IOT INTEGRATED ANTENNA ROTATOR

CHONG JIA XIN

Bachelor of Electronics Engineering Technology
(Computer System) with Honours

UNIVERSITI MALAYSIA PAHANG

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Bachelor of Electronics Engineering Technology (Computer System) with Honours.

_____

(Supervisor's Signature)

Full Name :

Position :

Date :

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

_____

(Student's Signature)

Full Name : CHONG JIA XIN

ID Number : TG18001

Date : 31 January 2022

IOT INTEGRATED ANTENNA ROTATOR

CHONG JIA XIN

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Electronics Engineering Technology (Computer System) with Honours

Faculty of Electrical & Electronics Engineering Technology

UNIVERSITI MALAYSIA PAHANG

JANUARY 2022

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Ir. Dr. Rosmadi Bin Abdullah, who has always impressed me with his outstanding professional conduct. I greatly appreciate his invaluable supervision and tutelage throughout my SDP. His insights on the project's development, particularly from the standpoint of a professional engineer, answered a lot of my concerns. I consider myself fortunate to have the opportunity to work with such an inspiring supervisor like him.

Very special thanks to Dr. Syukran Hakim Bin Norazman for his treasured guidance which was enormously influential and helpful in shaping my software development skills. I am truly grateful for his enlightening advise and providing opportunities for me to grow professionally. His profound knowledge and professional expertise in software engineering enabled me to effectively complete this project. It is always an honor to learn from Dr. Syukran.

My appreciation goes to the Faculty of Electrical & Electronics Engineering Technology, Universiti Malaysia Pahang for providing me with the funding assistance to complete this project. Additionally, I would like to thank my teammate, Dira Nadia Binti Padzil for her co-operation while conducting this SDP. I also want to express my gratitude to my family and friends for their consistent encouragement which was important for the successful completion of my studies.

ABSTRACT

This project deals with the design and fabrication of an IoT Integrated Antenna Rotator to upgrade an existing antenna rotator by integrating it with Internet of Things (IoT) technology. The objective of this thesis is to integrate Google Maps into web server with the antenna rotator system in order to achieve a speedy and high accuracy standalone antenna pointing system. This thesis describes the software development of the rotator system which includes the specification, system and software design, implementation and unit testing, integration and system testing, and finally the operation. NodeMCU ESP32 is implemented as the microcontroller in this SDP to control the system by using its central processor. A web server is developed in order to act as a user interface to interact with the antenna rotator system. Digital map is integrated with the system to attain IoT technology by using Google Maps API. The IoT Integrated Antenna Rotator is tested upon the integration and fabrication process. The antenna rotator is able to rotate to a desire location precisely through web server by using Google Maps API. The outcomes can result in significant cost and time savings, as well as increase the product reliability and user confidence in using IoT Integrated antenna pointing system.

ABSTRAK

Projek ini membentangkan reka bentuk dan fabrikasi IPB Integrasi Pemutar Antena untuk menaik taraf pemutar antena sedia ada dengan menyepadukannya dengan teknologi Internet Pelbagai Benda (IPB). Objektif tesis ini adalah untuk mengintegrasikan Google Map dengan sistem pemutar antena untuk mencapai sistem penunjuk antena *stand-alone* yang pantas dan ketepatan tinggi. Tesis ini menerangkan pembentukan dan pembangunan sistem pemutar yang merangkumi spesifikasi, reka bentuk sistem dan perisian, pelaksanaan dan ujian unit, integrasi dan ujian sistem, dan akhirnya operasi. NodeMCU ESP32 dilaksanakan sebagai mikropengawal dalam projek ini untuk mengawal sistem dengan menggunakan pemproses pusatnya. Pelayan web dibangunkan untuk bertindak sebagai antara muka pengguna untuk berinteraksi dengan sistem pemutar antena. Peta digital disepadukan dengan sistem untuk mencapai teknologi IPB dengan menggunakan API Peta Google. IPB Integrasi Pemutar Antena diuji atas proses penyepaduan dan fabrikasi. Pemutar antena dapat berputar ke lokasi yang diinginkan dengan tepat melalui pelayan web dengan menggunakan API Peta Google. Hasilnya dapat mencapai penjimatan kos dan masa dengan ketara, serta meningkatkan kebolehpercayaan produk dan keyakinan pengguna dalam menggunakan sistem penuding antena bersepadu IPB.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application programming interface
CSS	Cascading Style Sheets
CLI	Command-line interface
GUI	Graphical user interface
HTML	HyperText Markup Language
HTTP	Hypertext transfer protocol
IDE	Integrated development environment
IoT	Internet of things
JS	JavaScript
JSON	JavaScript object notation
LCD	Liquid crystal display
MCU	Microcontroller
PWM	Pulse-width modulation
RAM	Random-access memory
ROM	Read-only memory
SDP	Senior design project
SPI	Serial peripheral interface
SSID	Service set identifier
URL	Uniform resource locator
UART	Universal asynchronous receiver-transmitter
UX	User experience
UI	User interface
WLAN	Wireless local area network
WLAN	Wireless local area network

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

This chapter discussed the project background, problem statement of the project, objectives of the project, research scope and organization. The focus of this thesis is to develop an IoT Integrated Antenna Rotator which is able to transmit and receive signals in an effective way. In order to process the signals in different directions, the antenna needs to rotate to the appointed position. The existing antenna rotator is manually driven and it is less user friendly. As a solution, an IoT integrated antenna rotator that can regulate the antenna position using software is needed to be developed to ease the antenna orientation process.

1.1 Project Background

The performance of the antenna has a significant effect on the telecommunication system sustainability (Junfithrana et al., 2017). The direction of the antenna pointing area is crucial in receive or transmit the signal in the particular location. The majority of direction-finding antennas work throughout a broad frequency spectrum and estimate the direction of arrival of an incoming electromagnetic field using a 2-dimensional angular coverage (for instance, only the azimuth angle is estimated for a restricted range of elevation angles) (Duplouy et al., 2019). It would be preferable if the antenna's orientation could be controlled by an IoT integrated antenna rotator. The rotator is mounted beneath the antenna and connected to a controller and internet. The rotator is able to assist the antenna in receiving or transmitting signal in various directions without exerting additional forces.

The Internet of Things (IoT) is a new technology which allows a worldwide network of machines, appliances and devices capable of interacting and exchanging data with each other. The IoT is known as one of the greatest forthcoming technologies and is

gaining enormous attention from various industries (I. Lee & Lee, 2015). With IoT integration, most of the process is going through digital transformation. This allows a system to operate at an accelerated level and speed because it is connected to a software which could transfer information through a network and analyze the data in a short time. Hence, IoT is implemented in the antenna rotator system and is able to upgrade the existing antenna rotator into a speedy and high accuracy antenna pointing system.

The IoT Integrated Antenna Rotator is concerned with the design and development of an antenna rotator that will allow the antenna to be positioned according to the chosen azimuth. The rotator is able to be controlled automatically via software through internet. The user is able to alter the orientation of the antenna by selecting the desired location from the web server.

1.2 Problem Statement

The direction of the antenna plays a crucial role in obtaining signals. The antenna must point precisely to the area to transmit or receive a signal from a certain location. The existing antenna rotator is able to be controlled manually or remotely. However, the existing product requires the user to know the direction or coordinate of the desired signal source or destination, which causes the system to become less user-friendly and the direction of the antenna pointing area is not accurate. Besides, this will reduce the efficiency of the signal communication system since the transmitted or received signal cannot be sent to or accepted from the desired location accurately. Therefore, a IoT integrated antenna pointing system is required to overcome the issues. A 24VAC rotator is used as the primary driver in the antenna system for orientation and speed control, along with the necessary mechanical coupling. NodeMCU ESP32 is used as a controller to control the signal to shift the motor and stop in time in order to regulate the direction of the antenna rotator. A proper control algorithm must be built and implemented on the controller to provide the position control command in order to achieve a decent response. Also, a Graphical User Interface (GUI) is implemented into this system to achieve IoT integration. A digital map is integrated with the system thus the antenna rotator is able to adjust its angle and direction according to the chosen location from the web server. The

antenna rotator device, circuit construction, controller and software application must be correctly constructed and adhere to the antenna's specifications in order to achieve its functionalities.

1.3 Objectives

The main aim for our Senior Design Project is to fabricate an IoT Integrated Antenna Rotator to upgrade an existing antenna rotator with the integration of Internet of Things (IoT) technology. This ultimate aim can be accomplished through the following objectives:

i. To develop Graphical User Interface (GUI) by using web server and allow the system to communicate with microcontroller thru internet.

ii. To integrate Google Maps with the antenna rotator system.

1.4 Research Scope

In the beginning of this project's evolution, several factors were considered to evolve the IoT Integrated Antenna Rotator. For this project, the aim is to construct the IoT Integrated Antenna Rotator by upgrading the existing antenna rotator by integrating it with Internet of Things (IoT).

i. The microcontroller is used to control the angle of the antenna.

ii. Web server is used as a user interface to communicate with the microcontroller.

iii. Google Maps is integrated with the system to achieve IoT technology.

iv. Analyze and evaluate the most suitable material for this project.

1.5 Report Organization

This report consists of 4 chapters as a subtopic and the list of references. The first chapter discusses in detail of the background study of the project, the problem statement, the objectives, the research scope of the project for this SDP 2 and the report organization. For the second chapter, it is focusing more on the literature review that also includes a detailed background and related works written based on the research journals, articles and reports that have been cited and quoted for reference regarding the subjects of this project. Next, the third chapter is research methodology that includes methodology flowchart in addition to research activities such as design, method selection, modelling and simulations to achieve the objectives of this project. Then, the results or outcomes that have been achieved from the testing and analysis are discussed in chapter 4. A detailed explanation on the results can be obtained in this chapter. Lastly, the attachment of the list of references that were referring to when doing the research and analysis and the Gantt Chart as the timeline of the progress for this project.

CHAPTER 2

LITERATURE REVIEW

This chapter provides the literature review of antenna rotator, internet of things, microcontroller, network, software engineering and process, application programming interface, graphical user interface, web server and WebSocket protocol.

2.1 Antenna Rotator

An antenna rotator is typically used when signals arrive at a desired area from widely separated places and the relatively narrow bandwidth of a single fixed-position antenna prevents accurate reception or transmission. Using an antenna rotator, an observer may swiftly position the antenna in the direction of the desired channel's transmission tower or in the direction that guarantees the best reception of a given channel. An antenna rotator frequently enables for the easy "fine-tuning" of the antenna orientation to accept signals from various locations caused by changing atmospheric conditions or other signal-interfering circumstances. Additionally, the antenna rotator enables the reduction of multipath, adjacent-channel, and other types of interference that may be mitigated in some cases by reorienting the antenna slightly. Therefore, the antenna rotator device may be used to rotate a directional antenna. It is composed of two components which are a controller and a rotator. The term "rotator" refers to any machine capable of rotating something. The rotator plays a significant part in the antenna rotator project and must be designed perfectly for smoother movement and rotation. Gearbox and drive (geared motor), azimuth angle encoder, and rotary joint make up a basic motor (Muhammad Rusydi Bin Buchek & Darul Ridzuan, 2014).

Generally, the millimetre-wave rotator assembly was used to design the rotator. The design was particularly emphasizing the coaxial cable and coax rotary joints. Figure 2.1 illustrates the rotator bearing layout.

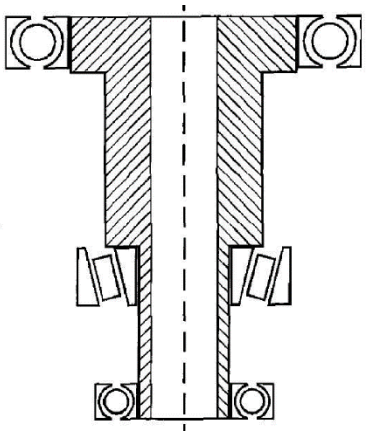


Figure 2.1 Rotator Bearing Layout

Source: (Muhammad Rusydi Bin Buchek & Darul Ridzuan, 2014).

The radial loads are carried by the upper and lower bearings, while the middle bearing offers axial support. The rotating table as known as the "lazy-Susan" turntable was used as a support for several of the antenna rotator projects. The motor shaft would be supported by the rotating table as it rotates (Kivinen et al., 1999).

The Dennard Type 2000 rotator (Figure 2.2) is selected in this project because it is one of the most advanced Pan and Tilts on the market, with clean designs and pressure die cast components to ensure high quality finishes and fits previously unavailable in a unit of this price range, as well as a striking and creative appearance to ensure it will remain the market's preferred option (*Dennard Type 2000*, n.d.) . This rotator was selected and will be attached to the antenna because it has features that are compatible with the antenna as listed as follow:

i. Components are pressure diecast to ensure high quality and tolerance fits.

ii. External friction cap change is shielded for ease of use.

iii. Socketed in depth Extremely strong Aluminium shafts that have been anodized for rigidity.

iv. Pan gearing with anti-backlash.

v. Clean wiring looms are made possible by modular printed circuit board electronics.

vi. Rear connector is angled and recessed for transit safety.

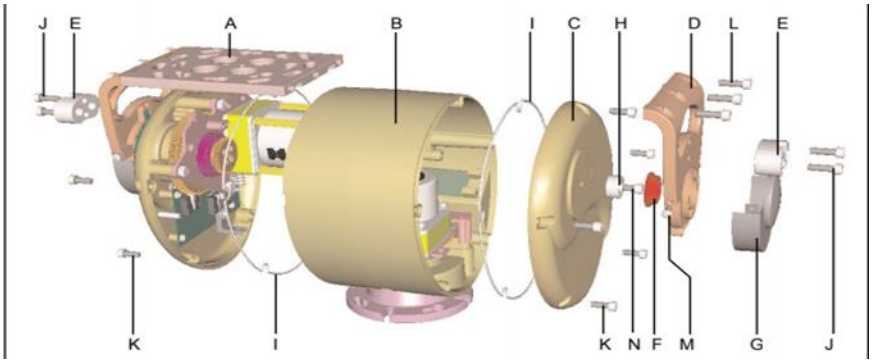


Figure 2.2 General Assembly Type 2000 Pan & Tilt Head

Source: (*Dennard Type 2000*, n.d.)

The Dennard Type 2000 rotator has built in potentiometer to act as a feedback mechanism for this IoT integrated Antenna Rotator system.

2.2 Internet of Things (IoT)

The fourth industrial revolution is currently underway—fourth in the sense that it is inventive and qualitative. Moreover, the quality of the variation could be observed in the integrated management and supervision of the entire manufacturing process, which is unified and agile (Nagy et al., 2018). To stay competitive in a worldwide economy, manufacturing firms require continually update their production methods to meet changing market needs (Pedersen et al., 2016).

The first foundation for industrial digitization is device networking. This is usually referred to as the Internet of Things, a notion that the profession cannot deny. IoT is a term that refers to "mobile devices" that are outfitted with a chip, RFID, sensor, or any other networking device that are capable of sharing and communicating data (Hermann et al., 2016).

With the advent of IoT, smart products develop that can communicate the present condition of production or process monitoring, the process characteristics, and the impending need for maintenance, as well as provide recommendations regarding the nature of the intervention or even intervene the system itself (Atzori et al., 2010; Weyer et al., 2015). With the proliferation of IoT and artificial intelligence, repetitive job is becoming more obsolete. Machines accomplish these operations precisely and at a substantially reduced financial cost (Gubbi et al., 2013). There are several advantages utilizing IoT, including the following:

i. The ability to retrieve data from any location, moment and gadget.

ii. Enhance communication between electronics devices which are connected.

iii. Able to save energy and cost by transferring data packets through a linked network

iv. Automating tasks contributes to the improvement of a business's service quality by removing the requirement for human intervention.

The applications of IoT technologies are several, since they are adaptable to nearly any technology capable of delivering pertinent information about its own performance, the accomplishment of an activity, or even the ambient conditions that need to supervise and regulate remotely.

One of the examples of IoT application is the implementation of IoT in manufacturing or industrial sector. Manufacturing has became the most popular IoT application category in the 2018 research. Technology company such as Microsoft and AWS, as well as significant industrial automation firms such as Siemens or Rockwell Automation, are among the drivers pushing the manufacturing industry's digital transformation. Industrial IoT applications encompass a broad variety of initiatives using linked "things" both within and outside the plant. For example, many IoT-based factory control and automation projects offer holistic smart factory solutions that incorporate a variety of components, including production floor monitoring, wearables and augmented reality on the shop floor, remote PLC control, and automated quality control systems. Remote control of linked machinery, equipment monitoring, and administration and control of complete remote industrial activities such as oil rigs are all examples of typical outside the factory projects. Numerous case studies cite "decreased operational downtime and cost savings" as primary reasons for OEMs to use industrial IoT solutions.

An IoT-enabled environment is also shown by an integrated transportation system that can dynamically route and rearrange itself in response to changing traffic requirements and circumstances (Zanella et al., 2014). In healthcare, IoT has been utilised to monitor patient recovery and to compare it to a number of patient-specific factors using IoT-enabled devices (Chen et al., 2014). Additionally, the data collected may be utilised to compare patient reactions to therapy in a variety of global environmental scenarios. Additionally, smart IoT devices may be utilised to monitor and regulate energy use.

In this project, IoT is integrated with an antenna rotator in order to enable user control and monitor the antenna pointing system remotely through online platform such as web server. This implementation enhances the precision, product reliability and increase the conveniency of the antenna rotator system by integrating with IoT technology.

2.2.1 IoT Integration

Several commercial devices have emerged in recent years that used as IoT general purpose devices, including Arduino, RaspberryPi, and NodeMCU. These devices share three characteristics where they include GPIO (General Purpose Inputs Outputs) ports that enable easy connection of virtually any type of sensors (Inputs) or actuators (Outputs), they include an IP-compatible network interface, typically wired 802.3 Ethernet with RJ45 connector or wireless 802.11 wifi, and they include an open-source operating system that enables development using industry-standard programming standards. In some cases, IoT devices are equipped with alternative LAN (or PAN) network technologies such as ZigBee or Bluetooth. While these technologies may perform better in some areas, such as energy consumption or cost, each of them requires an IP gateway to integrate the network with other devices or to send packets over long distances. In many instances, the manufacturer also provides dedicated gateways for their own IoT devices in order to increase the connection between the IoT devices (Diaz-Cacho et al., 2015).

IoT devices incorporate compact processors that enable them to transmit and receive data using the IP network protocol stack. While some of these devices may have sufficient processing capabilities to modify and process data before it is delivered to the network or after it is received, others encode or retrieve data from IP packet structures.

In order to fabricate this project, a microcontroller is selected to be the processor of the IoT Integrated Antenna Rotator. The microcontroller is required to be equipped with wireless Wi-Fi network in order to achieve IoT integration.

2.3 Microcontroller

Embedded systems instruction is diverse and heterogeneous (Subbian & Beyette, 2013). Firstly, embedded computing systems are developed and constructed at various levels of abstraction such as hardware, system software and application. Moreover, it incorporates principles from a variety of fields, including as electronics, computer science, software engineering, and control theory (Caspi et al., 2005). The embedded system is comprised of a microcontroller and auxiliary hardware components (Illera & Sepulveda, 2015). The instruction is able to be implemented to the microcontroller by program it and the code was written in C using the Arduino IDE.

A microcontroller (abbreviated MCU for microcontroller unit) is a miniature computer contained within a single metal-oxide-semiconductor (MOS) integrated circuit (IC) chip. A microcontroller is composed of one or more CPUs (processor cores), memory, and programmable I/O peripherals. Additionally, a tiny amount of RAM and programme memory in the form of ferroelectric RAM, NOR flash, or OTP ROM are frequently incorporated on the chip. Microcontrollers are intended for embedded applications, as opposed to microprocessors used in personal computers or other general-purpose applications, which are composed of a number of discrete chips.

2.3.1 NodeMCU ESP32

The ESP32 is a dual-core development board that integrates Wi-Fi and Bluetooth wireless capabilities. It integrates a broad range of peripherals, including capacitive touch, ADC, DAC, SPI, UART, PWM, I2C and I2S. The ESP32 is the successor to the ESP8266. It includes an additional touch-sensitive pins for awaking the device from deep sleep condition (*ESP-WROOM-32 Datasheet Espressif Systems*, 2017). The ESP32 consumes very little power because to its power-saving capabilities and it has security features on a microcontroller chip. NodeMCU ESP32 is selected as the microcontroller implemented in this project to achieve the communication between the software and hardware by using Arduino IDE and web server. The features of the ESP32 is listed in Table 2.1.

Table 2.1 NodeMCU ESP32 features

Features/ Properties	NodeMCU ESP 32
Microcontroller	ESP32
Operating Voltage	3.3V
Power supply	7V – 12V
Current consumption	20 mA – 240 mA
Current consumption Deep Sleep	5 µA
Digital I/O Pins	36
Digital I/O Pins with PWM	36
Analog Input Pins	15
SPI/I2C/I2S/UART	4/2/2/2
Flash Memory	4MB
SRAM	520KB
Size (Length x Width)	52mm x 31mm
802.11 b/g/n Wi-Fi	HT 40
Bluetooth	Yes
Touch sensor	10
CAN protocol	Yes
Ethernet MAC Interface	Yes
Security	Boot flash encryption. OTP 1024-bit
Hall effect sensor	Yes
Power jack	No
USB connection	Yes

Figure 2.3 NodeMCU

Source: *Cytron*.

NodeMCU ESP32 as shown in Figure 2.3 is used to receive the input signal and control the output signal in the system. The ESP32 is programmed to determine the direction of the antenna rotator and calculate the bearing to achieve the targeted azimuth angle once receive the coordinate from the user through internet. In this SDP, the standalone ESP32 microcontroller is implemented which allows access to web server from the microcontroller without any custom firmware modifications. This unique feature of the ESP32 grants the system to easily deploy as a standalone device to perform the task.

2.4 Wi-Fi Network

A wireless network (Wi-Fi) is a network that communicates between computers and other network devices utilizing radio signal frequency. It is also known as a WiFi network or a wireless local area network (WLAN). It allows devices to remain connected to the network while roaming without being attached with wires. Wi-Fi signals are amplified by access points, so a device can be far away from a router and still connect to the network. Wireless connection allows the received data to be communicated to a server, where it could be analysed and processed further by the application hence achieve IoT technology (Saloni & Hegde, 2016). In this project, the communication between the ESP32 and web server is deployed through Wi-Fi network. The user is able to access the Wi-Fi network by simply enter the correct SSID and password of the Wi-Fi in ESP32.

2.5 Software Engineering

Software development is a broad term that encompasses a range of computer science tasks concerned with the process of developing, designing, executing, and maintaining software. The systematic application of engineering principles to the production of software is referred to as software engineering (Bourque et al., 2002).

Several significant software innovations had their commercial debuts during this era. Of fact, several significant technological advances occurred before to 1984: IBM's OS/360 and the microprocessor, as well as other still-relevant software engineering principles, were established far earlier (Boehm, 2006; Redwine & Riddle, 1985). Initially, software spread from a few corporate desks to almost everyone's life. The personal computer, the Internet, and mobile phones are all examples of this phenomenal progress. Second, empirical evaluations surpassed subjective judgments. "Software engineering is not yet a genuine profession," Mary Shaw stated, "but it has the possibility to become one (Shaw, 1990)." During the early stages of technology development, various innovations were simply manufactured and distributed, but beginning in the 1980s, engineers reviewed and experimentally tested new technologies to determine their impact (Ebert, 2008).

The Figure 2.4 illustrates significant software technologies and the stages of maturity at which they achieved. It is based on a layout introduced by Sam Redwine and William Riddle (Redwine & Riddle, 1985). To keep things simple, the learning curve is devided into three phases which are foundations (when fundamental research and concepts were developed), limited usage (when concepts reached a few organisations and customers), and broad use (when the technology achieved approximately a third of its then-available market).

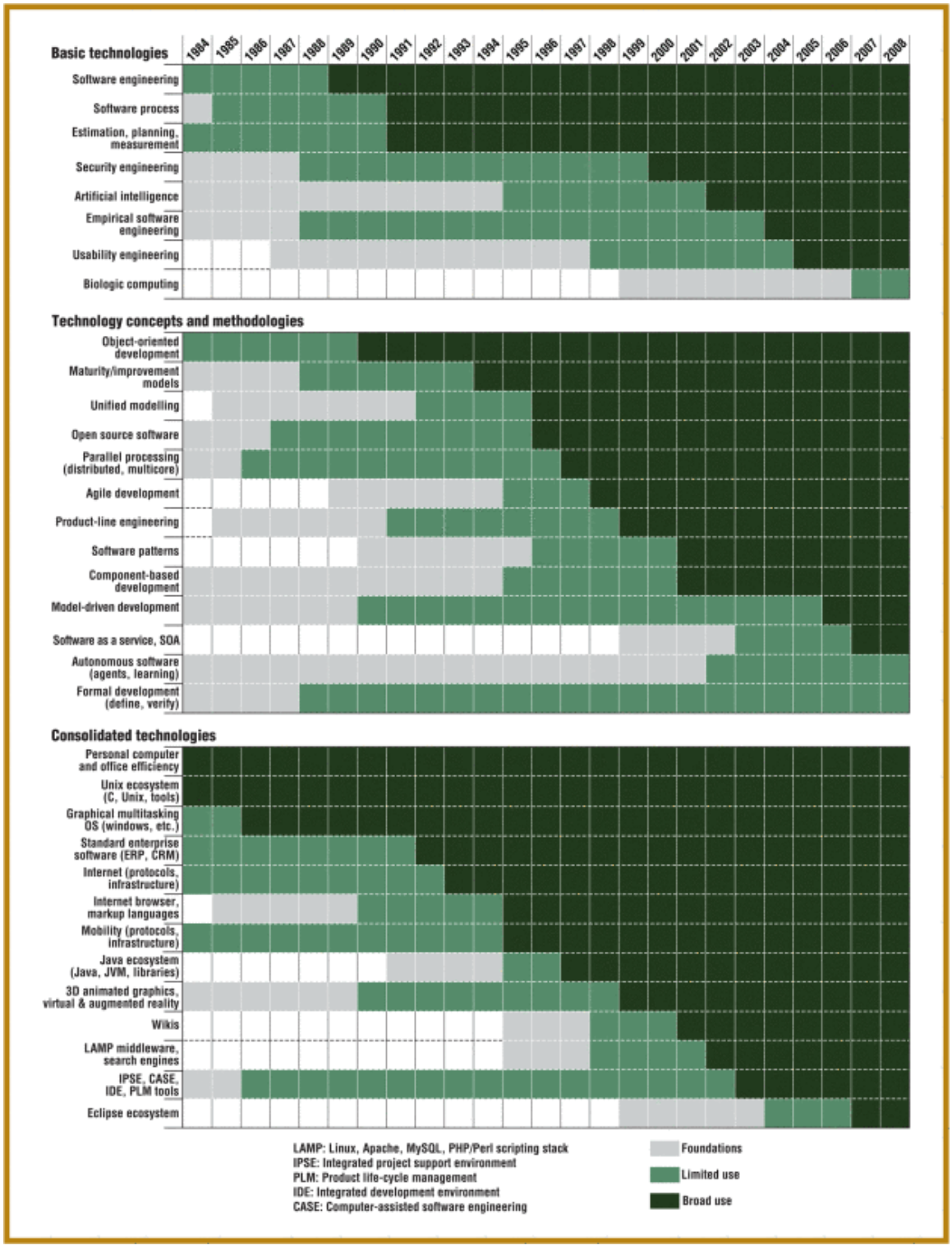


Figure 2.4 Evolution of software technology in the last twenty-five years

Source: (Redwine & Riddle, 1985)

Three clusters of software technology are depicted in Figure 2.4. Fundamental technologies contribute to the evolution of broad trends and disciplines, and they are applicable to all sectors and stages of software development. The majority of people were born during the last 25 years. Technology ideas and approaches integrate fundamental processes seen in a wide variety of businesses and products. Consolidated technologies are conceptual in nature and offer ready-to-use technical solutions.

As seen in Figure 2.4, numerous patterns describe the progress of software technology during the last 25 years:

i. Rather than individual corporations, software technologies are driven by ecosystems of researchers, suppliers, customers, and consumers.

ii. Before technologies succeed, they must undergo multiple trials with varying degrees of emphasis.

iii. Different sectors embrace a given technology at differing rates.

iv. A domain-specific emphasis enables users to tailor technology to their own requirements.

v. Process-driven design and delivery have supplanted ad hoc trial-and-error approaches.

vi. Previously fragmented and independent technologies are now interconnected.

vii. Every one of those developments had a significant impact on the engineering of products and on structuring the software industry.

Certain technologies take an abnormally long maturation periods or never completely develop. Their transition to widespread utilization follows an S-shaped pattern of innovation that goes from initial research and trials to widespread industrial adoption and then repeats again (Ebert & Dumke, 2007; Hamel, 2001). Software technologies are beneficial when they are widely used. However, certain sectors adopt a specific technology considerably more quickly than others. A excellent illustration is the tortuous path toward

usable code generation and engineering tool packages. These tool sets began as a result of technology being unprepared; subsequently, the market was unprepared. Artificial intelligence and expert systems both met the same end. Nowadays, they are nearly everywhere, since industry has realized that expert systems cannot exist as a stand-alone technology but must be integrated into products. Figure 2.5 illustrates this impact in further depth with regards to information security. Depending on the application domain, technologies encounter a variety of problems and are adopted at varying rates.

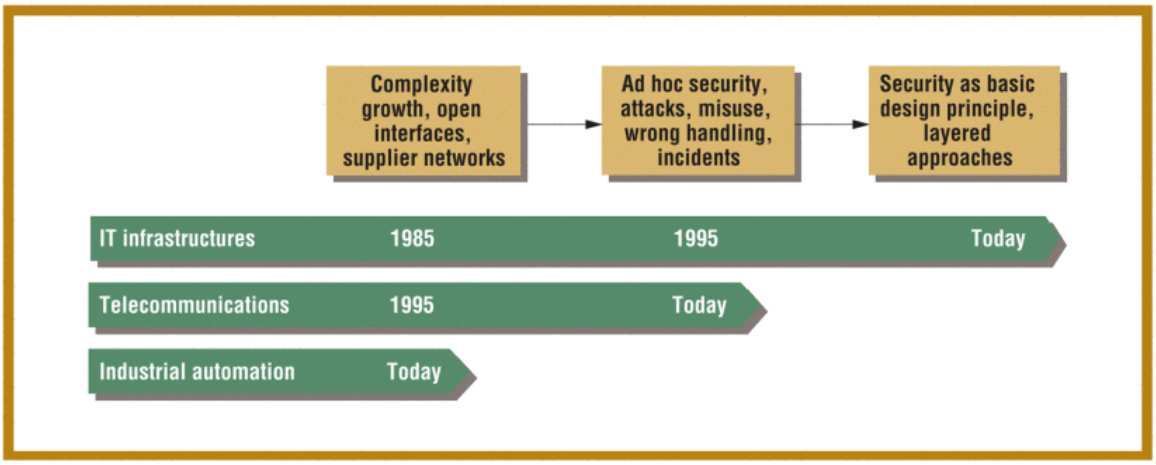


Figure 2.5 Industry-specific maturation points of IT security.
Source: (Ebert, 2008)

From the 1980s on, software processes, both engineering and management-related, accelerated technological advancement. The complexity of software systems continues to expand faster than humans can regulate it. People had previously encountered this bottleneck in the 1960s, but it began to recede as large industries shifted their focus to the process of software engineering. As a result, software development has evolved considerably over the last 25 years, transitioning from a highly individualistic creative activity to a mostly collaborative technical profession.

Integration of processes, tools, and people accelerate the adoption of new technologies. It's difficult to think that 25 years ago, the majority of software, its creators, and users operated in isolation. Software integration became most obvious with the birth and rapid expansion of the Internet, owing to the interaction and integration it enables.

Component frameworks and open standards provide further impetus for this movement. Adoption and integration are not simple tasks. To add value to engineers' work, new technology, methods, and engineering tools require extensive change management.

Engineering approaches are used to guide the software development process, which encompasses the act of defining, implementing, assessing, measuring, managing, changing, and improving the software life cycle process itself. It makes extensive use of software configuration management, which is concerned with regulating configuration changes in a systematic manner and with ensuring the integrity and traceability of the configuration and code throughout the system's life cycle. Contemporary methods implement software versioning.

2.6 Software Process

A long-standing legacy in software engineering and information systems has been devoted to the creation and implementation of approaches that facilitate successful software development (Hirschheim et al., 1995). Scholars have extensively analysed formulations and evaluations of prescriptive approaches during the last four decades, rather than examining processes based on those methods (Wynekoop & Russo, 1997).

Software methods were initially developed in the mid-1960s as an organization's defined approach to an anticipated set of software development activities in order to minimise or lessen the possibility of quality, cost, or time failure (Sommerville, 1996). Royce developed an early version of the software life cycle approach based on system engineering ideas (Fitzgerald, 2000; Rovce, 1970). This paradigm was eventually termed as 'traditional' or 'waterfall' model and is widely regarded as the foundational standard for software development processes. Waterfall is a term that refers to a succession of unidirectional, top-down, and non-iterative processes. On the other hand, the waterfall model is described as an iterative process (Rovce, 1970). After that, the approach faced criticism for a variety of reasons, including its treatment of iterations and inability to provide cost-effective, user-driven software (Lyytinen, 1987). Later techniques acknowledged the crucial importance of iterations in producing successful designs (Boehm

& Turner, n.d.), resulting in the development of non-'monolithic' perspectives that utilise iterations to gradually produce software (Graham, 1992).

With shifting trends in software development, it has not yet achieved the level of developing software that is correct, easy to use, works reliably, is maintainable, is cost effective, and is delivered on schedule (Lacerda & Furtado, 2018). Agile Software Development Process has exploded in popularity as a viable option for software development, allowing developers to efficiently simplify all of the above variables. Though the agile approach's quality has been demonstrated theoretically, it has not been quantified (Bhasin, 2012; Hsu & Lin, 2018; Jain et al., 2016).

2.6.1 Waterfall Model

Winston W. Royce introduced the iterative waterfall software process paradigm in 1970. This model gained popularity and served as a useful guide for designing software products. The term "structural specification" is taken from it. Each phase follows the completion of the previous one, and activities can be separated into stages. Although the result of one phase becomes the input of the following phase, developer have the option of revisiting stages in the subsequent cycle (Trivedi & Sharma, 2013).

This model is named as "waterfall model" because its graphical depiction resembles a cascade of waterfalls, as seen in Figure 2.6. This model is simple to comprehend and apply and it is one of the first models and is still frequently used in government projects and by a large number of large corporations (Kumar & Bhatia, 2014). The waterfall model serves as a foundation for the development of several additional lifecycle models. This approach reaffirms the value of define -before-design, design-before-code (Cohen et al., 2010).

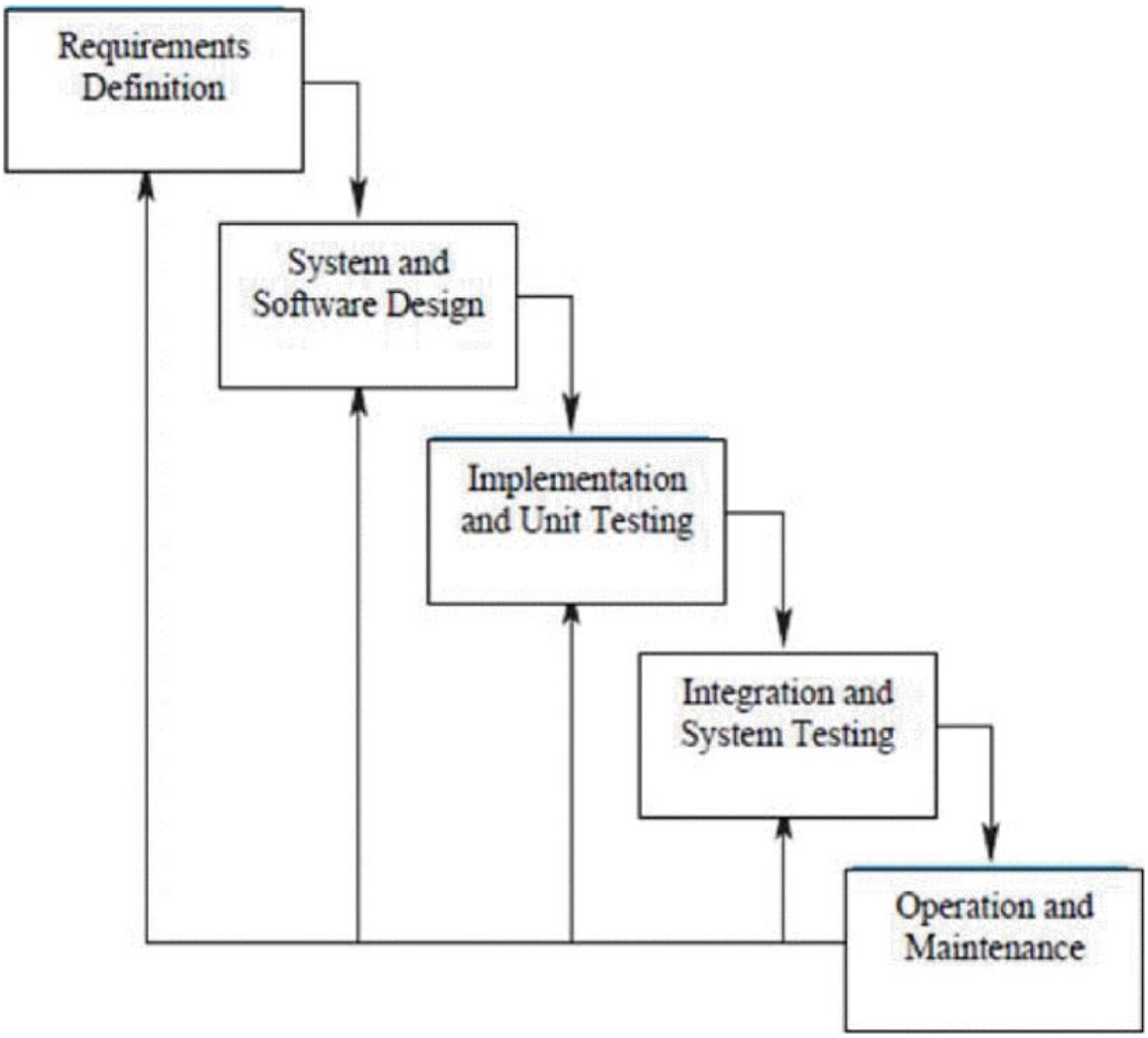


Figure 2.6 Waterfall model

Source: (Kumar & Bhatia, 2014)

The waterfall model is a project management method that relies on a sequential design process resembling the face of a waterfall. Beginning with requirements, since the waterfall model requires that specifications be thoroughly documented prior to the start of any other project phase, the project manager is keen to invest more time collecting requirements. Requirements begin with a concept or an idea of what the customer wishes to accomplish. The project manager will conduct a concept discussion with the client, subject matter experts, and other stakeholders in order to develop highly

precise business requirements. Before proceeding to the next step, the project team confirms and accomplishes the requirements phase (Sinha & Das, 2021).

The design process encompasses both the conceptual and physical design phases. The logical design is an abstract representation of how software data flows at the outputs. It is commonly shown visually as a data flow diagram. The physical design dictates the equipment, such as storage and network infrastructure, that will make the logical design a reality.

During the implementation phase, developers write the actual code needed to develop the software according to the specifications specified in the design document. To guarantee that the software requirements are satisfied, the separate programme modules or programmes are combined and tested as a whole system. This enables continuous implementation followed by developer-led unit-level verification.

Verification or testing process is the process of comparing the programme to the requirements specified in the first phase of the software development lifecycle. The project team selects a set of colleagues named Testers to test the programme during the verification phase. If the programme does not conform to the requirements specified in the requirements document, the tester returns it to the software engineers for additional adjustment. Once all validation tasks have been completed in the phase and the output has been approved by testers, the project proceeds to the next phase.

Following this stage of verification, the development team delivers the code to the production environment. The final step is maintenance, during which the project team fixes production defects or errors and pushes new builds with cleaner code into production.

The waterfall model is suitable to implement on certain system such as Critical systems that require significant investigation of the software definition and design for safety and security. To do this analysis on these systems, the specification and design papers must be comprehensive. Typically, safety-related issues in the specification and design are extremely costly to rectify during the implementation stage. This model is also appropriate for embedded systems in which software requires communicate with hardware.

Due to the inflexibility of hardware, it is typically not practical to postpone choices about software functionality until the software is implemented. Additionally, this methodology is applicable to large software systems that are a component of larger engineering systems produced by several partner organizations. The hardware in the systems may be created similarly, and businesses find it more convenient to adopt a single model for hardware and software. Additionally, if many businesses are involved, exhaustive specifications may be required to enable the separate creation of various subsystems.

2.6.2 Agile Methodologies

Agile techniques are software development approaches that adhere to the Agile Manifesto's software development values and principles. Agile techniques strive to deliver the best product possible through small cross-functional self-organizing teams that regularly deliver tiny bits of functionality, allowing for user feedback and correction.

Agile is about being flexible to the market and the consumer by quickly responding to their requirements and desires and adjusting course as necessary. Agile methods may be used to any sector that involves a continuous flow of work and the delivery of work products, such as information technology or software development. Agile technique contributes to risk reduction by rendering prospective products 'irrelevant' in the market. They do this by splitting the generally lengthy wait period (often using the classic "waterfall approach") into shorter cycles (called sprints or iterations). Each sprint is divided into three phases: requirements, implementation, and testing (Figure 2.7). Each sprint concludes with a client review to ensure that smaller vertical pieces of the product finally match their demands and are also market ready.

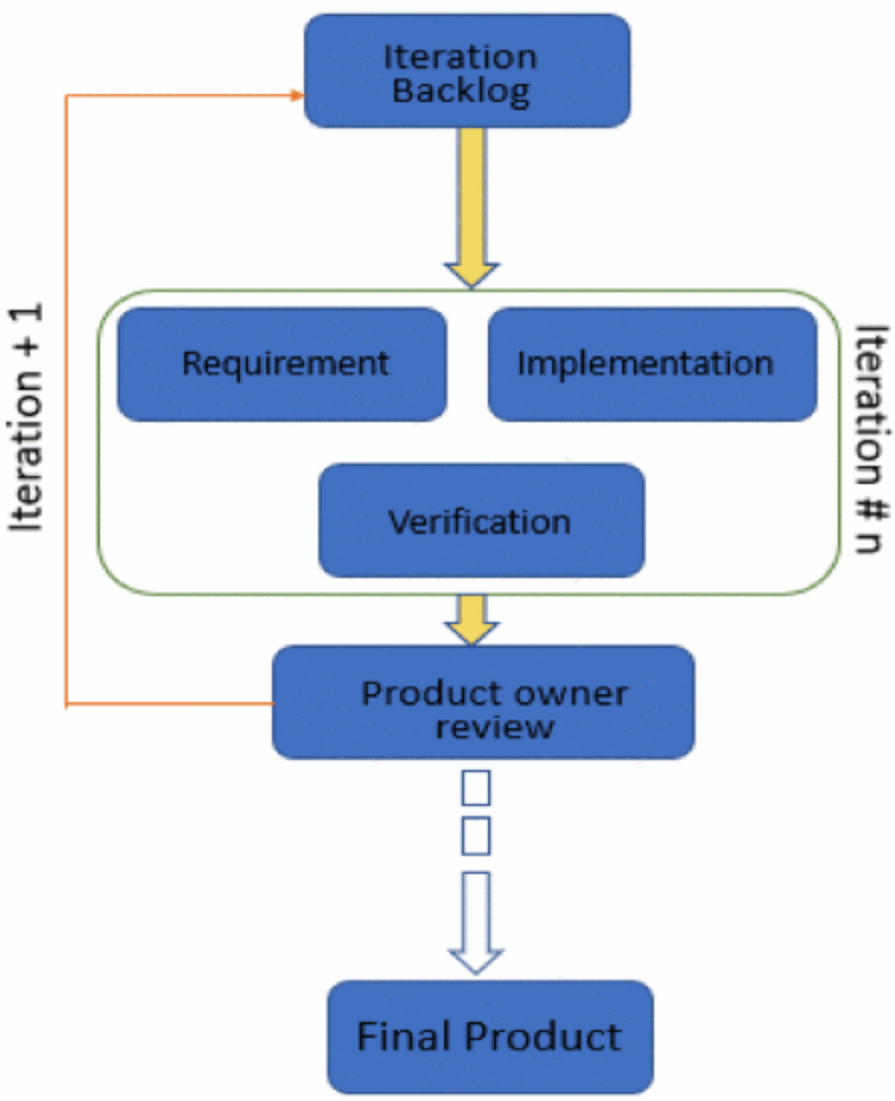


Figure 2.7 Agile methodology iteration

Source: (Sinha & Das, 2021)

Due to the iterative and incremental nature of the agile methodology, each stated phase is completed within a series of brief iterative cycles of the software development process referred to as sprints. While in a conventional process, also known as a lightweight development process, each step is carried out sequentially without any iteration defined. The incremental cycles result in a deployable product for the customer (Dingsøyr & Lassenius, 2016). Additionally, agile contains stages like as concept, genesis, release, production, and retirement that are not covered in traditional methodologies. Agile

development is a more rapid and sustainable kind of development than traditional development since it allows for customer contact throughout the process and the adaptation to needed adjustments. As seen in Figure 2.8, it is a customer-centred technique. Additionally, customer satisfaction is prioritised by connecting with them more frequently or virtually in every phase of a sprint than in a typical one (Jain et al., 2018).

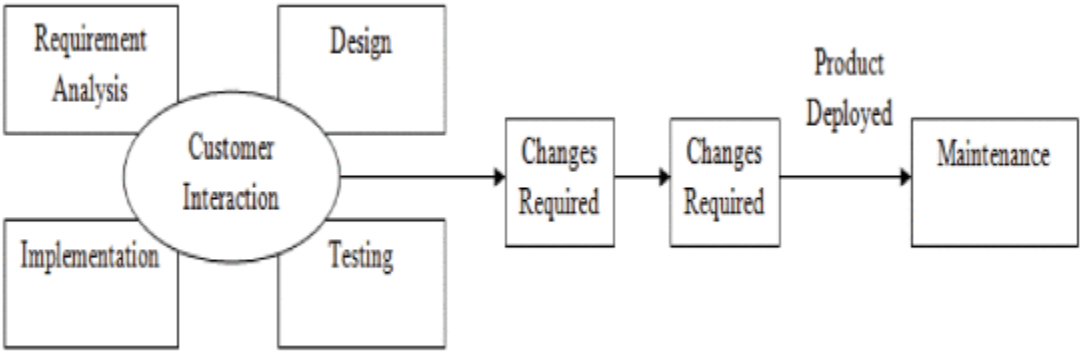


Figure 2.8 Customer centred agile methodology
Source: (Jain et al., 2018)

Agile approach provides several benefits such as the cost of adopting modifications to requirements is minimised. The amount of rework required for analysis and documentation is substantially less than what is necessary with the waterfall approach. Besides, it is easy to obtain consumer feedback on completed development activity. Customers may leave comments on software demonstrations and check what has been implemented. Customers have difficulty determining progress based on software design papers. Also, it is feasible to produce and deploy relevant software to the client quickly, even if not all functionality is available. Customers may begin utilizing and gaining value from the product far sooner than with a waterfall method.

However, a number of disadvantages also noticed from agile approach where the process is imperceptible. Managers require regular deliverables in order to track progress. When systems are built rapidly, producing publications that reflect each version of the system is not cost effective. Furthermore, a s new increments are introduced, the

system structure degrades. Regular modification results in a jumbled code base as new functionality is implemented in every manner feasible. Adding new features to a system gets increasingly complicated and expensive. To prevent structural deterioration and overall code messiness, agile methodologies recommend rework (modify and reorganize) the programme on a frequent basis.

2.6.3 Agile Methodology Vs. Traditional Waterfall Model

Various models are used for development purposes in both traditional and agile approaches. Various models such as the waterfall model, prototype model, spiral model, iterative enhancement model, and evolutionary model are used in conventional approach (Singh & Gautam, 2016; Singh & Prasad Kannojia, 2012). Kanban, scrum, extreme programming, and feature driven development are just a few of the techniques used in agile approach.

The approach to process maturity concentrated on process and project management enhancements, as well as the adoption of quality software engineering practices throughout a business. The maturity level of a process indicates how well-established technical and managerial practises have been integrated into an organization's software development processes. This strategy is primarily concerned with enhancing product quality and process predictability. The agile approach focused on iterative development and the minimization of overheads in the software process. Agile methodologies are defined by their ability to offer functionality quickly and adapt to changing client needs. The idea of improvement is that the best processes are those with the fewest overheads, which agile methodologies may accomplish.

The most appropriate model may be chosen based on the requirements analysis of the programme to be created and the available resources. In this project, waterfall model is deployed as the methodology for this project. The waterfall technique relied more on specifications, whereas agile relied more on prototypes. In this SDP, hardware is involved and an embedded system is built. According to the lack of flexibility of hardware, it is generally not possible to suspend decisions about software functionality until the software is delivered.

2.7 Application Programming Interface (API)

An application programming interface (API) is a way for computers or computer programmes to communicate with one another. It is a form of software interface that provides a service to other software components (Reddy, 2011). An API specification is a document or standard that explains how to create or use such a connection or interface. The term "implement" or "expose" an API refers to a computer system that adheres to this standard. The word application programming interface (API) can apply to either the specification or the implementation. Nowadays, APIs are frequently utilised in today's software development industry (Zhong & Mei, 2019).

Unlike a user interface, which links a computer to a human, an API connects computers or pieces of software together without direct usage by end user. An API is frequently composed of many components that serve as tools or services to the programmer. When a programme or programmer makes use of one of these components, it is referred to as calling that section of the API. The API's calls are sometimes referred to as subroutines, methods, requests, or endpoints. A specification for an API defines these calls, i.e. it describes how to utilize or implement them.

One of the purposes of APIs is to obscure the technical details of how a system works, revealing just the portions that a programmer would find helpful and ensuring that they remain consistent even if the internal details change in the future. An API may be tailored to a specific pair of systems, or it may be a shared standard that enables interoperability across several systems.

2.7.1 Web API

Web Application Programming Interfaces are the specified interfaces that enable interactions between an organisation and the apps that consume its assets. They also serve as a service-level agreement (SLA) for the functional provider to specify and disclose the service path or URL for its API consumers. An API approach is an architectural approach focusing on the provision of a programme interface to a collection of services to a variety of applications servicing a variety of users.

Web APIs enable programmatic access to remote data or functionality through a network. For example, applications can make use of the Google Places API to discover local establishments, the Twitter, Instagram, or Facebook APIs to connect users with friends and family, or the Stripe API to collect end-user payments. Usually, programmes are composed of microservices that communicate with one another via web APIs (Wittern, 2018). When used in conjunction with web development, an API is commonly specified as a collection of specifications, such as HTTP request messages, and a definition of the structure of response messages, typically in the Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format. A shipping business API, for example, might be integrated into an eCommerce website to enable customers to request shipping services and to automatically include current shipping prices, without the site developer having to manually insert the shipper's rate table into a web database. While the term "web API" was historically used interchangeably with "web service," a recent trend (dubbed Web 2.0) has been a shift away from SOAP-based web services and service-oriented architectures (SOA) toward more direct representational state transfer (REST)-based web resources and resource-oriented architectures (ROA) (Benslimane et al., 2008).

In this project, Google Maps API is implemented in order to add a digital map in the web server to allow user to choose locations from the map easily. The Google Maps API is a collection of JavaScript classes that enable users to alter and incorporate Google Maps into their websites. Google Maps API enables the entire globe of Google Maps enthusiasts link the free electronic map to their web pages and utilize the free electronic map for a range of geographic information system applications (Li, 2011). For instance, users may utilise Google Maps to see data points connected with certain geolocations. Additionally, users may generate personalised routes (Zhu, 2012). Additionally, the Google Maps API is fully documented, with several courses and samples available online for students to study. Additionally, there is an active user forum where users may ask and answer questions. Although the Google Maps API is a free service, it does have certain technological limitations. For example, a user's ability to make queries to Google Maps services per day and per second is limited. Paid memberships allow users to increase their limit, however the great majority of users seldom exceed the pre-set capacity.

2.8 Graphical User Interface (GUI)

The graphical user interface (GUI) is a sort of user interface that enables users to interact with electronic devices through the use of graphical symbols (Martinez, 2011)and audible indicators such as main notation, rather than using text-based user interfaces, written command labels, or text navigation. GUIs were developed in response to the perceived steep learning curve associated with command-line interfaces (CLIs), which require users to write instructions onto a computer's keyboard. It has always been necessary for applications to design a pleasing and intuitive user interface (UI) and user experience (UX) that allows people to utilize them easily (Alfaridzi & Yulianti, 2020). A user interface is described as a medium that a system provides for users to engage with it and vice versa, whether they need to control the system, enter data, or consume the system's contents (Joo, 2017). The term "user experience" refers to the impressions that individuals receive when they use or engage with a product. The user experience refers to how consumers feel about a product in terms of how satisfying, pleasant, and simple to use it (Kurniawan, 2004).

Interfaces are inextricably linked to both design and interaction. The interface design process contributes to graphically connecting system functionalities. Additionally, the UX interface is influenced by the system's usability, its contents, and services, the affinity of users, and the value of users. Developing a consistent interface is a critical task to do since it defines the user's comfort level when managing a system. When consistency is understood, it may result in a more effective objective (NIELSEN, 1989). According to Jonathan Grudin, there are three processes involved in building a consistent interface: the first step is to define the consistency; the second step is to ensure that the consistency is solid; and the third step is to check for undesirable consistency (Grudin, 1989). It is critical for designers to provide a consistent interface because while they may not be aware of the context in which the user is operating, they are aware of the conversation that will occur between the user and the system. Numerous trade-offs must be made in order to create a user interface that fulfils all of the computer system's requirements and goals while adhering to a set of principles (Johnson, 2010).

There are ideas referred to as "golden rules" that may be applied to a variety of interactive systems that include user involvement (Shneiderman & Plaisant, 2010). All of the concepts are refined as following:

i. Consistency

This criterion is now being violated, consistency manifests itself in a variety of ways. In comparable instances, consistent sequences of activities must be done. Situations such as concealing the password while entering and requesting confirmation before to deleting an item would improve the system's quality.

ii. Offering universal usability

The developer should consider the user's demands and build the interface in such a manner that further transformations may be made when the user's requirements vary. Integrating features for novice users, such as adding explanations and giving shortcuts for experts, results in an increase in system quality.

iii. Informative feedback

Each user action should be accompanied by informative feedback. For activities that are performed infrequently but regularly, a modest reaction should be supplied; however, for actions that are performed infrequently but frequently, a more significant response should be offered.

By segregating by layer from the tactic of evaluating the user's intent with the UX interface to the cognitive and sensory attributes that cause user behaviour to the surface, Jesse James Garrett demonstrated the framework of UX by splitting it into Strategy, Scope, Structure, Skeleton, and Surface. Recently, the design interface for UI/UX has placed a premium on user demands (JOO, 2017; H. E. Lee, 2014). Once user demonstrate an action plan, it empathises immediately. This is for the purpose of comprehending the users. The second step is to establish the objective in terms of a project or business by identifying the issue. The following stage, Ideate, aims to generate new concepts and solutions. The subsequent stage is to develop a prototype of the UI/UX that was offered as a concept or

solution in the previous phase. The final stage is to finish the UI/UX by examining and making decisions. Figure 2.9 illustrates this execution strategy as UI/UX interface thinking.

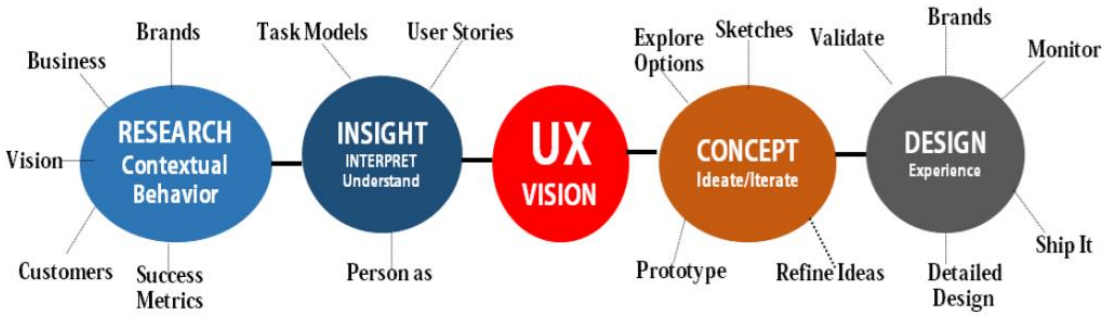


Figure 2.9 Interface thinking of UI/UX design

Source: (Joo, 2017)

In this project, web server is implemented in GUI to allow user interact with the antenna rotator remotely through internet.

2.9 Web Server

The requirement for remote monitoring and accessibility for a variety of embedded applications has raised the demand for cost-effective and power-efficient techniques (Bammidi & Kundala, n.d.). Numerous remote monitoring and control solutions are investigated, and it is shown that the greatest outcomes are achieved when the web server are properly created for embedded applications (Tian-huang & Jia-xi, 2008).

A web server is a piece of software or hardware that responds to client requests made over the World Wide Web using HTTP (Hypertext Transfer Protocol) and other protocols. A web server's primary responsibility is to show website content by storing, processing, and distributing webpages to users. Apart from HTTP, web servers offer the SMTP (Simple Mail Transfer Protocol) and FTP (File Transfer Protocol) protocols, which are used for email, data transfer, and storage, respectively.

The web server can deliver services needed to web clients and website hosting. Web clients can connect to the web server through a router and the internet. This is a low-cost web server solution that allows for local data storage and access via a segment of web clients (MacHeso et al., 2021). The client server architecture greatly contributes to the

development of Internet of Things devices, which have become the topic of discussion in the computer industry (Limpraptono et al., 2011). Figure 2.10 illustrates a typical client-server architecture in which a request is made to the server whenever a web client wishes to reach the server.

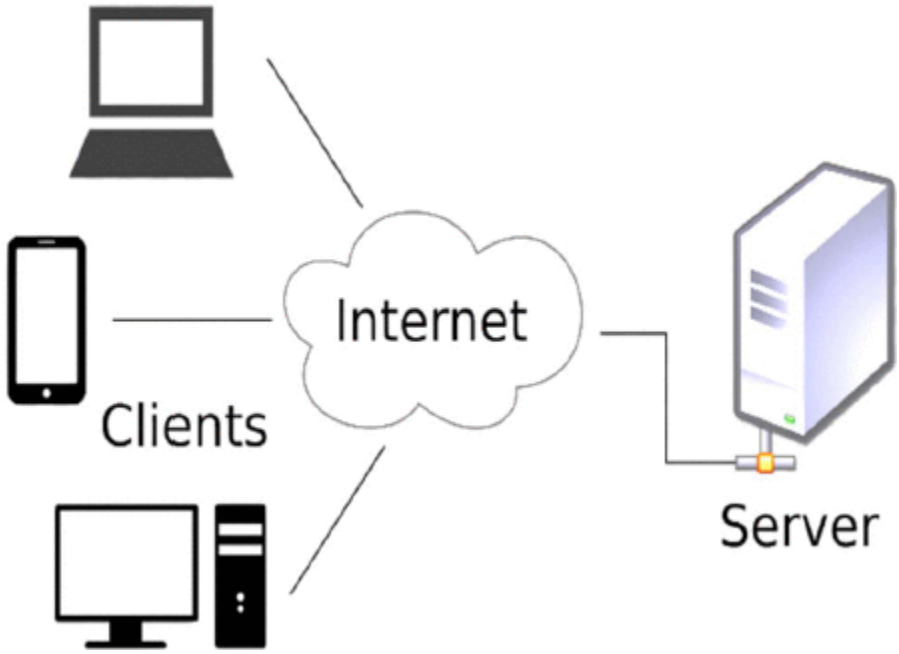


Figure 2.10 Server architecture

Source: (Limpraptono et al., 2011)

Between the web client and the web server, a particular protocol called Hypertext Transfer Protocol (HTTP) is utilised. In this project, a real-time monitoring system is necessary to analyse, investigate, and make judgments on environmental characteristics such as the feedback potentiometer value from antenna rotator. This task could be achieved with the deployment of NodeMCU ESP32. The advantages of embedding a web server into a device are listed as following (Can Filibeli et al., 2007):

- Users can interact with appliances using a web browser on a device. These technologies, which range from personal computers to mobile phones, are widely available and widely used.

- Costs associated with user interface hardware (electromechanical) can be removed, allowing for the construction of more user-friendly interfaces at a cheap cost.
- Controlling, monitoring, and upgrading are all possible from any location on Earth.
- Developers may maintain products throughout their life cycles by uploading new software versions, which reduces maintenance expenses.
- Updates and extensions to user interfaces are possible.

2.10 WebSocket Protocol

In HTML5, the WebSocket protocol is a TCP-based application-layer communication protocol that corresponds to the concept of a socket and creates a full-duplex communication channel between the server and Web client. The server and client is able to communicate via the WebSocket channel (Mei & Long, 2020). The most distinguishing characteristic of the WebSocket protocol is that it enables the server to actively deliver data to the client, which is extremely difficult with the HTTP protocol, and its minimal header information makes it ideal for real-time communication (Qin, 2017).

With the tremendous growth of the Internet, the Browser/Server mode has become the standard for modern applications, and developers have steadily prioritised Web communication security in recent years. The system operations are performed on the browser side, which has the advantages of high real-time speed, cross-platform compatibility, and ease of use. Simultaneously, the WebSocket protocol transmits data 500 times faster than the HTTP standard (Lubbers, 2011). Indeed, WebSocket-based communication interfaces have been commonly applied to overcome issues involving real-time communication (Zha et al., 2014). As a result, a real-time monitoring system for wireless coverage data are developed using the WebSocket protocol, and the acquired wireless coverage data is shown in real time on the browser side. This technology provides a simple and dependable foundation for planning or post-maintenance wireless network coverage (Liu et al., 2018).

It only requires a simple "handshake" action between the browser and the server in the WebSocket, then a fast track is formed between the browser and the server. Once linked, WebSockets act as data frames, transmitting and receiving data in dual channel mode.

Between the server and the client, a WebSocket connection is established through the WebSocket protocol during the initial "handshake." Additionally, the protocol is based on the TCP/IP underlying protocol. The WebSocket protocol is straightforward; clients such as standard browsers communicate with servers over the 80 or 443 port. According to HttpHeader, the server determines whether a connection is a WebSocket request and, if so, upgrades it to a connection. Following the handshake's success, the connection enters the two-way long connection data transmission phase.

WebSocket data transfer is frame-based: 000 signifies the start of data, 0xff denotes the end of data, and the data is encoded in utf-8. In comparison to standard HTTP queries, the WebSocket handshake processes merely two bytes of request header information between the server and the client, significantly reducing the amount of bandwidth consumed by requests and server resources (Zhang & Shen, 2013).

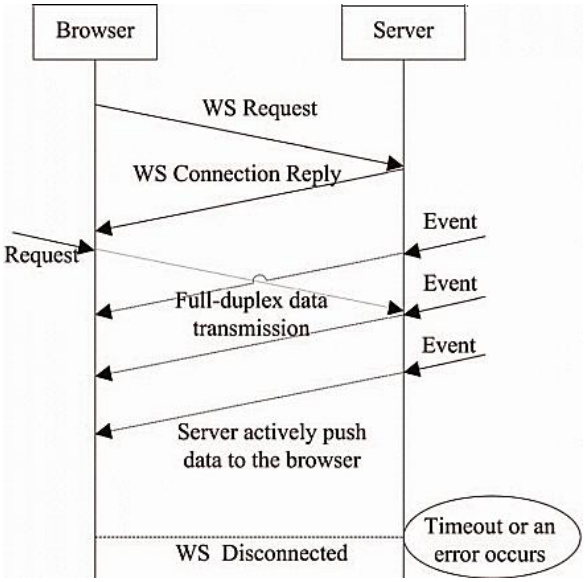


Figure 2.11 WebSocket communication model

Source: (Zhang & Shen, 2013)

At the moment, the need for real-time information transmission is increasing. HTML5 recommends the use of WebSockets to enable real-time online communication. Figure 2.11 illustrates the WebSocket communication paradigm. WebSocket is a new HTML5 protocol that enables full-duplex communication between the browser and the server. In comparison to previous real-time communication technologies, WebSocket enables real-time communication while reducing server resources and bandwidth.

WebSockets provide the following benefits over traditional real-time communication. WebSocket significantly reduces network bandwidth usage by appending more than 800 bytes of HTTP headers to a single HTTP request. When the WebSocket protocol is used instead of HTTP, each message is sent across the network in the form of a WebSocket framework, with a total overhead of around 2B (Yang & Yang, 2016). In comparison to conventional HTTP, each request-response pair requires a mode in which the client connects to the server. Each time data is exchanged, in addition to the actual data, the server and client transfer a huge number of HTTP headers, resulting in a low efficiency of exchange of information. Similar to Socket, WebSocket is a TCP long-connection communication mechanism. Following the establishment of the WebSocket connection, further data is transferred in the form of a frame sequence. The client and server are not needed to re-initiate the connection request until either the client or the server disconnects from the WebSocket connection. The use of network bandwidth resources is significantly reduced in the case of excessive concurrency and high load traffic between the client and the server. The Figure 2.12 compares network traffic generated by the system while it interacts in real time via polling and WebSocket. When traffic and load rise, the WebSocket approach outperforms previous polling techniques significantly.

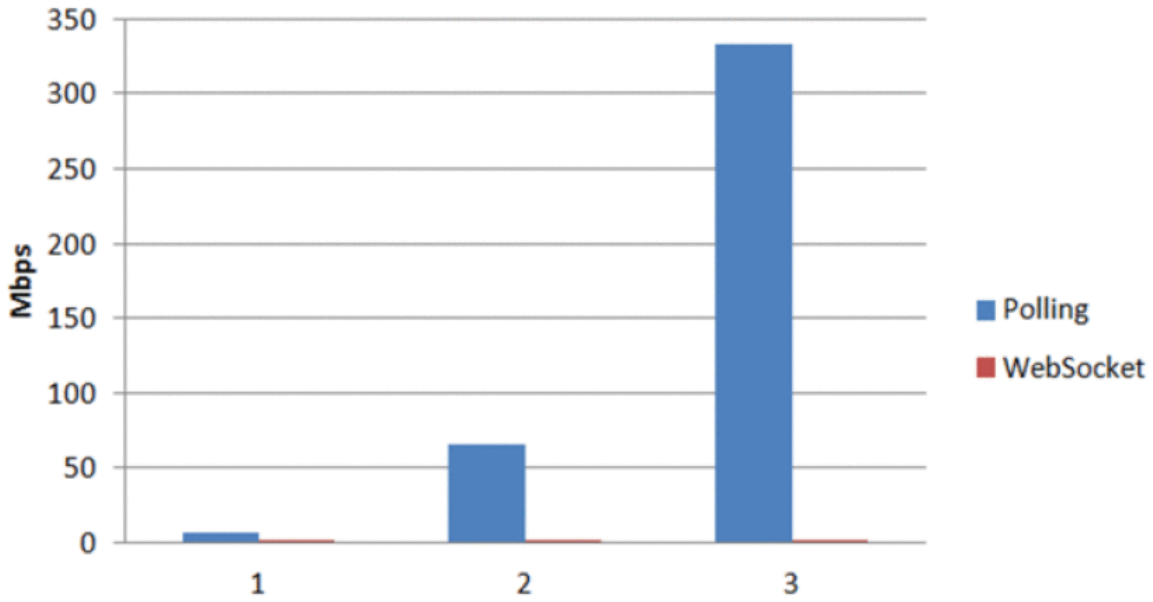


Figure 2.12 Network load comparison graph

Source: (Liu et al., 2018)

WebSocket is a true full-duplex communication protocol that provides superior real-time performance which it enables full-duplex communication between the server and the browser. Once the connection is established, the server and client are equal and may communicate with one another. The HTTP long connection is based on HTTP, the standard client-server request protocol. Due to the full-duplex nature of the protocol, the server can send information to the client at any time. Due to the client sends and receives messages over the same persistent connection, the advantage of real-time communication is obvious. In comparison to an HTTP request, which requires the client to request to the server, the latency is greatly decrease, and the data can be transferred more frequently in a short period of time.

CHAPTER 3

METHODOLOGY

This chapter details the order, technique, and procedure for constructing the project. Understanding the methodologies and technology that will be employed is critical while developing the project. This is a hardware and software integration design project. The project must be completed in stages and in accordance with the process flow outlined below (Figure 3.2) in order to accomplish the stated aim.

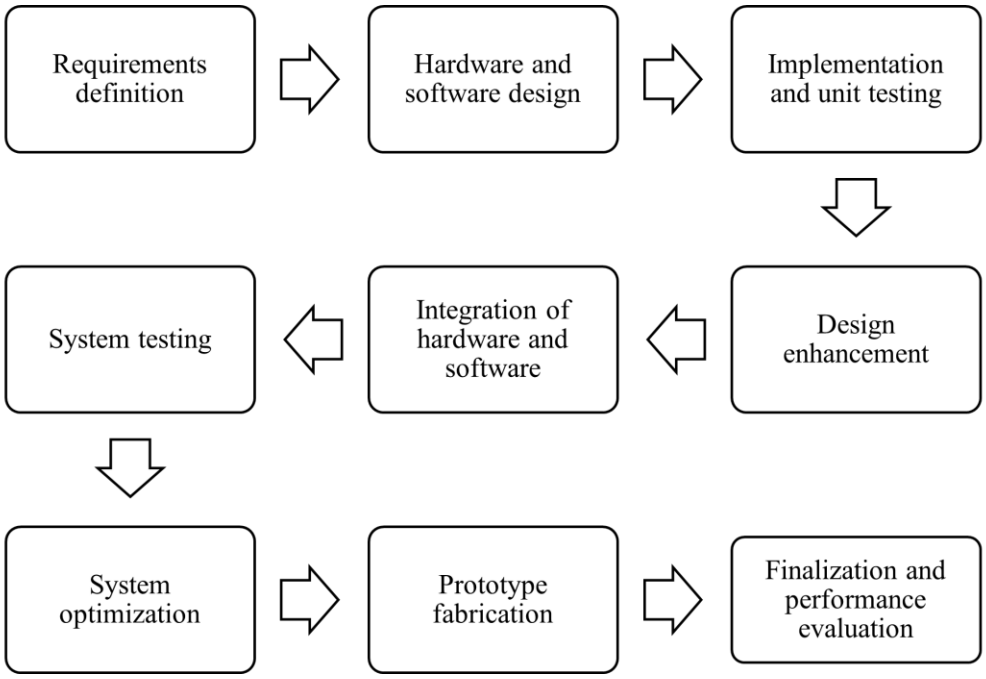


Figure 3.1 Process flow for the fabrication of IoT Integrated Antenna Rotator

3.1 Flowchart of Methodology

Figure 3.2 shows the methodology flowchart in order to complete this project.

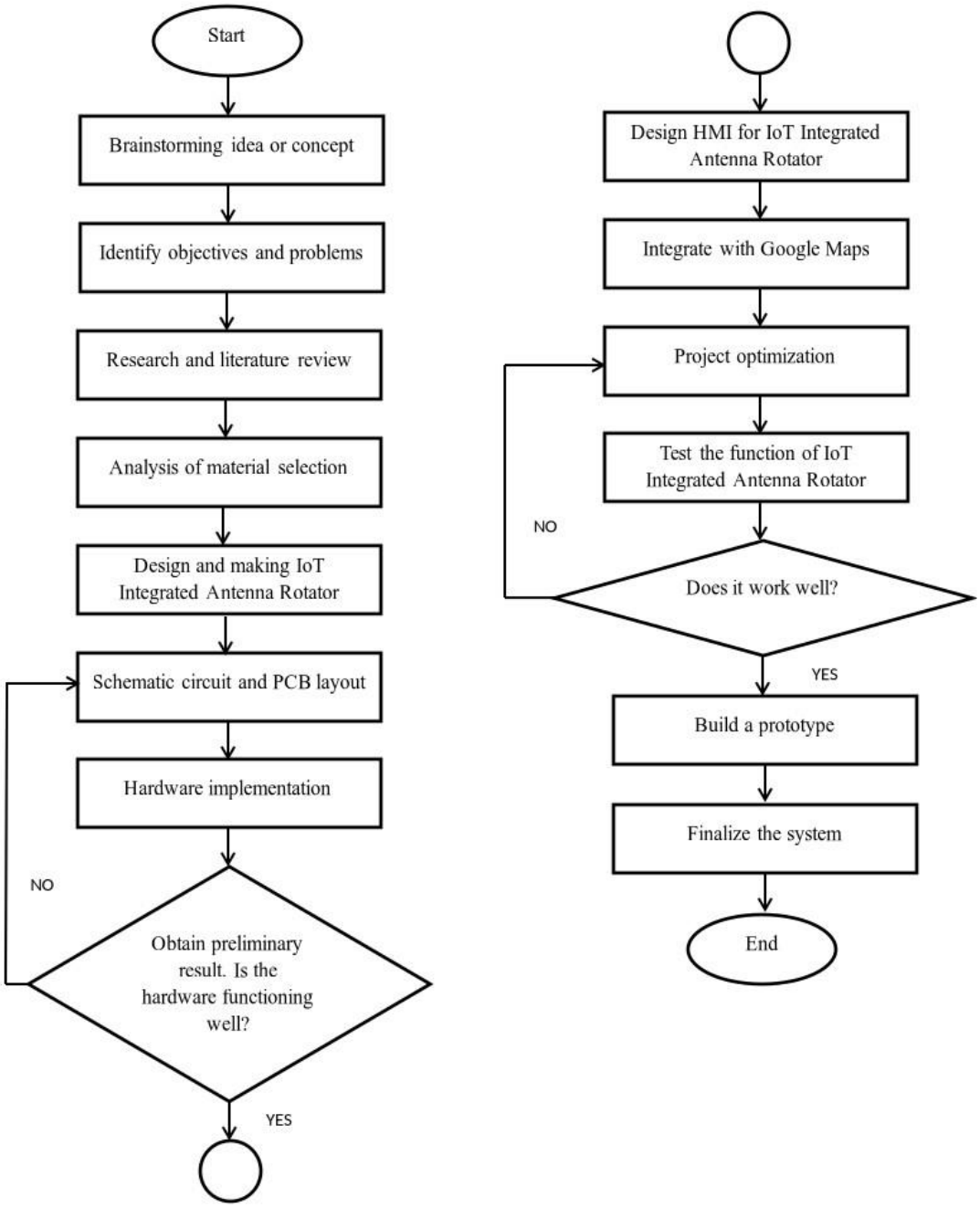


Figure 3.2 Flowchart of methodology

3.2 Project Layout

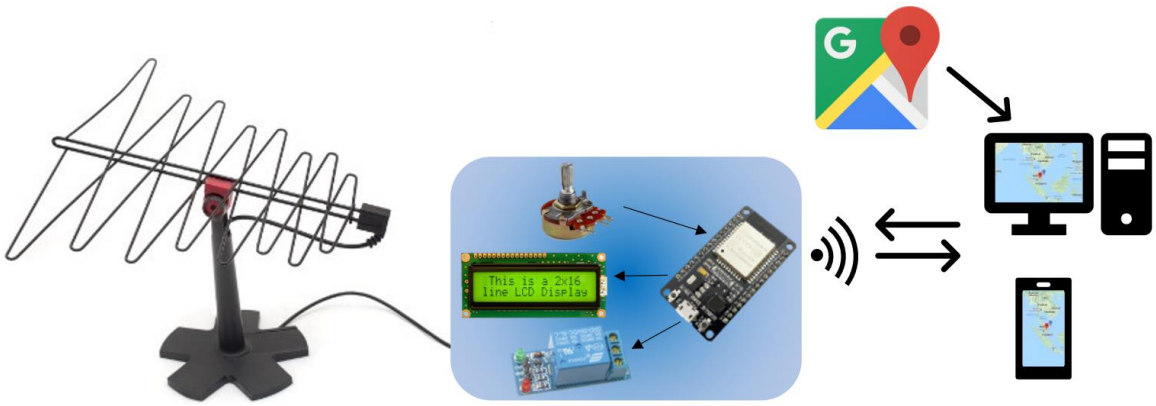


Figure 3.3 Expected project layout of IoT integrated Antenna Rotator

Figure 3.3 illustrate the expected layout of the IoT integrated antenna rotator. An antenna is mounted on the antenna rotator while the rotator is installed on a solid base. The rotator is connected to a control box which inside it contains the microcontroller, electronic components and electrical circuit. The microcontroller inside the control box is connected to a Wi-Fi network to enable the system to access internet. Users are allowed to utilize the web server as the GUI to control or monitor the orientation of the antenna rotator. By implementing the Google Maps API, user can easily pin the places in Google Maps to obtain the correct coordinate of the locations. Alternatively, user also can manually alter the angle of the antenna rotator by adjusting the potentiometer at the control box.

3.3 Block Diagram of IoT Integrated Antenna Rotator (Software Development)

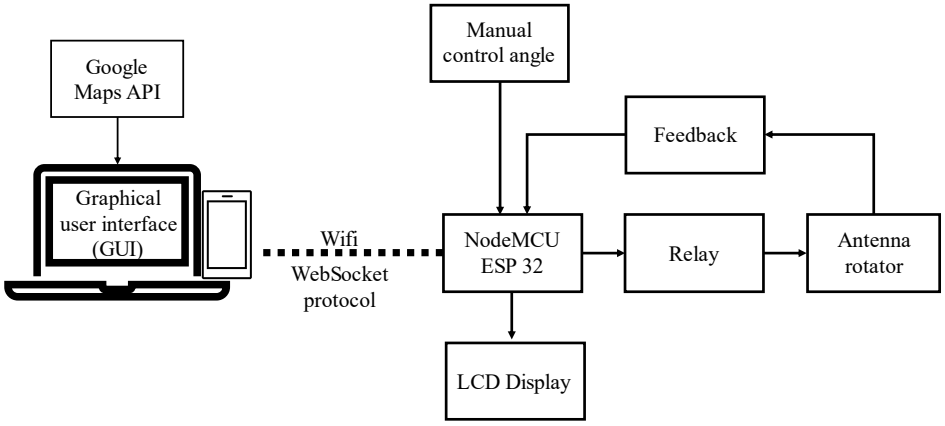


Figure 3.4 Block diagram of project (software development)

Figure 3.4 illustrates the block diagram of IoT Integrated Antenna Rotator which is emphasise on the software development. The Google Maps API is embedded into the web server which allows user to select the current or desire location from the digital map. The user is also able to select a certain angle from the web server to control the angle of the rotator. The input data is then sent to the NodeMCU ESP32 controller via Wi-Fi using WebSocket protocol to control the output of the system. ESP32 will calculate the bearing based on the input data from web server and decide the antenna rotator to turn clockwise or anti-clockwise by switching the relay. A feedback mechanism is implemented in this system to confirm the antenna rotator is correctly pointing to a certain direction. The ESP32 ensure the current angle (feedback angle) of the antenna rotator is always within the range or close to the desire angle entered by user. The real-time current angle of the antenna rotator is displayed on the web server as well to allow user remotely monitor the system. User is allowed to manually control the orientation of the antenna rotator using potentiometer by simply switch on the manual mode hence the antenna rotator could rotate according to the manual rotation. The current angle and desire angle of the system is also displayed on the LCD.

3.4 Project Flow

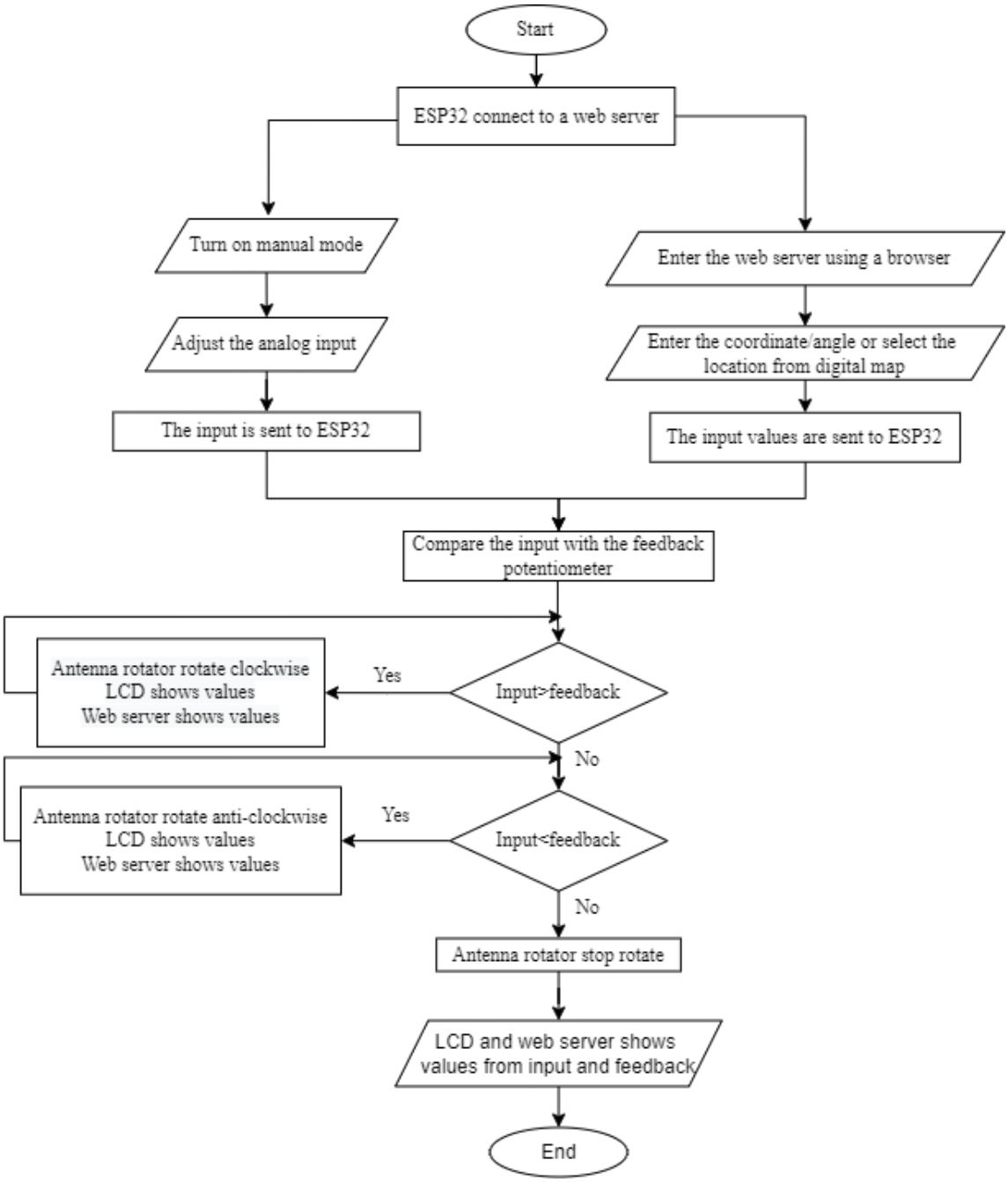


Figure 3.5 Project flowchart of IoT Integrated Antenna Rotator

Figure 3.5 shows the flowchart for IoT Integrated Antenna Rotator. There are 2 method that user can adjust the orientation of the antenna rotator which are analog control or select the location thru web server. When the coordinate of the location is inserted, the data of the coordinate is transmitted to the ESP32. The ESP32 controls the movement of the rotator and causes the motor to rotate. The inserted value is compared with the feedback value from antenna rotator, in order to decide the orientation of the motor. The motor keeps rotating until it reaches the desired angle determined by the controller. Finally, the rotator faces the selected location entered by the user.

3.5 Hardware Components

The selected hardware components for design the IoT Integrated Antenna Rotator are listed as following.

3.5.1 NodeMCU ESP32

Figure 3.6 NodeMCU ESP32

Figure 3.6 shows an ESP32 is selected as the controller in this project which it is a dual-core development board that integrates Wi-Fi features. It integrates a broad range of peripherals, including capacitive touch, ADC, DAC, SPI, UART, PWM, I2C and I2S.

3.5.2 Antenna Rotator

Figure 3.7 Dennard Type 2000

Dennard Type 2000 in Figure 3.7 is used as the rotator to rotate the position of an antenna. The cables of the Dennard Type 2000 are connected to the system to control the orientation movement and provide the feedback of the position to the controller.

3.5.3 LCD Display

Figure 3.8 LCD Display

Figure 3.8 shows a liquid crystal display (LCD) which is a light-modulating flat-panel display device that makes use of liquid crystals' capabilities. Liquid crystals do not generate light directly; rather, they create monochromatic contents via a backlight or reflector. It is used in this project to display the current and desire angles of the system.

3.5.4 Potentiometer 10kΩ

Figure 3.9 Potentiometer 10kΩ

Figure 3.9 shows potentiometer of 10kΩ. It is a manually adjustable variable resistor with three terminals to rotate the azimuth angle of the antenna rotator.

3.5.5 Transformer

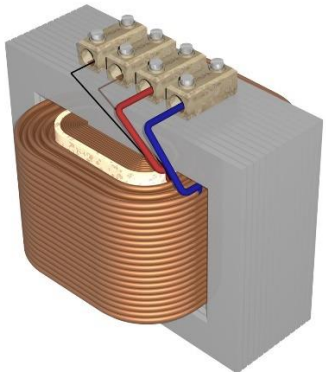


Figure 3.10 Transformer

A transformer as illustrated in Figure 3.10 is used in the project to convert and step down the source from 240VAC to 24VAC.

3.5.6 Relay 5V Module

Figure 3.11 5V relay Module

Figure 3.11 shows a 5V relay module. The relay is controlled by a low-level triggered control signal. When the relay is activated, it operates in one of two modes, either normally open or normally closed to control the orientation of antenna rotator.

3.5.7 Toggle Switch

Figure 3.12 Toggle switch

Figure 3.12 shows a 3-pins toggle switch that is activated by back-and-forth movement of a lever to open or close an electrical circuit. It is used in this project to turn on or off the manual rotation mode of the IoT Integrated Antenna Rotator.

3.5.8 Optocoupler

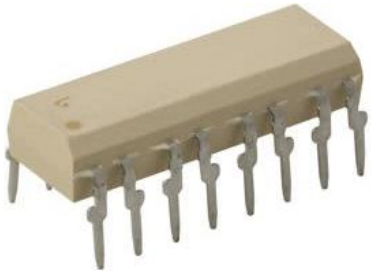


Figure 3.13 Optocoupler

Figure 3.13 shows an optocoupler that connects two isolated circuits together. It used to protect the system from damaging the circuit and devices by receiving the signal with excessive voltages.

3.5.9 Resistor

Figure 3.14 Resistor

Figure 3.14 shows resistors which are used in electrical circuit to limit the current flow or voltage in the system. 220Ω and 3.8kΩ resistors are used in this project to adjust the current flow in this system.

3.6 Software Tools

The following are the selected software tools for develop the IoT Integrated Antenna Rotator software system.

3.6.1 Arduino IDE

The Arduino Integrated Development Environment (IDE) is a cross-platform programme developed in C and C++ functions. It is used to build and upload programmes to Arduino compatible boards. In this project it used to program the ESP32.

3.6.2 Visual Studio Code

Visual Studio Code is a simplified code editor that includes debugging, task execution, and version control capabilities. It is designed to give only the tools necessary for a speedy code-build-debug cycle, leaving more complicated processes to more feature-rich IDEs. Visual Studio Code is used in this project to develop the front-end of the system.

3.6.3 Web script

Scripting is a prevalent technique in web development. Web development encompasses all activities involved in creating an Internet-based website, including web design, web content creation, and programming. Scripting enables the transformation of a static HTML page into a more dynamic one. It enables people to engage with a website. In this project, HTML, CSS and JavaScript are used to design the web server. HTML establishes the foundation for websites, which is expanded and updated by other technologies such as CSS and JavaScript. CSS is used to manage the appearance, formatting, and layout of web pages. JavaScript is used to programmatically control the behaviour of certain components.

3.6.4 WebSocket

WebSocket is an IETF-standardized transport protocol that permits the usage of multiplex messages on both the client and server side of a single TCP connection (Fette & Melnikov, 2011). WebSocket is intended to assist the web browser and web server in

communicating bidirectionally many messages concurrently with the least amount of latency and packet loss (Nakajima et al., 2013).

3.6.5 Google Maps API

Google Maps API is a collection of application programming interfaces that allow user to communicate with the web services. It enables developers to create simple to highly advanced location-based applications for the Web, iOS, and Android. In this project, the geographic coordinates are obtained by using Google Maps API.

3.7 Circuit Design

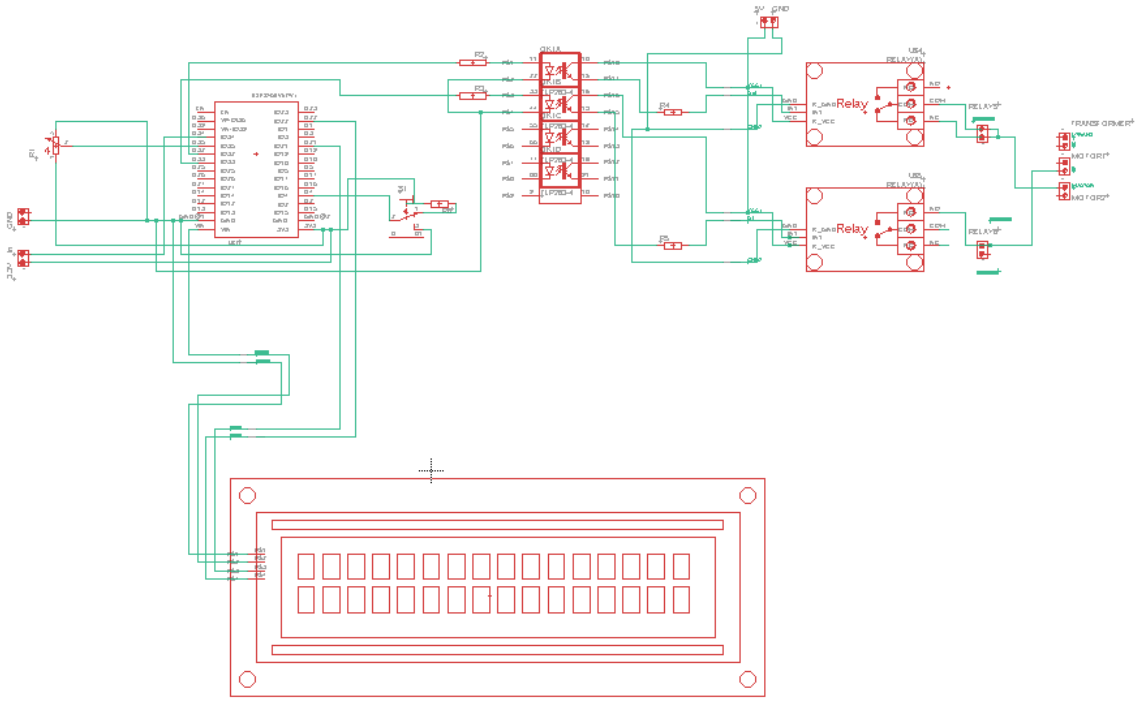


Figure 3.15 IoT Integrated Antenna Rotator schematic circuit

Figure 3.15 illustrate the schematic circuit connection of IoT Integrated Antenna Rotator using NodeMCU ESP32 as the controller. The feedback signal and manual rotation signal are connected to A6 (D34) and A7 (D35) correspondingly, input of relay 1 and relay 2 are connected to D32 and D33 respectively, SDA is connected to D21 and SCL connected to D22. The ESP32 and relays are 2 isolated circuit and they are connected using

optocoupler to protect the system from harming the circuit and devices by receiving the signal with extreme voltages. The relays are used to control the orientation of the antenna rotator where one relay is used to control the clockwise rotation, another relay is used to control the anti-clockwise rotation. A toggle switch is connected to D4 as an input signal to let user to switch on the manual rotation mode.

3.8 Software Development

The waterfall model is a software development methodology plan that is based on the concept of following a predefined sequence of events from top to bottom in the shape of a cascade. Its stages are closely associated with the software development life cycle (SDLC), from which it originated. Iterative waterfall software development processes gained popularity because they gave realistic instructions for producing software products. Each phase follows the completion of the previous one, and activities can be separated into stages. Even though the result of previous stage becomes the input of the following stage, developers have the option of reviewing stages in the iteration cycle (Trivedi & Sharma, 2013). Figure 3.16 illustrate the iterative waterfall model which have been implemented in this project.

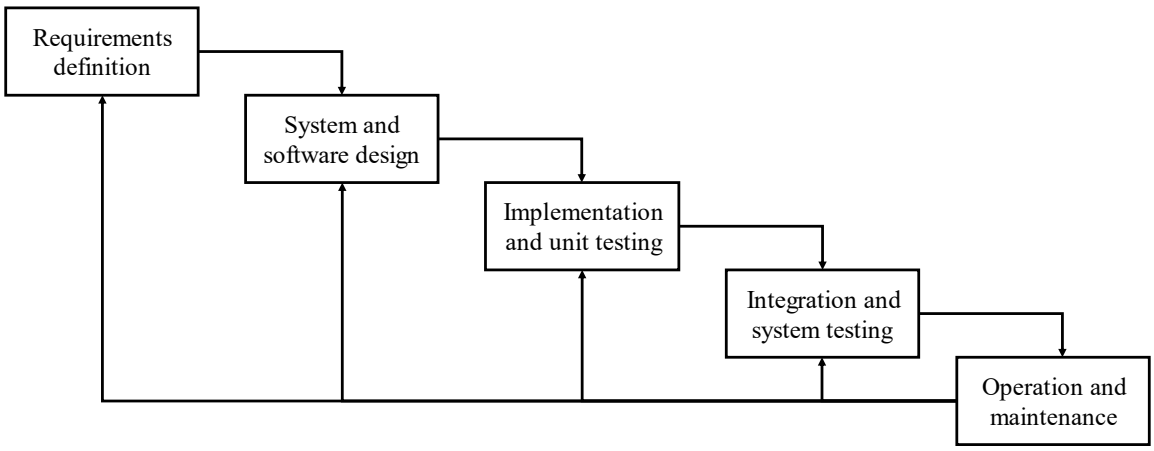


Figure 3.16 Iterative waterfall model

3.8.1 Requirements Analysis and Definition

The definition and analysis of requirements are concerned with consultation with system users establishes the system's services, restrictions, and aims. They are then detailed specified and serve as the basis for the system specification. In this project, GUI is used to enable the user to select the desired location or insert coordinate of a specific area to determine the direction of the antenna rotator. The software is designed using HTML, CSS and JavaScript scripting to build a web server for the system. Also, the Google Map is

integrated with the web server in order to achieve the objective of this project which is IoT integration technology. The user could access the web server by using browser. The data is then sent to the ESP32 and automatically be converted to the desire rotation angle to control the orientation of the antenna rotator. The value of the angles will be displayed on the LCD and the web server also able to show the real-time angle of the antenna rotator. The user also can manually control the antenna rotator either thru potentiometer or web server. This IoT integration can help users to control the direction of the antenna rotator in a most effective way.

The elements and the functions of the web server include:

- Integrated with Google Maps API.
- Selection of current and desire location in Google Maps.
- Get current location from Google Maps.
- Key in desire coordinate.
- Manually define rotation angle.
- Display real-time current angle of antenna rotator.

The features and the tasks of the ESP32 involve:

- Convert the coordinate information to angles.
- Decide the orientation of the antenna rotator.
- Compare the rotation angle with feedback angle (current angle).
- Allows user manually control rotation angle.
- Display current angle and desire angle on LCD.

3.8.2 System and Software Design (GUI)

A web server is built by using HTML, CSS and JavaScript scripting to act as a GUI for this system. The program is scripted in Visual Studio Code before being integrated with backend program (ESP32). Figure 3.17 until Figure 3.30 show the coding for constructing the web server. However, the full backend scripting is written in Appendix B.

Firstly, JavaScript scripting is built to define the makers' locations and prepare for allowing the users to drag the markers in Google Maps in order to obtain the coordinates. These steps are shown in Figure 3.17, Figure 3.18 and Figure 3.19.

```
<script>

    /* Global variables */
    let map = null;
    let marker1 = null;
    let marker2 = null;


    const curLocation = {
      lat: 3.221101104201,
      lng: 101.63787669557,

      get getGeo() {
        return {lat: this.lat, lng: this.lng};
      },
      set setGeo(cood) {
        this.lat = cood.lat;
        this.lng = cood.lng;
        // Side effect
        document.getElementById("myLat").value = this.lat;
        document.getElementById("myLng").value = this.lng;
        marker1.setPosition(this.getGeo);
      }
    };

    const desLocation = {
      lat: 3.221101104201,
      lng: 101.63787669557,
```

Figure 3.17 JavaScript scripting 1

```
  get getGeo() {
    return {lat: this.lat, lng: this.lng};
  },
  set setGeo(cood) {
    this.lat = cood.lat;
    this.lng = cood.lng;
    document.getElementById("lat").value = this.lat;
    document.getElementById("lng").value = this.lng;
    marker2.setPosition(this.getGeo);
  }
};

function initMap() {

  // Create map
  map = new google.maps.Map(document.getElementById("map"), {
    zoom: 4,
    center: curLocation.getGeo,
    pixelRatio: window.devicePixelRatio || 1
  });


  // Create markers
  marker1 = new google.maps.Marker({
    position: curLocation.getGeo,
    map,
    draggable:true,
    title: "Current location",
  });
```

Figure 3.18 JavaScript scripting 2

```
  marker2 = new google.maps.Marker({
    position: curLocation.getGeo,
    map,
    draggable:true,
    title: "Destination",
    icon: {url: "http://maps.google.com/mapfiles/ms/icons/blue-dot.png"},
  });
  currentLot()
  google.maps.event.addListener(marker1, 'dragend', function() {
    var coordinate = marker1.getPosition();
    curLocation.setGeo = {lat:coordinate.lat(), lng: coordinate.lng()}
  });

  google.maps.event.addListener(marker2, 'dragend', function() {

    var coordinate = marker2.getPosition();
    desLocation.setGeo = {lat:coordinate.lat(), lng: coordinate.lng()}
  });
}

function handleLocationError(browserHasGeolocation, infoWindow, pos) {
  infoWindow.setPosition(pos);
  infoWindow.setContent(
    browserHasGeolocation
      ? "Error: The Geolocation service failed."
      : "Error: Your browser doesn't support geolocation."
  );
  infoWindow.open(map);
}
```

Figure 3.19 JavaScript scripting 3

Figure 3.20 shows the coding for allowing the users to get their current location by using Google Maps API. For security purpose, the browser will ask for the user's permission before getting the user's current location.

```
function currentLot() {
    // Get current location using GMAP
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(
      (position) => {


        const markerDist = 2.0;
        curLocation.setGeo = {lat:position.coords.latitude, lng: position.coords.longitude}
        desLocation.setGeo = {lat:position.coords.latitude+markerDist, lng: position.coords.longitude+markerDist}
        map.panTo(curLocation.getGeo);



      },
       () => {
         handleLocationError(true);
       });
  } else {
    // Browser doesn't support Geolocation
    handleLocationError(false);
  }
}

</script>
```

Figure 3.20 JavaScript scripting 4

Figure 3.21, Figure 3.22, Figure 3.23, Figure 3.24 and Figure 3.25 show the CSS scripting describing the for the appearance of the web server. Figure 3.21 shows the dynamic bar to represent the real-time current angle of the antenna rotator. Figure 3.23, Figure 3.24 and Figure 3.25 show the design of the dynamic slider to enable user to choose the desire rotation angle.

```
<style>
#dynRectangle
{
    width:0px;
    height:20px;
    top: 9px;
    background-color: #7ccfbd;
    z-index: -1;
    margin-top:8px;
}
body   {background-color:#2b6777; font-family: 'Lato', sans-serif;}
h1     {font-size: 40px; color: #2b6777; text-align: center;}
h2     {font-size: 30px; color: rgb(255, 255, 255)}
h3     {font-size: 17px; color:rgb(255, 255, 255)}
.header {
background-color: #c8d8e4;;
padding: 0.5px;
text-align: center;
}

/* Google Map */
    #map {
  height: 50%;
  border-radius: 5px;
  margin-top: 0px;
  margin-left: 80px;
  margin-right: 80px;
  margin-bottom: 0px;
  padding: 40px;
}
```

Figure 3.21 CSS scripting 1

```
html,
body {
  height: 100%;
  margin: 0;
  text-align: center;
}
div.input {
  border-radius: 5px;
  background-color: #c8d8e4;
  margin-top: 0px;
  margin-left: 80px;
  margin-right: 80px;
  margin-bottom: 0px;
  padding-top: 40px;
  padding-bottom: 10px;
}
input[type=text], select {
  width: auto;
  padding: 5px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
}
```

Figure 3.22 CSS scripting 2

```css
button[type=button] {
  width: 100%;
  background-color: #52ab98;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  width: 15%;
}
button[type=button]:hover {
  background-color: #3a796b;
}
div.dynamic {
  margin-left: 205px;
  margin-right: 205px;
  padding-left: 205px;
  padding-right: 205px;
  color: white;

}
.slidecontainer {
  width: auto;
  padding: 0px 400px;
  color: white;
}
```

Figure 3.23 CSS scripting 3

```css
.slider {
  -webkit-appearance: none;
  width: 100%;
  height: 15px;
  border-radius: 5px;
  background: #d3d3d3;
  outline: none;
  opacity: 0.7;
  -webkit-transition: .2s;
  transition: opacity .2s;
}

.slider:hover {
  opacity: 1;
}

.slider::-webkit-slider-thumb {
  -webkit-appearance: none;
  appearance: none;
  width: 25px;
  height: 25px;
  border-radius: 50%;
  background: #04AA6D;
  cursor: pointer;
  transition: height 0.2s ease-in-out;
 }
```

Figure 3.24 CSS scripting 4

```css
.slider::-webkit-slider-thumb:hover {
  -webkit-appearance: none;
  appearance: none;
  width: 25px;
  height: 35px;
  border-radius: 50%;
  background: #04AA6D;
  cursor: pointer;
}

.slider::-moz-range-thumb {
  width: 25px;
  height: 25px;
  border-radius: 50%;
  background: #04AA6D;
  cursor: pointer;
}
```

Figure 3.25 CSS scripting 5

Figure 3.26 shows the function for initialize the web socket once the user enters the web server by using browser. Figure 3.26 and Figure 3.27 show the functions which are triggered when user click the button to send the data from web server to backend (ESP32).

```
InitWebSocket()
function InitWebSocket()
{
  websock = new WebSocket('ws://'+window.location.hostname+':81/');
  websock.onmessage=function(evt)
  {
    JSONobj = JSON.parse(evt.data);
    document.getElementById('POTvalue').innerHTML = JSONobj.POT;
    var pot = parseInt(JSONobj.POT*2);//length
    var elem = document.getElementById("dynRectangle");
    var width = pot;
    elem.style.width = pot+"px";

  }
}

function sendLocation(){

 var new_lot={
   my_lat:document.getElementById("myLat").value,
   my_lng:document.getElementById("myLng").value,
   new_lat:document.getElementById("lat").value,
   new_lng:document.getElementById("lng").value,
   manual_angle:'0',
 };
 console.log(new_lot);
 websock.send(JSON.stringify(new_lot));
 alert("Coordinates has been sent.");

}
```

Figure 3.26 JavaScript scripting 5

```
function manualRotation(){

  var manual_lot={
    my_lat:'0',
    my_lng:'0',
    new_lat:'0',
    new_lng:'0',
    manual_angle:document.getElementById("myRange").value,
    };
    console.log(manual_lot);
  websock.send(JSON.stringify(manual_lot));
  alert("Rotation angle has been sent.");

}
```

Figure 3.27 JavaScript scripting 6

Figure 3.28 shows the HTML scripting for displaying the real-time updating the current angle of antenna rotator and presenting the Google Maps in the web server.

```
<body>
  <div class="header">
    <h1>IoT Integrated Antenna Rotator</h1>
  </div>
    <h2>
    Current Bearing: <span style="color:#7ccfbd" id="POTvalue">0</span> degree
    </h2>
    <div style="color:white;">
     0°             
                     
                      360°
    </div>
   <div class="dynamic">


        <div id="dynRectangle"></div>
    </div>
    <br>

    <!-----------------------------------------Google Map---------------------------------------------------------->
    <div id="map"></div>

    <!-- Async script executes immediately and must be after any DOM elements used in callback. -->
    <script
      src="https://maps.googleapis.com/maps/api/js?key==initMap&v=weekly"
      async
    ></script>
```

Figure 3.28 HTML scripting 1

Figure 3.29 shows the HTML scripting for displaying and inserting the coordinate information as well as the manual control rotation angle from web server.

```
  <br>
  <div class="input">
  <form name="input">
  Current Latitude: <input type="text" name="myLat" id="myLat" value="">  
  Current Longitude: <input type="text" name="myLng" id="myLng" value="">  

  New Latitude: <input type="text" name="lat" id="lat" placeholder="Type or drag the cursor">  
  New Longitude: <input type="text" name="lng" id="lng" placeholder="Type or drag the cursor">  
     
  <button type="button" onclick="currentLot()">Reset Location</button>
  <button type="button" onclick="sendLocation()">Send</button>
  </form>
</div>

<!--Manually control rotation angle with range slider-->
<h2>Manually Control Rotation Angle</h2>
<div class="slidecontainer">
  <input type="range" min="0" max="290" value="50" class="slider" id="myRange">

  <p>Rotation Angle: <span id="m_angle"></span>°</p>
  <button type="button" onclick="manualRotation()">Rotate</button>
</div>
```

Figure 3.29 HTML scripting 2

Figure 3.30 shows the JavaScript to enable the user to use the slider and obtain the slider value as the angle for manual control rotation.

```
<script>
  var slider = document.getElementById("myRange");
  var output = document.getElementById("m_angle");
  output.innerHTML = slider.value;

  slider.oninput = function manual() {
    var manualVal = this.value;
    output.innerHTML = manualVal;
  }
</script>
```

Figure 3.30 JavaScript scripting 7

3.8.3 System and Software Design (ESP32)

ESP32 is programmed in C/C++ with Arduino IDE. Figure 3.31 until Figure 3.39 show the coding for constructing the system in ESP32. However, the full backend scripting is written in Appendix C. Figure 3.31 shows the header files and libraries used in this project. JSON library is used to prepare the static memory which will be used in transmitting data between frontend and backend. Web server and web socket server is located at port 80 and 81 respectively after insert the SSID and password of the Wi-Fi network.

```
#include <WiFi.h>
#include <WebServer.h>
#include <WebSocketsServer.h>
#include <ArduinoJson.h>
#include <LiquidCrystal_I2C.h>

// set the LCD number of columns and rows
int lcdColumns = 16;
int lcdRows = 2;
LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);

// The JSON library uses static memory, so this will need to be allocated:
StaticJsonDocument<200> doc_tx;                     // provision memory for about 200 characters
StaticJsonDocument<200> doc_rx;

// We want to periodically send values to the clients, so we need to define an "interval" and remember the last time we sent data to the client (with "previousMillis")
int interval = 1000;                                // send data to the client every 1000ms -> 1s
unsigned long previousMillis = 0;                   // we use the "millis()" command for time reference and this will output an unsigned long

//-----------------------------------------------
const char* ssid = "";
const char* password = "";
//-----------------------------------------------
WebServer server(80);
WebSocketsServer webSocket = WebSocketsServer(81);
```

Figure 3.31 ESP32 coding 1

Figure 3.32 shows the coding for declaring the variables and include the html sketch file named “webpage.h” which is the GUI for the project. The scripting descripted in System and Software Design (GUI) is saved as “webpage.h” in the same sketch folder and to be uploaded into ESP32 later.

```
//-----------------------------------------------
String JSONtxt;
const char* PARAM_INPUT_3 = "input3";//D34
const char* PARAM_INPUT_4 = "input4";//D35
const int RELAY1_PIN = 32;//cw
const int RELAY2_PIN = 33;//ccw
const int SWpin = 4; //SW manual/online

float c_lat;
float c_lng;
float n_lat;
float n_lng;
int manual_angle;
int angle;
int rotation_angle;
String POTvalString;
float POTvalue;
int manual_lot;
//int SW_POT;

// variable for storing the pushbutton status
int SWstate = 0;

//-----------------------------------------------
#include "webpage.h"
//-----------------------------------------------
void handleRoot()
{
  server.send(200,"text/html", webpageCont);
}
```

Figure 3.32 ESP32 coding 2

Figure 3.33 shows the program in setup() which only runs once when the system is activated. The direction of antenna rotator remains unchanged when the system is turned on and the system is connected to the Wi-Fi once the system is started. The web server, web socket and LCD is initialized at this stage too.

```
void setup()
{
  Serial.begin(115200);
  /*Initial rotation angle*/
  rotation_angle = int(0.113961*(analogRead(A6))+10.7873);//initial state of the rotator remain unchanged.
 pinMode (RELAY1_PIN, OUTPUT);
 pinMode (RELAY2_PIN, OUTPUT);
 pinMode(SWpin, INPUT);
  WiFi.begin(ssid, password);
  while(WiFi.status() != WL_CONNECTED)
  {
    Serial.print("."); delay(500);
  }
  WiFi.mode(WIFI_STA);
  Serial.print(" Local IP: ");
  Serial.println(WiFi.localIP());

  server.on("/", handleRoot);
  server.begin(); webSocket.begin();
  webSocket.onEvent(webSocketEvent);

  // initialize LCD
  lcd.init();
  // turn on LCD backlight
  lcd.backlight();
  lcd.clear();

}
```

Figure 3.33 ESP32 coding 3

Inside the loop function as shown in Figure 3.34, the program loops sequentially, which enables the program to adapt and respond. The system utilizes it to exert active control over the ESP32 board. The ESP32 will keep reading the feedback value from antenna rotator and updating to frontend (GUI) using JSONtxt. The manual rotation is enabled if the toggle switch is turned to “HIGH” condition. The LCD will continuously display the real-time current angle and desire angle of the system.

```
void loop()
{
  webSocket.loop(); server.handleClient();

unsigned long now = millis();                     // read out the current "time" ("millis()" gives the time in ms since the Arduino started)
  if ((unsigned long)(now - previousMillis) > interval) { // check if "interval" ms has passed since last time the clients were updated

  POTvalue=(0.113961*(analogRead(A6))+10.7873);//D34
  POTvalString = String(POTvalue);
  //Serial.println(POTvalString);
  JSONtxt = "{\"POT\":\""+POTvalString+"\"}";
  webSocket.broadcastTXT(JSONtxt);
 previousMillis = now;                            // reset previousMillis
  }

  SWstate=digitalRead(SWpin);

  if (SWstate == HIGH) {
    Serial.println("******Manual rotation activated******");
    Serial.println("");
    // turn on manual rotation
    POTvalue=(0.113961*(analogRead(A6))+10.7873);//D34
    rotation_angle=(((analogRead(A7))*290)/4096);//D35
    rotation();

  //lcd.clear();
    // set cursor to first column, first row
  lcd.setCursor(0, 0);
  lcd.print("Current: ");
  lcd.print(POTvalue);
  lcd.setCursor(0, 1);
  lcd.print("Desire: ");
  lcd.print(rotation_angle);
  delay(100);
  }
```

Figure 3.34 ESP32 coding 4

Figure 3.35 show the web socket event which is triggered when data is sent from frontend to backend.

```
void webSocketEvent(uint8_t num, WStype_t type, uint8_t *payload, size_t length){
    switch (type) {                                 // switch on the type of information sent
    case WStype_DISCONNECTED:                       // if a client is disconnected, then type == WStype_DISCONNECTED
      Serial.println("Client " + String(num) + " disconnected");
      break;
    case WStype_CONNECTED:                          // if a client is connected, then type == WStype_CONNECTED
      Serial.println("Client " + String(num) + " connected");
      // optionally you can add code here what to do when connected
      break;
    case WStype_TEXT:                               // if a client has sent data, then type == WStype_TEXT
      // try to decipher the JSON string received
      DeserializationError error = deserializeJson(doc_rx, payload);//catch if there's error or not
      if (error) {
        Serial.print(F("deserializeJson() failed: "));
        Serial.println(error.f_str());
        return;
      }
```

Figure 3.35 ESP32 coding 5

Figure 3.36 show the data from GUI is being stored in backend and the program will differentiate either user choose coordinate information or manual rotation angle to control the antenna rotator.

```
    else {
      // JSON string was received correctly, so information can be retrieved:
        c_lat = doc_rx["my_lat"];
        c_lng = doc_rx["my_lng"];
        n_lat = doc_rx["new_lat"];
        n_lng = doc_rx["new_lng"];
        manual_lot = doc_rx["manual_angle"];

      Serial.println("Received coordinate from user: ");
      Serial.println("Current Latitude: " + String(c_lat));
      Serial.println("Current Longitude: " + String(c_lng));
      Serial.println("New Latitude: " + String(n_lat));
      Serial.println("New Longitude: " + String(n_lng));
      Serial.println("Manual rotation angle: " + String(manual_lot));
      Serial.println("");
      //Serial.println(payload[0]);

      if (manual_lot==0){
        rotation_angle=(int)getBearing();
      }
      else{
        rotation_angle=manual_lot;
      }
      lcd.clear();
    }
    break;
  }
}
```

Figure 3.36 ESP32 coding 6

Figure 3.37 shows the function to calculate the bearing based on the coordinate information. The angle is being used as the desire angle when user select the locations from Google Maps in web server and send to ESP32.

```
float getBearing(){

    float theta1 = c_lat;
    float theta2 = n_lat;
    float delta1 = n_lat-c_lat;
    float dL = n_lng-c_lng;

    //--------------Calculation--------------//

     float x = cos(theta2) * sin(dL) ;
    float y = cos(theta1)*sin(theta2) - sin(theta1)*cos(theta2)*cos(dL);

    float bearing = atan2(x,y);
    bearing = degrees(bearing);// radians to degrees

    angle = (((int)bearing + 360) % 360); //change the range from -180~180 to 0~360

    Serial.print("Bearing: ");
    Serial.println(angle);

    return angle;

  }
```

Figure 3.37 ESP32 coding 7

Figure 3.38 and Figure 3.39 shows the function of rotation which control the orientation of the antenna rotator by switching the relays. The desire rotation angle is compared with the current (feedback) angle of antenna rotator. Based on the conditions, the relays would be turned off or on in order to control the antenna rotator to turn clockwise , anticlockwise or stop. A range of 10 degree is program due to the detection of fluctuation in feedback mechanism.

```
void rotation(){
int current_angle=(int)(POTvalue);
Serial.println("current: "+String(current_angle)+" vs rotation: "+String(rotation_angle));
int Xval = rotation_angle - current_angle;
if(rotation_angle==current_angle || (Xval>-10 && Xval<10) ){//if manual=feedback
 digitalWrite(RELAY1_PIN, LOW);
 digitalWrite (RELAY2_PIN, HIGH);
 Serial.println("***Rotation Complete***");
  Serial.println(" ");
   lcd.setCursor(0, 0);
   lcd.print("Current: ");
   lcd.print(POTvalue);
   lcd.setCursor(0, 1);
   lcd.print("Desire: ");
   lcd.print(rotation_angle);
   delay (500);
}

 else if(rotation_angle>current_angle){ //new location > current location, turn RIGHT
  digitalWrite (RELAY1_PIN, LOW);
  digitalWrite (RELAY2_PIN, LOW);
  Serial.println("Rotating... ...(left)");
  Serial.println(" ");
   lcd.setCursor(0, 0);
   lcd.print("Current: ");
   lcd.print(POTvalue);
   lcd.setCursor(0, 1);
   lcd.print("Desire: ");
   lcd.print(rotation_angle);
   delay (500);
 }
```

Figure 3.38 ESP32 coding 8

```
else if (rotation_angle<current_angle){//new location < current location, turn LEFT
  digitalWrite (RELAY1_PIN, HIGH); //D32
  digitalWrite (RELAY2_PIN, HIGH); //D33
  Serial.println("Rotating... ...(right)");
  Serial.println(" ");
   lcd.setCursor(0, 0);
   lcd.print("Current: ");
   lcd.print(POTvalue);
   lcd.setCursor(0, 1);
   lcd.print("Desire: ");
   lcd.print(rotation_angle);
   delay (500);
 }
}
```

Figure 3.39 ESP32 coding 9

3.9 Calibration and Calculations

Due to the orientation limitation of the Dennard 2000, the antenna rotator is only able to rotate from 0° until 290°. The antenna rotator is then calibrated to make sure the angles and the required input voltage for the orientation. Figure 3.40 shows the circular protractor pasted on the antenna rotator in order to measure the relationship between angle of the antenna rotator and the required voltage. The data is recorded and plotted into a graph. Figure 3.41 illustrate the graph of the angle of antenna rotator versus ADC (analog to digital conversion) value.

Figure 3.40 Antenna rotator calibration

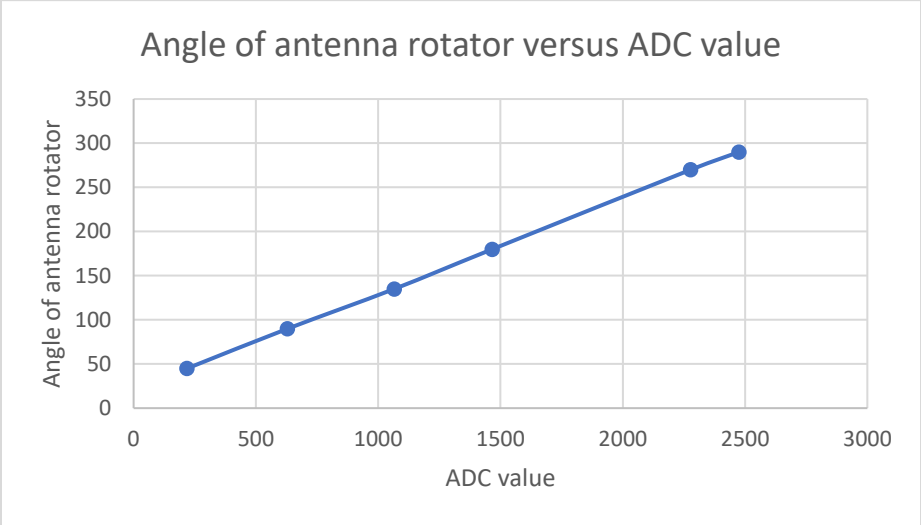


Figure 3.41 Angle of antenna rotator versus ADC value

Based on the relationship shown in Figure 3.41, Eq. (3.1) is expressed for calculating the feedback value or current angle of antenna rotator.

$$Current\ angle = (0.113961 \times ADC\ value\ of\ feedback\ potentiometer) + 10.7873 \quad 3.1$$

The angle of the manual rotation via potentiometer is expressed in Eq. (3.2).

$$Manual\ rotation\ angle = \frac{(ADC\ value\ of\ potentiometer \times 290)}{4096} \quad 3.2$$

The bearing between two points a and b is calculated mathematically by obtaining the inverse tan function of X and Y as in Eq. (3.3).

$$\text{bearing} = \tan^{-1}(X, Y) \quad 3.3$$

X and Y are defined as Eq. (3.4).

$$X = cos\ \theta b \times sin\ \Delta L \quad 3.4$$

$$Y = cos\ \theta a \times sin\ \theta b - sin\ \theta a \times cos\ \theta b \times cos\ \Delta L$$

a and b represent the two coordinates, and their prefixes are given in Eq. (3.5).

$$L = Longitude \quad 3.5$$

$$\theta = Latitude$$

$and\ \Delta L\ is\ the\ difference\ between\ the\ Longitudal\ values\ of\ the\ two\ points.$

The vector direction of the calculated bearing is illustrated as in Figure 3.42.

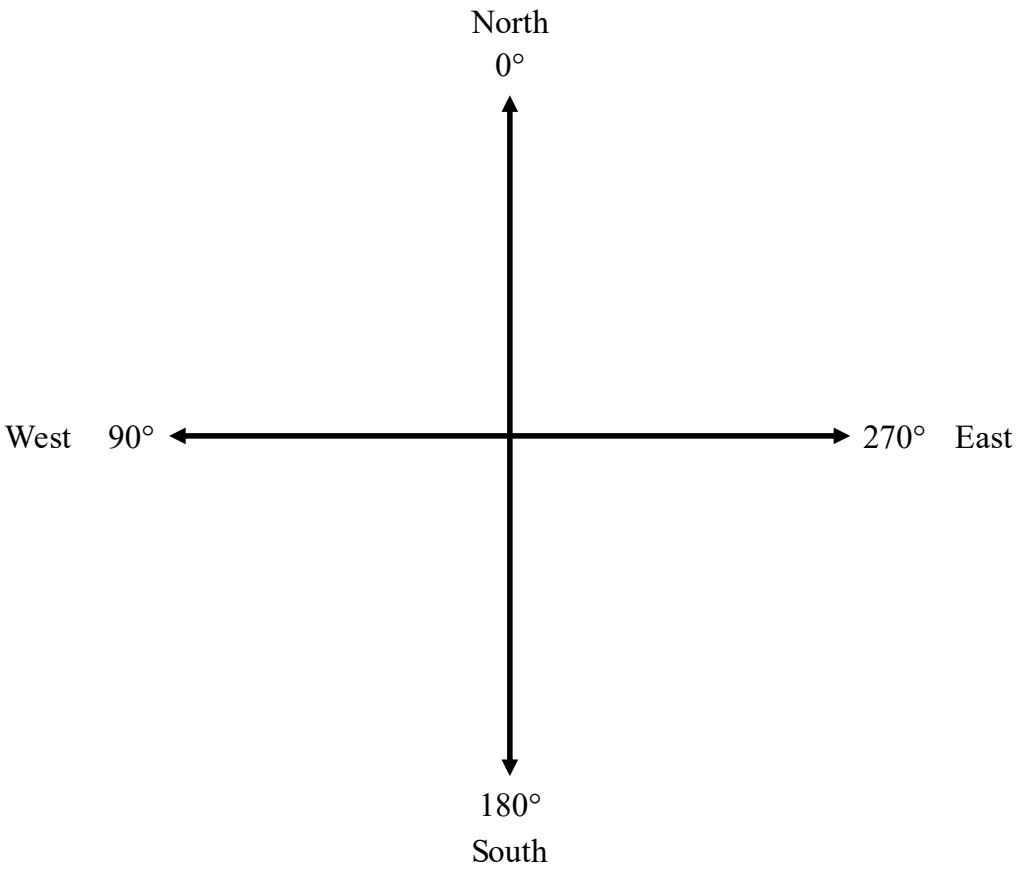


Figure 3.42 Vector direction of the calculated bearing

3.10 Project Cost and Material List

The estimated project cost and the selected materials are listed in Table 3.1. The total cost is around RM74.72. Every member of the project is allowed to claim the cost of material up to RM300.00. Hence, the project cost is within the claiming range.

Table 3.1 Project cost and material list

No	Material / Component	Specification	Supplier/ Source	Quantity	Estimated cost (RM)
1	NodeMCU ESP32	WROOM DEVKIT	Cytron	1	38.72
2	Potentiometer	10kΩ	Reuse from previous project	2	0.00
3	LCD	JHD162A with driver	Reuse from previous project	1	0.00
4	Prototype PCB board	72mm x 47mm	Apply from faculty	1	0.00
5	Connector	2-pin	Apply from faculty	10	0.00
7	Transformer	240VAC/24AC	Reuse from previous project	1	0.00
9	Relay	SRD-5VDC-SL-C	Reuse from previous project	2	0.00
10	Optocoupler	TLP621-4	Apply from faculty	1	0.00
11	PCB Stand	-	Reuse from previous project	20	0.00
12	Antenna rotator	Dennard Type 2000	Reuse from previous project	1	0.00
13	Ethernet Shield	Model V2.0	Cytron	1	36.00
14	Resistor	220Ω	Apply from faculty	4	0.00
15	Resistor	3.8kΩ	Apply from faculty	4	0.00
TOTAL ESTIMATED COST = RM 74.72					

CHAPTER 4

RESULTS AND DISCUSSION

This chapter details the implementation and unit testing, integration and system testing and discussion of the outcomes in order to verify that each unit meets its specification and validate this project achieves the objectives.

4.1 Implementation and Unit Testing

At this phase, the software design is accomplished as a set of programs or program units. The required hardware to conduct the unit testing are assembled as in Figure 4.1, Figure 4.2 and Figure 4.3. The assembled hardware is known as control box for this project.

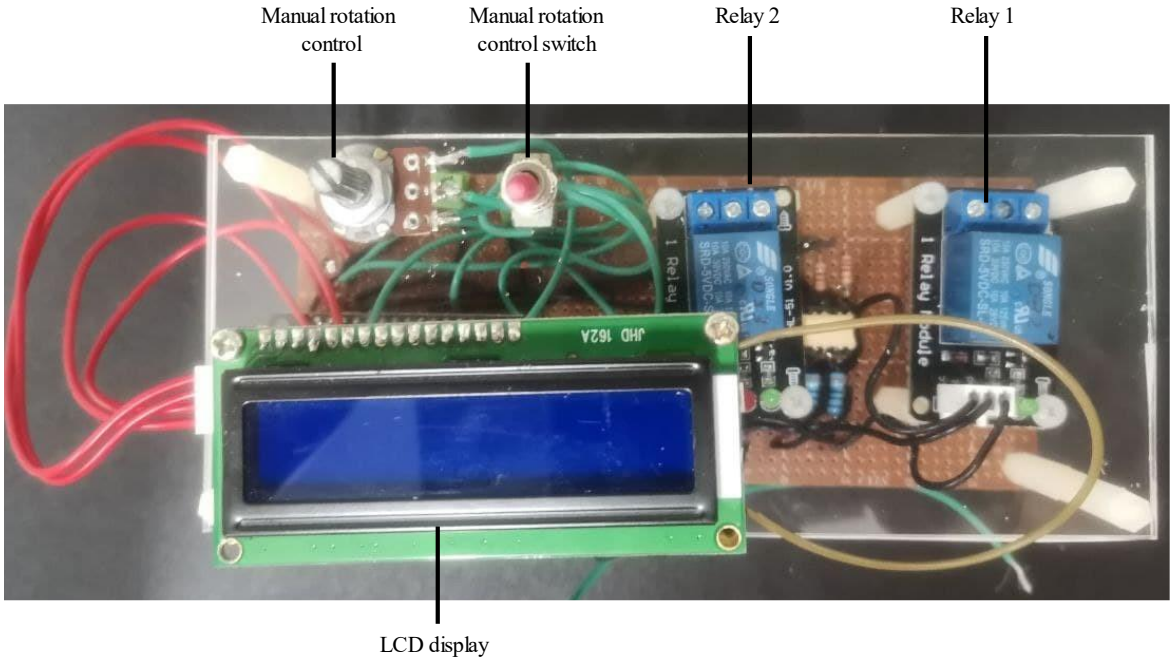


Figure 4.1 Top view of hardware assembly

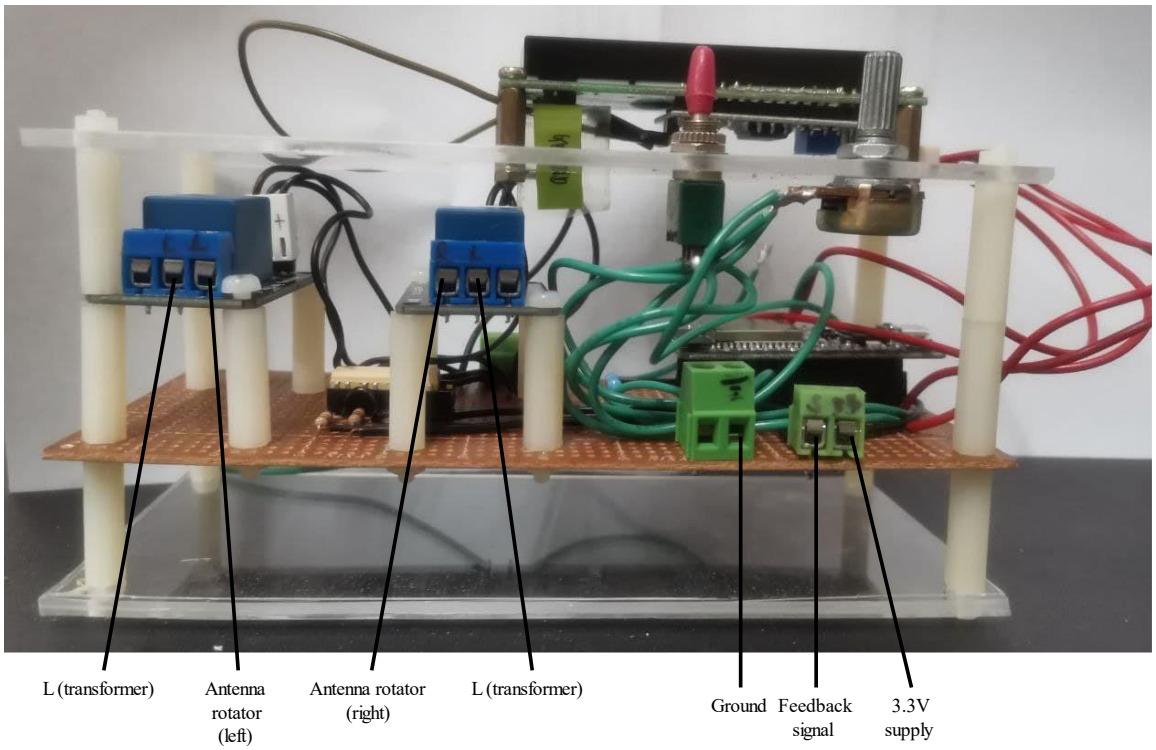


Figure 4.2 Side view of hardware assembly 1

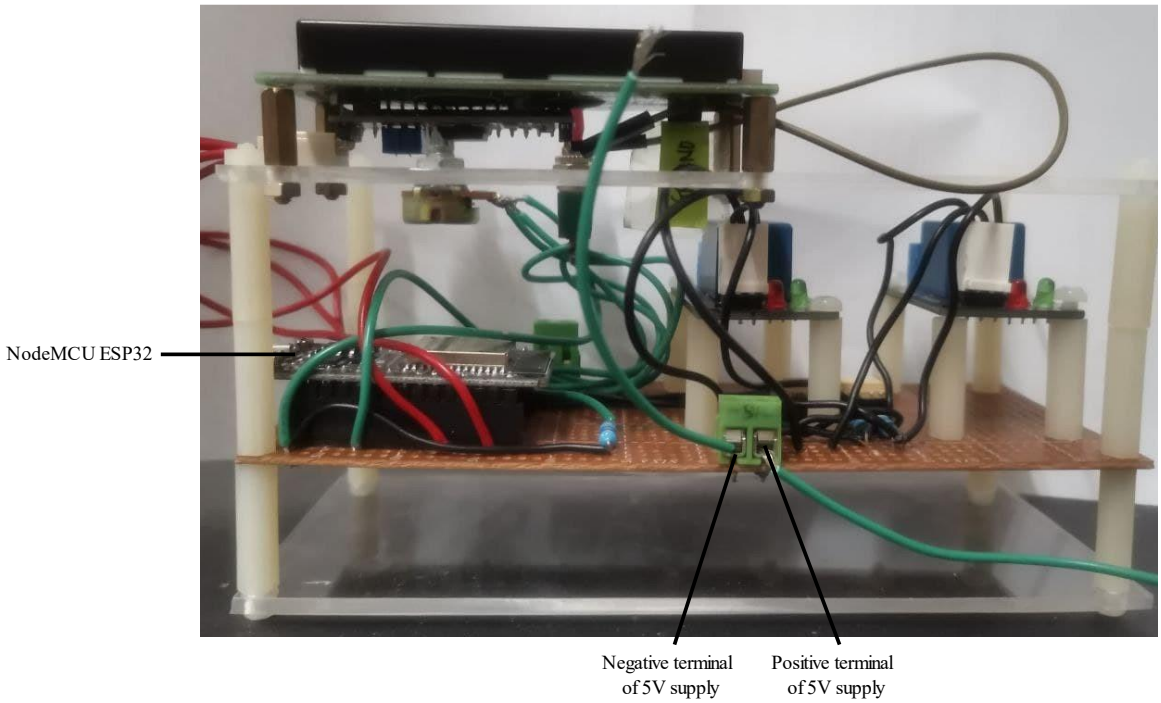


Figure 4.3 Side view of hardware assembly 2

For unit testing purpose, a potentiometer is connected to simulate the feedback mechanism of the antenna rotator as in Figure 4.4.

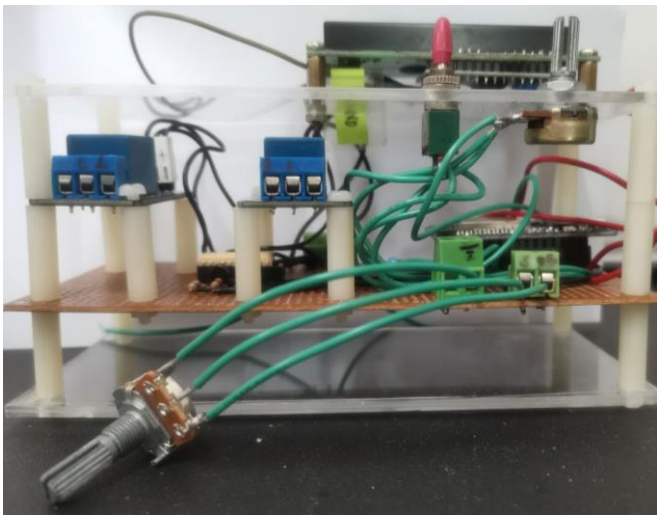


Figure 4.4 Simulation of feedback mechanism

The software program is uploaded into ESP32 by using USB cable as in Figure 4.5, and the Arduino IDE shows message after the upload process is done.

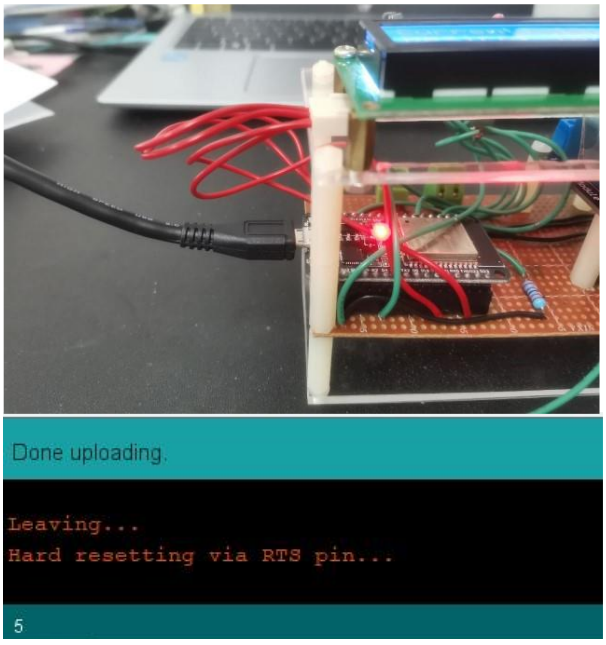


Figure 4.5 Upload program to ESP32

Figure 4.6 shows the serial monitor in Arduino IDE to observe the parameters and results of the system. When the system is started, the ESP32 is connecting to the Wi-Fi and web server. After the connection is completed, an IP address showed up at serial monitor. At beginning stage, the direction of the antenna rotator remains unchanged and the serial monitor will display “Rotation Complete” to indicate the antenna rotator is in stop mode.

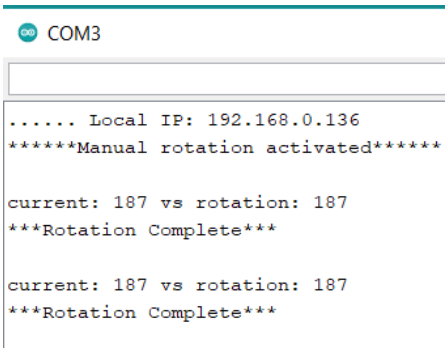


Figure 4.6 Getting IP address

User can access the web server of IoT Integrated Antenna Rotator by typing the IP address inside a web browser. The layout of the GUI is shown in Figure 4.7. The real-time current bearing of antenna rotator can be observed in web browser and the LCD displays the current angle and desire angle of the system as in Figure 4.8.

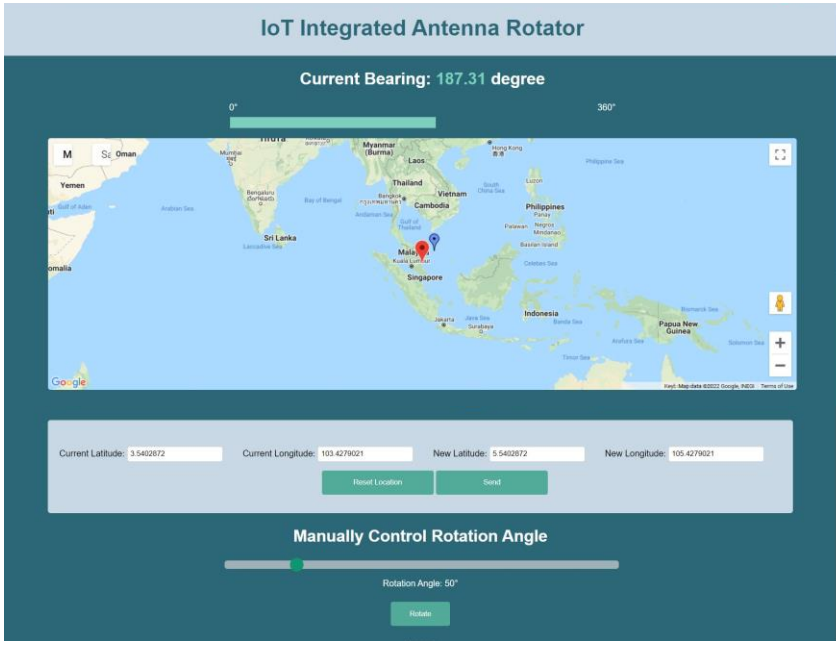


Figure 4.7 GUI layout (web server)

Figure 4.8 LCD display

For enhance the security feature and protect the privacy of the user, the web browser will request the permission from user before the system obtain the current location of the user. A pop-up window appears as in Figure 4.9 when user first entering the web server or click the “Get Current Location” button as in Figure 4.10. The result of the action is shown in Figure 4.11.

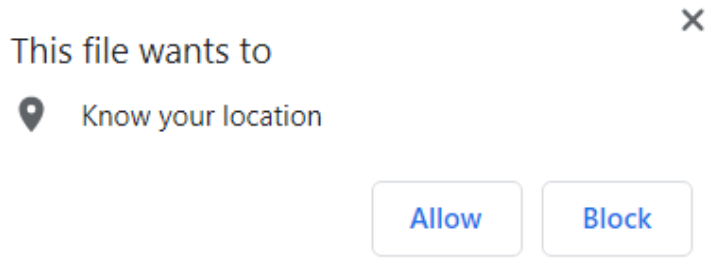


Figure 4.9 Request current location permission

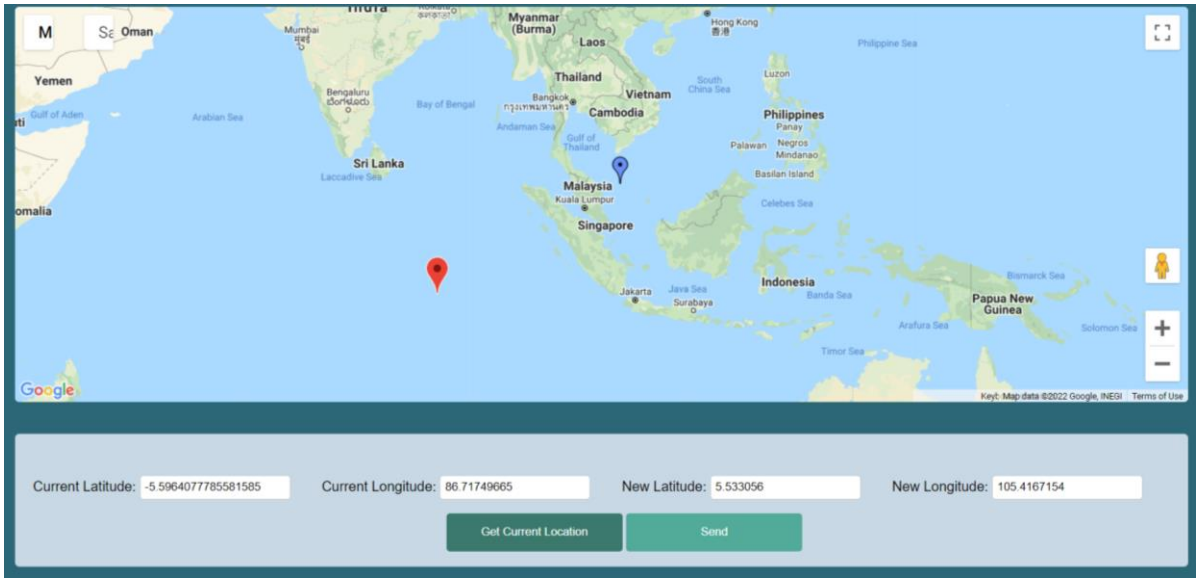


Figure 4.10 Get current location (before)

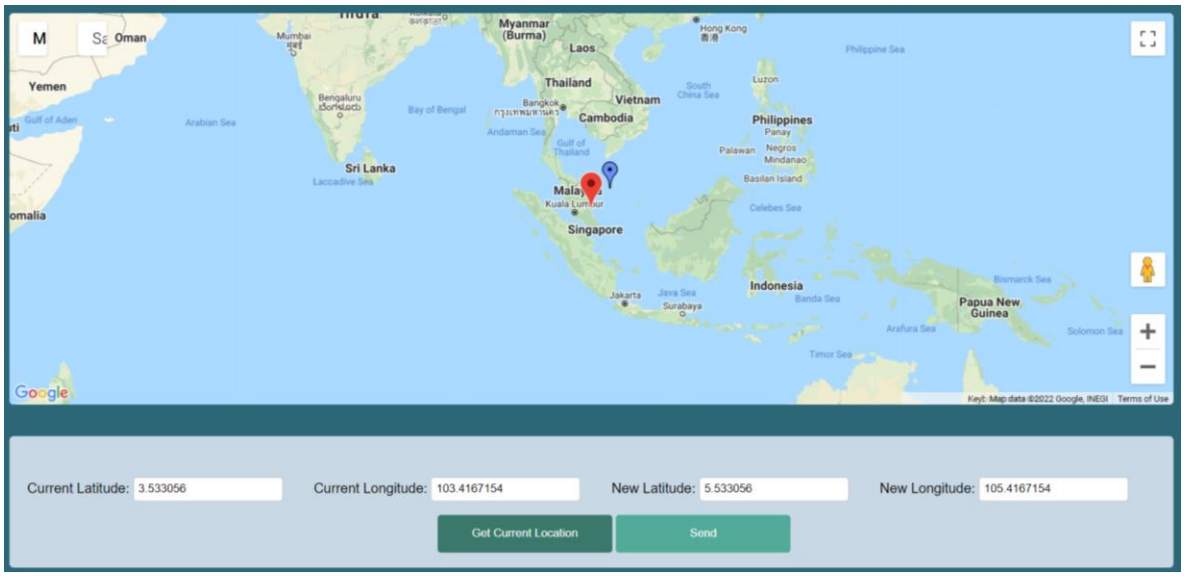


Figure 4.11 Get current location (after)

The red marker indicates the current location of the user and the blue marker represents the desire location that antenna rotator is required to point. Both markers are draggable, and the coordinates information is updated in the text boxes. User can type the coordinate information if they know or simply utilize the Google Maps. When user click “Send” button as in Figure 4.12, an alert window shows up indicates the coordinate data has been sent to ESP32. Figure 4.13 shows the serial monitor when the data has been received in backend and the bearing of the desire rotation value is calculated. The antenna rotator starts to rotate until the current angle and desire angle are close to each other. Web server and LCD will keep update the latest angles of the system as in Figure 4.14.

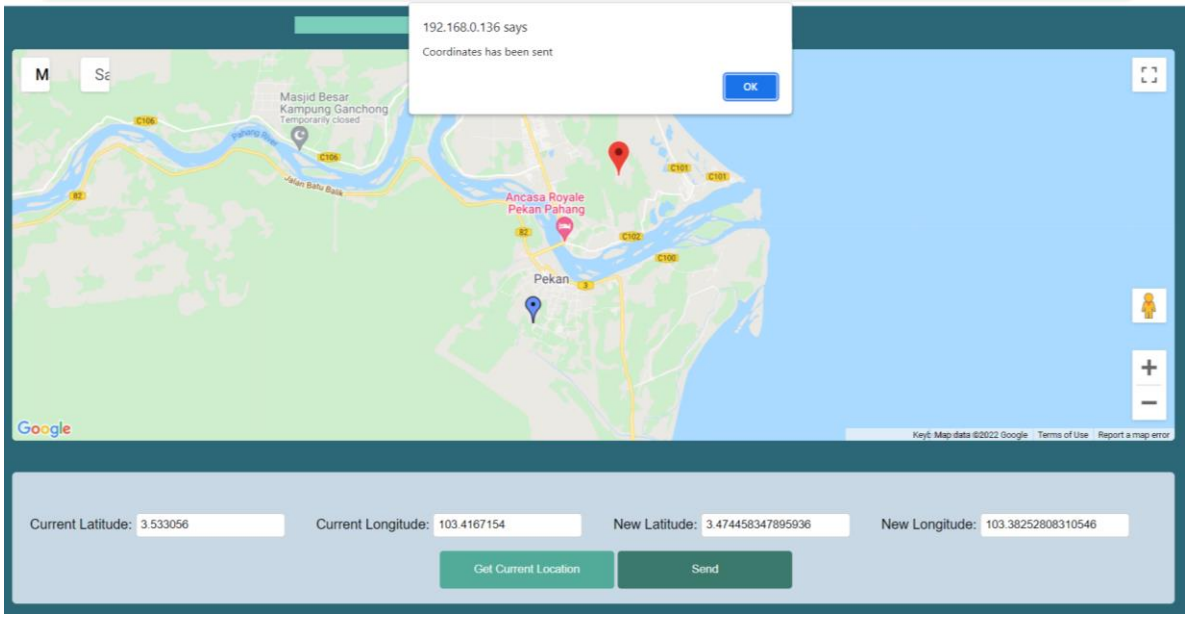
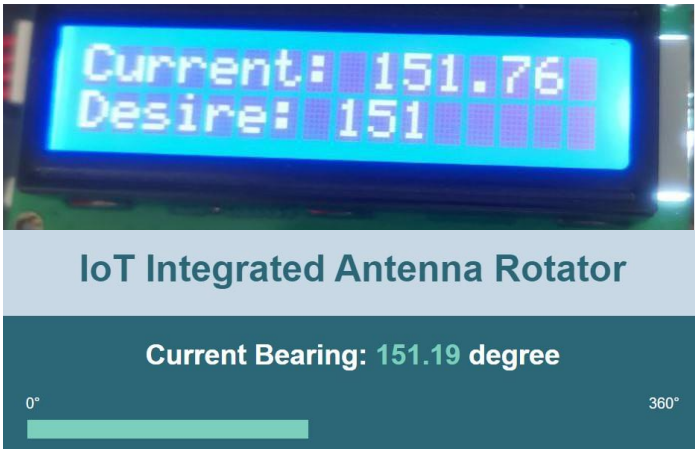


Figure 4.12 Sending coordinate information from web browser to ESP32

```
Received coordinate from user:
Current Latitude: 3.53
Current Longitude: 103.42
New Latitude: 3.47
New Longitude: 103.38
Manual rotation angle: 0

Bearing: 151
current: 187 vs rotation: 151
Rotating... ...(right)
```

Figure 4.13 Received coordinate information in ESP32




```
current: 151 vs rotation: 151
***Rotation Complete***

current: 151 vs rotation: 151
***Rotation Complete***

current: 151 vs rotation: 151
***Rotation Complete***
```

Figure 4.14 Angles update 1

User also can utilize the slider and send the desire angle from web server to ESP32 to rotate the antenna rotator as in Figure 4.15. Figure 4.16 shows the backend information during the data is received, and Figure 4.17 shows the angles update in LCD, web server and serial monitor.

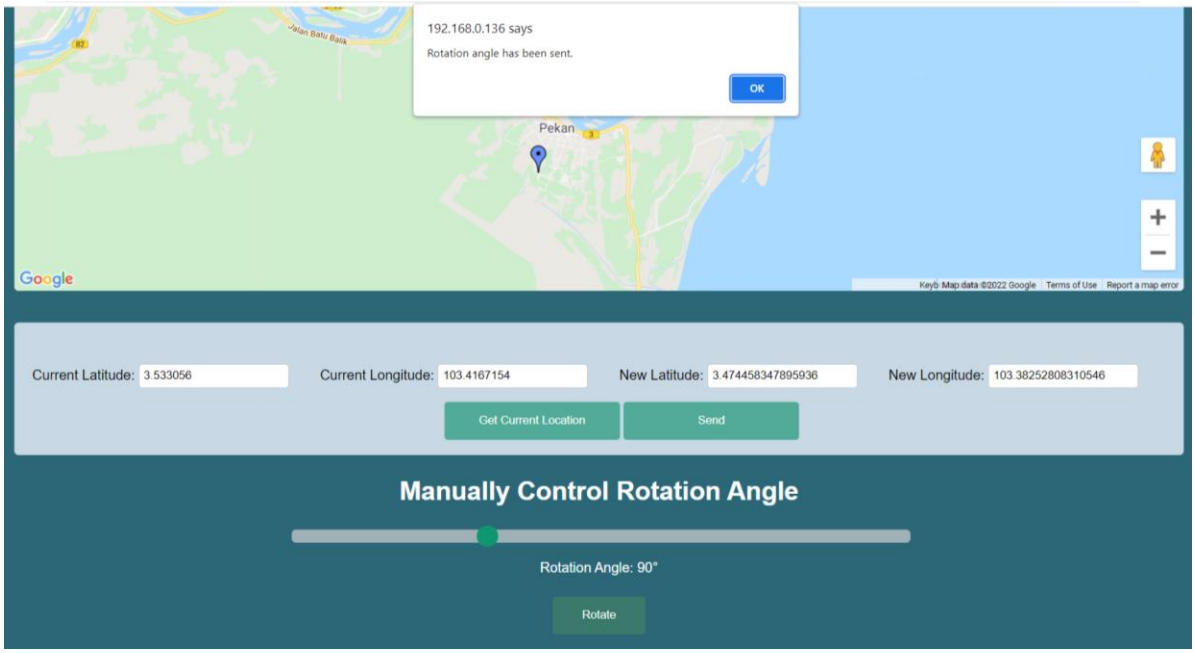


Figure 4.15 Sending desire rotation angle from web server to ESP32

```
Received coordinate from user:
Current Latitude: 0.00
Current Longitude: 0.00
New Latitude: 0.00
New Longitude: 0.00
Manual rotation angle: 90

current: 151 vs rotation: 90
Rotating... ...(right)
```

Figure 4.16 Received desire rotation angle in ESP32

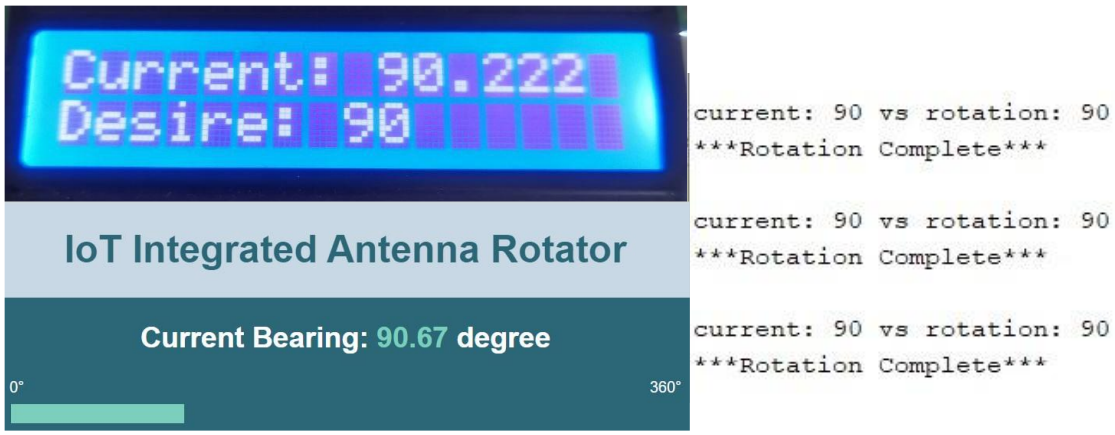


Figure 4.17 Angles update 2

User can manually control the orientation of antenna rotator using potentiometer by simply toggle the switch to “ON”. When the manual mode is activated, serial monitor shows “Manual rotation activated” as in Figure 4.18 and the antenna rotator rotates according to the potentiometer. Figure 4.19 shows the angles update in LCD, web server and serial monitor, the antenna rotator stop when the current angle reaches the desire angle.

```
******Manual rotation activated******

current: 89 vs rotation: 185
Rotating... ...(left)
```

Figure 4.18 Manual rotation mode being activated

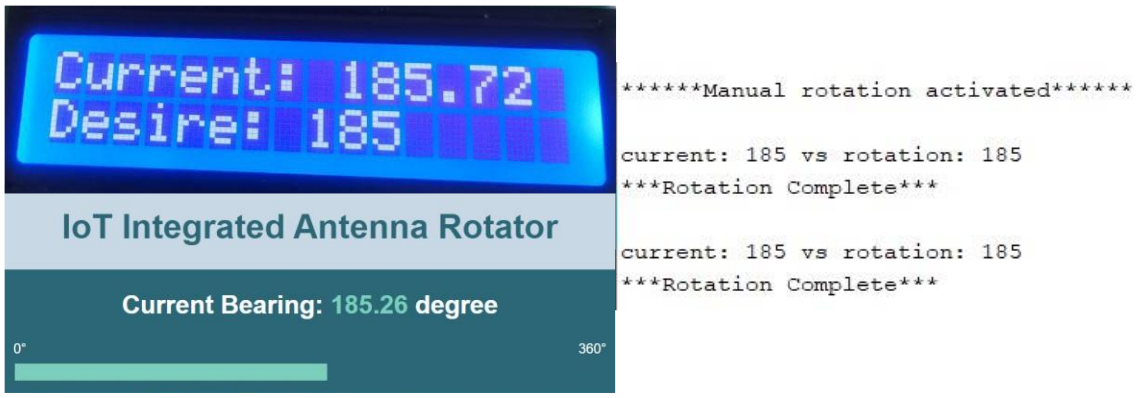


Figure 4.19 Angles update 3

Figure 4.20 shows the web server layout by using web browser in a smart phone. Users are able to access the web server by simply enter the IP address in the web browser of phone and control the antenna rotator or monitor the system.

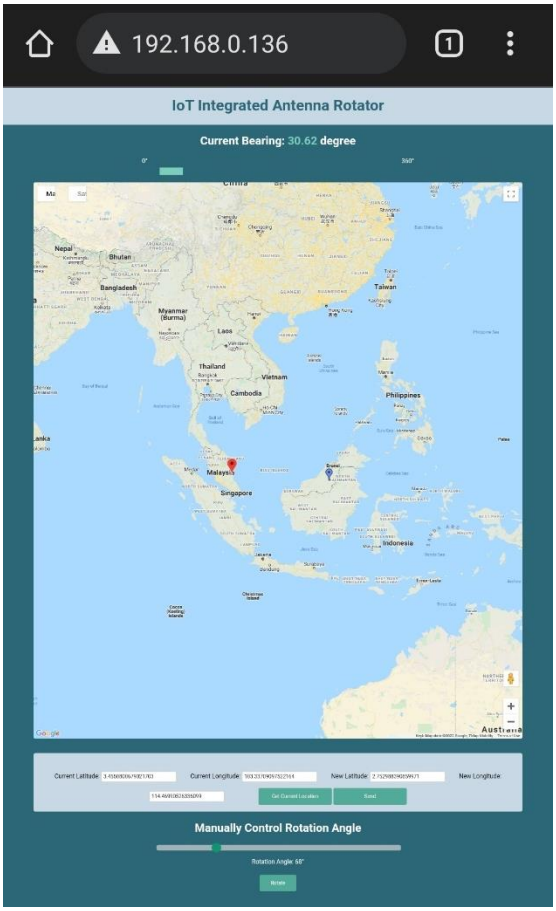


Figure 4.20 Web server layout in smartphone

4.2 Integration and System Testing

In this stage, the individual program units or the software programs are integrated with the hardware components. The full system is tested to assure compliance with the software requirements and achieve the objective of the project.

Figure 4.21 shows the hardware connection of this project. In order to increase the accuracy of the rotation angle of antenna rotator, 2 protractors are pasted at the rotation plate of antenna rotator as in Figure 4.22 to form a circular protractor. The calibration of the system has been made as stated in Chapter 3.9 with the assistance of the protractor.

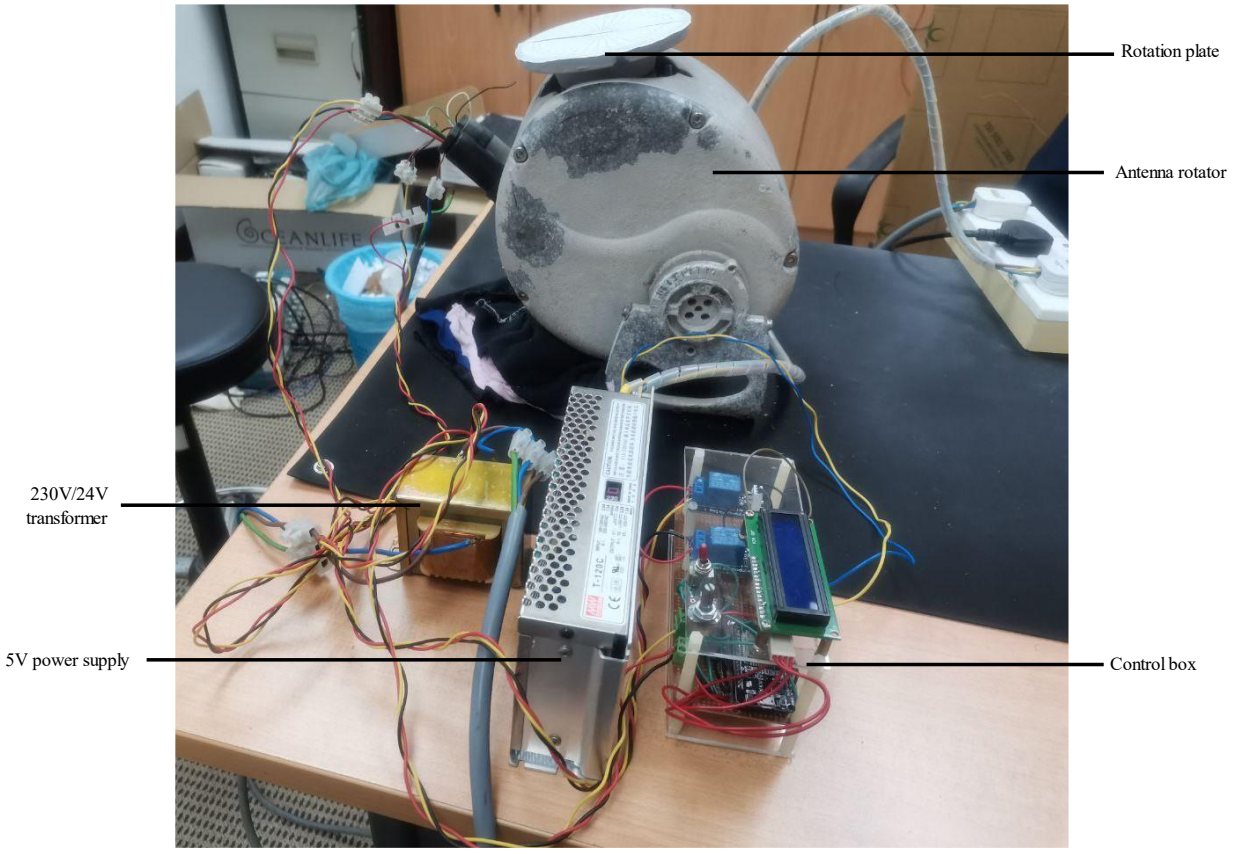


Figure 4.21 Hardware connection of IoT Integrated Antenna Rotator

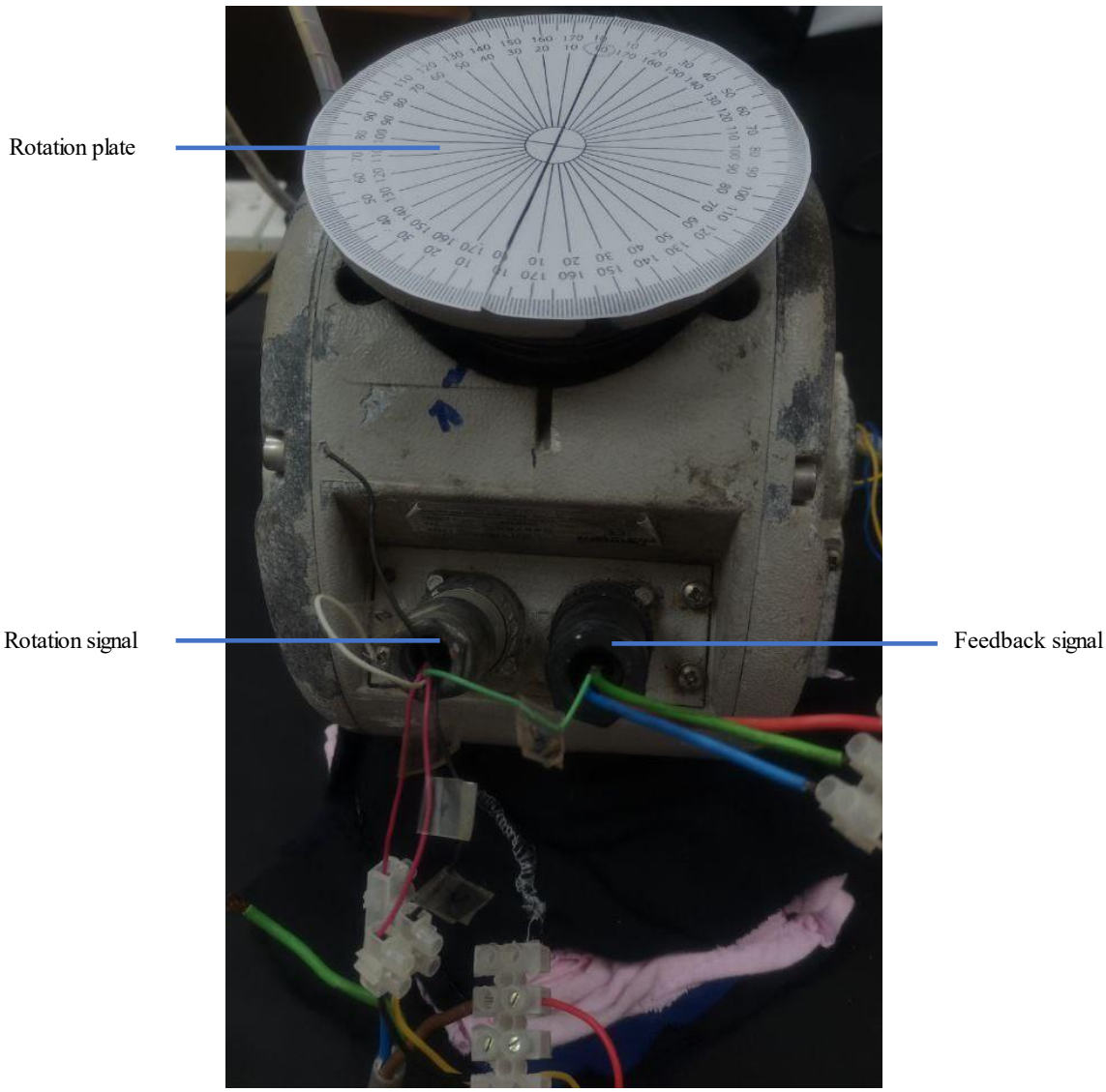


Figure 4.22 Antenna rotator

The software is integrated with all the hardware material as in Figure 4.23. The programs are uploaded into ESP32 via micro-USB cable. The IoT Integrated Antenna Rotator is a standalone system and it is able to operate without connect to the laptop. In Figure 4.23, after the program has been uploaded, the laptop is simply provide the 5V power supply to ESP32 to allow ESP32 function continuously. Instead of using laptop, there are a lot of devices or equipment could be used to provide the 5V power supply such as power bank or phone charger.

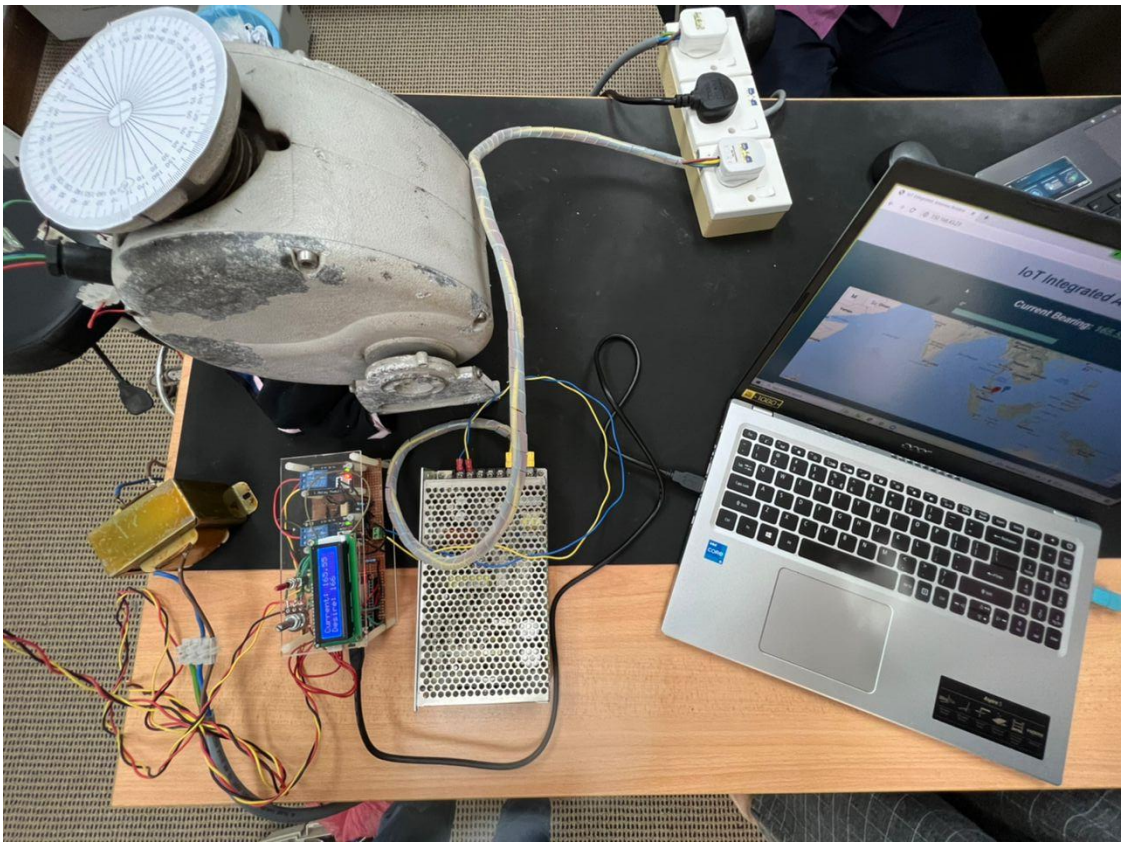


Figure 4.23 System integration

When the system is integrated, the user is able to control the antenna rotator thru web server (GUI). Figure 4.24 shows the real-time current rotation angle update of the antenna rotator in web server. Figure 4.25 shows the user pin the location in Google Maps and send the information to ESP32 to control the antenna rotator. The rotation plate starts to move upon received the instruction from the controller. At the same time, LCD displays the real-time current angle of the antenna rotator and the desire rotation angle selected by user. User can control the antenna rotator with a specific rotation angle by adjust the slider in the web server as in Figure 4.26. Besides, the antenna rotator is able to be manually regulated by switching to manual rotation mode and adjusting the potentiometer as in Figure 4.27. All the process executed by web server can be done using mobile phone as well in Figure 4.28.

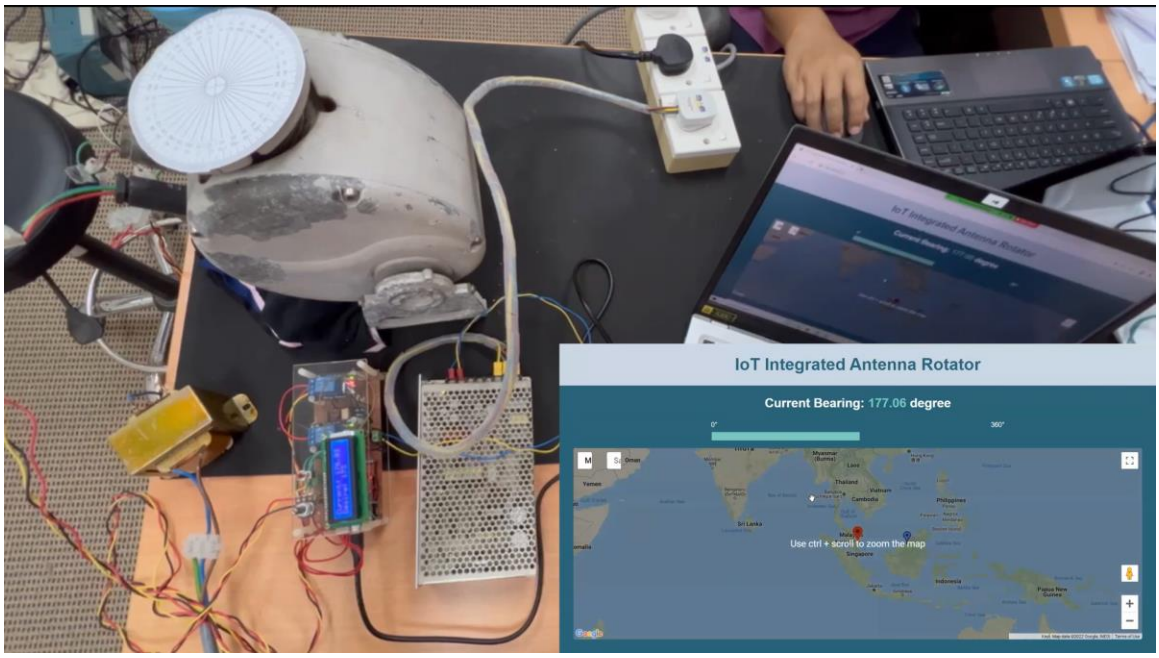


Figure 4.24 System integration real-time update current angle

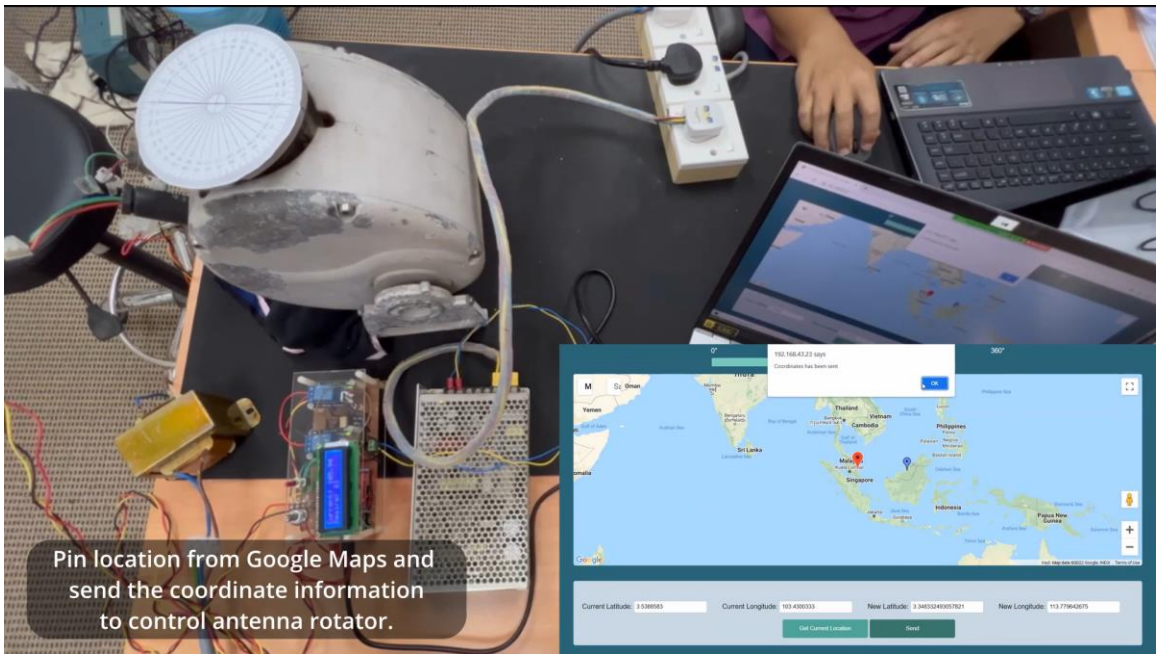


Figure 4.25 Control system using Google Maps

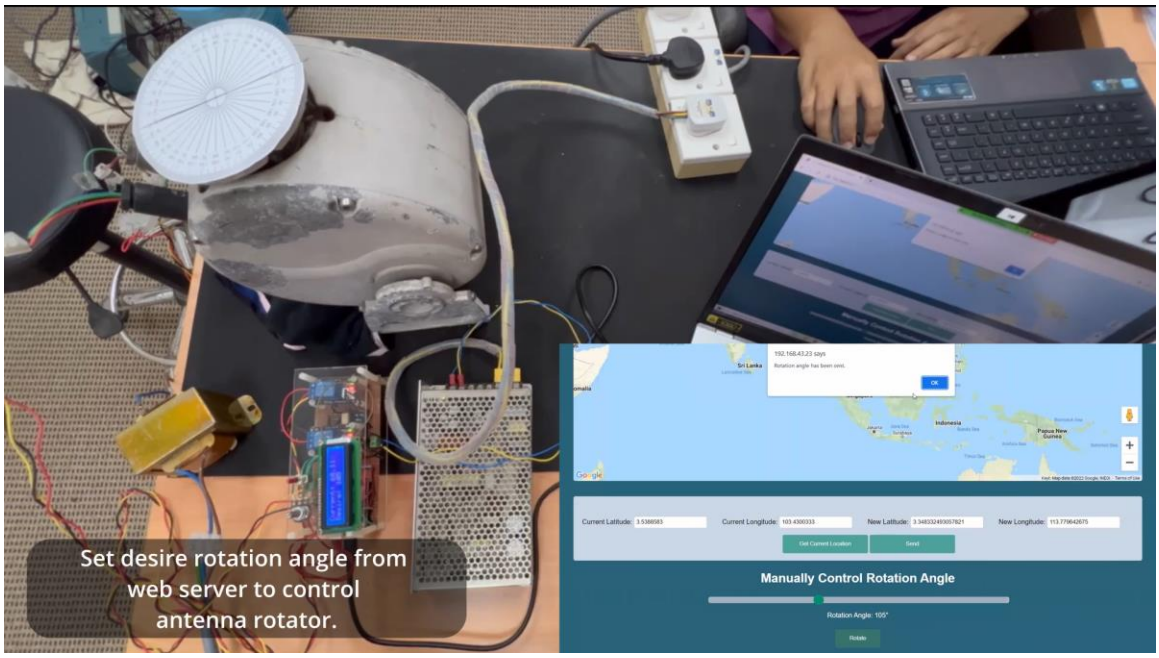


Figure 4.26 Control system by setting angle from GUI

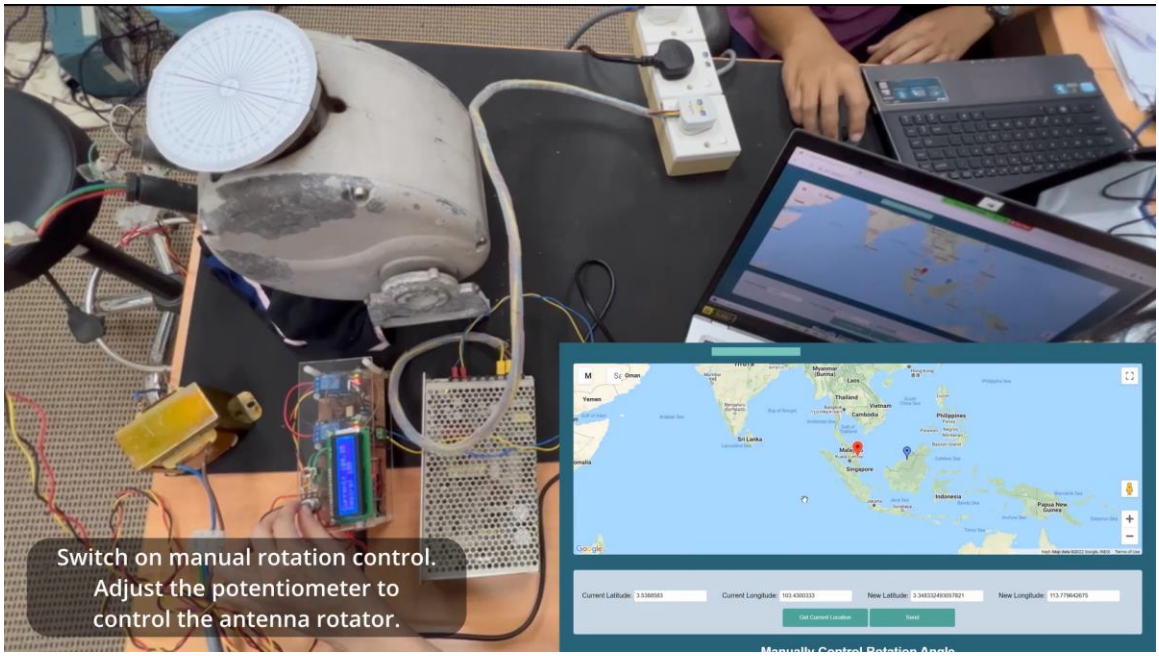


Figure 4.27 Manual rotation control

Figure 4.28 System testing using mobile phone

4.3 Prototype

After the system testing, all the hardware is assembled and a complete prototype is fabricated. Figure 4.29 and Figure 4.30 shows the top view and side view of the IoT Integrated Antenna Rotator control box respectively. The prototype is then being tested as shown in Figure 4.31 to ensure the system functionality. The testing results have achieved all the objectives and the characteristics and benefits of the IoT Integrated Antenna rotator are listed as following:

- Display real-time current angle of antenna rotator in GUI and LCD.
- Pinpoint locations in Google Maps via GUI.
- Regulate the orientation of antenna rotator through internet or by adjusting the potentiometer.
- Accurate geographic coordinates.
- Enhance the transmission and reception of signal.

Figure 4.29 Top view of the IoT Integrated Antenna Rotator control box

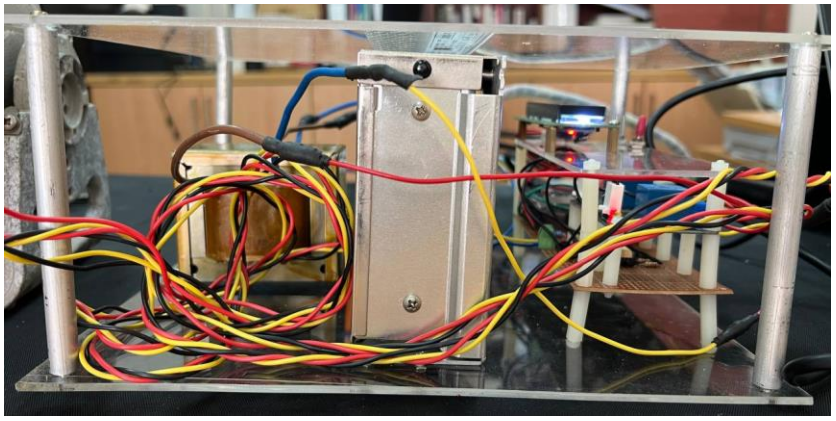


Figure 4.30 Side view of the IoT Integrated Antenna Rotator control box

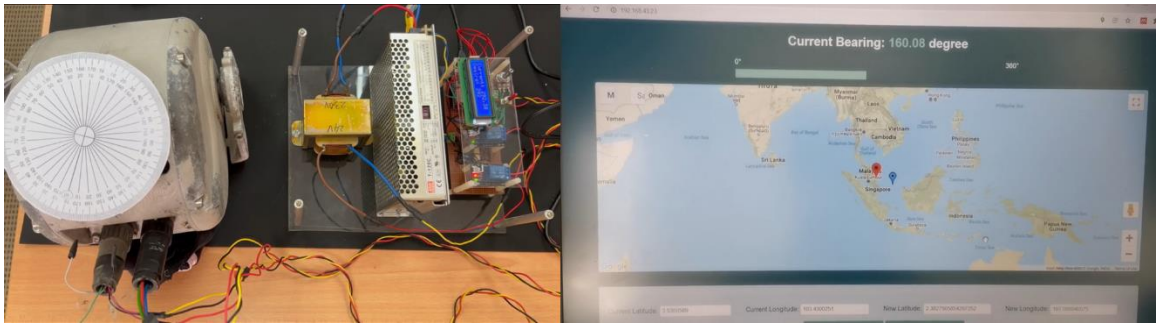


Figure 4.31 Prototype testing

Figure 4.32 shows the fully functioning prototype of IoT integrated Antenna Rotator with the web server control using laptop and mobile phone.

Figure 4.32 IoT Integrated Antenna Rotator prototype

4.4 Ethical Consideration

Ethical issues are explicitly considered throughout the software development life cycle in ethics-aware software engineering (Rashid et al., 2009). To strengthen security and preserve the user's privacy, the web browser will request permission from the user before obtaining the user's current location. A pop-up window appears as in Figure 4.33 when user first entering the web server or click the “Get Current Location” button.

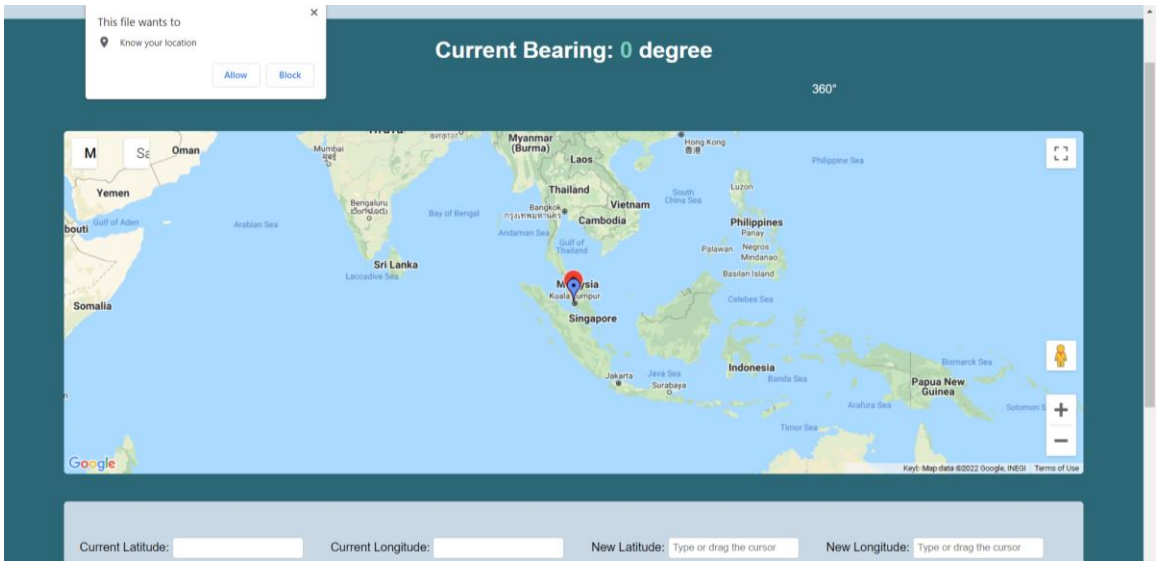


Figure 4.33 Request for location information

When user agree to let the website to access location services, the coordinate information of current location of the user will be updated in the text box and Google Maps as in Figure 4.34. The Internet Service Provider assigns the device an IP address which is required to access the internet. IP addresses establish a link between the device and the websites and services that the user accesses. IP addresses are assigned on an approximately geographical basis. This implies that every website that a user visits may obtain information about the user's general location.

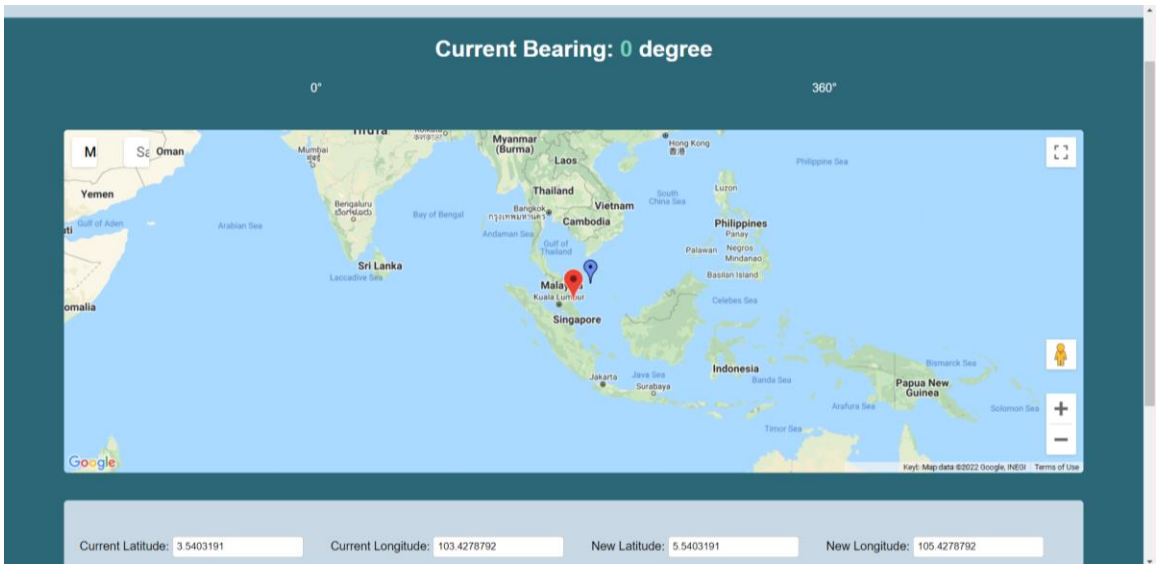


Figure 4.34 Location information permission is allowed

If user deny the permission, the current location of the user will not be obtained by the website as in Figure 4.35. However, user still can set the current location by pinpoint the location manually in Google Maps or simply enter the latitude and longitude in the text box.

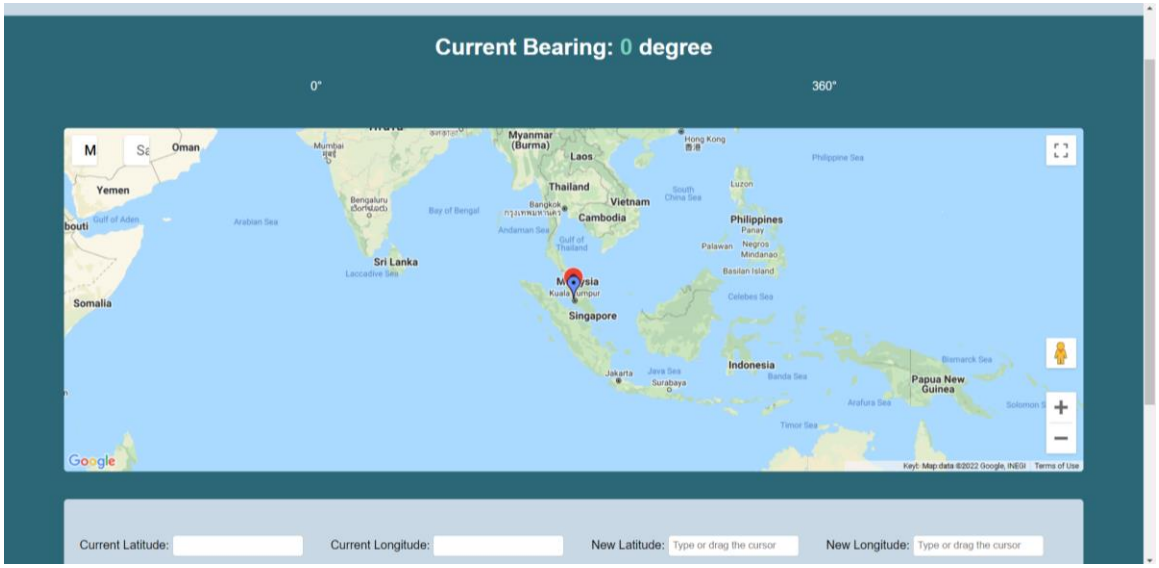


Figure 4.35 Location information permission is denied

CHAPTER 5

CONCLUSION AND RECOMMENDATION

This chapter concludes the essential features of the design and the meaningful outcomes of the project. The limitation and the recommendation for the future work of this project also included in this chapter.

5.1 Conclusion

An IoT Integrated Antenna Rotator is successfully fabricated in this project to upgrade an existing antenna rotator by integrating it with Internet of Things (IoT) technology. The objective of this thesis is to integrate Google Maps into GUI (web server) with the antenna rotator system in order to achieve a speedy and high accuracy standalone antenna pointing system. The real-time current rotation angle of the antenna rotator is able to be displayed at the web server and LCD. User is able to control the antenna rotator by 3 types of modes which are sending coordinate information from Google Maps, transmitting certain desire rotation angle remotely from web server and manually regulate the antenna rotator by adjusting the potentiometer. The objectives are achieved and the results are validated. IoT Integrated Antenna Rotator can give significantly impact to the communication system with the convenience and precision in controlling antenna directionality in this system.

5.2 Recommendation for Future Work

Although the IoT Integrated Antenna Rotator is successfully fabricated, there are still few limitations have been found. Firstly, the Dennard Type 2000 rotator only can rotate in a range from 0° to 290°. A limit switch inside the Dennard Type 2000 rotator will limit the orientation of the rotator to prevent it from damaging the device. To make the rotation range from 0° to 360°, a continuous rotation 360° servo motor could be considerate as the rotator.

Besides, this project only controls the azimuth angle of the antenna rotator. This project could be enhanced by adding feature of controls the elevation angle of the antenna rotator. By adding this feature, the antenna rotator will have the ability to easily point at the satellite.

Lastly, the feedback value of the antenna rotator is not stable and affect the accuracy of the rotator angle. To resolve it, an external analog signal filter circuit could be applied at the feedback mechanism to eliminate the ripple of the signal. Also, an amplifier could be used and regulate the voltage range from 0V to 3.3V to enable it compatible with the ESP32 ADC analog input voltage and avoid floating value.

REFERENCES

Alfaridzi, M. D., & Yulianti, L. P. (2020). UI-UX design and analysis of local medicine and medication mobile-based apps using task-centered design process. *2020 International Conference on Information Technology Systems and Innovation, ICITSI 2020 - Proceedings*, 443–450. https://doi.org/10.1109/ICITSI50517.2020.9264947

Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, *54*(15), 2787–2805. https://doi.org/10.1016/j.comnet.2010.05.010

Bammidi, R., & Kundala, D. (n.d.). *Remote Monitoring and Control by Embedded Database Design and Web Server Implementation*.

Benslimane, D., Dustdar, S., & Sheth, A. (2008). Services mashups: The new generation of web applications. In *IEEE Internet Computing* (Vol. 12, Issue 5, pp. 13–15). https://doi.org/10.1109/MIC.2008.110

Bhasin, S. (2012). Quality assurance in agile: A study towards achieving excellence. *Proceedings - Agile India 2012, AgileIndia 2012*, 64–67. https://doi.org/10.1109/AgileIndia.2012.18

Boehm, B. (2006). A view of 20th and 21st century software engineering. In *Proceedings - International Conference on Software Engineering* (Vol. 2006). https://doi.org/10.1145/1134285.1134288

Boehm, B., & Turner, R. (n.d.). *Balancing Agility and Discipline: A Guide for the Perplexed*.

Bourque, P., Robert, F., Lavoie, J. M., Lee, A., Trudel, S., & Lethbridge, T. C. (2002). Guide to the Software Engineering Body of Knowledge (SWEBOK) and the Software Engineering Education Knowledge (SEEK) - A preliminary mapping. *Proceedings - 10th International Workshop on Software Technology and Engineering Practice, STEP 2002*, 8–23. https://doi.org/10.1109/STEP.2002.1267595

Can Filibeli, M., Ozkasap, O., & Reha Civanlar, M. (2007). Embedded web server-based home appliance networks. *Journal of Network and Computer Applications*, *30*(2), 499–514. https://doi.org/10.1016/j.jnca.2006.04.001

Caspi, P., Sangiovanni-Vincentelli, A., Almeida, L., Benveniste, A., Bouyssounouse, B., Buttazzo, G., Crnkovic, I., Damm, W., Engblom, J., Folher, G., Garcia-Valls, M., Kopetz, H., Lakhnech, Y., Laroussinie, F., Lavagno, L., Lipari, G., Maraninchi, F., Peti, Ph., Puente, J. de la, … Yi, W. (2005). Guidelines for a Graduate Curriculum on Embedded Software and Systems. *ACM Trans. Embed. Comput. Syst.*, *4*(3), 587–611. https://doi.org/10.1145/1086519.1086526

Chen, S., Xu, H., Liu, D., Hu, B., & Wang, H. (2014). A vision of IoT: Applications, challenges, and opportunities with China Perspective. In *IEEE Internet of Things Journal* (Vol. 1, Issue 4, pp. 349–359). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/JIOT.2014.2337336

Cohen, S., de Haan, U., & Dori, D. (2010). A Software System Development Life Cycle Model for Improved Stakeholders' Communication and Collaboration. *Int. J. of Computers, V*, 20–41. https://doi.org/10.15837/ijccc.2010.1.2462

Dennard Type 2000. (n.d.).

Diaz-Cacho, M., Delgado, E., Falcon, P., & Barreiro, A. (2015). IoT integration on industrial environments. *IEEE International Workshop on Factory Communication Systems - Proceedings, WFCS*, *2015-July*. https://doi.org/10.1109/WFCS.2015.7160553

Dingsøyr, T., & Lassenius, C. (2016). Emerging themes in agile software development: Introduction to the special section on continuous value delivery. *Information and Software Technology*, *77*, 56–60. https://doi.org/10.1016/j.infsof.2016.04.018

Duplouy, J., Morlaas, C., Aubert, H., Potier, P., & Pouliguen, P. (2019). Wideband Vector Antenna for Dual-Polarized and Three-Dimensional Direction-Finding Applications. *IEEE Antennas and Wireless Propagation Letters*, *18*(8), 1572–1575. https://doi.org/10.1109/LAWP.2019.2923531

Ebert, C. (2008). A brief history of software technology. *IEEE Software*, *25*(6), 22–25. https://doi.org/10.1109/MS.2008.141

Ebert, C., & Dumke, R. (2007). Software measurement: Establish - Extract - Evaluate - Execute. In *Software Measurement: Establish - Extract - Evaluate - Execute*. https://doi.org/10.1007/978-3-540-71649-5

ESP-WROOM-32 Datasheet Espressif Systems. (2017).

Fette, I., & Melnikov, A. (2011). *The websocket protocol*. RFC 6455, December.

Fitzgerald, B. (2000). Systems development methodologies: the problem of tenses. In *ITP* (Vol. 13, Issue 3). # MCB University Press. http://www.emerald-library.com

Graham, D. R. (1992). *INCREMENTAL DEVELOPMENT AND DELIVERY FOR LARGE SOFTWARE SYSTEMS*.

Grudin, J. (1989). The Case against User Interface Consistency. *Commun. ACM*, *32*(10), 1164–1173. https://doi.org/10.1145/67933.67934

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, *29*(7), 1645–1660. https://doi.org/10.1016/j.future.2013.01.010

Hamel, G. (2001). Leading the revolution: *Strategy & Leadership*, *29*(1), 4–10. https://doi.org/10.1108/10878570110367141

Hermann, M., Pentek, T., & Otto, B. (2016). Design principles for industrie 4.0 scenarios. *Proceedings of the Annual Hawaii International Conference on System Sciences*, *2016-March*, 3928–3937. https://doi.org/10.1109/HICSS.2016.488

Hirschheim, R., Klein, H. K., & Lyytinen, K. (1995). *Information Systems Development and Data Modeling: Conceptual and Philosophical Foundations*. Cambridge University Press. https://doi.org/DOI: 10.1017/CBO9780511895425

Hsu, H. J., & Lin, Y. (2018). How Agile Impacts a Software Corporation: An Empirical Study. *Proceedings - International Computer Software and Applications Conference*, *2*, 20–25. https://doi.org/10.1109/COMPSAC.2018.10197

Illera, M. J., & Sepulveda, S. B. (2015). Embedded system based on microcontroller for generating I-V curves of electronic devices. *2014 IEEE 33rd International Performance Computing and Communications Conference, IPCCC 2014*, *2014-January*. https://doi.org/10.1109/PCCC.2014.7017033

Jain, P., Ahuja, L., & Sharma, A. (2016). *Current State of Research in Agile Quality Development*.

Jain, P., Sharma, A., & Ahuja, L. (2018). The Impact of Agile Software Development Process on the Quality of Software Product. *2018 7th International Conference on Reliability, Infocom Technologies and Optimization: Trends and Future Directions, ICRITO 2018*, 812–815. https://doi.org/10.1109/ICRITO.2018.8748529

Johnson, J. (2010). *Designing with the Mind in Mind: Simple Guide to Understanding UI Design Rules*.

Joo, H. (2017). A Study on Understanding of UI and UX, and Understanding of Design According to User Interface Change. In *International Journal of Applied Engineering Research* (Vol. 12). http://www.ripublication.com

JOO, H. S. (2017). A Study on UI/UX and Understanding of Computer Major Students. *International Journal of Advanced Smart Convergence*, *6*(4), 26–32. https://doi.org/10.7236/IJASC.2017.6.4.4

Junfithrana, A. P., Rahardjo, E. T., Zulkifli, F. Y., & Basari. (2017). Development of automated antenna radiation pattern measurement using rotator application model to increase accuracy. *2017 International Conference on Computing, Engineering, and Design (ICCED)*, 1–5. https://doi.org/10.1109/CED.2017.8308101

Kivinen, J., Zhao, X., & Vainikainen, P. (1999). Wideband indoor radio channel measurements with direction of arrival estimations in the 5 GHz band. *IEEE Vehicular Technology Conference*, *4*, 2308–2312. https://doi.org/10.1109/vetecf.1999.797350

Kumar, G., & Bhatia, P. K. (2014). Comparative analysis of software engineering models from traditional to modern methodologies. *International Conference on Advanced Computing and Communication Technologies, ACCT*, 189–196. https://doi.org/10.1109/ACCT.2014.73

Kurniawan, S. (2004). Interaction design: Beyond human?computer interaction by Preece, Sharp and Rogers (2001), ISBN 0471492787. *Universal Access in the Information Society*, *3*(3–4), 289–289. https://doi.org/10.1007/s10209-004-0102-1

Lacerda, L. L., & Furtado, F. (2018). Factors that help in the implantation of agile methods: A systematic mapping of the liteature. *Iberian Conference on Information Systems and Technologies, CISTI*, *2018-June*, 1–6. https://doi.org/10.23919/CISTI.2018.8399406

Lee, H. E. (2014). A Study on the concept and types of UX design in the smart product field. *Journal of Korea Design Knowledge*, *null*(30), 289–299. https://doi.org/10.17246/jkdk.2014..30.027

Lee, I., & Lee, K. (2015). The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons*, *58*(4), 431–440. https://doi.org/10.1016/j.bushor.2015.03.008

Li, S. (2011). A method for building thematic map of GIS based on Google Maps API. *Proceedings - 2011 19th International Conference on Geoinformatics, Geoinformatics 2011*. https://doi.org/10.1109/GeoInformatics.2011.5980798

Limpraptono, F. Y., Sudibyo, H., Ratna, A. A. P., & Arifin, A. S. (2011). The design of embedded web server for remote laboratories microcontroller system experiment. *TENCON 2011 - 2011 IEEE Region 10 Conference*, 1198–1202. https://doi.org/10.1109/TENCON.2011.6129302

Liu, Q., Yang, G., Zhao, R., & Xia, Y. (2018). Design and Implementation of Real-time Monitoring System for Wireless Coverage Data Based on WebSocket; Design and Implementation of Real-time Monitoring System for Wireless Coverage Data Based on WebSocket. In *2018 IEEE 3rd International Conference on Cloud Computing and Internet of Things (CCIOT)*.

Lubbers, P. (2011). Html5 web sockets: A quantum leap in scalability for the web. *Http://Www. Websocket. Org/Quantum. Html*.

Lyytinen, K. J. (1987). *A taxonomic perspective of information systems development: theoretical constructs and recommendations*.

MacHeso, P., Chisale, S., Daka, C., Dzupire, N., Mlatho, J., & Mukanyirigira, D. (2021). Design of Standalone Asynchronous ESP32 Web-Server for Temperature and Humidity Monitoring. *2021 7th International Conference on Advanced Computing and Communication Systems, ICACCS 2021*, 635–638. https://doi.org/10.1109/ICACCS51430.2021.9441845

Martinez, W. (2011). Graphical user interfaces. *Wiley Interdisciplinary Reviews: Computational Statistics*, *3*. https://doi.org/10.1002/wics.150

Mei, W., & Long, Z. (2020). *Research and Defense of Cross-Site WebSocket Hijacking Vulnerability*. Proceedings of 2020 IEEE International Conference on Artificial Intelligence and Computer Applications : ICAICA 2020 :

Muhammad Rusydi Bin Buchek, by, & Darul Ridzuan, P. (2014). *ANTENNA ROTATOR DESIGN AND CONTROL*.

Nagy, J., Oláh, J., Erdei, E., Máté, D., & Popp, J. (2018). The role and impact of industry 4.0 and the internet of things on the business strategy of the value chain-the case of hungary. *Sustainability (Switzerland)*, *10*(10). https://doi.org/10.3390/su10103491

Nakajima, H., Isshiki, M., & Takefuji, Y. (2013). WebSocket proxy system for mobile devices. *2013 IEEE 2nd Global Conference on Consumer Electronics, GCCE 2013*, 315–317. https://doi.org/10.1109/GCCE.2013.6664841

NIELSEN, J. (1989). 1 - Executive Summary: Coordinating User Interfaces for Consistency. In J. Nielsen (Ed.), *Coordinating User Interfaces for Consistency* (pp. 1–7). Morgan Kaufmann. https://doi.org/https://doi.org/10.1016/B978-0-08-050315-8.50007-9

Pedersen, M. R., Nalpantidis, L., Andersen, R. S., Schou, C., Bøgh, S., Krüger, V., & Madsen, O. (2016). Robot skills for manufacturing: From concept to industrial deployment. *Robotics and Computer-Integrated Manufacturing*, *37*, 282–291. https://doi.org/https://doi.org/10.1016/j.rcim.2015.04.002

Qin, J. (2017). Research and performance analysis of instant messaging based on WebSocket [J]. *Mobile Communication*, *41*(12), 44–48.

Rashid, A., Weckert, J., & Lucas, R. (2009). Software Engineering Ethics in a Digital World. *Computer*, *42*(6), 34–41. https://doi.org/10.1109/MC.2009.200

Reddy, Martin. (2011). *API design for C++*. Morgan Kaufmann.

Redwine, S., & Riddle, W. (1985). Software Technology Maturation. In *Proceedings - International Conference on Software Engineering*.

Rovce, W. W. (1970). *MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS*.

Saloni, S., & Hegde, A. (2016). WiFi-aware as a connectivity solution for IoT: Pairing IoT with WiFi aware technology: Enabling new proximity based services. *2016 International Conference on Internet of Things and Applications, IOTA 2016*, 137–142. https://doi.org/10.1109/IOTA.2016.7562710

Shaw, M. (1990). Prospects for an Engineering Discipline of Software. *IEEE Software*, *7*(6), 15–24. https://doi.org/10.1109/52.60586

Shneiderman, Ben., & Plaisant, Catherine. (2010). *Designing the user interface : strategies for effective human-computer interaction*. Addison-Wesley.

Singh, B., & Gautam, S. (2016). Hybrid Spiral Model to Improve Software Quality Using Knowledge Management. In *International Journal of Performability Engineering* (Vol. 12, Issue 4).

Singh, B., & Prasad Kannojia, S. (2012). A Model for Software Product Quality Prediction. *Journal of Software Engineering and Applications*, *05*(06), 395–401. https://doi.org/10.4236/jsea.2012.56046

Sinha, A., & Das, P. (2021). *Agile Methodology Vs. Traditional Waterfall SDLC: A case study on Quality Assurance process in Software Industry*. 1–4. https://doi.org/10.1109/iementech53263.2021.9614779

Sommerville, I. (1996). *Software Process Models*.

Subbian, V., & Beyette, F. R. (2013). Developing a new advanced microcontrollers course as a part of embedded systems curriculum. *Proceedings - Frontiers in Education Conference, FIE*, 1462–1464. https://doi.org/10.1109/FIE.2013.6685076

Tian-huang, C., & Jia-xi, H. (2008). *Design and Realization of CGI in Embedded Dynamic Web Technology*. 774–777. https://doi.org/10.1109/npc.2007.39

Trivedi, P., & Sharma, A. (2013). *A Comparative Study between Iterative Waterfall and Incremental Software Development Life Cycle Model for Optimizing the Resources Using Computer Simulation*.

Weyer, S., Schmitt, M., Ohmer, M., & Gorecky, D. (2015). Towards Industry 4.0 - Standardization as the crucial challenge for highly modular, multi-vendor production systems. *IFAC-PapersOnLine*, *48*, 579–584.

Wittern, E. (2018). Web APIs-Challenges, design points, and research opportunities. *Proceedings - International Conference on Software Engineering*, 19–22. https://doi.org/10.1145/3194793.3194801

Wynekoop, J. L., & Russo, N. L. (1997). Studying system development methodologies: an examination of research methods. *Information Systems Journal*, *7*.

Yang, L., & Yang, G. (2016). Real-time wireless signal testing and analyzing system based on websocket. *Proceedings - 2016 6th International Conference on Instrumentation and Measurement, Computer, Communication and Control, IMCCC 2016*, 648–652. https://doi.org/10.1109/IMCCC.2016.179

Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of things for smart cities. *IEEE Internet of Things Journal*, *1*(1), 22–32. https://doi.org/10.1109/JIOT.2014.2306328

Zha, X., Wu, R., & Gao, Y. (2014). Technology research on conversion from C/S to B/S. *Comput Eng*, *40*, 263–267.

Zhang, L., & Shen, X. (2013). Research and development of real-time monitoring system based on WebSocket technology. *Proceedings - 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer, MEC 2013*, 1955–1958. https://doi.org/10.1109/MEC.2013.6885373

Zhong, H., & Mei, H. (2019). An Empirical Study on API Usages. *IEEE Transactions on Software Engineering*, *45*(4), 319–334. https://doi.org/10.1109/TSE.2017.2782280

Zhu, Y. (2012). Introducing Google Chart Tools and Google Maps API in Data Visualization Courses. *IEEE Computer Graphics and Applications*, *32*(6), 6–9. https://doi.org/10.1109/MCG.2012.114

APPENDICES

A. Gantt Chart for SDP 2

WEEK		1	2	3	4	5	6		7	8	9	10		10	11	12	13	14
DATE		(11.10.21 - 15.10.21)	(18.10.21 - 22.10.21)	(25.10.21 - 29.10.21)	(1.11.21 - 5.11.21)	(8.11.21 - 12.11.21)	(15.11.21 - 19.11.21)		(29.11.21 - 3.12.21)	(6.12.21 - 10.12.21)	(13.12.21 - 17.12.21)	(20.12.21 - 21.12.21)		(29.12.21 -30.12.21)	(3.01.22 - 7.01.22)	(10.01.22 -14.01.22)	(17.01.22 -21.01.22)	(24.01.22 -28.01.22)
1	Material preparation							Mid-term break (20.11.21 – 28.11.21)					Postponement Pdp due to flood disaster (22.12.21 – 28.12.21)					
2	Define requirements																	
3	Front-end system design																	
4	Back-end system design																	
5	Software implementation and unit testing																	
6	Hardware integration and system testing																	
7	Build up prototype																	
8	Recheck and revise project																	
9	Report and thesis writing																	
10	Project presentation																	
11	Report submission																	

B. Back End Scripting

```
/************************************/
/*---------------- Attribution Notice ----------------*/
/*  This program was written by:                     */
/*  Chong Jia Xin (TG18001)                          */
/*  Date : 9th January 2022                          */
/*  Email : tg18001@student.ump.edu.my               */
/*  Intended for SDP use.                            */
/************************************/
//==========================================
//ESP32 WebSocket Server
//==========================================
#include <WiFi.h>
#include <WebServer.h>
#include <WebSocketsServer.h>
#include <ArduinoJson.h>
#include <LiquidCrystal_I2C.h>

// set the LCD number of columns and rows
int lcdColumns = 16;
int lcdRows = 2;
LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);

// The JSON library uses static memory, so this will need to be allocated:
StaticJsonDocument<200> doc_tx;                 // provision memory for about 200
characters
StaticJsonDocument<200> doc_rx;

// We want to periodically send values to the clients, so we need to define an "interval"
and remember the last time we sent data to the client (with "previousMillis")
int interval = 1000;                            // send data to the client every 1000ms -> 1s
unsigned long previousMillis = 0;               // we use the "millis()" command for time
reference and this will output an unsigned long

//------------------------------------------
const char* ssid = "";
const char* password = "";
//------------------------------------------
WebServer server(80);
WebSocketsServer webSocket = WebSocketsServer(81);
//------------------------------------------
String JSONtxt;
```

```
const char* PARAM_INPUT_3 = "input3";//D34
const char* PARAM_INPUT_4 = "input4";//D35
const int RELAY1_PIN = 32;//cw
const int RELAY2_PIN = 33;//ccw
const int SWpin = 4; //SW manual/online

float c_lat;
float c_lng;
float n_lat;
float n_lng;
int manual_angle;
int angle;
int rotation_angle;
String POTvalString;
float POTvalue;
int manual_lot;
//int SW_POT;

// variable for storing the pushbutton status
int SWstate = 0;

//---------------------------------------------
#include "webpage.h"
//---------------------------------------------
void handleRoot()
{
  server.send(200,"text/html", webpageCont);
}
//====================================================================
void setup()
{
  Serial.begin(115200);
  /*Initial rotation angle*/
  rotation_angle = int(0.113961*(analogRead(A6))+10.7873);//initial state of the rotator
remain unchanged.
 pinMode (RELAY1_PIN, OUTPUT);
 pinMode (RELAY2_PIN, OUTPUT);
 pinMode(SWpin, INPUT);
  WiFi.begin(ssid, password);
  while(WiFi.status() != WL_CONNECTED)
  {
    Serial.print("."); delay(500);
  }
  WiFi.mode(WIFI_STA);
```

```
  Serial.print(" Local IP: ");
  Serial.println(WiFi.localIP());

  server.on("/", handleRoot);
  server.begin(); webSocket.begin();
  webSocket.onEvent(webSocketEvent);

  // initialize LCD
  lcd.init();
  // turn on LCD backlight
  lcd.backlight();
  lcd.clear();

}
//=======================================================================
void loop()
{
  webSocket.loop(); server.handleClient();

unsigned long now = millis();                     // read out the current "time" ("millis()" gives
the time in ms since the Arduino started)
  if ((unsigned long)(now - previousMillis) > interval) { // check if "interval" ms has passed
since last time the clients were updated

  POTvalue=(0.113961*(analogRead(A6))+10.7873);//D34
  POTvalString = String(POTvalue);
  //Serial.println(POTvalString);
  JSONtxt = "{\"POT\":\""+POTvalString+"\"}";
  webSocket.broadcastTXT(JSONtxt);
 previousMillis = now;                          // reset previousMillis
  }

  SWstate=digitalRead(SWpin);

  if (SWstate == HIGH) {
    Serial.println("******Manual rotation activated******");
    Serial.println("");
    // turn on manual rotation
    POTvalue=(0.113961*(analogRead(A6))+10.7873);//D34
    rotation_angle=(((analogRead(A7))*290)/4096);//D35
    rotation();

  //lcd.clear();
    // set cursor to first column, first row
```

```
 lcd.setCursor(0, 0);
 lcd.print("Current: ");
 lcd.print(POTvalue);
 lcd.setCursor(0, 1);
 lcd.print("Desire: ");
 lcd.print(rotation_angle);
 delay(100);
 }


   //-------------Rotation----------------------//

    rotation();


}

void webSocketEvent(uint8_t num, WStype_t type, uint8_t *payload, size_t length){
  switch (type) {                              // switch on the type of information sent
  case WStype_DISCONNECTED:                    // if a client is disconnected, then type ==
WStype_DISCONNECTED
    Serial.println("Client " + String(num) + " disconnected");
    break;
  case WStype_CONNECTED:                       // if a client is connected, then type ==
WStype_CONNECTED
    Serial.println("Client " + String(num) + " connected");
    // optionally you can add code here what to do when connected
    break;
  case WStype_TEXT:                            // if a client has sent data, then type ==
WStype_TEXT
    // try to decipher the JSON string received
    DeserializationError error = deserializeJson(doc_rx, payload);//catch if there's error
or not
    if (error) {
      Serial.print(F("deserializeJson() failed: "));
      Serial.println(error.f_str());
      return;
    }

    else {
      // JSON string was received correctly, so information can be retrieved:
        c_lat = doc_rx["my_lat"];
        c_lng = doc_rx["my_lng"];
        n_lat = doc_rx["new_lat"];
```

```
        n_lng = doc_rx["new_lng"];
        manual_lot = doc_rx["manual_angle"];

       Serial.println("Received coordinate from user: ");
       Serial.println("Current Latitude: " + String(c_lat));
       Serial.println("Current Longitude: " + String(c_lng));
       Serial.println("New Latitude: " + String(n_lat));
       Serial.println("New Longitude: " + String(n_lng));
       Serial.println("Manual rotation angle: " + String(manual_lot));
       Serial.println("");
       //Serial.println(payload[0]);

       if (manual_lot==0){
         rotation_angle=(int)getBearing();
       }
       else{
         rotation_angle=manual_lot;
       }
       lcd.clear();

     }


     break;
  }

}
float getBearing(){

   float theta1 = c_lat;
   float theta2 = n_lat;
   float delta1 = n_lat-c_lat;
   float dL = n_lng-c_lng;

   //--------------Calculation--------------//

    float x = cos(theta2) * sin(dL) ;
   float y = cos(theta1)*sin(theta2) - sin(theta1)*cos(theta2)*cos(dL);

   float bearing = atan2(x,y);
   bearing = degrees(bearing);// radians to degrees

   angle = (((int)bearing + 360) % 360); //change the range from -180~180 to 0~360
```

```
  /* //Alternate way to change the range from -180~180 to 0~360
  if(bearing<0){
    angle=360+((int)(bearing));
  }
  else{
    angle= (int)(bearing);
  }
  */

  Serial.print("Bearing: ");
  Serial.println(angle);

  return angle;



}

void rotation(){
int current_angle=(int)(POTvalue);
Serial.println("current: "+String(current_angle)+" vs rotation: "+String(rotation_angle));
int Xval = rotation_angle - current_angle;

if(rotation_angle==current_angle || (Xval>-10 && Xval<10) ){//if manual=feedback
 digitalWrite(RELAY1_PIN, LOW);
 digitalWrite (RELAY2_PIN, HIGH);

 Serial.println("***Rotation Complete***");
  Serial.println(" ");
   lcd.setCursor(0, 0);
   lcd.print("Current: ");
   lcd.print(POTvalue);
   lcd.setCursor(0, 1);
   lcd.print("Desire: ");
   lcd.print(rotation_angle);

    delay (500);

}

else if(rotation_angle>current_angle){ //new location > current location, turn RIGHT
  digitalWrite (RELAY1_PIN, LOW);
  digitalWrite (RELAY2_PIN, LOW);
```

```
 Serial.println("Rotating... ...(left)");
 Serial.println(" ");
  lcd.setCursor(0, 0);
  lcd.print("Current: ");
  lcd.print(POTvalue);
  lcd.setCursor(0, 1);
  lcd.print("Desire: ");
  lcd.print(rotation_angle);

   delay (500);

 }
else if (rotation_angle<current_angle){//new location < current location, turn LEFT
 digitalWrite (RELAY1_PIN, HIGH); //D32
 digitalWrite (RELAY2_PIN, HIGH); //D33

 Serial.println("Rotating... ...(right)");
 Serial.println(" ");
  lcd.setCursor(0, 0);
  lcd.print("Current: ");
  lcd.print(POTvalue);
  lcd.setCursor(0, 1);
  lcd.print("Desire: ");
  lcd.print(rotation_angle);

   delay (500);

 }
}
```

C. Front End Scripting

```
/***************************************/
/*------------------ Attribution Notice -----------------*/
/*  This program was written by:                        */
/*  Chong Jia Xin (TG18001)                             */
/*  Date : 9th January 2022                             */
/*  Email : tg18001@student.ump.edu.my                  */
/*  Intended for SDP use.                               */
/***************************************/
//=====================
//HTML code for webpage
//=====================
const char webpageCont[] PROGMEM =
R"=====(
<!DOCTYPE HTML>
<html>
<head>
<title>IoT Integrated Antenna Rotator</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=0.1">
<!--------------------------Javascript (G Map)------------------------->
<script src="https://polyfill.io/v3/polyfill.min.js?features=default"></script>


<script>

   /* Global variables */
   let map = null;
   let marker1 = null;
   let marker2 = null;


   const curLocation = {
     lat: 3.221101104201,
     lng: 101.63787669557,

     get getGeo() {
       return {lat: this.lat, lng: this.lng};
     },
     set setGeo(cood) {
       this.lat = cood.lat;
       this.lng = cood.lng;
       // Side effect
       document.getElementById("myLat").value = this.lat;
       document.getElementById("myLng").value = this.lng;
       marker1.setPosition(this.getGeo);
```

```
  }
};

const desLocation = {
  lat: 3.221101104201,
  lng: 101.63787669557,

  get getGeo() {
    return {lat: this.lat, lng: this.lng};
  },
  set setGeo(cood) {
    this.lat = cood.lat;
    this.lng = cood.lng;
    document.getElementById("lat").value = this.lat;
    document.getElementById("lng").value = this.lng;
    marker2.setPosition(this.getGeo);
  }
};


function initMap() {

  // Create map
  map = new google.maps.Map(document.getElementById("map"), {
    zoom: 4,
    center: curLocation.getGeo,
    pixelRatio: window.devicePixelRatio || 1
  });


  // Create markers
  marker1 = new google.maps.Marker({
    position: curLocation.getGeo,
    map,
    draggable:true,
    title: "Current location",
  });

  marker2 = new google.maps.Marker({
    position: curLocation.getGeo,
    map,
    draggable:true,
    title: "Destination",
    icon: {url: "http://maps.google.com/mapfiles/ms/icons/blue-dot.png"},
  });

  currentLot()
```

```
  // Subscribe to marker drag-end events
 google.maps.event.addListener(marker1, 'dragend', function() {


  var coordinate = marker1.getPosition();

  curLocation.setGeo = {lat:coordinate.lat(), lng: coordinate.lng()}



 });

 google.maps.event.addListener(marker2, 'dragend', function() {


  var coordinate = marker2.getPosition();

  desLocation.setGeo = {lat:coordinate.lat(), lng: coordinate.lng()}


 });


}

function handleLocationError(browserHasGeolocation, infoWindow, pos) {
 infoWindow.setPosition(pos);
 infoWindow.setContent(
  browserHasGeolocation
   ? "Error: The Geolocation service failed."
   : "Error: Your browser doesn't support geolocation."
 );
 infoWindow.open(map);
}


function currentLot() {
    // Get current location using GMAP
 if (navigator.geolocation) {
  navigator.geolocation.getCurrentPosition(
   (position) => {


      const markerDist = 2.0;
      curLocation.setGeo = {lat:position.coords.latitude, lng: position.coords.longitude}
```

```
            desLocation.setGeo = {lat:position.coords.latitude+markerDist, lng:
position.coords.longitude+markerDist}
            map.panTo(curLocation.getGeo);



        },
         () => {
           handleLocationError(true);
         });
      } else {
         // Browser doesn't support Geolocation
         handleLocationError(false);
      }
    }

  </script>
<!--------------------------CSS-------------------------->
<style>
    #dynRectangle
    {
       width:0px;
       height:20px;
       top: 9px;
       background-color: #7ccfbd;
       z-index: -1;
       margin-top:8px;
    }
    body   {background-color:#2b6777; font-family: 'Lato', sans-serif;}
    h1     {font-size: 40px; color: #2b6777; text-align: center;}
    h2     {font-size: 30px; color: rgb(255, 255, 255)}
    h3     {font-size: 17px; color:rgb(255, 255, 255)}
    .header {
    background-color: #c8d8e4;;
    padding: 0.5px;
    text-align: center;
    }

/* Google Map */
    #map {
  height: 50%;
  border-radius: 5px;
  margin-top: 0px;
  margin-left: 80px;
  margin-right: 80px;
  margin-bottom: 0px;
  padding: 40px;
}
```

```
/* Optional: Makes the sample page fill the window. */
html,
body {
  height: 100%;
  margin: 0;
  text-align: center;
}
div.input {
  border-radius: 5px;
  background-color: #c8d8e4;
  margin-top: 0px;
  margin-left: 80px;
  margin-right: 80px;
  margin-bottom: 0px;
  padding-top: 40px;
  padding-bottom: 10px;
}
input[type=text], select {
  width: auto;
  padding: 5px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
}
button[type=button] {
  width: 100%;
  background-color: #52ab98;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  width: 15%;
}
button[type=button]:hover {
  background-color: #3a796b;
}
div.dynamic {
  margin-left: 205px;
  margin-right: 205px;
  padding-left: 205px;
  padding-right: 205px;
  color: white;
```

```
}
.slidecontainer {
  width: auto;
  padding: 0px 400px;
  color: white;
}

.slider {
  -webkit-appearance: none;
  width: 100%;
  height: 15px;
  border-radius: 5px;
  background: #d3d3d3;
  outline: none;
  opacity: 0.7;
  -webkit-transition: .2s;
  transition: opacity .2s;
}

.slider:hover {
  opacity: 1;
}

.slider::-webkit-slider-thumb {
  -webkit-appearance: none;
  appearance: none;
  width: 25px;
  height: 25px;
  border-radius: 50%;
  background: #04AA6D;
  cursor: pointer;
  transition: height 0.2s ease-in-out;
 }

.slider::-webkit-slider-thumb:hover {
  -webkit-appearance: none;
  appearance: none;
  width: 25px;
  height: 35px;
  border-radius: 50%;
  background: #04AA6D;
  cursor: pointer;
}

.slider::-moz-range-thumb {
  width: 25px;
  height: 25px;
  border-radius: 50%;
```

```
  background: #04AA6D;
  cursor: pointer;
}

</style>
<!----------------------JavaScript---------------------->
<script>
  InitWebSocket()
  function InitWebSocket()
  {
    websock = new WebSocket('ws://'+window.location.hostname+':81/');
    websock.onmessage=function(evt)
    {
      JSONobj = JSON.parse(evt.data);
      document.getElementById('POTvalue').innerHTML = JSONobj.POT;
      var pot = parseInt(JSONobj.POT*2);//length
      var elem = document.getElementById("dynRectangle");
      var width = pot;
      elem.style.width = pot+"px";
     // elem.innerHTML = (width/2) + '°';
    }
  }
 /* function sendText(){
  websock.send(document.getElementById("txBar").value);
  document.getElementById("txBar").value ="";
}
*/

 function sendLocation(){

  var new_lot={
    my_lat:document.getElementById("myLat").value,
    my_lng:document.getElementById("myLng").value,
    new_lat:document.getElementById("lat").value,
    new_lng:document.getElementById("lng").value,
    manual_angle:'0',
  };
  console.log(new_lot);
  websock.send(JSON.stringify(new_lot));
  alert("Coordinates has been sent");

}



function manualRotation(){

  var manual_lot={
```

```
   my_lat:'0',
   my_lng:'0',
   new_lat:'0',
   new_lng:'0',
   manual_angle:document.getElementById("myRange").value,
   };
   console.log(manual_lot);
  websock.send(JSON.stringify(manual_lot));
  alert("Rotation angle has been sent.");

}
</script>
</head>
<!-------------------------HTML------------------------->
<body>
  <div class="header">
    <h1>IoT Integrated Antenna Rotator</h1>
  </div>
    <h2>
    Current Bearing: <span style="color:#7ccfbd" id="POTvalue">0</span> degree
    </h2>
    <div style="color:white;">
     0°
            

            &e
msp;

            &e
msp;  360°
    </div>
   <div class="dynamic">


       <div id="dynRectangle"></div>
    </div>
    <br>

    <!-------------------------------------------Google Maps---------------------------------------------------->
    <div id="map"></div>

    <!-- Async script executes immediately and must be after any DOM elements used in callback.
-->
    <script
      src="https://maps.googleapis.com/maps/api"
      async
    ></script>
      <br>
```

```
 <br>
 <div class="input">
 <form name="input">
 Current Latitude: <input type="text" name="myLat" id="myLat" value="">  
 Current Longitude: <input type="text" name="myLng" id="myLng" value="">  

 New Latitude: <input type="text" name="lat" id="lat" placeholder="Type or drag the
cursor">  
 New Longitude: <input type="text" name="lng" id="lng" placeholder="Type or drag the
cursor">  
    
 <button type="button" onclick="currentLot()">Get Current Location</button>
 <button type="button" onclick="sendLocation()">Send</button>
 </form>
</div>

<!--Manually control rotation angle with range slider-->
<h2>Manually Control Rotation Angle</h2>
<div class="slidecontainer">
 <input type="range" min="0" max="290" value="0" class="slider" id="myRange">

 <p>Rotation Angle: <span id="m_angle"></span>°</p>
 <button type="button" onclick="manualRotation()">Rotate</button>
</div>

<script>
 var slider = document.getElementById("myRange");
 var output = document.getElementById("m_angle");
 output.innerHTML = slider.value;

 slider.oninput = function manual() {
  var manualVal = this.value;
  output.innerHTML = manualVal;
 }
</script>

<br>
</body>

</html>
)=====";
```