charge: CHAN Tai Man

Work Done:

GY25z functions:

The communication protocol of GY25z is UART with a default baud rate of 115200 bps, so the UART2 channel in ESP32 is used. First, we wrote an init function, several commands provided in the user manual [10] are used for initialization:

● Select output data type: 0xA5+0x55+0xXX+sum, byte of 0xXX has the meaning as shown in Table 3, following:

Table 4. Meaning of bits.

Bit:	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Meaning:	/	temperature	/	tilt angle	/	/	angular velocity	acceleration

*1 = output, 0 = no output

As we only need the tilt angle, the byte will be 0001 0000 so 0xXX = 0x10. “sum” is the hexadecimal number of XX, thus sum = 0x0A in this case. Therefore, the whole command will be 00xA5+0x55+0x10+0x0A.

● Set query mode: 0xA5+0x56+0x01+0xFC
● Save configuration: 0xA5+0x5A+0x01+0x00

The following is the part of the init function gy25z_init():

```
  void
gy25z_init(){ Serial2.begin(115200);
//start UART2

Serial2.write(0XA5);  //Select output data type
Serial2.write(0X55);
Serial2.write(0X10);
Serial2.write(0X0A);
delay(3000);

//repeat above Serial2.write() steps to set query mode and save configuration
.
.
.
```

Next, we wrote the get roll angle function gy25z_getRoll(). First serial write the query mode command; the module will send back an 11 bytes data frame which has the following format (for obtaining tilt angle only):

Table 5. Format of 11-byte data frame.

Byte	Value	Meaning
Byte 0:	0x5A	Starting flag
Byte 1:	0x5A	Starting flag
Byte 2:	0x10	Output data type (0xXX)
Byte 3:	0x06	Number of data bytes
Byte 4:	0x00~0xFF	Roll angle high byte
Byte 5:	0x00~0xFF	Roll angle low byte
Byte 6:	0x00~0xFF	Pitch angle high byte
Byte 7:	0x00~0xFF	Pitch angle low byte
Byte 8:	0x00~0xFF	Yaw angle high byte
Byte 9:	0x00~0xFF	Yaw angle low byte
Byte 10:	0x00~0xFF	Sum of data bytes

To obtain all 11 bytes values, create an array variable Re_buf[11] and write a while loop. In the loop: when the MCU receives a byte, assign the received value into Re_buf[0]. If Re_buf[0] is not equal to 0x5A, it means the received message is wrong so report the error and break the loop. If yes, assign the next received value into Re_buf[1] in the next loop and so forth until it has looped 11 times.

After getting all bytes, check if the first bit and second bit are equal to 0x5A. If so, left shift Re_buf[4] 8 bits and or bitwise the shifted Re_buf[4] with Re_buf[5], the roll angle is this 16-bit value divided by 100.0.

Detail code can be viewed in ”/Switch/src/switch_gy25z.cpp” on the Github repository [8]

Touch sensor functions:

Below is the sample code of touchRead and touchInterrupt functions provided by the ESP32 library []. The first function returns the pin's capacitive value. The second one triggers touch interruption and changes the corresponding boolean value when the capacitive reading is lower than the threshold value.

However, when using multiple touch pins and touching them by turns quickly, false triggers will happen due to noise interruption during debounce. For example, touch T8 pin then T9 pin will result in T8 T9 T8 interrupt, T9 T8 are triggered at the same time when T9 is touched. Therefore, we added a referenced time interval, if the time between the last interrupt and current interrupt is shorter than the referenced time interval, the current interrupt will be considered as a false trigger and will be ignored. Added lines are in purple.

```
//touchRead function: return capacitive value of T8 pin
int T8Value = touchRead(T8);

//
bool triggeredT8 = false;
bool triggeredT9 = false;
void IRAM_ATTR T8wasActivated() { triggeredT8 = true; } //put callback function in IRAM
void IRAM_ATTR T9wasActivated() { triggeredT9 = true; }

volatile unsigned long lastT8 = 0;
volatile unsigned long lastT9 = 0;

const int threshold = 20;
const long minInterval = 350; //referenced time interval

   bool        touchIntervalComp(unsigned        long
 lastTouch){ if ((millis() - lastTouch) < minInterval)
 return false; return true;
}

   void setup()
 { Serial.begin(115200
 ); delay(1000);
 touchAttachInterrupt(T8, T8wasActivated, threshold);
 touchAttachInterrupt(T9, T9wasActivated, threshold);
}

//example touch interrupt triggered function
if (triggeredT8){
 triggeredT8 = false;
 if (touchIntervalComp(lastT8)){
   //Do something
   lastT8 = millis();
 }
}

   if
 (triggeredT9){ trigg
 eredT9 = false;
 if (touchIntervalComp(lastT9)){
   //Do something
   lastT9 = millis();
 }
}
```

Application of the function can be viewed in "/Switch/src/motor_touch.cpp" on the Github repository [8]. They will

also be explained in the next task.

Task 3

Aim: Define main functions for ELSA's configurations and routine

Expected Outcome: Functions that allow ELSA to calibrate on/off positions, perform auto and manual switching.

Member in charge: CHAN Tai Man

Work Done:

Calibrate function:

The proposed method of calibration is to ask users to toggle the switch to the on and off position once during calibrate mode; the MCU will automatically record its roll readings. However, we cannot press the light switch directly because the motor locks the moving 3D printed parts. To toggle the switch 'manually', we use the touch interrupt function defined in the last tasks. When the touchpad on the 3D printed part is pressed, it triggers the MCU to let the motor rotate to the corresponding direction to toggle the switch. When the touchpad is not pressed afterwards, read the roll readings. The roll value increases from 0 to 655 when it rotates clockwise around the x-axis. Figure 13 is the diagrammatic sketch.

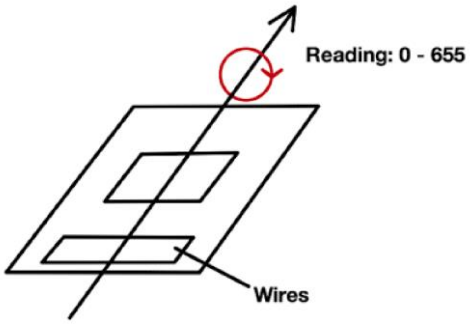

Figure 13. Diagrammatic sketch of GY25z roll value variations

In addition, we considered two calibration scenarios because ELSA can be placed in two orientations, we named Figure 14a orientation 1 and Figure 14b orientation 2.

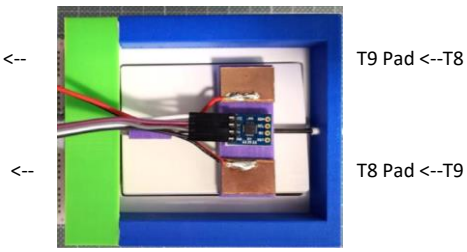

Figure 14a.Orientation 1

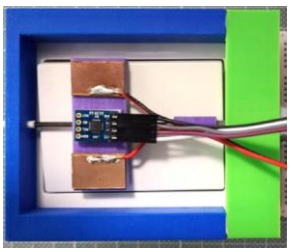

Figure 14b. Orientation 2

There are a several variables in the function:

Float: T8Roll (roll value when T8 pad is pressed), T9Roll (roll value when T9 pad is pressed)

Boolean: cal (switch is calibrated), calT8 (T8Roll is calibrated), calT9 (T9Roll is calibrated), T8IsOn (switch is on when T8 pad is pressed)

The properties of the two scenarios are reversed.

Table 6. ELSA orientation

Orientation	On rotate direction	Off rotate direction	T8Roll > T9Roll	T8IsOn
1	Clockwise	Anticlockwise	True	True
2	Anticlockwise	Clockwise	False	False

Based on the properties, we made the function flowchart in figure 15.

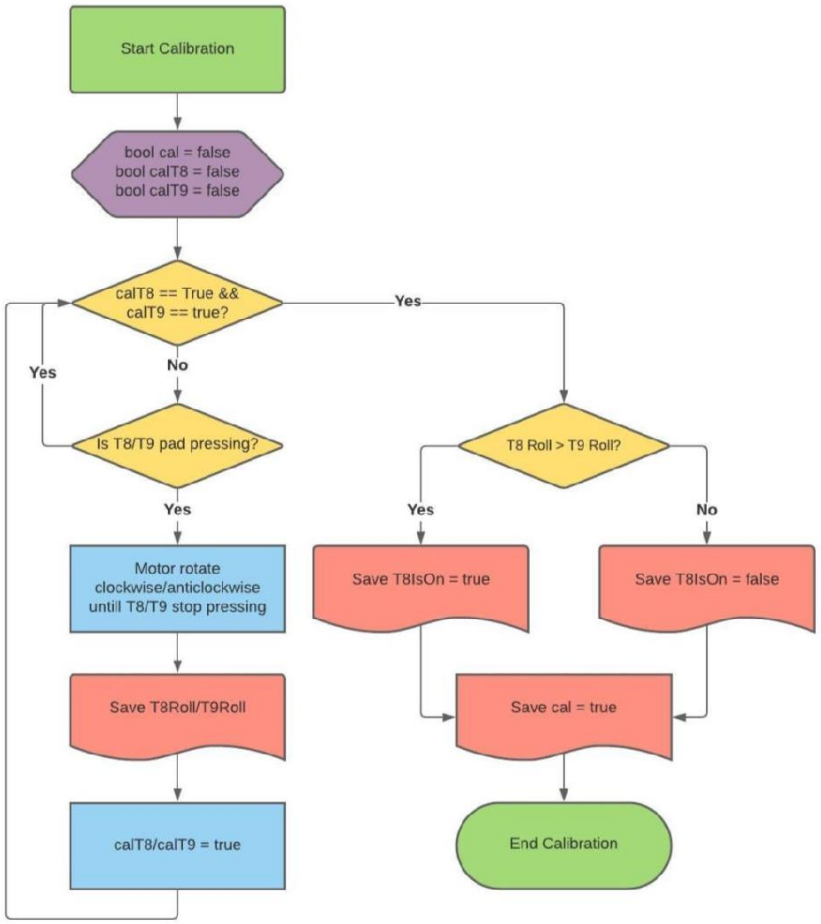

Figure 15. Flowchart of function calibrate()

"Preference.h" library [11] is used to save the roll values permanently so they will not be erased after reset. The code

can be found in the function "calibrate" in "/Switch/src/switch_motor.cpp" on the Github repository [8].

Remote and manual on/off functions:

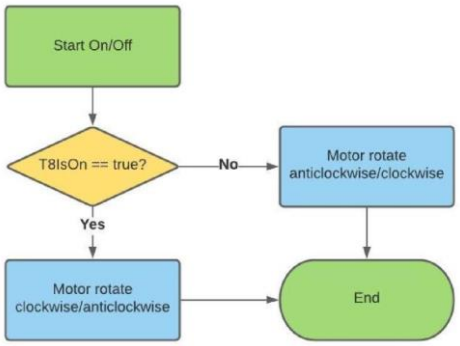

Figure 16. Flowchart of function on() and off()

Figure 16 is the flowchart of remote on/off functions. Manual on/off functions used the touchInterrupt functions in the last task. If the T8 pad is touched, the motor rotates clockwise, otherwise rotates anticlockwise.

The motor is controlled by GPIO pins EN, AIN1 and AIN2. When EN is 0, the motor driver is in sleep mode, the motor will not rotate even AIN1 and AIN2 have inputs. Below table shows the logic of AIN1 and AIN2.

Table 7. Logic of AIN1 and AIN2.

AIN1	AIN2	AOUT1 (Motor+)	AOUT2 (Motor-)	Function
0	0	Z	Z	Stop
0	1	L	H	Rotate Clockwise
1	0	H	L	Rotate Anticlockwise

This is the code of clockwise rotation, it keeps rotating until it reaches the target roll value.

```
  void
turnClkwise(){ preferences.begin("switchS
etting");
if(preferences.getBool("calibrated",false)){
   preferences.putBool("switchIsOn", (preferences.getBool("T8IsOn", false)?true:false));
   digitalWrite(EN, HIGH);
   float current_roll = gy25z_getRoll();
   float target_roll = preferences.getFloat("T8Roll",-1);
   while(current_roll< target_roll && abs(current_roll - target_roll)>2){
      digitalWrite(AIN1, LOW);
      digitalWrite(AIN2, HIGH);
   }
   digitalWrite(EN, LOW);
}else{
   Serial.println("Plz calibrate first");
}
preferences.end();
}
```

Codes of above functions can be viewed "/Switch/src/switch_motor.cpp" on the Github repository [8].

Task 4

Aim: Create a simple web server with existing WiFi to prove the idea of remote control of ELSA through a web browser.

Expected Outcome: ESP32 board connects to an existing WiFi and generates a webpage, other clients who connected with the same WiFi can access the webpage to on/off the switch

Member in charge: CHAN Tai Man

Work Done: We created a simple web server that can handle on/off requests by using the Adruino sample code on the Last Minute Engineers tutorial page [12]. The code connects the ESP32 board to assigned WiFi, then it sets up HTTP requests, corresponding functions, and designs the web page by HTML. When the ESP board successfully connects to WiFi, it will send a message to the console and report light switch status. Figure 17 shows the console and figure 18 is the view of the webpage.

Figure 17. Arduino IDE Console of ESP32

Figure 18. View of Webpage

Some main functions are used from WiFi.h and WebServer.h libraries.

`WiFi.begin(ssid, password);` connect to assigned WiFi

`server.on("/", handle_OnConnect);` create http request on root path and trigger corresponding function

`server.begin();` start web server

`server.handleClient();` start handling incoming request The code can be found in Appendix G.

Testing and results:

First, we test the GY25z get roll angle function alone, we use a simple test code to request the module keep sending out values while rotating the 3D printed moving part. It successfully gets the readings; they are reasonable and the error is small (±2). We also successfully trigger the MCU when we touch the touchpads. After the web interface in the third objective is built, we tested the main functions. We sent calibration and on/off requests several times, they both worked fine. The demo video is on the Github repository [8] under folder "/video".

Commented [21]: Testing and results can be reported as a separate task, or be a part of a larger task, as shown in this model report.

2.2.2.1 Evaluation of the hardware and software

Expected outcome:

ELSA can connect to the web server and complete calibrations, on/off requests accurately.

The actual outcome:

All elements in the prototype did their task mentioned in figure 10 well. The prototype can create a web server and fulfil all basic requests (calibrate position, remote and manual on/off) so the results meet the objective.

To design functions that are feasible for both scenarios made the function more complex. Also, formerly the calibrate function triggered the watchdog and forced the ESP32 board to reset. We discovered that it was not

suitable to put interrupt while loops and serial write functions in a while loop, it caused too much work. We used if conditions instead and the problem was solved.

2.2.3 Set up a configuration system and implement other features

In the last task, a simple interface is created with the function of turning on and off the switch. The code in the last task only allows the user to change the configuration in the source code. However, the targeted users are non- technical users. They are not supposed to learn to edit and compile the code themselves. To make ELSA more user- friendly, a way to configure ELSA is needed. Moreover, ELSA in the last task is lacking in features. It should be able to perform more complex tasks, like task scheduling. In this work, we program the microprocessor to link ELSA to a web server so that ELSA can be controlled through a web browser.

A top-down approach is used to design the main structure of the system.

The main working environment should be at home or other places with a stable WiFi network. ELSA does not require an authorization system as the "Sesame Lock" [6] reviewed in the literature review, but it needs to be able to communicate with other switches in the LAN. Therefore, WiFi network access and a method to communicate with other switches are needed.

The targeted users are non-technical users. The web page of ELSA should be accessible from different devices without searching for the IP address. Therefore, multicast-DNS and DNS should be implemented. Also, the web page should be responsive, so different devices can access it with the proper layout.

In this objective, all the code can be classified into two classes, frontend, and backend. The frontend is the user interface, part of them are visual elements, the remaining is event handling and communication with the server. The backend involves the network connection, hosting a web server, and scheduled tasks. Both parts have nothing to do with the circuit. Most of the code should run even without the circuit.

The IDE used is Visual Studio Code with PlatformIO plugin [13]. The first reason for choosing this IDE is because both the frontend language and backend language are supported by the editor, so development can be done without switching between editors. Second, the PlatformIO project is more suitable for programming in a group. It has config file saving configurations for the project. For example, the platform name, board, framework, and library used in the project. Others can load the config and compile the project without extra configuration. If we use ArduinoIDE, all the config needs to be done manually.

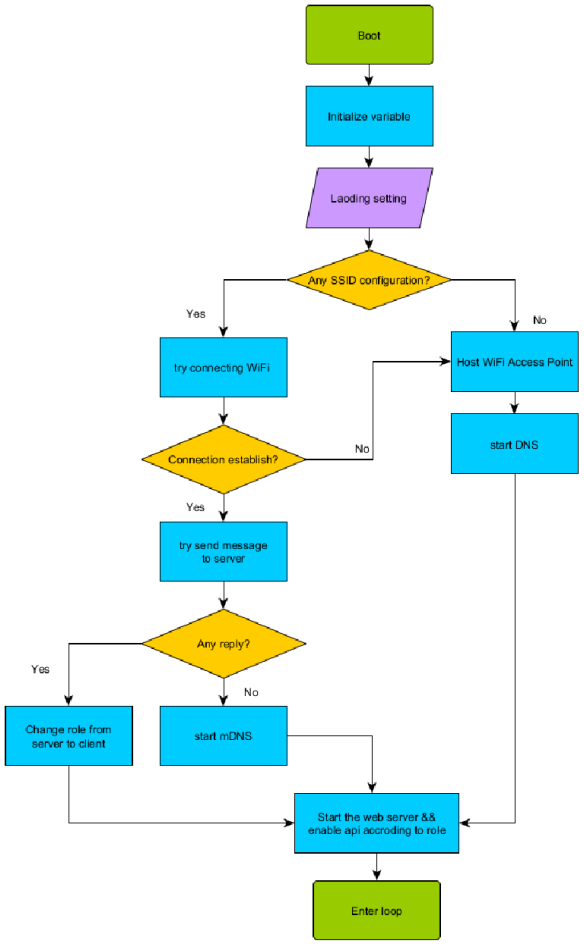

Figure 19. A flowchart showing the rough idea of the backend boot process

The flowchart in figure 19 shows the rough idea about the order of service and the decision needed to be made when ELSA boots up. The tasks in the objective will show the build process of the program according to this flowchart.

Task 1

The first task is setting up a web server. Building the server in the early stage is because it can help us to debug other parts. Most of the settings in ELSA should not be hardcoded in the source code, such as the WiFi setting. Setting up API in the early stage allows us to perform certain tasks with the “GET” or ”POST” command.

In the last objective, there is already a web server built using TCP. However, the web page content is hardcoded into the source code. It is hard to read and maintain. This method is suitable for smaller projects that contain few lines of code and doesn't plan to do further development. For this objective, we are going to build more web page content and API endpoint. Therefore, using a library, which supports reading from files and creating API endpoints, is a better choice.

Aim: Setting up the web server that can handle GET and POST requests.

Expected Outcome: A web server serving a web page and can handle GET and POST requests.

Member in charge: CHAN Siu Ming

Work Done: Using the "Async Web Server" library [14], an asynchronized web server can be hosted using a few lines. It can read from the file system and extend the API easily. It adds a new class called "AsyncWebServer". Here are lines from the source code, showing the usage of this class.

```
AsyncWebServer web_server(80);
```

It creates a new "AsyncWebServer" object and hosts it on port 80.

```
web_server.on("/restart", HTTP_GET, responses_restart);
```

This line creates a request handler. It accepts three augments, URI, type of request, and the function that handles the request. URI refers to the string after the URL. Assume the server of this example is hosting on 127.0.0.1. When a user sends a GET request to "127.0.0.1/restart", the handle will run the function "responses_restart".

```
web_server.serveStatic("/", SPIFFS, "/").setDefaultFile("index.html");
```

This line of code does two things. The first half will add a request handler to the root of the web page and serve all files in the root directory in SPIFFS. The second part sets the default file that returns to the user, in this case, "index.html". When a user enters the URL in the browser, the handler will return the default file, the HTML file. In the HTML files, there are lines requesting files like JavaScript and CSS style sheet. They are placed in the root directory and served by the first half of the line.

For testing, WiFi connection is established by hard coding SSID and WiFi information. The IP address can be obtained by either printing the information to the serial monitor or checking the client list of the access point. By typing the IP address into the browser, the content of the "index.html" should be shown. When typing the IP with the URI, the browser will send a GET request and the server should run the corresponding function. It can be checked by printing information to the serial monitor.

Task 2

In the last task, the web server is hosted. However, there is no content on the page. In this task, the content will be added to the page. The content of the page needs to fulfill one important requirement, user friendly on both mobile and desktop devices. In other words, the layout of the content should be changed according to different devices.

Also, there should be some kind of tab system to keep content organized instead of placing them on one page.

Aim: Designing a responsive web page that has an extensible tab layout

Expected Outcome: A responsive page with a tab layout.

Member in charge: CHAN Siu Ming

Work Done: For this task, some codes from w3school are used as references. They are responsive top navigation [15] and menu icon [16].

The example code of the HTML, JavaScript, and CSS can be found in Appendix H. Some JavaScript code is modified using JQuery library [17] instead of pure JavaScript so that the code is more readable. Therefore, to use the example code, jQuery library (version 3.5.1) is needed.

The tab system in the example works based on the location hash. When the user clicks on the tab, the hash after the address will change. It will trigger an event, and then the CSS of the tab and content will be changed according to that hash.

To test the layout without uploading the file to the ELSA. A simple python script is created. By using this script and the developer tool from the browser, the visual effect on different devices can be tested easily.

localServer.py

```
import  HTTP.server
import socketserver
import os


PORT = 8000


web_dir = os.path.join(os.path.dirname(    file__), 'data')
os.chdir(web_dir)
```

The “web_dir” is the variable pointing to the folder that contains HTML, JavaScript, and CSS files. In our case, the

“data” folder. The code can also be found on the Github repository [8] under the folder “/Switch/src”.

The result should be a page looking like figure 20a and 20b:

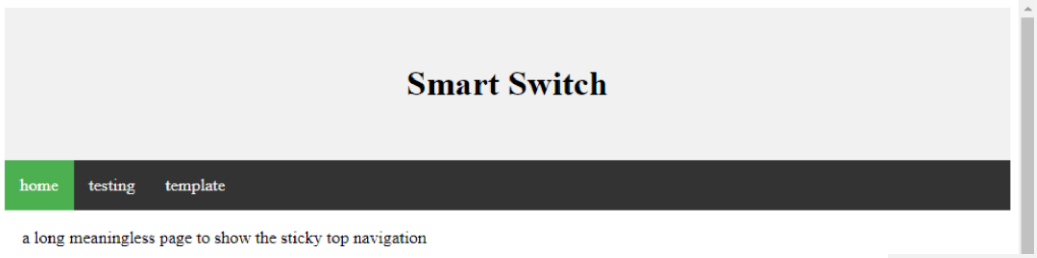

Figure 20a. The view on a wider screen

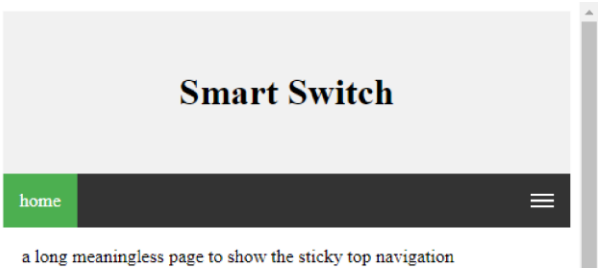

Figure 20b. The view on a smaller screen

Task 3

This task is going to implement the WiFi part. ELSA needs to read data from the configuration and decide to connect to an access point or host an access point.

Aim: Connect to WiFi network if there is configuration. Otherwise, host a LAN.

Expected Outcome: Connecting to WiFi if there are any configurations. If it fails to connect or no configuration, host a LAN so the user can connect to the LAN and set configuration.

Member in charge: CHAN Siu Ming

Work Done: The main libraries related to this task are “Preference” and “WiFi” from the Arduino framework [11]. “Preference.h” is a library that stores data in a non-volatile way and accesses data through user-defined keys. “WiFi” is the library to handle the WiFi connection or hosting.

In the beginning, an API endpoint, “/wifi_setting”, is added to receive two important data, SSID and the password. For this request, POST is used instead of getting for security purposes. POST hides the data from the URL and also the data can be encrypted. The data carried by the request will be in serialize array because the web server library and C++ do not support deserializing JSON useless other libraries are used. The code can be found in the function “responses_wifi_setting” in “/Switch/src/responses.cpp” on the Github repository [8].

Then, a simple form is created on the page to allow input of the SSID and password. Here is the HTML and the JavaScript code used.

part of index.html	part of script.js

<form id="Wifi_form">
<label for="SSID">SSID:</label>
 	
<input name="SSID" type="text">
 	
<label for="Password">Password:</label>
 	
<input type="password" name="passwd">
 	
<input type="submit" value="Submit">
</form>
	$(function () {
	$('#Wifi_form').on('submit', function (e)
	{ e.preventDefault();
	$.post("/wifi_setting",
	$("#Wifi_form").serializeArray(), function(data,
	status){
	$(Wifi_respond).html(data);
	});
	});
	}

After that, the main part is implemented. The program structure is the same as the flowchart shown in figure 19 at the beginning of the objective.

Three main functions are used in this part and here is the example from the source code: preferences.getString("SSID")

It is from the “Preference.h”. Bypassing the key to the augment, it returns a string that is stored with the same key. In this example, the string stored with the key ”SSID” is returned.

WiFi.begin(SSID.c_str(), PASSWD.c_str());

It is from “WiFi.h”. It tries to establish a connection with the given SSID and password.

wifi_ap_start(PRODUCT_NAME+"-"+DEVICE_ID);

It is from “WiFi.h”. It is used for creating WiFi access points with the given SSID and password. In this example, no password is passed to augment, so WiFi with no password will be hosted.

The details of the program can be reviewed in the Github repository [8]. The code related to this part are placed in

“service_Wifi_client.cpp” and “service_WiFi_AP.cpp” under “/Switch/src”

To test this part, an extra API endpoint is created to clear the setting in the “Preference.h”. This part of the code is tested in four different cases. The cases and result are shown in Table 7, following:

Table 8. Result of testing the code.

Cases	Result
No configuration	The device hosts an access point and can be searched by other devices.
Configuration with wrong SSID or password	The device hosts an access point and can be searched by other devices.
Configuration with correct SSID and password	Connection established, can access Web server through the IP address
Configuration with hidden SSID and password	Connection established, can access Web server through the IP address

The outcome matches the expected result from the flowchart.

The main target of the task has been fulfilled, but there is a minor detail that can be improved. In most WiFi settings pages, there is a list showing the WiFi that has been broadcasted. To achieve that, some more code is added to achieve that function.

First, a new API endpoint, ”/SSIDlist”, is added. It will return a list of SSID that is broadcasting in JSON format. Then is

modifying the web page, the modified code is as follow: (new code are coloured in purple):

part of index.html

```
<form id="Wifi_form">
    <label for="SSID">SSID:</label>
    <br>
    <input list="ssid-list" name="SSID" type="text">
    <datalist id='ssid-list'>
     <option>list loading.....</option>
    </datalist>
    <br>
```

part of script.js

```
function getSSID(){
    $.get("/SSIDlist",function(data,status,xhr){  if(
  xhr.status  ==  202){setTimeout(getSSID,500);}
  if(xhr.status == 200){
    //console.log(data);
    $("#ssid-list").empty();
    var content = ""
          for (item  of
     data){              if
     (item.ssid){
     content += "<option value='"+item.ssid+"'>"+item.ssid+"</option>";
     }
    }
    $("#ssid-list").html(content);
    $("#ssid-list").change();
    ssidlock = false;
  }
 });
}
```

Task 4

Aim: Use DNS and mDNS to create a domain name that is easier to access

Expected Outcome: Hosting DNS server for access point mode and multicast-DNS for client mode.

Member in charge: CHAN Siu Ming

Work Done: For multicast DNS, the library “ESPmDNS.h'' from Arduino framework [17] is used. It adds a class called “MDNSResponder”. There are two functions from this class being called in the project.

bool MDNSResponder::begin(const char* hostName)

This line will take the augment as the domain name for multicast DNS. For example, if “switch” is the input of the augment, the URL will be “switch.local/”. In this project, it will be the product name.

bool MDNSResponder::addService(char *name, char *proto, uint16_t port)

This line task adds the service and the port, it advertises information about network services that the device offers. For this project, (“http”,”tcp” 80) is the input to assign the HTTP service at port 80.

For DNS, the library “DNSServer.h” from the Arduino framework is used. It adds a class called DNSServer. The class

can handle the DNS request. Two functions from this class are being used in this project, they are:

bool DNSServer::start(const uint16_t &port, const String &domainName, const IPAddress &resolvedIP)

This function accepts three arguments, the port, domain name, and the IP address that hosts the server. In this project, we are using the default port, port 53. The domain name will be the product name. The IP address will be the device IP address.

void DNSServer::processNextRequest()

This line is placed at the loop. it checks if there is any DNS request and returns the IP address.

The details of the program can be reviewed in the Github repository [8]. The code related to this part are placed in

“service_mDNS.cpp” and “service_DNS.cpp” under “/Switch/src”

To check if multicast-DNS is working, we use another device that connects to the same LAN and types the domain name in the browser. To check the DNS function, we clear the setting using the API endpoint created at task2 and connect a device to the Access Point.

The result of DNS is fine. All devices can reach the web page.

For multicast-DNS, it succeeded on devices with IOS, Mac OS X, Windows, and Linux. However, it fails on android because android does not support multicast DNS in the default DNS lookups. The only way to use multicast-DNS is to write an app for android. This will be one of the tasks that need to be done if there is further development.

Task 5

Assuming there is more than one device in the same network, then a control system is needed to control each device individually. Otherwise, there will be multiple devices using the same domain name and may cause problems. The target for these tasks is to figure out a way to solve this problem. The problem contains two parts. First, the method to control the device individually.

Second, let the devices discover each other.

Aim: Find a way to control clients individually and set up a discovering system for detecting other switches in the same network.

Expected Outcome: The device can find other clients and users can control each client on the webpage.

Member in charge: CHAN Siu Ming

Work Done: The way to solve the first part problem is to create a list on one of the devices. To make the explanation easier, the device storing the list will be the “main device”, others are “Client”. In this case, the first device joining the local area network will be the main client. It will store the ID and IP address of the client.

When a user accesses the page, the page will send a request to get the list. Then using those IP addresses to get other information, like on/off status. The following diagram shows the flow of information.

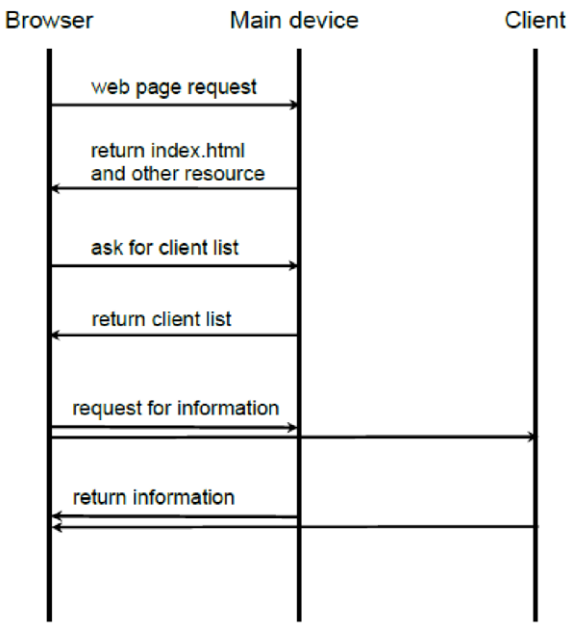

Figure 21. Sequence UML Diagram showing the communication process

One thing to be noticed is that getting information from another domain name server will trigger an error. Shown on the right side of Figure 22. It can be fixed by adding a header to the response from the client.

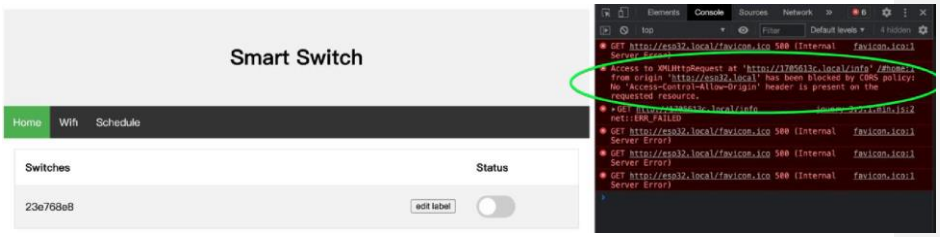

Figure 22. The error message when getting information from different domains

The remaining problem is to get the IP and ID from the client, which joins the network later than the main device. The first idea is utilizing the multicast-DNS. As the main client has a fixed domain name. When other clients join the network, it can send a packet to that domain name. If there are responses, servers exist, and vice versa. However, it fails because multicast-DNS lookup is not implemented in the ESP32 network library.

The second idea is using UDP to broadcast messages on the local area network. The idea is inspired by DHCP. It uses the broadcast address to message the server. Once the client confirms the existence of the main device, it will broadcast the message periodically, in case there are changes in IP address or disconnection. The details of the code are in the repo, most of the functions used can be found in “service_udp.cpp”.

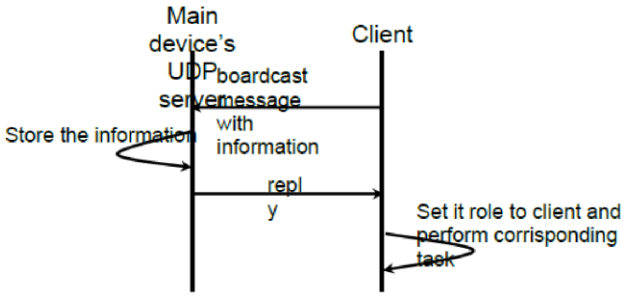

Figure 23. Sequence UML Diagram showing how messages are sent and processed

To debug the code, a software called “node-red” [18] is used. Using node-red, the computer can work as a UDP client, it can receive broadcast messages and send custom UDP messages. There are two files on the Github repository [8] under folder “/node-red-config”. The file “fake-server.json” contains the setting that emulates the existence of the server and ”fake-client” contains the setting that emulates the existence of the client.

To check the actual result, two development boards are used. Both are connected to the computer and printing some messages on the serial monitor.

```
WiFi:           OK
IP address:     192.168.1.33
Role:           Server
ID:             esp32
label:          23e768e8
```

Figure 24a. Message on serial monitor of the main device

```
WiFi:           OK
IP address:     192.168.1.13
Server found
Role:           Client
ID:             1705613c
label:          1705613c
```

Figure 24b. Message on serial monitor of the client.

Task 6

Aim: Perform scheduled tasks

Expected Outcome: The devices can perform tasks according to time. Users should be able to configure tasks on the web page.

Member in charge: CHAN Tai Man

Work Done: To perform scheduled tasks, the library ”CronAlarm”[19] is being used. It provides a convenient way to create a timed task using cron expression [20].

The next problem is to save the schedule, so the device can continue running the task after a restart. To achieve this, the “Preference” library is used. The key we designed is a String with a fixed prefix “Schedule_” plus with a number, which counts from 0. The value stored is a modified cron expression, which is obtained by joining the cron expression string and the action. For example, “ */15 * * * * * on” mean turn on every 15 seconds. With these two custom formats, saving and reading data can be done by a loop.

Example for reading from config:

```
String cron_key(int num){return "Schedule_" + String(num);}

    void
  cron_load_from_setting(){ pr
  eferences.begin("setting");
  int failed = 0;
  int i = 0;
  while(preferences.isKey(cron_key(i).c_str())){
    String data = preferences.getString(cron_key(i).c_str());
    if(cron_add(data)==255){++failed;};
    ++i;
  }
  Serial.println(String(i-failed)+" cron task loaded, "+String(failed)+" failed");
  preferences.end();
}

uint8_t cron_add(String custom_cron){
  String timing = custom_cron.substring(0,custom_cron.lastIndexOf(' '));
  return Cron.create((char *)timing.c_str(),(custom_cron.endsWith("on")?turnON:turnOFF),false);
}
```

Example for writing to config:

```
String cron_key(int num){return "Schedule_" + String(num);}

  void
 cron_write_to_setting(){ int
 size_of_array = 2;
 String data[ size_of_array ] ={“*/15 * * * * * on”, “0 */2 * * * * off” };
        for(int i =0;
     i<size_of_array;++i){ preferences.putString(cron_
     key(i).c_str(), data[i]);
 }
```

The last problem is to communicate with the web page, so the web page can get and set the scheduled task. For getting a schedule, a new API endpoint, “getSchedule”, is added. It will return a JSON with a list of modified cron expressions.

The JSON is generated by the following code:

```
  String
cron_get_list_json(){ prefere
nces.begin("setting"); String
json = "[";
int i = 0;
while(preferences.isKey(cron_key(i).c_str())){
   String data = preferences.getString(cron_key(i).c_str());
   if(i) {json += ",";};
   json += "{\"value\":\"";
   json += data.c_str();
   json += "\"}";
   ++i;
}
json += "]";
preferences.end();

return json;
```

For setting the schedule, a new API endpoint, “responses_setSchedule”, is added. This endpoint received a table of modified cron expressions and set them to the cron. Because it might contain a large number of data, POST with a serialized array is used.

In the Web page, a new table with id “schedule-table” is created for showing and editing the schedule. When the user adds a new task to the table, the JavaScript code will create a string in the modified cron expression and store it with the data-* attribute. When the JavaScript code needs to send the post request, it will read from the HTML data-* attribute, the code is as follows:

```
function applyTable(){
 var entry = $("#schedule-table tbody tr");
 var data = [];
 var num = 0;
     for(var item of
  entry){ temp =
  { name:num.toString(10),
  value:item.dataset.cron
  }
  data[num++]=temp;
 }
 var id = $("#client").val()
 var url = $("#client option[value='"+id+"']")[0].dataset.ip
 $.post("http://"+url+"/setSchedule",data);
}
```

The API endpoint function is as follows:

```
String cron_key(int num){return "Schedule_" + String(num);}

    void responses_setSchedule (AsyncWebServerRequest
  *request){ Serial.println("\nPOST:\t\tSet Schedule");
  preferences.begin("setting");
  cron_clear();
  int i = 0;
  while(request->hasParam(String(i).c_str(), true)){
    String data = request->getParam(String(i).c_str(), true)->value();
    cron_add(data);
    preferences.putString(cron_key(i).c_str(), data);
    ++i;
  }
  preferences.end();
  AsyncWebServerResponse *response = request->beginResponse(200);
  response->addHeader("Access-Control-Allow-Origin","*");
  request->send(response);
}
```

This detailed code for this part can be found in “service_cron.cpp” under the folder ”/Switch/src”

The testing at this stage involves configuring the WiFi setting through the web browser then connecting more than one device to the same local network to test the function of controlling multiple switches. Finally, set a timed task through the pages. The result of both tests is good, we can connect two switches to one LAN, turn on the switches separately, and both of them together after 1 minute using the timed task and switch them off remotely in the web browser.

2.2.3.1 Evaluation of setting up a configuration system and implementing other features

Expected outcome:

Set up a configuration system, implement other features, and create a user interface for ELSA so that ELSA can discover new devices and the user can configure WiFi settings and set automated tasks

The actual outcome:

Users can configure WiFi settings through the web server. If there is no configuration, it will become a WiFi access point and let users access the setting page.

Some User-friendly features have been implemented. Users can access with a domain name instead of the IP address. The device can also discover a new device by itself and let the user control them individually.

One advanced feature has been implemented. Users can set simple automated tasks through the web page. To conclude, it meets the expected outcome.

The biggest challenge was finding a way to control multiple switches. For other tasks, the communication involves two devices and the address is fixed. However, this task involves more than one device with unknown addresses. The first idea should have worked if multicast-DNS is implemented in the DNS lookup process, but unfortunately, it didn’t. One way to solve this problem is modifying the library, but it takes time and further knowledge to analyze the code. Therefore, we used an approach that is similar to the DHCP, which is less time-consuming and easier.

2.3 ELSA Evaluation & Discussion

Our main objective was to create an external switching device that does not require modification of the current switch. This allows users to control their light switch remotely without any re-wiring during installation. Also provide IoT-related features, like schedule and timed lighting.

The first thing we evaluated was the mechanism and frame. First, it needs to be able to stick to the current switch and toggle the switch. According to the result in Subsection 2.2.1, the first designed model can stick to the European-style rocker switch and is able to toggle the light switch using a motor.

Second, it should support different kinds of switches. In the final task of Subsection 2.2.1, a modified version is made and the Australian rocker switch is also supported. At the end of the task, the two most common switches in Hong Kong are supported.

The next thing to evaluate is the device's hardware and its control program, the program should be able to control the hardware to toggle the light switch accurately. This needs the web server created in subsection 2.2.3 because most of the main functions such as remote on/off and calibrate needs to be called in the web browser. Therefore the second and third objectives, to program the microprocessor to link ELSA to a web server and control ELSA through a web browser, and to set up a configuration system, implement other features, and create a user interface for ELSA, respectively, need to be evaluated together. As stated in testing results in subsection 2.2.2, the device can refer to the tilting position marked in calibration to turn on and off remotely and manually. This means the hardware and the program work as intended. When more than one device was connected to the same local network the system was able to control multiple switches. Finally, a timed task demonstrated that the switches could function on a timer. We can connect two switches to one LAN, turn on the switches separately, and together both of them after 1 minute using the timed task and switch them off remotely in the web browser.

Overall, the development achieves the main objective. We made an IoT lighting device that can install easily by attaching to existing switches and control lights remotely and automatically. The functions it provides are not as many as the products mentioned in the literature review but it solves the inconvenient installation problem.

SECTION 3—CONCLUSION

This project aimed to design an external device that sticks to an existing switch to add IoT-related functions to the switch. We used Fusion 360 modelling and simulation software to create a simplified model. Then we used 3D printing technology to build the switching mechanism including two frames to fit European and Australian switches. We wrote the program to handle requests, to report switch status and to control the switch on and off using C++ with open source frameworks and libraries. We also configured a system and user interface. In the end, two switching mechanisms were designed and fabricated, so the two common types of switches in Hong Kong are supported and the program allows remote control in LAN, configures WiFi settings, and schedules tasks.

However, this is not a complete product yet. There are still a lot of flaws that should be fixed and improved.

For the switching mechanism and the model, four things could be improved. First, the models designed did not go through any calculation. We believe that the model can be more simplified and more efficient if there is a more accurate analysis. Second, the product printed by a 3D printer using PLA can be worn down easily. This part can be improved by using another material or adding some mechanical parts like a bearing. Third, the kind of supported switch. Currently, it supports only two kinds of switches which is far not enough for daily usage. Fourth, the current model does not include a slot or space to hold the electronic components, electrical circuit, and wires in place. For example, the tilt module is fixed on the 3D print by double-side adhesive tape. It is not secure and will affect the accuracy of the readings.

For the program, two things can be done. First, redesigning the system for handling more than one switch. The current system solves the problem, but it can be improved. The current page will send a request to each switch when it is loading which can cause a problem. And it can only solve by redesigning the system. Second, add more features. There are a lot more features that should be implemented. For example, adding the “Wake on Wireless LAN” feature to save power. However, this might require changing the development board and rewriting some code.

Commented [22]: Restate the project objective(s).

Commented [23]: Briefly state the methodology used.

Commented [24]: State your main results.

Status logs are another feature that should implement. Users can make use of these statistics to show daily or weekly usage to let users understand their using habits.

Commented [25]: Communicate work or future that could be done.

REFERENCES

[1] B. Zhou, et al. “Smart home energy management systems: Concept, configurations, and scheduling strategies.”, Renewable and Sustainable Energy Reviews, Vol. 61, P.30-40, Aug 2016.

[2] R. Crist, “Smart bulbs vs. smart switches: The pros and cons of connected lighting”, Cnet, Sep 2015, Available: https://www.cnet.com/news/smart-bulbs-vs-smart-switches-the-pros-and-cons-of-connected-lighting/

[3] iDISRUPTED, “How Do Smart Switches Work”, Jan 2019, [Online]. Available: https://idisrupted.com/how-do- smart-light-switches-work/#cheaper-electricity-bills

[4] iDISRUPTED, “Smart Bulbs Vs Smart Switches”, Nov 2018, [Online]. Available: https://idisrupted.com/smart-bulbs-vs-smart-switches/#Which_To_Choose_%E2%80%93_Smart_Switch,_Dimmer,_Bulb_Or_Plug

[5] K. Sandburg, “Smart Lighting & Wireless Future”, Strategy Dynamic, Sep 2018, Available: https://medium.com/strategy-dynamics/smart-lighting-wireless-future-9b67eca73984

[6] Candy House, "CandyHouse," [Online]. Available: https://candyhouse.co/pages/sesame-features.

[7] August Home, "Wi-Fi Smart Lock,", [Online]. Available: https://august.com/products/august-wifi-smart-lock. [8] O. K. Cheng and M. C. IP, “FYP - Smart Light Switch“, Apr 2021, [Source code]. Available: https://github.com/JobyCheng/FYP-Smart-Light-Switch

[9] Espressif Systems,“ESP32-WROOM-32 Datasheet”, [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf [10] Anonymous, “GY25z User Manual”, Available: https://drive.google.com/file/d/1bTs86isK0wGnsllb6imhOohNSqhkRfK9/view?usp=sharing

[11] Espressif, et al. “Arduino core for the ESP32“, [Source code]. Available: https://github.com/espressif/arduino-esp32

[12] Last Minute Engineers, “Create A Simple ESP32 Web Server In Arduino IDE”, [Online]. Available: https://lastminuteengineers.com/creating-esp32-web-server-arduino-ide/

[13] PlatformIO Labs, “PlatformIO”, [Online]. Available: https://platformio.org

[14] me-no-dev, et al. “ESPAsyncWebServer”, [Source code]. Available: https://github.com/me-no-dev/ESPAsyncWebServer

[15] w3schools, “How TO - Responsive Top Navigation”, [Online]. Available: https://www.w3schools.com/howto/howto_js_topnav_responsive.asp
[16] w3schools, “How TO - Menu Icon”, [Online]. Available: https://www.w3schools.com/howto/howto_css_menu_icon.asp

[17] OpenJS Foundation, “jQuery”, [Online]. Available: https://jquery.com

[18] OpenJS Foundation, “Node-RED”, [Online]. Available: https://nodered.org

[19] Martin-Laclaustra, “CronAlarms“, [Source code]. Available: https://github.com/Martin-Laclaustra/CronAlarms

[20] Wikipedia, “cron”, [Online]. Available: https://en.wikipedia.org/wiki/Cron#CRON_expression

APPENDICES

Appendix A – Final Project Schedule

Table 9. ELSA schedule.

Objective Statements	Task	Group Member in charge	WK31 Oct	WK48 Oct	WK525 Oct	WK61 Nov	WK78 Nov	WK815 Nov	WK922 Nov	WK1029 Nov	WK116 Dec	WK1213 Dec	WK1320 Dec	WK1427 Dec	WK153 Jan	WK1610 Jan	WK1717 Jan	WK1824 Jan
Mechanism and frame design																		
	Research	CHAN Tai Man CHAN Siu Ming																
	Fabricate mechanism	CHAN Tai Man CHAN Siu Ming																
	Fabricate outer frame for European switch	CHAN Siu Ming																
	Testing the device	CHAN Tai Man																
	Fabricate outer frame for Australian switch	CHAN Tai Man CHAN Siu Ming																
Set up Hardware and Software																		
	Set up hardware	CHAN Tai Man																
	Define sensor functions	CHAN Tai Man																
	Define major functions	CHAN Tai Man																
	Set up simple web server	CHAN Tai Man																

Continued on the next page

Commented [26]: Ensure table numbers and headings are used in this section.

Table 10. ELSA schedule continued.

Commented [27]: Ensure table numbers and headings are used in this section.

Objective Statements	Task	Group Member in charge	WK1824Jan	WK1931Jan	WK207Feb	WK2114Feb	WK2221Feb	WK2328Feb	WK247Mar	WK2514Mar	WK2621Mar	WK2728Mar
Set up Hardware and Software (continued)												
	Set up hardware	CHAN Tai Man										
	Define sensor functions	CHAN Tai Man										
	Define major functions	CHAN Tai Man										
	Set up simple web server	CHAN Tai Man										
Configuration and advanced features												
	Setting up the web server	CHAN Siu Ming										
	Designing a web page	CHAN Siu Ming										
	Connect to WiFi	CHAN Siu Ming										
	Host DNS and multicast-DNS	CHAN Siu Ming										
	Multi-client in LAN	CHAN Tai Man CHAN Siu Ming										
	Scheduling and configuration of tasks	CHAN Tai Man CHAN Siu Ming										

Appendix B – Budget

Table 11. Budget.

Items	Cost
ESP-32 Devkit V1 development board	~RMB 26
GA12-N20 (Motor) (gear reduction rate 1:250, with encoder)	~RMB 30
DRV8833 (H-bridge Motor Driver)	~RMB 0.98
GY25z (tilt module)	~RMB 21.5
Hexagon Shaft	HKD 18 (for 4 pieces)
SAFT LS14250 (3.6V Battery)	~RMB 13
Total	RMB 91.48 HKD 18

Appendix C – Meeting Minutes

Meeting 1 Date: 3/9/2020 Time: 10:30am

Location: Zoom meeting

Attendees: CHAN Tai Man CHAN Siu Ming
None

Minutes taken by: CHAN Tai Man

- We all agree to use the laser as the optical communication source
- The main objective of the project is confirmed
- IP suggested to use solar panels as signal receivers and position sensors
- The alignment method is decided (3x3 solar board represent different directions) Table 1.

Action Items from the Previous Meeting

Action Item to be completed	By when	By whom	Status
UOWC Research	Sept 3rd	All	Completed
Proposal Report introduction	Sep 12th	CHAN Tai Man	In progress
Literature review	Sep 12th	CHAN Siu Ming	In progress

Table 2. Action Items for Next Meeting

Action Item to be completed	By when	By whom
Proposal Report objectives 1,2	Sep 14th	CHAN Tai Man
Proposal Report objectives 3,4	Sep 14th	CHAN Siu Ming
Finalize the proposal report	Sep 16th	All

Next Meeting: Oct 9, 15:00, Zoom

Meeting 2

Date: 9/10/2020 Time:
3:00pm

Location: Zoom meeting

Attendees: Professor A, CHAN Tai Man CHAN Siu Ming
Absent: None

Minutes taken by: CHAN Tai Man

- Professor A commented that without the help from TAs, the UOWC project may be too complicated for us, we might not be able to finish it. Therefore he suggested we propose another topic.
- We both interested in IoT projects, Professor A suggested we can make home automation products, like smart door lock
- We decided to make a smart light switch and we will start do research work

Table 1. Action Items from the Previous Meeting

Action Item to be completed	By when	By whom	Status
Proposal Report introduction	Sep 12th	CHAN Tai Man	Completed
Literature review	Sep 12th	CHAN Siu Ming	Completed
Proposal Report objectives 1,2	Sep 14th	CHAN Tai Man	Completed
Proposal Report objectives 3,4	Sep 14th	CHAN Siu Ming	Completed
Finalize the proposal report	Sep 16th	All	Completed

Table 2. Action Items for Next Meeting

Action Item to be completed	By when	By whom
Research on light switch device mechanism	Oct 30th	All

Next Meeting: Nov 2, 13:00, Zoom

Appendix D – Group Members' Contribution

CHAN Tai Man

I have taken research on using a motor to toggle the switch. I discovered a small DC gear motor (N20) that can provide large torque, so I decided to use it in the first prototype. In the first prototype, I used a motor with a gear reduction rate of 1:100 and a stick whose one end is inserted with the motor shaft and the other end is tied up with a thread and is stuck to one side of a light switch. The torque of the motor gives force to push or pull the side of the light switch to complete the on/off behaviour mechanically. This prototype can barely press and pull the light switch, a larger gear reduction gearbox is needed, and it works with the motor with a gear reduction rate of 1:250. The advantage of this design is it does not need to cover the whole light switch (compared with the current design) so the size of the smart device can be smaller. But the force arm of this design is short so it does not convert the torque into push/pull force efficiently, also the position of the force point needs to be precise (better as close to the edge of the switch), this may cause inconvenience to the user when installing the device. Therefore, I redesign the mechanism into the current see-saw design, the force arm is much longer and no need to consider the pulling part.

When building the actual 3D printed prototype to finish object 1, I have suggested printing a component to connect the D-shaft and hexagon instead of using gears to get a simplified prototype. I keep improving the whole design, such as adding a motor encoder to increase the accuracy of the motor rotation and the state of the switch, utilizing the touch sensor provided by ESP32 to perform 'manual switching' when we find the motor gearbox locks the shaft and cannot press the 3D print part by hand. Also, I was responsible for searching and buying suitable components for the device (3D print shaft, motor, sensors and MCU modules).

After planning and building the circuit. I followed the sample code online and built a simple web server. I spent quite a lot of time thinking about the method to do the calibration. My first idea is to add a new webpage and create four buttons inside, they are used to rotate the motor clockwise, anticlockwise and tell the MCU to record the on/off roll value. But it is too complicated to program the page and the GET functions and there are noticeable delays after pressing buttons on the web page. Then I created the current method which is more direct and simple. Then I encountered the watchdog problem. This let me know we should keep the task in interrupt routines as short as possible and using serial functions in interrupt routines will stall the program.

For objective 3, I was responsible for testing and reporting bugs of the web user interface on different platforms (PC, Android, IOS). I also tested using multiple switches.

CHAN Siu Ming

Currently, we have finished the first objective of the project. We are currently planning for the second and third objectives. My contribution is focused on 3D model design and printing. I spend quite a lot of time on design and printing.

In the beginning, I tried to use"FreeCAD", an open-source software, to design the model. However, after I finish the simplified model of the switch, I notice that it cannot simulate some physical joint and collision. I think it is a huge problem, cause we are printing some moving parts. If there is no simulation, I cannot know if the parts move correctly until printing it out. Therefore, I did some research and figured out that "Fusion360", a cloud-based CAD. It provides simple physics moving simulation which meets our requirement and also provides a student license.

Then, the printing process of the middle part. At first, we did not do any test print. We printed the middle part that connects to the hexagon shaft and it fails. We tried a few more with a different setting in the printer, but they also failed. We do some searching online and notice the problem may be on the horizontal expansion of the material.

Because we are printing 2mm diameter holes, 0.1mm expansion can affect a lot. Usually, the problem can be fixed by setting parameters in the slicing software, but I did not see this parameter in slicing software for “UP Box+”. To fix

the problem, I need to manually tweak the model. To find out the offset distance, we do a test print that prints 5 holes from 1.6 mm to 2.4mm, so we can know the correct offset. Finally, we get the middle piece with the correct hole size and I apply the same offset for the part that connects the D-shaft and hexagon shaft.

After that, is the outer frame. In the task, I break the model into 2 pieces. There are two slots on both sides for combining the pieces. The reason I am doing this is that I don’t want to reprint the whole frame if the hole for the motor does not fit. If I print it in 2 pieces, I can replace the one that doesn’t fit the motor easily. One small detail is that I set the offset of the connecting slots a little bit larger as the slot does not affect the result. Even though they are loose, they can still align the 2 pieces. Luckily, the first print was a success, and the hole fit the motor perfectly, so we do not need to print another one.

For objective 3, I found some useful libraries that might help us finish the task. For example, “Preferences.h” for saving some important settings, like wifi SSID and password. The possibility of using an async server instead of the normal way for better code reading. Also the DNS and multicast-DNS setting, so we can use a better address name for accessing the device.

Then, I also think of a way to control multiple switches. The first idea using multicast-DNS should work, but I did expect that multicast-DNS is not implemented. Therefore, I start to research other ways of doing this. I remember DHCP having a similar case, where the server and client have no information about each other's mac address and ip address. I look at how it works and use it as a reference to create a way to control multiple switches. I look at different resources to understand how the library works and the way to test them. As a result, I found a software -- node-red, it set up UDP or TCP socket easily.

Appendix E – Deviation(s) from the proposal and progress report and supporting reason(s)

In the beginning, we try to use electromagnetic as the main method to toggle the switch. However, after the component arrives, we notice that the component doesn't work as we expected. The electromagnet on the market is specialized for other purposes. It specializes in attractive force in a short distance, the trade-off is a weak repulsive force. This is an unexpected situation. We think of a few backup plans and choose the most reliable one which is swapping to the motor approach. Because we need a frame that works to start the next objective.

Appendix F – Monthly Reports

Monthly Report for ECE FYP/FYT

Project Code:	AB04a-20	**Supervisor(s):**	**Prof A**
Project Title:	**Underwater Communication for Robots**		
Group Member(s):	**1)** CHAN Tai Man **2)** CHAN Siu Ming **3)**		
Reporting Period: **According to your FYP study pattern, select the reporting period. (Refer to the table in the guidelines)**	**Report #1** ☒ **Oct (Fall)** **Report #2** ☐ **Nov (Fall)** **Report #3** ☐ **Jan (Winter)** **(please attach Reports #1-3 to the Progress Report to be submitted in Feb)**		

Project Code:	**AB04a-20**	**Supervisor(s):**	**Prof A**
Project Title:	**Smart Light Switch**		
Group Member(s):	1) CHAN Tai Man 2) CHAN Siu Ming 3)		
Reporting Period: According to your FYP study pattern, select the reporting period. (Refer to the table in the guidelines)	**Report #1** ☐ **Oct (Fall)** **Report #2** ☒ **Nov (Fall)** **Report #3** ☐ **Jan (Winter)** **(please attach Reports #1-3 to the Progress Report to be submitted in Feb)**		
Progress Report: List the work completed in this reporting period. Identify the major difficulties encountered. Comment on the overall progress.	**We have built the second prototype which is using a motor to control the switch.** **https://drive.google.com/file/d/18AZEr64DdAgEUOds5Jx9R7U1rjYa-_G5e/view?usp=sharing** **In this video, the motor is operated by around 7V and the gear ratio is 1:250, it needs larger torque to ensure it works constantly and the material of the stick needs to be tougher otherwise it will break.** **After comparing the two approaches, we decided to choose the magnet approach as it has a few advantages compared to the motor one.** **It produces less noise.** **Power can be adjusted by tweaking the current and voltage of the electromagnet.** **nodel can be modified to fit on different switches (motor approach cannot switch small switches well)**		

Future Plan: **Write down the working plan for the next reporting period.**	**Buying the components for the next prototype, including magnet, electromagnet, Arduino, wifi module for Arduino.** **Use 3D printing to print the parts out.** **Start building circuit and programming Arduino to change output in GPIO, DAC with a web server.**
Group Representative's Signature:	CHAN Tai Man CHAN Siu Ming

Project Code:	**AB04a-20**	**Supervisor(s):**	**Prof. A**
Project Title:	**Smart Light Switch**		
Group Member(s):	**1)** CHAN Tai Man **2)** CHAN Siu Ming **3)3)**		
Reporting Period: **According to your FYP study pattern, select the reporting period. (Refer to the table in the guidelines)**	**Report #1** ☐ **Oct (Fall)** **Report #2** ☐ **Nov (Fall)** **Report #3** ☒ **Jan (Winter)** **(please attach Reports #1-3 to the Progress Report to be submitted in Feb)**		
Progress Report: **List the work completed in this reporting period. Identify the major difficulties encountered. Comment on the overall progress.**	**We found several problems when we tried to build the electromagnetic approach prototype.** **Recent electromagnets sold in the market can only attach metal things when they are very close with each other and cannot generate repulsive force. We need to make coils by ourselves to achieve expected results, but it will be hard and have many errors.** **The magnetic field of electromagnets may affect other modules (e.g., WIFI module) and cause damages.** **We think the electromagnet approach is unachievable, so we started working on the motor approach. Similar to the electromagnetic approach, the motor turns the center of a board like a seesaw to flip the switch. We are working on the following things.** **Making 3D print parts** **Finding better motor and components** **Writing Arduino web control program**		
Future Plan: **Write down the working plan for the next reporting period.**	**Finish the prototype, at least the switch can be controlled by the web page and is able to flip large surface light switches.**		

Group Representative's Signature:	**CHAN Tai Man** **CHAN Siu Ming**

Appendix G – Source code for 2.2.2 Task 2

```
#include <WiFi.h>
#include <WebServer.h>

/*Put your SSID & Password*/
const char* ssid = " YourNetworkName";  // Enter SSID here
const char* password = " YourPassword";  //Enter Password here

WebServer server(80);

uint8_t LED1pin = 4;
bool LED1status = LOW;
uint8_t LED2pin = 5;
bool LED2status = LOW;

void setup() {
  Serial.begin(115200);
  delay(100);
  pinMode(LED1pin, OUTPUT);
  pinMode(LED2pin, OUTPUT);

  Serial.println("Connecting to ");
  Serial.println(ssid);

  //connect to your local wi-fi network
  WiFi.begin(ssid, password);

  //check wi-fi is connected to wi-fi network
  while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected..!");
  Serial.print("Got IP: ");  Serial.println(WiFi.localIP());

  server.on("/", handle_OnConnect);
  server.on("/led1on", handle_led1on);
  server.on("/led1off", handle_led1off);
  server.on("/led2on", handle_led2on);
  server.on("/led2off", handle_led2off);
  server.onNotFound(handle_NotFound);

  server.begin();
  Serial.println("HTTP server started");
}
void loop() {
  server.handleClient();
  if(LED1status)
  {digitalWrite(LED1pin, HIGH);}
  else
  {digitalWrite(LED1pin, LOW);}

  if(LED2status)
  {digitalWrite(LED2pin, HIGH);}
  else
  {digitalWrite(LED2pin, LOW);}
}

void handle_OnConnect() {
  LED1status = LOW;
  LED2status = LOW;
  Serial.println("GPIO4 Status: OFF | GPIO5 Status: OFF");
```

```
  server.send(200, "text/html", SendHTML(LED1status,LED2status));
}

void handle_led1on() {
  LED1status = HIGH;
  Serial.println("GPIO4 Status: ON");
  server.send(200, "text/html", SendHTML(true,LED2status));
}

void handle_led1off() {
  LED1status = LOW;
  Serial.println("GPIO4 Status: OFF");
  server.send(200, "text/html", SendHTML(false,LED2status));
}

void handle_led2on() {
  LED2status = HIGH;
  Serial.println("GPIO5 Status: ON");
  server.send(200, "text/html", SendHTML(LED1status,true));
}

void handle_led2off() {
  LED2status = LOW;
  Serial.println("GPIO5 Status: OFF");
  server.send(200, "text/html", SendHTML(LED1status,false));
}

void handle_NotFound(){
  server.send(404, "text/plain", "Not found");
}

String SendHTML(uint8_t led1stat,uint8_t led2stat){
  String ptr = "<!DOCTYPE html> <html>\n";
  ptr +="<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";
  ptr +="<title>LED Control</title>\n";
  ptr +="<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}\n";
  ptr +="body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;} h3 {color: #444444;margin-bottom: 50px;}\n";
  ptr +=".button {display: block;width: 80px;background-color: #3498db;border: none;color: white;padding: 13px 30px;text-
decoration: none;font-size: 25px;margin: 0px auto 35px;cursor: pointer;border-radius: 4px;}\n";
  ptr +=".button-on {background-color: #3498db;}\n";
  ptr +=".button-on:active {background-color: #2980b9;}\n";
  ptr +=".button-off {background-color: #34495e;}\n";
  ptr +=".button-off:active {background-color: #2c3e50;}\n";
  ptr +="p {font-size: 14px;color: #888;margin-bottom: 10px;}\n";
  ptr +="</style>\n";
  ptr +="</head>\n";
  ptr +="<body>\n";
  ptr +="<h1>ESP32 Web Server</h1>\n";
   ptr +="<h3>Using Station(STA) Mode</h3>\n";

   if(led1stat)
  {ptr +="<p>LED1 Status: ON</p><a class=\"button button-off\" href=\"/led1off\">OFF</a>\n";}
  else
  {ptr +="<p>LED1 Status: OFF</p><a class=\"button button-on\" href=\"/led1on\">ON</a>\n";}

  if(led2stat)
  {ptr +="<p>LED2 Status: ON</p><a class=\"button button-off\" href=\"/led2off\">OFF</a>\n";}
  else
  {ptr +="<p>LED2 Status: OFF</p><a class=\"button button-on\" href=\"/led2on\">ON</a>\n";}

  ptr +="</body>\n";
  ptr +="</html>\n";
  return ptr;
}
```

Appendix H – Example file for 2.2.3 Task 2

index.html

```
<html lang="en">
   <head>
      <title>Smart Switch</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet" href="style.css">
      <script src="jquery-3.5.1.min.js"></script>
      <script src="script.js"></script>
   </head>
<body>
   <div class="header">
      <h1>Smart Switch</h1>
   </div>

   <div id="navbar" class="navbar" >
      <a href="#home" id="home" class="tab">home</a>
      <a href="#testing2" id="testing2" class="tab">testing</a>
      <a href="#template" id="template" class="tab">template</a>
      <a href="javascript:void(0);" class="icon" onclick="ResponsiveNav()">
         <div></div>
         <div></div>
         <div></div>
      </a>
   </div>

   <div id="home-content" class="content">
   <!--put code under this div-->
   a long meaningless page to show the sticky top navigation
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><
br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><b
r><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
   end
   <!--put code above this div-->
   </div>

   <div id="testing2-content" class="content">
   <!--put code under this div-->
   testing2
   <!--put code above this div-->
   </div>

   <div id="template-content" class="content">
   <!--put code under this div-->
   template
   <!--put code above this div-->
   </div>

</body>
</html>
```

script.js

```
// Responsive Webpage
function ResponsiveNav() {
  $("#navbar").toggleClass("responsive");
}

function openTab() {
  var tab_id = location.hash.replace("#", "");

  $(".active").removeClass("active");
  $("#" + tab_id).addClass("active");

  $(".content").css("display","none")
  $("#"+tab_id+"-content").css("display","block")

  if ($("#navbar.responsive").length == 1) { $(".navbar .icon").click(); }
}
// End Responsive Webpage

window.onhashchange = openTab;

//shorthand document.ready function
$(function () {
       if (location.hash == "")
    { location.hash =
    "#home"
       } else
    { openTab(
    );
```

style.css

```
.header {
  background-color: #f1f1f1;
  padding: 30px;
  text-align: center;
}

.navbar {
 overflow: hidden;
 background-color: #333;
 position:sticky;
 top: 0;
}

.navbar a {
 float: left;
 display: block;
 color: #f2f2f2;
 text-align: center;
 padding: 14px;
 text-decoration: none;
}

.navbar a:hover {
 background-color: #ddd;
 color: black;
}

.navbar a.active {
 background-color: #4CAF50;
 color: white;
}

.navbar .icon {
 display: none;
}

.navbar .icon div {
  width: 20px;
  height: 2px;
  background-color: white;
  margin: 3px 0;
}

@media screen and (max-width: 600px) {
 .navbar a:not(.active) {display: none;}
 .navbar a.icon {
  float: right;
  display: block;
 }
}

@media screen and (max-width: 600px) {
 .navbar.responsive a.icon {
  position: absolute;
  right: 0;
  top: 0;
 }
 .navbar.responsive a {
  float: none;
  display: block;
  text-align: left;
 }
}
```

```
  .content
{ display: none;
padding: 16px;
```