Faculty of Electronics and Computer Technology and Engineering

DEVELOPMENT OF WIRELESS CONTROL FOR RETROFIT WHEELCHAIR SYSTEM USING ARDUINO-BASED MICROCONTROLLER WITH IOT

DONACIUS

Bachelor of Electronics Engineering Technology (Industrial Electronics) with Honours

2024

DEVELOPMENT OF WIRELESS CONTROL FOR RETROFIT WHEELCHAIR SYSTEM USING ARDUINO-BASED MICROCONTROLLER WITH IOT

DONACIUS

A project report submitted
in partial fulfillment of the requirements for the degree of
Bachelor of Electronics Engineering Technology (Industrial Electronics) with Honours

Faculty of Electronics and Computer Technology and Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2024

UNIVERSITI TEKNIKAL MALAYSIA MELAKA
FAKULTI TEKNOLOGI DAN KEJURUTERAAN ELEKTRONIK DAN KOMPUTER

BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II

Tajuk Projek : Development of Wireless Control for Retrofit Wheelchair using Arduino-Based Miicrocontroller with IOT

Sesi Pengajian :1-2023/2024

Saya DONACIUS mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓):

☐	**SULIT***	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	**TERHAD***	(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)
/	**TIDAK TERHAD**	

Disahkan oleh:

[illegible] [illegible]

(TANDATANGAN PENULIS) (COP DAN TANDATANGAN PENYELIA)

Alamat Tetap: 1940, LORONG 4, TAMAN TAWAU LAMA, BATUU 2 ½, JALAN APAS 91000 TAWAU, SABAH

Ts. KHAIRUL AZHA BIN A AZIZ
Pensyarah Kanan
Fakulti Teknologi Dan Kejuruteraan Elektronik Dan Komputer
Universiti Teknikal Malaysia Melaka

Tarikh:7 February 2024 Tarikh: 7 February 2024

*CATATAN: Jika laporan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh laporan ini perlu dikelaskan sebagai SULIT atau TERHAD.

DECLARATION

I declare that this project report entitled “Development of Wireless Control for Retrofit Wheelchair System Using Arduino-based microcontroller with IoT” is the result of my own research except as cited in the references. The project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature	:	[illegible]
Student Name	:	DONACIUS
Date	:	7 February 2024

APPROVAL

I approve that this Bachelor Degree Project 2 (PSM2) report Development of Wireless Control for Retrofit Wheelchair System Using Arduino-based microcontroller with IoT is sufficient for submission.

Signature	:	[illegible]
Supervisor Name	:	TS KHAIRUL AZHA BIN A AZIZ
Date	:	7 February 2024

DEDICATION

To my beloved parents,
Mother ESTER RUBEN, and father DONATIAN BIN JUNSUN
Thank you for your constant love and support.

ABSTRACT

A manual wheelchair is a standard wheelchair that can be self-propelled or pushed by others. However, it may present difficulties in certain scenarios, particularly for individuals with physical injuries or those living alone. The problem arises when users lose the ability to move around independently. This project aims to develop Wireless Control for Reftrofit Wheelchair System Using Arduino-base microcontroller with IoT. The android app are develop using Mit App Inventor platfrom. The wheelchair is equipped with two motor driver MD30C that control the DC motors attached to the wheels, which can be controlled via touch or voice commands on the smartphone. Additionally, the system allows wireless control of the household appliances and includes an IoT feature that enables users to remotely check whether the lights are on or off. Bluetooth communication is utilized between the smartphone and the arduino Uno . The smartphone interface offers two control options: voice commands and touch input. The wheelchair can be moved forward, backward, left, right, or stopped, and it features a proximity sensor to detect obstacles and automatically stop the wheelchair. Additionally, the inclusion of an IoT function allows for real-time monitoring of household appliances, enabling users to conveniently monitor and control their appliances remotely. The ability to wirelessly control appliances brings numerous benefits in terms of convenience and accessibility. In conclusion, this project empowers users with wireless control over their wheelchairs, enabling them to navigate their surroundings more effectively.

ABSTRAK

Kerusi roda manual ialah kerusi roda standard yang boleh digerakkan sendiri atau ditolak oleh orang lain. Walau bagaimanapun, ia mungkin menimbulkan kesukaran dalam senario tertentu, terutamanya bagi individu yang mengalami kecederaan fizikal atau mereka yang tinggal bersendirian. Masalah timbul apabila pengguna kehilangan keupayaan untuk bergerak secara bebas. Projek ini bertujuan untuk membangunkan Kawalan Tanpa Wayar untuk Sistem Kerusi Roda Reftrofit Menggunakan mikropengawal asas Arduino dengan IoT. Apl android dibangunkan menggunakan platfrom Mit App Inventor. Kerusi roda ini dilengkapi dengan dua pemandu motor MD30C yang mengawal motor DC yang dipasang pada roda, yang boleh dikawal melalui sentuhan atau arahan suara pada telefon pintar. Selain itu, sistem ini membenarkan kawalan wayarles ke atas perkakas rumah dan termasuk ciri IoT yang membolehkan pengguna menyemak dari jauh sama ada lampu dihidupkan atau dimatikan. Komunikasi Bluetooth digunakan antara telefon pintar dan arduino Uno. Antara muka telefon pintar menawarkan dua pilihan kawalan: arahan suara dan input sentuh. Kerusi roda boleh digerakkan ke hadapan, ke belakang, kiri, kanan atau berhenti, dan ia mempunyai penderia jarak untuk mengesan halangan dan memberhentikan kerusi roda secara automatik. Selain itu, kemasukan fungsi IoT membolehkan pemantauan masa nyata perkakas rumah, membolehkan pengguna memantau dan mengawal peralatan mereka dari jauh dengan mudah. Keupayaan untuk mengawal perkakas secara wayarles membawa banyak faedah dari segi kemudahan dan kebolehcapaian. Kesimpulannya, projek ini memperkasakan pengguna dengan kawalan wayarles ke atas kerusi roda mereka, membolehkan mereka menavigasi persekitaran mereka dengan lebih berkesan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

The wheelchair is the most commonly utilised piece of equipment for persons with lower limb disabilities. In comparison to others who have both upper and lower limb limitations, it allows them some movement and independence.The majority of wheelchairs on the market are manual, with some offering a motorised alternative. Existing wheelchairs can be modified to provide a more cost-effective option. The system includes a microprocessor, wireless module, motor driver, and rechargeable battery, which all work together to improve the wheelchair's functionality. The microcontroller controls the movement of the wheelchair by processing signals received from the wireless module and converting them into motion commands for the motor driver.This system enables users to control their wheelchairs wirelessly through a smartphone or tablet application, eliminating the need for physical switches and buttons.

1.2 Problem statement

In the current scenario, with the increasing number of elderly individuals, there is a growing need for assistive technologies that enhance their mobility and independence. Many elderly individuals live alone or in adult foster homes, and when they lose their ability to move around independently, they face significant challenges. Traditional manually-operated wheelchairs can be difficult to use, especially for individuals with physical injuries or disabilities but good mental strength. Moreover, in situations where assistance is required, patients have to rely on others to bring the wheelchair within their reach. Furthermore, individuals dependent on wheelchairs for mobility encounter difficulties in manually operating electrical outlets or lighting, leading to inconvenience and impracticality. Therefore, there is a pressing need to develop a solution that addresses these challenges and enables elderly individuals with limited mobility to navigate their surroundings more effectively.

1.3 Project Objective

The objective of this study is as follows :

a) To develop a system that can control the movement of a wheelchair using smartphone via Bluetooth .
b) To design an android control interface that incorporates both a D-pad and voice commands.
c) To design android system that control the electrical appliances.
d) To provide real-time status of the electrical appliances power state.

1.4 Scope of Project

This project primarily focuses on enhancing the usability of wheelchairs within controlled environments, such as private houses. The target audience for this project is elderly individuals who have lost the ability to navigate their surroundings and live independently. The wheelchair's control system is designed to cater to five essential movements: moving forward, moving backward, moving to the left, moving to the right, and stopping The system utilizes an Arduino Uno microcontroller, which serves as the central control unit for managing these movements. The inclusion of a proximity sensor serves the purpose of detecting obstacles and automatically halting the wheelchair's movement to ensure user safety. Additionally, the project incorporates functionality to control specific power outlet within the house, specifically the lights. The Internet of Things (IoT) aspect of the project enables users to monitor the status of the lights, whether they are turned on or off.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

In this chapter will focus more on background research and literature review for the whole project. The major goal of this chapter is to discuss previous research and current project. The philosophy and concept utilized to tackle the project's challenge are discussed in this project. The whole sources of material are journal, paper and case studies that are chosen for their resemblance to the project aim.

2.2 History of Wheelchair

The first self-propelled wheelchair was invented in the 1655 by a clock-maker in Nurembered,Germany. Stephan Farller (1633-1689), built his own wheelchair to ease the difficulty of walking.

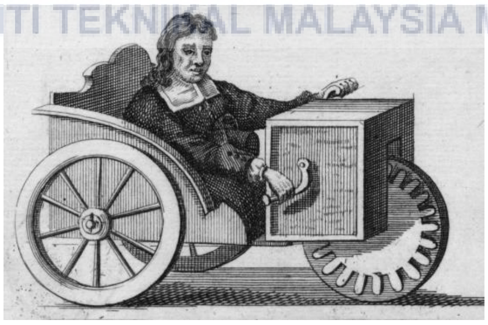


Figure 2.2.1 Fafler in his 50s sitting in his vehicle

From then the development of various way to control a wheelchair has been develop. The need of new method to control wheelchair arise as not every user or patient can used a conventional wheelchair where it is self-propelled by user or pushed by a companion. In a program for injured veterans returning from World war II ,Canadian inventor George Klien invented the first electric-powered wheelchair.[9]

2.3 Types of Wireless Communication

Wireless communications refers to the transmission of voice and data without the need for cables or wires. Instead of relying on physical connections, information is sent through electromagnetic signals that are broadcasted from transmitting facilities to intermediate and end-user devices.[6]

2.3.1 Infrared Communication

Infrared communication involves the use of IR radiation to transmit information within a device or system.[6] IR refers to electromagnetic energy with a longer wavelength than red light. It is commonly used for tasks like security control, TV remote control, and short-range communication. In the electromagnetic spectrum, IR radiation falls between microwaves and visible light, making it suitable for communication purposes.

To establish successful infrared communication, two essential components are needed: a photo LED transmitter and a photo diode receptor. The LED transmitter emits IR signals as non-visible light, which are then captured and detected by the photoreceptor. In this way, information is transferred between the source and the target. Common examples of sources and targets for infrared communication are mobile phones, TVs, security systems, laptops, and other devices that support wireless communication.[6]

2.3.2 Wi-Fi

Wi-Fi is a wireless communication technology that is utilized by gadgets like computers and smartphones. It makes it possible for various devices to connect wirelessly. In a Wi-Fi arrangement, a router serves as the primary hub for communication. Wi-Fi networks can only be accessed by users who are close to the router.

Because they provide the convenience of wireless communication, Wi-Fi networks are popular for networking applications. But it's crucial to use passwords to secure these networks. This guarantees the network's security and deters unauthorized access.[6]

2.3.3 Bluetooth Technology

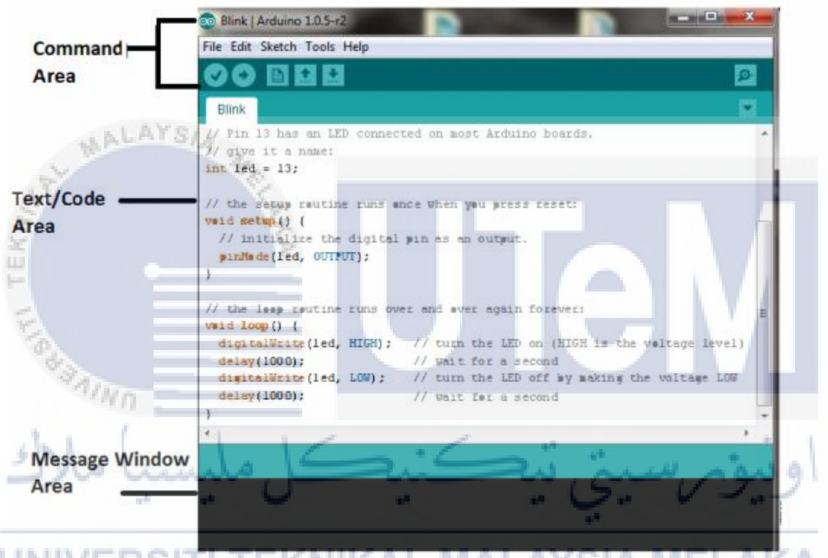
A popular short-range technology for Wireless Personal Area Networks (WPAN) is Bluetooth.[7] The primary purpose of Bluetooth technology is to enable wireless data transfer between various electronic devices and a system. Hands-free earbuds, a mouse, and a wireless keyboard are all connected to cell phones. Information can be transferred from one device to another using a Bluetooth device. This technology serves several purposes and is widely utilised in the wireless communication market. Despite the fact that development platforms make the process easier, many tools do not work with the Bluetooth specification.[7]

2.4 Study Related to Wireless Control Wheelchair

In this section, we will look on the components that are widely used by researchers. The specification of components are referred on the datasheet of the component to avoid wrong information.

2.4.1 Arduino IDE

Arduino is an open-source gadget based on simple hardware and software that may be used to create electrical appliance projects. The Arduino IDE Software compiles the programming code, converts it to binary, and transfers it to the circuit board through serial port connection, as illustrated in Figure . Furthermore, the Arduino employed the C++ programming language, which is relatively easy and makes learning to code simpler since it uses conventional serial protocol communication, which connects to a computer through USB.



```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Figure 2.4.1.1 Arduino IDE

The Arduino software is easy to use for beginners since it makes working with microcontrollers simpler. In addition, Arduino has the advantage of being a less expensive programmable board than other microcontroller platforms. Because it is straightforward and practical, it is a common initial choice. Users of the Arduino program can also attach shields, which are pre-built circuit boards with additional capabilities that let them experiment with more sensors, displays, and inputs. There are numerous Arduino boards available that can be utilised for various tasks. The widely used Arduino family board and a suitable starting point is the Arduino UNO, which is shown in Figure .[17]

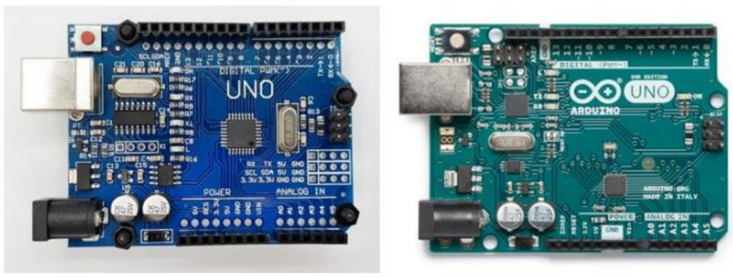


Figure 2.4.1.2 Arduino Uno

2.4.2 NodeMCU

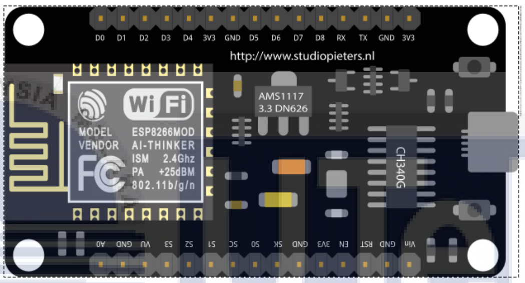


Figure 2.4.2.1 NodeMCU ESP8266

The module discussed in the paragraph is primarily based on the ESP8266 microchip, which is a cost-effective Wi-Fi module that integrates a full TCP/IP stack and microcontroller capabilities. It was developed by Espressif Systems. The ESP8266 NodeMcu is a versatile device that combines certain features of the Arduino board with the ability to connect to the internet.[18]

While Arduino modules and microcontrollers have been popular choices for automation projects, they lack built-in Wi-Fi capabilities. To overcome this limitation and enable internet connectivity, an external Wi-Fi protocol needs to be added to these devices.

The NodeMCU is a well-known development board that utilizes the ESP8266 Wi-Fi System-on-Chip (SoC). The version mentioned in the paragraph is version 3, based on the ESP-12E module. NodeMCU is an open-source firmware and development kit that facilitates prototyping of IoT (Internet of Things) products using LUA scripting. It can also be programmed using the Arduino IDE, providing flexibility for different programming approaches.[14]

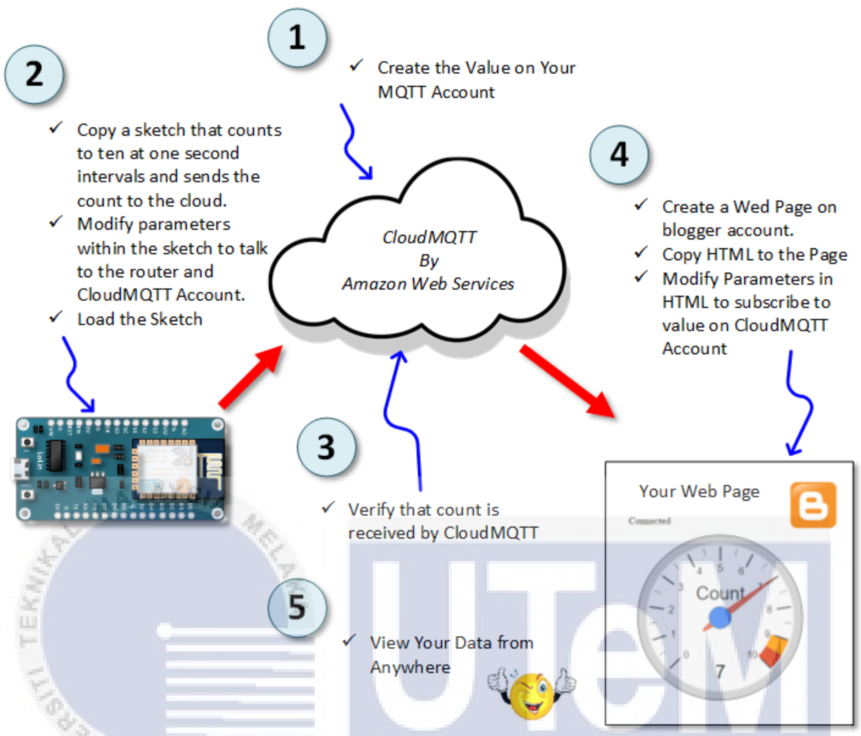

1
✓ Create the Value on Your MQTT Account
2
✓ Copy a sketch that counts to ten at one second intervals and sends the count to the cloud.
✓ Modify parameters within the sketch to talk to the router and CloudMQTT Account.
✓ Load the Sketch
CloudMQTT By Amazon Web Services
4
✓ Create a Wed Page on blogger account.
✓ Copy HTML to the Page
✓ Modify Parameters in HTML to subscribe to value on CloudMQTT Account
3
✓ Verify that count is received by CloudMQTT
Your Web Page
Count
7
5
✓ View Your Data from Anywhere

Figure 2.4.2.2 IOT Application

2.4.3 MIT-APP Inventor

MIT App Inventor is a web-based platform for building mobile apps for Android devices. It uses a graphical drag-and-drop interface to allow users to design and build apps without requiring any prior programming experience.

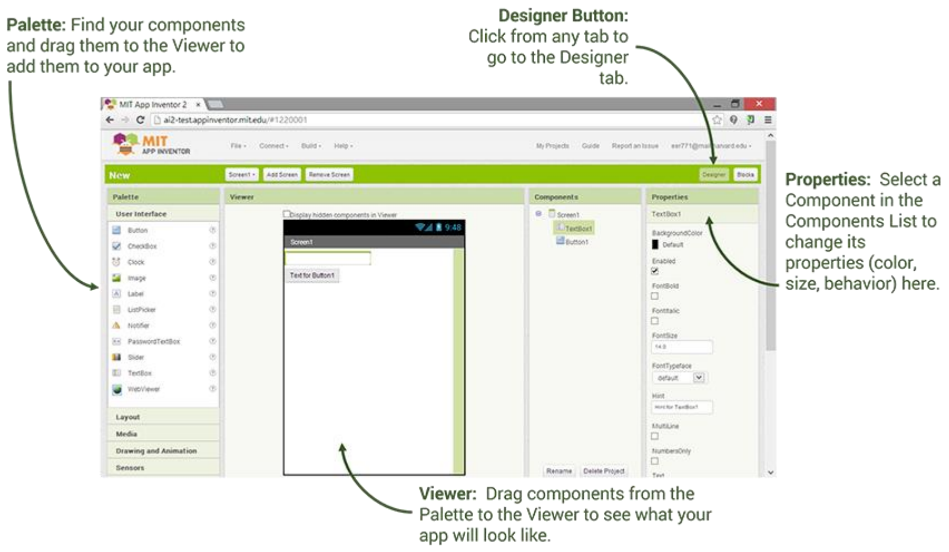

Palette: Find your components and drag them to the Viewer to add them to your app.
Designer Button: Click from any tab to go to the Designer tab.
Properties: Select a Component in the Components List to change its properties (color, size, behavior) here.
Viewer: Drag components from the Palette to the Viewer to see what your app will look like.
MIT APP INVENTOR
Palette
Viewer
Components
Properties

Figure 2.4.3.1 MIT App inventor IDE

2.5 Previous Related Research Work

Before moving forward with the development of wireless control for retrofit wheelchair system employing microcontroller with IoT, past research or articles are revised. Reading earlier research is mostly done to explore the theories and concepts that were utilized to address the issues that would arise throughout the project's development. The linked sources that have been chosen for this project are listed below.

2.5.1 Wheelchair Control by Head Motion

A. Pajkanović and B. Dokić presents a microcontroller-based system that enables head movements to control a typical electric wheelchair. Both mechanical and electronic components make up the system. A novel head motion identification technique is developed using accelerometer data processing. The system's mechanical actuator manages the wheelchair joystick. The technology is compatible with a variety of common electric wheelchair models.

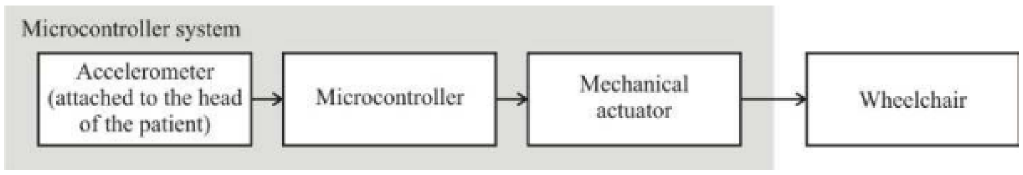


Figure 2.5.1.1 Microcontroller system block diagram

The system allows individuals to control a standard electric wheelchair using head motion. The system prototype, comprising an accelerometer, a microcontroller, and a mechanical actuator, is implemented and tested. The accelerometer collects data on head motion, which is processed by a novel algorithm implemented on the microcontroller. The resulting output is then connected to the mechanical actuator, enabling the positioning of the wheelchair joystick based on the user's commands. The algorithm within the microcontroller translates the sensor data into the appropriate joystick position. Furthermore, the mechanical actuator is designed to be compatible with various types of standard electric wheelchairs.

Based on the force measurements provided by an accelerometer mounted to the head, head motion recognition is performed. As previously indicated, the motion set simply consists of four members, each of which represents a conceivable head lean. As a result, the algorithm must make an educated guess as to which of the four directions the head is leaning. In other words, reading the accelerometer data for just two axes in this case, x and y is adequate.[1]

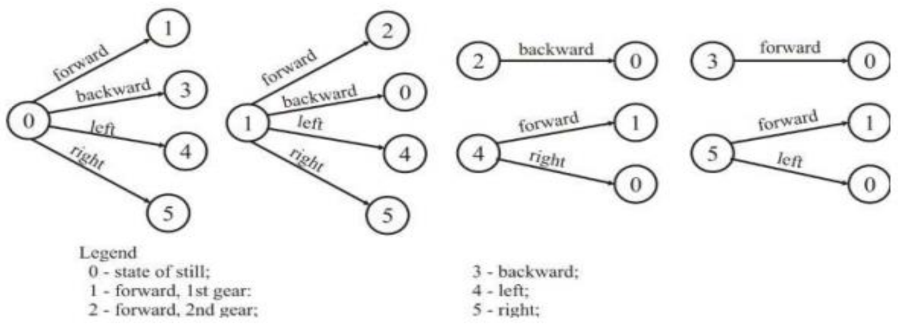


Figure 2.5.1.2 Wheelchair state diagram and relative meaning of user commands.

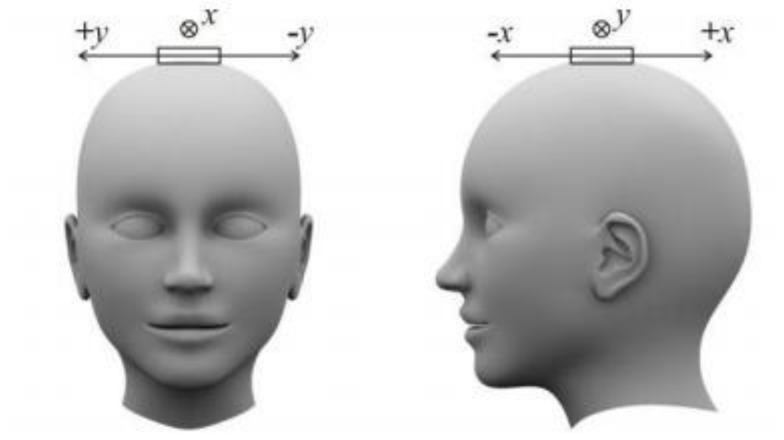


Figure 2.5.1.3 The position of the accelerometer relative to the head

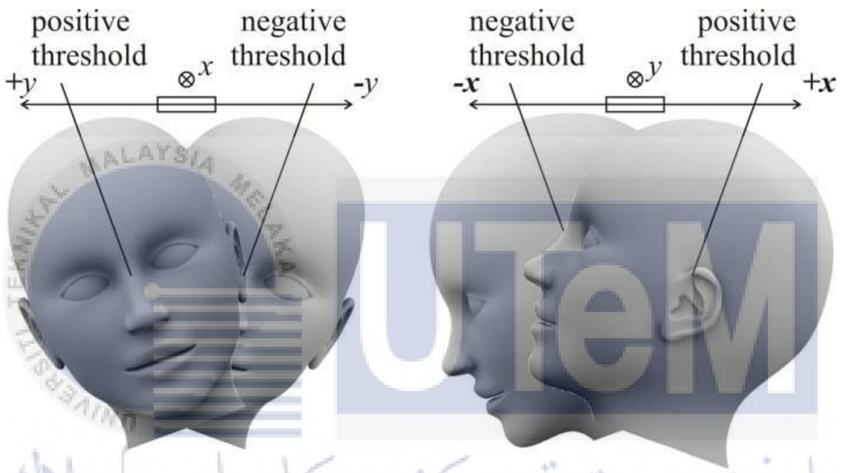


Figure 2.5.1.4 An example of threshold setting

2.5.2 Wheelchair Infrared Sensor Controlled Wheelchair for Physically Disabled People

Nowshin, N., Rashid, M. M., Akhtar, T., & Akhtar, N. proposed an automated wheelchair specifically designed for individuals with one side or partial body paralysis. The wheelchair is controlled by the user's eyes using an infrared (IR) sensor. The system is implemented using an Arduino Uno and Infrared Sensor to enable directional movement. A notable feature of this design is the inclusion of an emergency contact system, utilizing a GSM module and a compact solar-powered battery system. In summary, the proposed wheelchair design is hands-free, operates on renewable energy, and incorporates an emergency system that sends text messages to the user's emergency contacts in case of emergencies.

Nowshin, N., Rashid, M. M., Akhtar, T., & Akhtar, N. proposed model includes two push buttons and an IR sensor as inputs, with an Arduino Uno serving as the main controller. The movement of the wheelchair is controlled through a relay module. Button 1 is used to activate the system for movement, and the wheelchair's wheels respond to the number of eye blinks detected. In case of an emergency, Button 2 triggers the emergency message feature. Pressing it once sends a message to a family member's mobile phone, while pressing it twice sends a message to a physician, informing them about the user's health condition via a GSM module. To ensure power supply for the model, a solar panel and a rechargeable battery are used.[2]

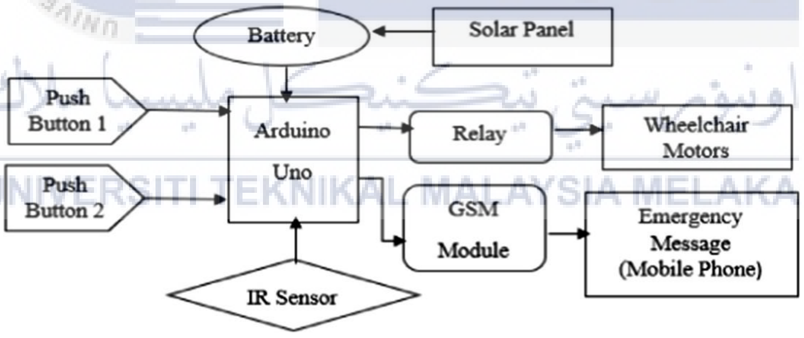


Figure 2.5.2.1 Block diagram of an automated wheelchair working process.

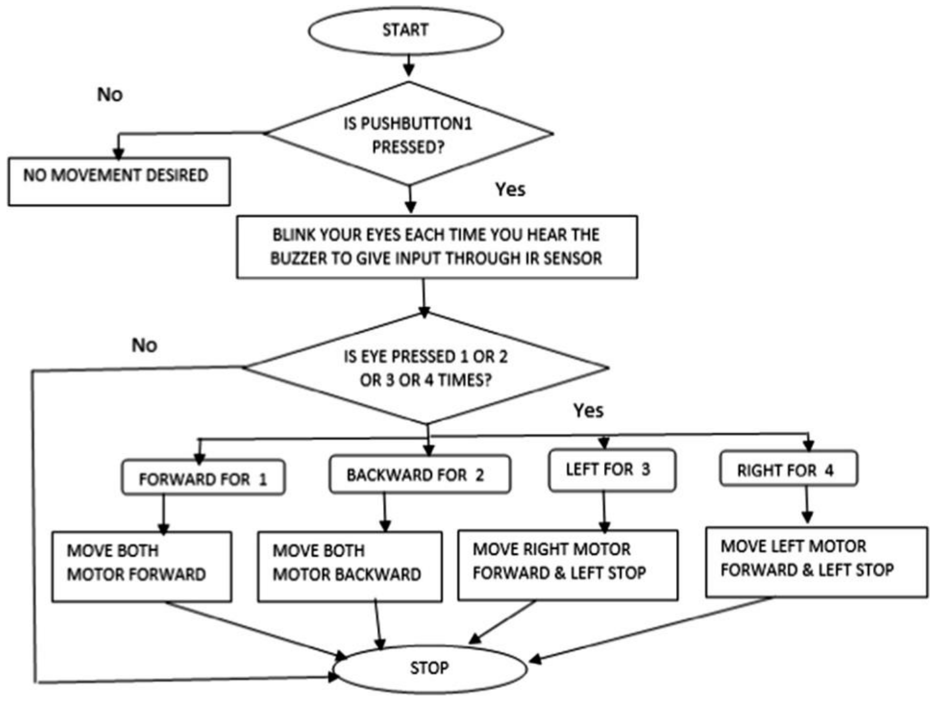

START
IS PUSHBUTTON1 PRESSED?
No
NO MOVEMENT DESIRED
Yes
BLINK YOUR EYES EACH TIME YOU HEAR THE BUZZER TO GIVE INPUT THROUGH IR SENSOR
IS EYE PRESSED 1 OR 2 OR 3 OR 4 TIMES?
No
Yes
FORWARD FOR 1
BACKWARD FOR 2
LEFT FOR 3
RIGHT FOR 4
MOVE BOTH MOTOR FORWARD
MOVE BOTH MOTOR BACKWARD
MOVE RIGHT MOTOR FORWARD & LEFT STOP
MOVE LEFT MOTOR FORWARD & LEFT STOP
STOP

Figure 2.5.2.2 Flowchart for controlling the movement of the wheelchair.

The IR sensor used in the system operates as a comparator using an Operational amplifier (LM 358). It compares the analog voltages from a potentiometer and the voltage generated by the photodiode. These voltages are applied to the terminals of the IC, which produces a digital output on the output pin, indicated by a red LED. When the voltage (Vd) generated by the photodiode is higher than the set voltage on the potentiometer, the output is HIGH, and vice versa. This IR sensor is specifically utilized to detect variations per eye blink. When the eye is closed, more IR light is reflected, whereas when the eye is open, less IR light is reflected due to the darker pupil/iris compared to the eyelid. By employing this logic, the IR sensor, fixed in a spectacle, can detect the number of times the eye is pressed.The Arduino digital pins are connected to the IR sensor to receive inputs from the user, and the output provided by the sensor is further processed. Based on the information in Table 1, the movement of the wheelchair motors is determined.

Table 2.5.2-1. Different states of motor direction

Number of eye blink	State of the movement
0	Stop
1	Forward
2	Reverse
3	Right
4	Left

2.5.3 Eye Controlled Wheelchair Based On Arduino Circuit

The concept presented by Reona Cerejo,Valentine Correi,and Neil Pereira in this paper is to develop an Eye Controlled System that enables the movement of a patient's wheelchair based on the movements of their eyeballs. For individuals with quadriplegia, who have limited mobility and can only move their eyes and partially tilt their heads, detecting these movements provides an opportunity for wheelchair control. Although there are various interfaces and techniques available for powered wheelchairs, they tend to be expensive and not affordable for economically disadvantaged individuals.

In this paper, a simpler and cost-effective method for developing a wheelchair is proposed. The system involves a person sitting on an automated wheelchair with a camera mounted on it. By looking in a specific direction and making eye movements, the person can control the wheelchair's movement in that direction. The camera signals captured are then sent to a PC and controlled by MATLAB. The control signals are then transmitted to an Arduino circuit through a Serial Interface. The Arduino circuit utilizes these signals to control the wheelchair's motors, enabling movement in the desired direction. [3]

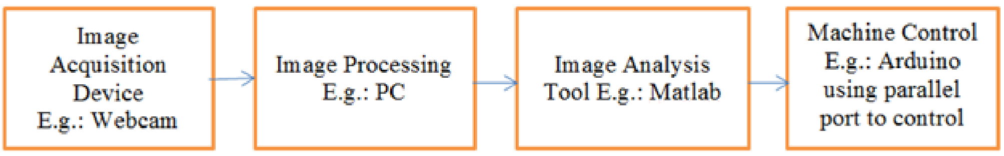


Image Acquisition Device E.g.: Webcam
Image Processing E.g.: PC
Image Analysis Tool E.g.: Matlab
Machine Control E.g.: Arduino using parallel port to control

Figure 2.5.3.1. Block Diagram

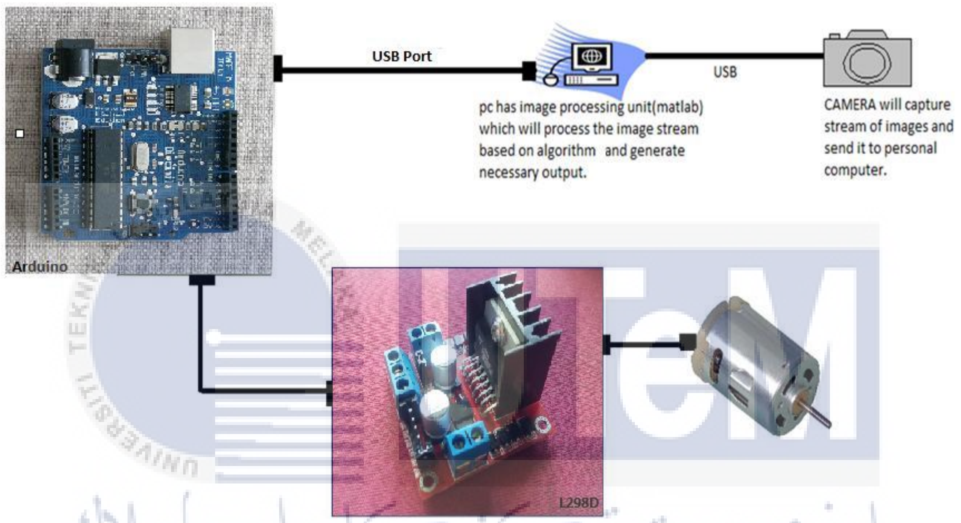
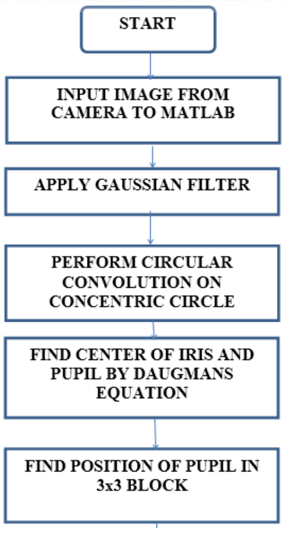

USB Port
USB
pc has image processing unit(matlab) which will process the image stream based on algorithm and generate necessary output.
CAMERA will capture stream of images and send it to personal computer.
Arduino
L298D

Figure 2.5.3.2. System Design Diagram


START
INPUT IMAGE FROM CAMERA TO MATLAB
APPLY GAUSSIAN FILTER
PERFORM CIRCULAR CONVOLUTION ON CONCENTRIC CIRCLE
FIND CENTER OF IRIS AND PUPIL BY DAUGMANS EQUATION
FIND POSITION OF PUPIL IN 3x3 BLOCK

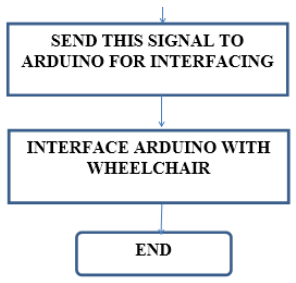

SEND THIS SIGNAL TO ARDUINO FOR INTERFACING
INTERFACE ARDUINO WITH WHEELCHAIR
END

Figure 2.5.3.3. System Flow graph

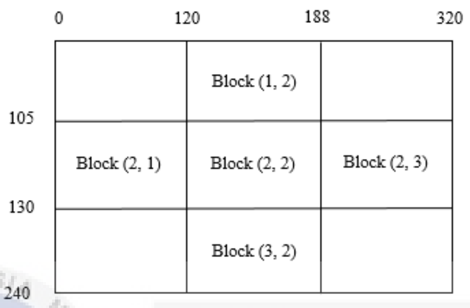


Figure 2.5.3.4. Image Processing Diagram

2.5.4 Voice Controlled Intelligent Wheelchair

Masato Nishimori, Takeshi Saitoh, and Ryosuke Konishi proposed an improvement to the previous researcher voice controlled wheelchair that uses voice command as the interface that has basic modes which are running until next command is input, turning or rotation and stopong. The voice command serves as the interface for the voice-controlled wheelchair being developed in this study. The wheelchair can operate in three fundamental modes: running until the next instruction is input, turning or rotating, and stopping. Voice commands have been implemented to control the wheelchair. These commands are divided into two categories: reaction commands and verification commands. There are a total of nine voice commands, with five being basic reaction commands and four being short moving reaction commands, which cause the wheelchair to move short distances. The specifics of each voice command and their corresponding reactions are provided in Table 2. [4]

Table 2.5.4-1 voice command and reaction

No	Command	Reaction(mode)
2	Susume	run forward
3	sagare	run backward
4	migi	turn right
5	Hidari	Turn left
6	Sukoshi-susume	Run forward about 30cm
7	Sukoshi-sagare	Run backward about 30cm
8	Sukoshi-migi	Rotate right about 30 degrees
9	Sukoshi-hidari	Rotate left about 30 degrees
A	OK/yes	Acceptance command
B	Torikeshi/no/cancel	Rejection command

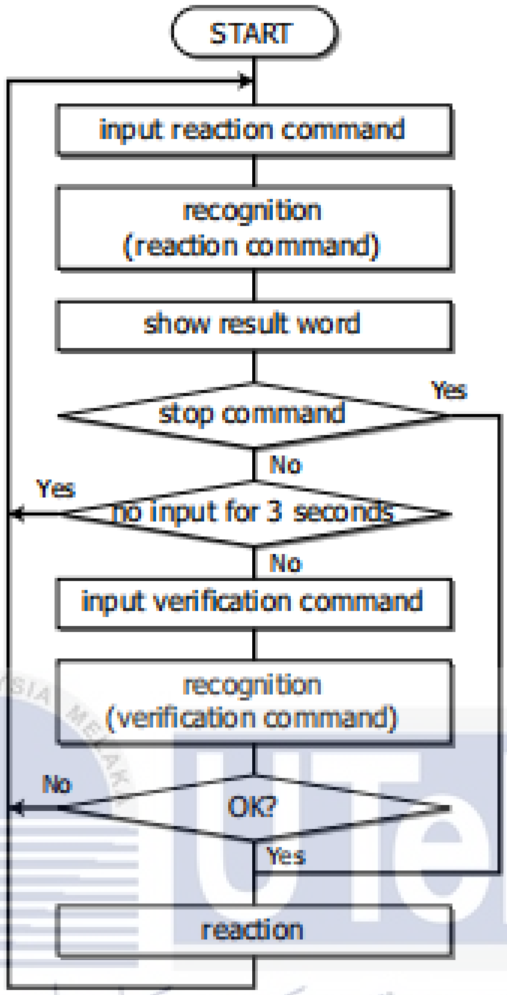


Figure 2.5.4.1. Control algorithm

2.5.5 Wireless Smart Wheelchair

Sirisha, P & Bethapudi, Prakash & Meghana, Ratna & Yamini, C & Lakshmi, E proposed an approach that offers a low-cost, simple, and user-friendly solution for a voice-controlled platform. This solution is fully customizable based on the user's spoken language and aims to enhance the user's independent mobility. The research explores the utilization of smartphones as the central control unit for a robot, which is an active research field with numerous opportunities and potential advancements. Additionally, Bluetooth technology is employed to enable wireless communication between the wheelchair and the mobile platform, providing a convenient remote control solution.

The project also incorporates the use of ultrasonic sensors to detect obstacles within a range of 4 meters. When an obstacle is detected, the system is notified and the wheelchair comes to a stop until further commands are given. The proposed system utilizes an Arduino Uno microcontroller and a Bluetooth module, which are controlled through an Android application. This enables individuals with disabilities to have control over their wheelchair according to their specific needs, even when caregivers are unable to actively monitor them.

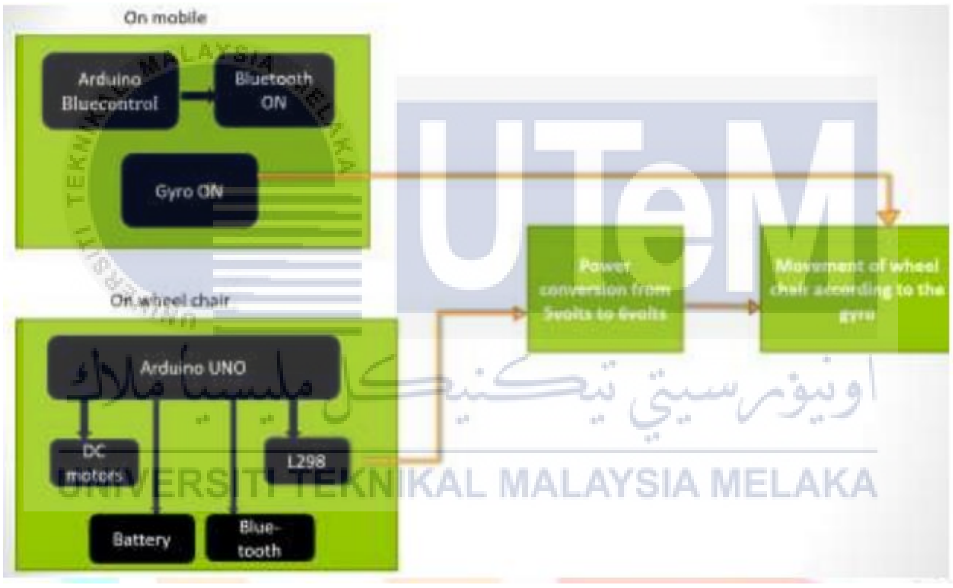


Figure 2.5.5.1 Architecture of wheel Chair with Pulse playground sensor

The provided figure depicts the model of a wireless smart wheelchair designed to be highly efficient .This wheelchair requires minimal effort from the user to initiate movement. The built-in accelerometer in a mobile device plays a significant role in controlling the wheelchair's movements in all four directions. By capturing the user's movements through the mobile device, commands are sent to the wheelchair, allowing it to move accordingly. A slight movement by the user triggers the desired direction in the wheelchair.

This wheelchair operates through a simple Bluetooth pairing. Tilting the mobile device to the left causes the wheelchair to turn left, tilting it to the right makes the chair turn right, tilting it forward moves the wheelchair in the forward direction, and tilting it backward moves the wheelchair backward. When there is no change in the mobile device's position and it remains at rest, the wheelchair will also come to a stop. [5]

2.6 Previous Researcher Works Comparison

Table 2.6-1 Previous Researcher Comparison

No	Author	Year	Title	Method	IOT Capability	Remote control of the wheelchair
1	Aleksandar Pajkanović, and Branko Dokić	2013	WHEELCHAIR CONTROL BY HEAD MOTION	Atmega 19, accelerometer, Mechanical actuator, Servo motor	Not stated	No. Sensor used to control wheelchair move is wired to control box.
2	Nadia Nowshin(&) ,Md Moont	2019	INFRARED SENSOR CONTROLLED WHEELCHAIR	Arduino Uno, IR sensor (to detect eye	Used GSM Module to send emergency	No .Sensor used to control the wheelchair movement is

	asir Rashid , Tasneema Akhtar , and Nafisa Akhtar		FOR PHYSICALLY DISABLED PEOPLE	blink),GSM Module	assistance message	wired to control box.
3	Reona Cerejo ,Valentine Correi, and Neil Pereira	2015	EYE CONTROLLED WHEELCHAIR BASED ON ARDUINO CIRCUIT	Arduino Uno, Camera (to detect eye movement) , Matlab (image processing unit)	Not stated	No .Sensor used to control the wheelchair movement is wired to control box.
4	Masato Nishimori, Takeshi Saitoh	2007	VOICE CONTROLLED INTELLIGENT WHEELCHAIR	PIC, Microphone, Laptop (control system)	Not stated	No .Sensor used to control the wheelchair movement is wired to control box.

	and Ryosuke Konishi					
5	P Sirisha ,Dr Prakash Bethapudi, Ratna Meghana M, CH Yamini, and E NaveenaLakshmi	2020	WIRELESS SMART WHEELCHAIR	Arduino UNO, NodeMCU, Mobile Device (index finger sensor)	Uses Node MCU to monitor the user heartbeat through Blynk app	Using android app from smartphone.
6	Ibrahim Bin	2015	WIRELESS ANDROID-BASED	IOIO ,RF receiver, Motor	Not Stated	Using Android App to control the

	Abu Bakar		WHEELCHAIR CONTROL AND SYSTEM	Driver and motor		Wheelchair movement.

2.7 Summary

The information relevant to this project learned about the techniques employed by the prior researchers based on their ideas and works, as well as their research. In order to demonstrate this, a comparison of the methods used and their benefits and drawbacks is done by the approaches used by earlier scholars, both in terms of parallels and differences. To give a comprehensive picture of how each component works, the ideas underlying these components are also described. The summary consist on what the method use to control the wheelchair, type of microcontroller and input sensor.

According to the provided table, four different methods were employed by researchers in the study. The first researcher utilized head motion, the second and third researchers employed eye motion and movement, the fourth researcher used voice control, and the last researcher utilized finger motion. The microcontrollers primarily used were Arduino, with some researchers using a PIC-type microcontroller. The choice of sensor depended on the specific method used. In the case of head motion, an accelerometer sensor was employed to detect the user's direction, which then controlled a mechanical actuator connected to the wheelchair joystick. The second and third researchers used different approaches for eye motion. The second researcher used an IR sensor to detect the motion caused by blinking, with each consecutive blink representing a different instruction for the microcontroller. The third researcher used a camera to detect the user's eye direction for wheelchair control. The fourth researcher utilized voice commands and a microphone to detect the user's instructions for wheelchair movement. The last researcher employed a smartphone as an input device, where the user would use their index finger to set the wheelchair's direction based on a virtual joystick in a mobile app.

In conclusion, based on the methods and modules used by previous researchers, it can be inferred that several features will be implemented in the development of the Wireless Control for Retrofit Wheelchair System using Microcontroller with IoT. The chosen methods for the project will be voice command and mobile applications, and the Arduino Uno microcontroller will be used as the processing unit for the project.

CHAPTER 3

METHODOLOGY

3.1 Introduction

The project flow was covered in great detail in this chapter. A portion of this chapter discussed the methods used to carry out this project throughout time. The goal of this chapter is to describe in detail and validate how the project was executed. The design, development, and usage in the development of Wireless Control for Retrofit Wheelchair system using Arduino-based microcontroller with IoT. These techniques were effectively carried out, producing a suitable mechanism and part for a wireless control for wheelchair.

3.2 Project Milestone

Project milestones play a crucial role in project management as they mark significant points or achievements throughout the project's lifecycle. These milestones serve as essential markers, providing a clear indication of progress, key deliverables, and important events within a project. Understanding project milestones is vital for effective project planning, execution, and control. They serve as strategic checkpoints that help break down the project into manageable phases and provide an overview of the project's timeline.

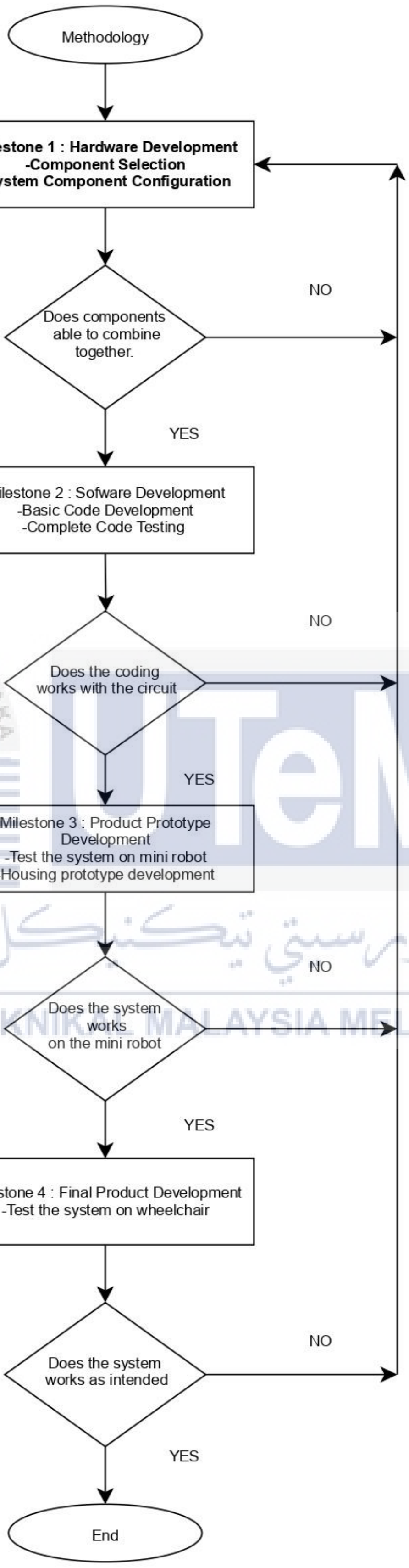

Methodology
Milestone 1 : Hardware Development
-Component Selection
-System Component Configuration
Does components able to combine together.
NO
YES
Milestone 2 : Sofware Development
-Basic Code Development
-Complete Code Testing
Does the coding works with the circuit
NO
YES
Milestone 3 : Product Prototype Development
-Test the system on mini robot
-Housing prototype development
Does the system works on the mini robot
NO
YES
Milestone 4 : Final Product Development
-Test the system on wheelchair
Does the system works as intended
NO
YES
End

Figure 3.2.1. Milestone Flowchart

3.3 Milestone 1: Hardware development

The first milestone was established on the basis of the aim within the methodology, which is to build a system by selecting appropriate and practical components and arranging them in an appropriate circuit architecture. Figure below show flowchart for this first milestone in the project.

3.3.1 System Design

The system consistof several small component parts, each of which requires specific connection for it to properly function. The component include a ESP8266, smartphone, relay, motor driver , lamp and the Arduino as the microcontroller. Firstly, connection between smartphone and the ESP8266 must be establish via Bluetooth. Once the connection is done there are two method to control the motor that is by using a touch pad and voice command.

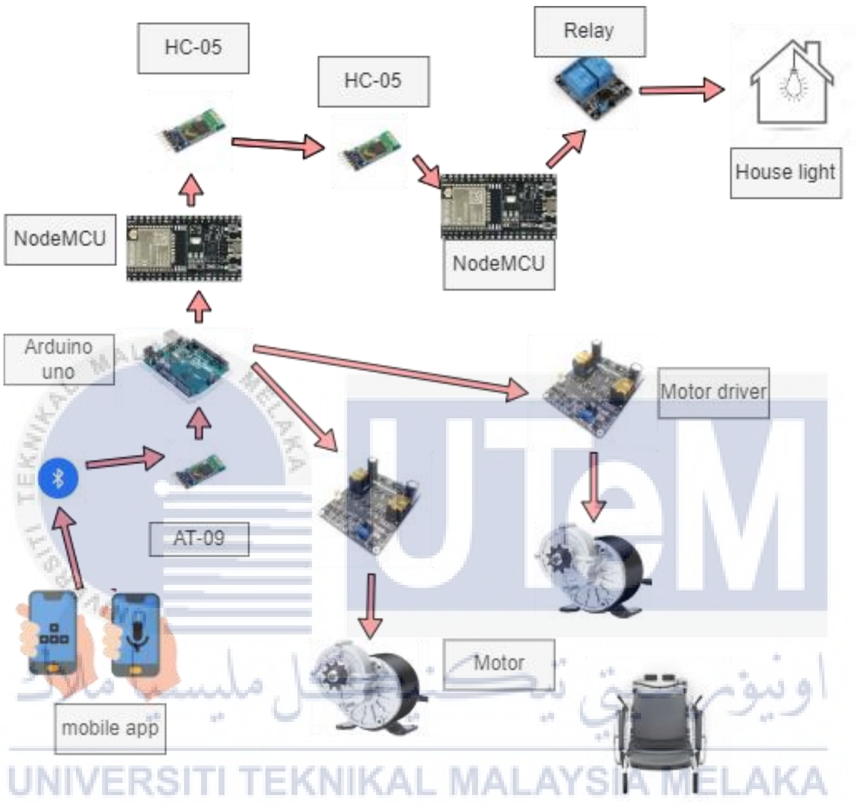


Figure 3.3.1.1 Block Diagram

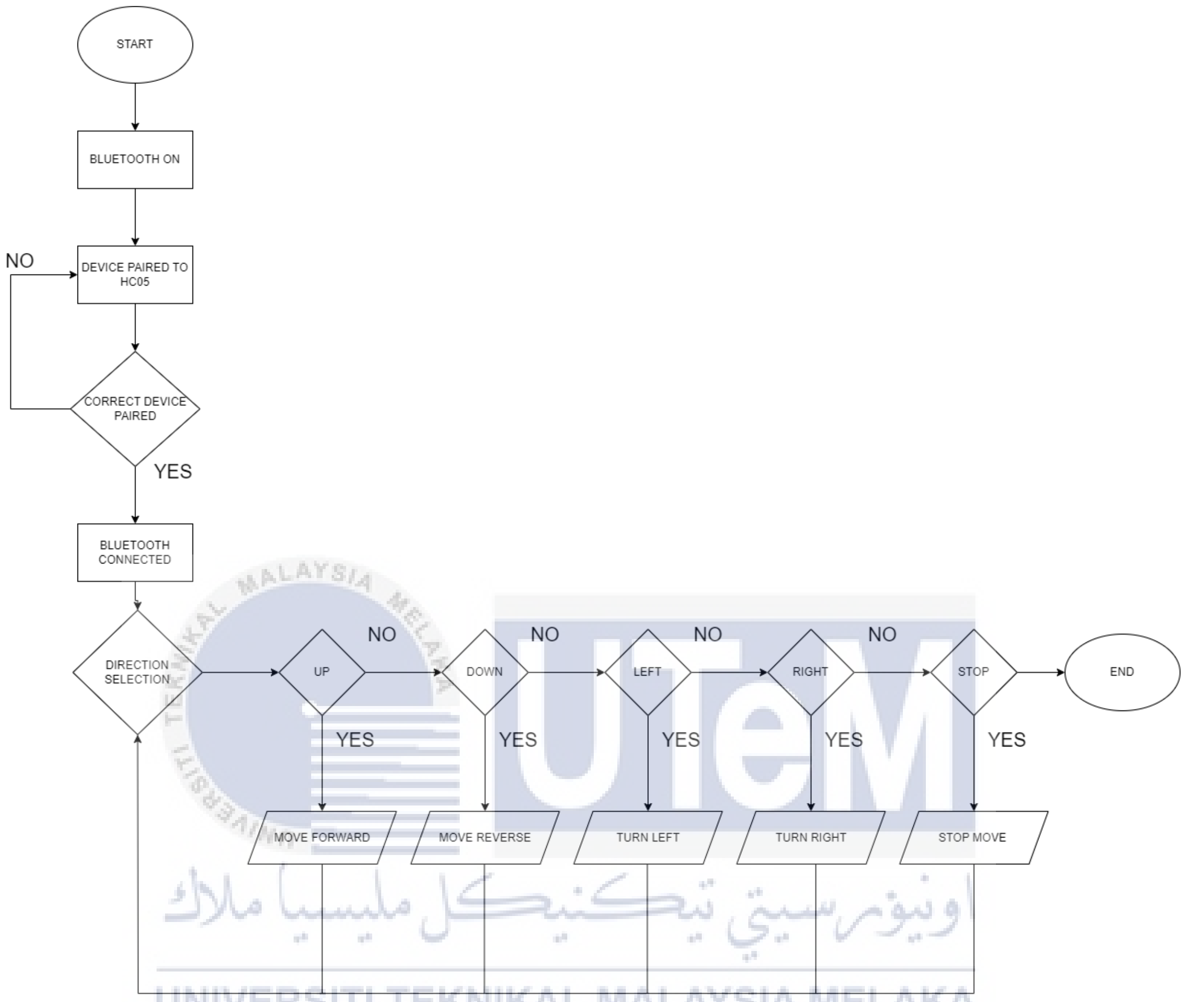


Figure 3.3.1.2. Flowchart for Touch Control

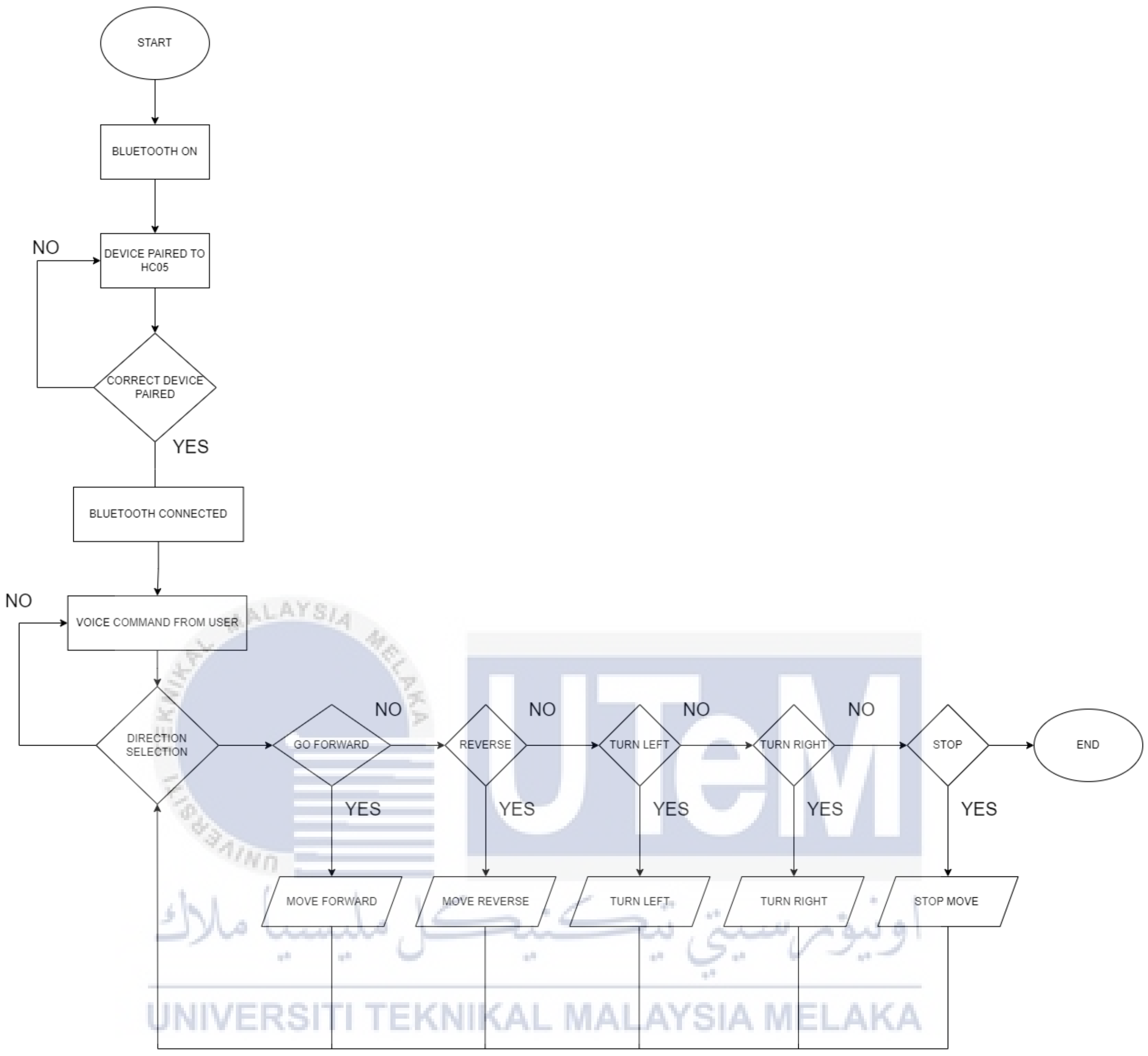


Figure 3.3.1.3. Flowchart for Voice Control

3.3.2 Component Selection

The project is comprised of a small number of component parts, each of which requires a specific connection in order to function properly. These include a ESP32, relay module ,MD30C Motor Driver that is controlled by a microcontroller. After the process of studying the related literature, a decision regarding the selection of components is made. According to the findings of the literature analysis conducted on Chapter 2, the components that were chosen are as follows: a microcontroller Arduino Uno as the main processing unit and a ESP 32 as the communication between smartphone and the Arduino Uno .For the purpose of developing the prototype, these components are practical, suited for the task at hand, and simple to implement.

3.3.2.1 Arduino Uno

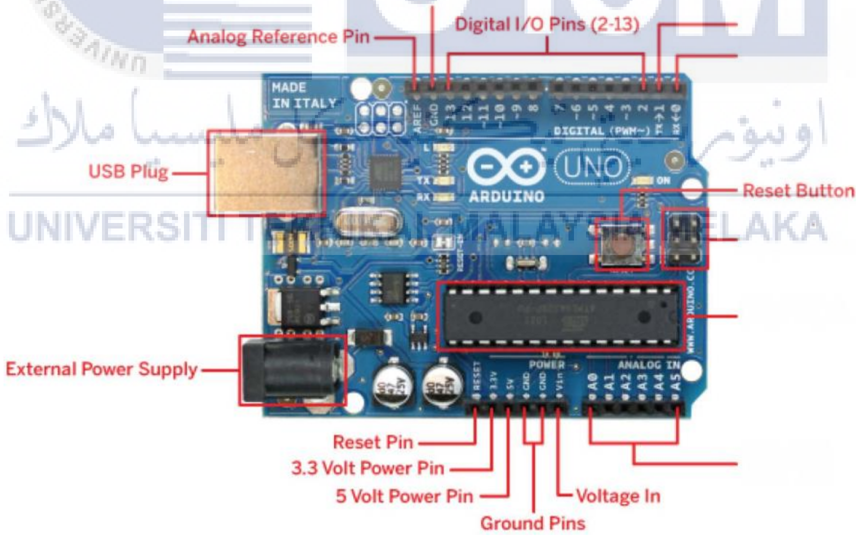


Figure 3.3.2.1.1. Arduino Uno

Arduino is an open-source platform widely used for constructing and programming electronics. It enables communication and data exchange with various devices, including the ability to command specific electronic devices over the internet. The Arduino platform utilizes a hardware component called Arduino Uno, which is a circuit board, and a software program written in a simplified version of the C++ programming language, to program the board.

Arduino can be used to read information from various input devices such as sensors, antennas, and potentiometers (trimmers), allowing for data acquisition. Additionally, it can send information to output devices such as LEDs, speakers, LCD screens, and DC motors, facilitating control and interaction with these devices.

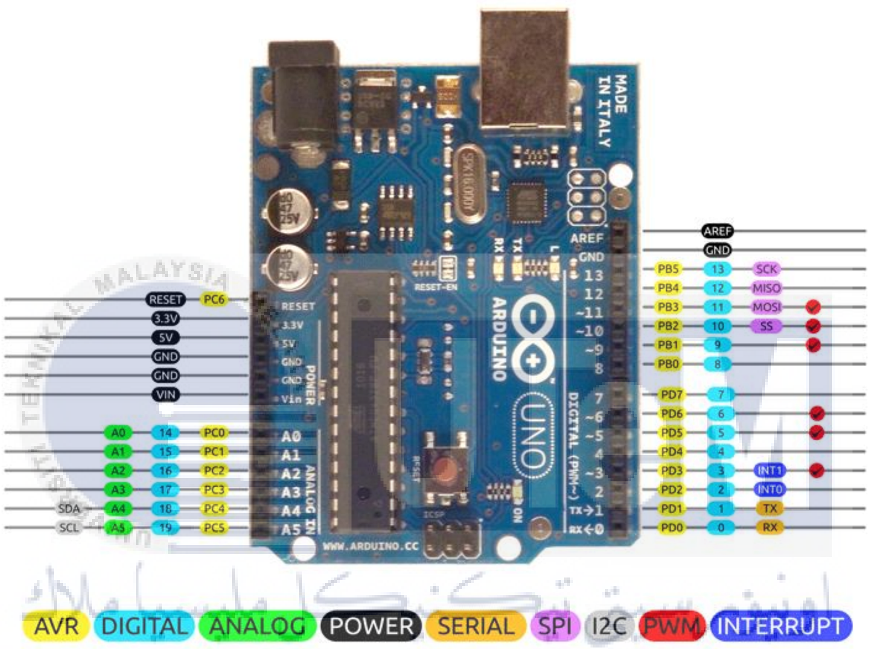


Figure 3.3.2.1.2. Arduino Uno pinout

3.3.2.2 NodeMCU

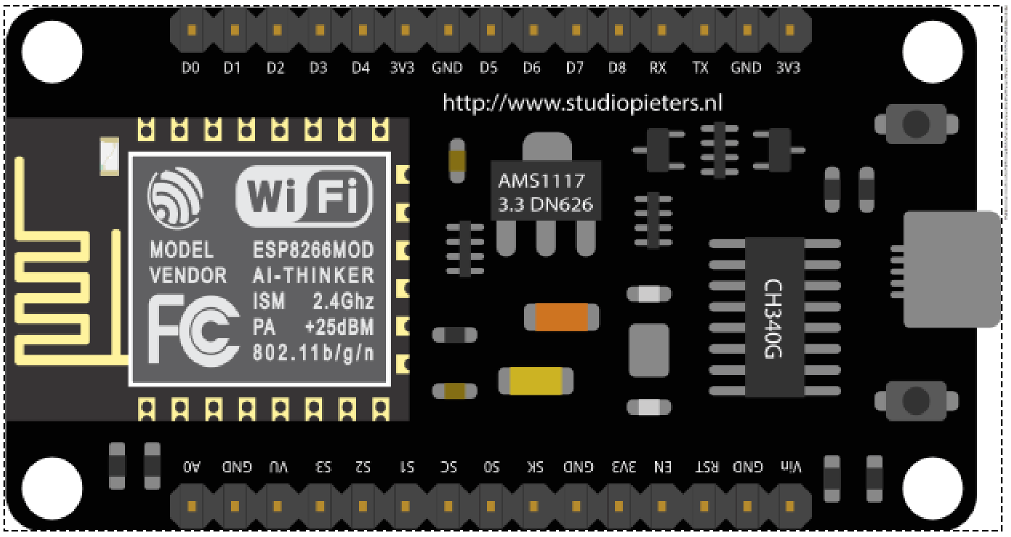


Figure 3.3.2.2.1. NodeMCU

The module discussed in the paragraph is primarily based on the ESP8266 microchip, which is a cost-effective Wi-Fi module that integrates a full TCP/IP stack and microcontroller capabilities. It was developed by Espressif Systems. The ESP8266 NodeMcu is a versatile device that combines certain features of the Arduino board with the ability to connect to the internet.

While Arduino modules and microcontrollers have been popular choices for automation projects, they lack built-in Wi-Fi capabilities. To overcome this limitation and enable internet connectivity, an external Wi-Fi protocol needs to be added to these devices.

The NodeMCU is a well-known development board that utilizes the ESP8266 Wi-Fi System-on-Chip (SoC). The version mentioned in the paragraph is version 3, based on the ESP-12E module. NodeMCU is an open-source firmware and development kit that facilitates prototyping of IoT (Internet of Things) products using LUA scripting. It can also be programmed using the Arduino IDE, providing flexibility for different programming approaches.

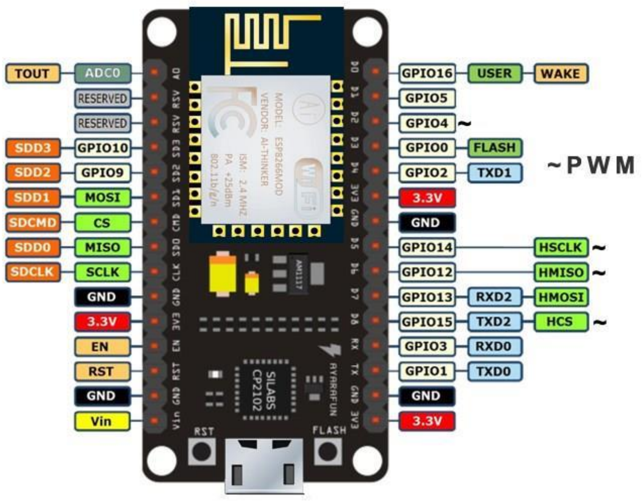


Figure 3.3.2.2.2. NodeMCU pinout

3.3.2.3 Relay

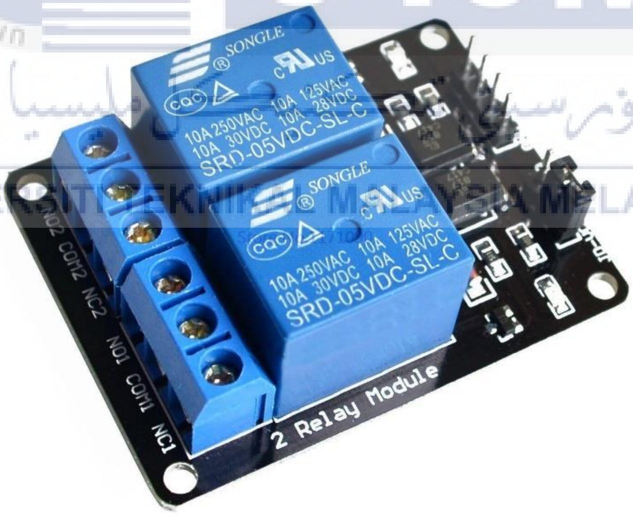


Figure 3.3.2.3.1. Two Channel Relay

A 2-channel relay module is an electronic component that allows you to control two separate circuits using a single module. It consists of two relays, each capable of switching high voltage or high current loads on and off.

The relay module typically has input pins that can be connected to a microcontroller, Arduino board, or any other digital output device. These input pins control the activation and deactivation of the relays. When the input signal is received, the relay switches its contacts, either connecting or disconnecting the load circuit.

Each channel of the relay module usually has three output pins: a normally open (NO) pin, a normally closed (NC) pin, and a common (COM) pin. The NO and COM pins are connected when the relay is activated, while the NC and COM pins are connected when the relay is inactive.

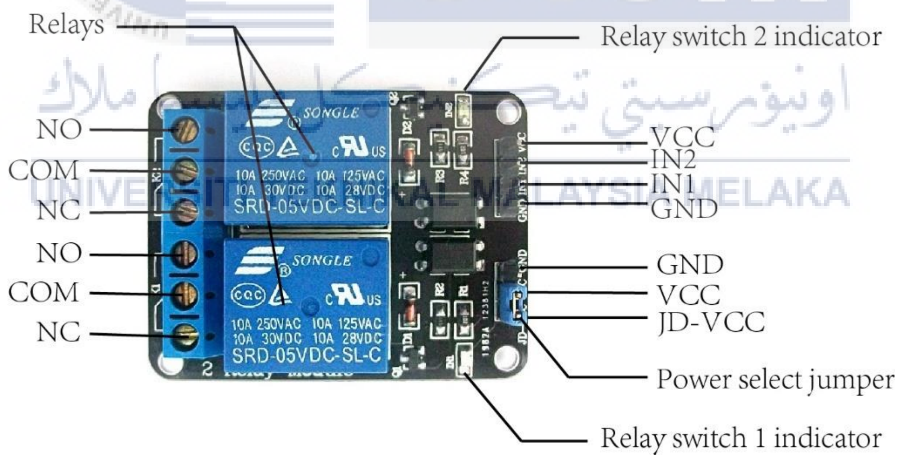


Figure 3.3.2.3.2. Relay pinout

3.3.2.4 MD30C 30A DC Motor Driver

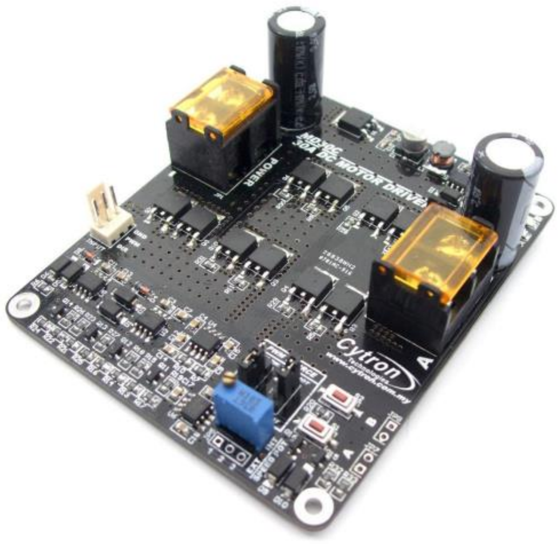


Figure 3.3.2.4.1. MD30C DC Motor Driver

Motor driver is an electronic device or circuit that controls the speed, direction, and operation of an electric motor. Motor drivers are commonly used in robotics, automation systems, and other applications where precise control of motor movement is required.

Features:

● Bi-directional control for 1 brushed DC motor.

● Motor Voltage: 5V - 25V30V.

● Maximum Current: 80A peak (1 second), 30A continuously.

● Reverse polarity protection.

● 3.3V and 5V logic level input.

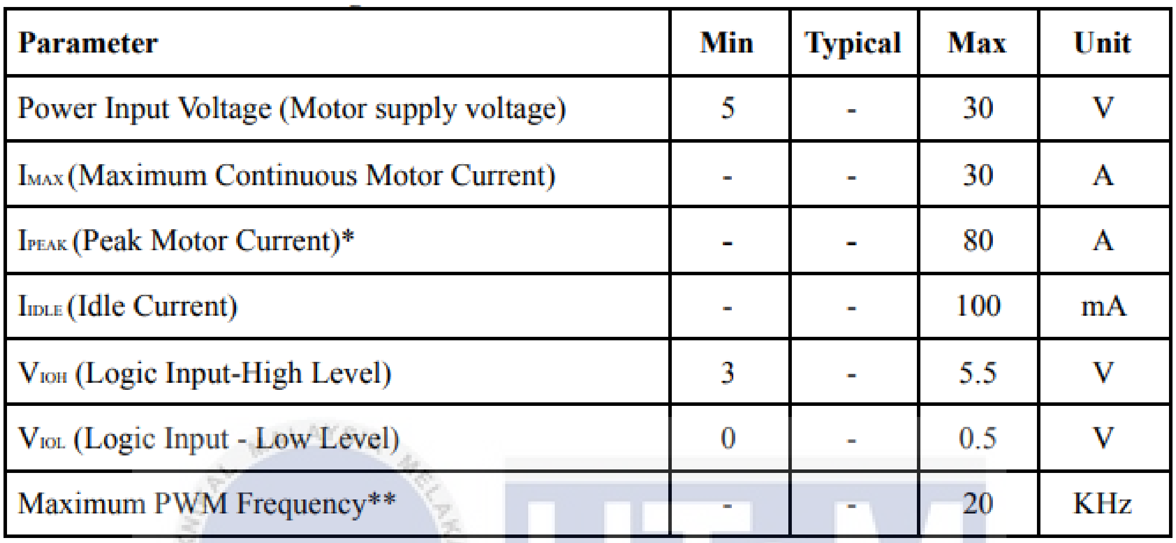
● Fully NMOS H-Bridge for better efficiency and no heatsink is required.

● Speed control PWM frequency up to 20KHz (Actual output frequency is same as input frequency when external PWM is selected).

● Onboard PWM generator with switches and potentiometer for standalone operation.

● Support both locked-antiphase and sign-magnitude for external PWM operation.

Table 3.3.2.4-1. Motor driver MD30C



Parameter	Min	Typical	Max	Unit
Power Input Voltage (Motor supply voltage)	5	-	30	V
I_{MAX} (Maximum Continuous Motor Current)	-	-	30	A
I_{PEAK} (Peak Motor Current)*	-	-	80	A
I_{IDLE} (Idle Current)	-	-	100	mA
V_{IOH} (Logic Input-High Level)	3	-	5.5	V
V_{IOL} (Logic Input - Low Level)	0	-	0.5	V
Maximum PWM Frequency**	-	-	20	KHz

3.4 Milestone 2: Software development

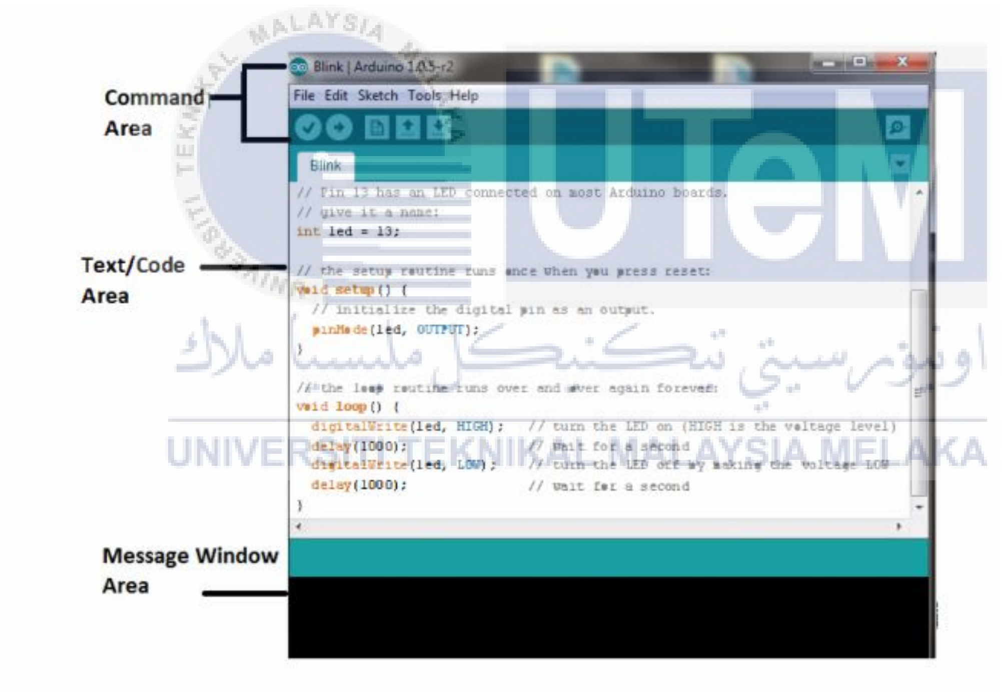
Within this part, milestone 2 is structured by making reference to the second aim within the methodology, which begins with the development of a fundamental program that is compatible with the circuit that was established in the previous milestone. Following the conclusion of the fundamental test, a comprehensive software program together with the circuit configuration is built, and the flowchart for the second milestone of the project is shown below.

3.4.1 Integrated Development Environment (IDE)

Integrated Development Environment is referred to as IDE. It is a piece of software that offers many capabilities and tools to help with software development. To speed up the development process, an IDE often comes with a source code editor, a compiler or interpreter, debugging tools, and other utilities.

3.4.1.1 Arduino CC

The software is a set of instructions that informs the hardware of what to do and how to do it. The Arduino IDE (Integrated Development Environment) is divided into three main parts:



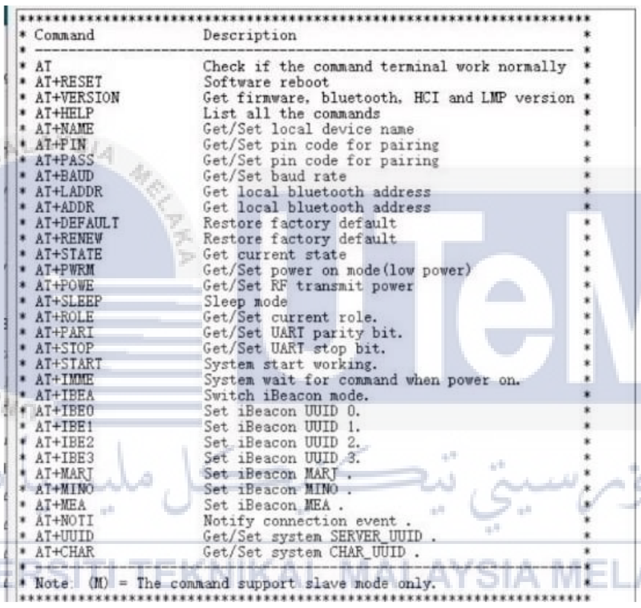
```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Figure 3.4.1.1.1. Arduino.cc Interface

3.4.2 Program development

3.4.2.1 AT-09 Bluetooth Module

The At-09 bluetooth module is configured to the desire setting and preferences. In order to change the setting of the Bluetooth module, the Bluetooth Module needs to be connected to Arduino board in order to enter the AT mode.



```
*********************************************************************
* Command               Description                                 *
* ----------------------------------------------------------------- *
* AT                    Check if the command terminal work normally *
* AT+RESET              Software reboot                             *
* AT+VERSION            Get firmware, bluetooth, HCI and LMP version *
* AT+HELP               List all the commands                       *
* AT+NAME               Get/Set local device name                   *
* AT+PIN                Get/Set pin code for pairing                *
* AT+PASS               Get/Set pin code for pairing                *
* AT+BAUD               Get/Set baud rate                           *
* AT+LADDR              Get local bluetooth address                 *
* AT+ADDR               Get local bluetooth address                 *
* AT+DEFAULT            Restore factory default                     *
* AT+RENEW              Restore factory default                     *
* AT+STATE              Get current state                           *
* AT+PWRM               Get/Set power on mode(low power)            *
* AT+POWE               Get/Set RF transmit power                   *
* AT+SLEEP              Sleep mode                                  *
* AT+ROLE               Get/Set current role.                       *
* AT+PARI               Get/Set UART parity bit.                    *
* AT+STOP               Get/Set UART stop bit.                      *
* AT+START              System start working.                       *
* AT+IMME               System wait for command when power on.      *
* AT+IBEA               Switch iBeacon mode.                        *
* AT+IBE0               Set iBeacon UUID 0.                         *
* AT+IBE1               Set iBeacon UUID 1.                         *
* AT+IBE2               Set iBeacon UUID 2.                         *
* AT+IBE3               Set iBeacon UUID 3.                         *
* AT+MARJ               Set iBeacon MARJ .                          *
* AT+MINO               Set iBeacon MINO .                          *
* AT+MEA                Set iBeacon MEA .                           *
* AT+NOTI               Notify connection event .                   *
* AT+UUID               Get/Set system SERVER_UUID .                *
* AT+CHAR               Get/Set system CHAR_UUID .                  *
* ----------------------------------------------------------------- *
* Note: (M) = The command support slave mode only.                  *
*********************************************************************
```

Figure 3.4.2.1.1 Sample of AT command

The VCC of the Bluetooth Module is connected to 5V of the Arduino. Ground pin of the Bluetooth module to the ground on the Arduino board. It is important to connect the RX and TX pin of the Bluetooth module vise versa with the Arduino RX and TX pin.

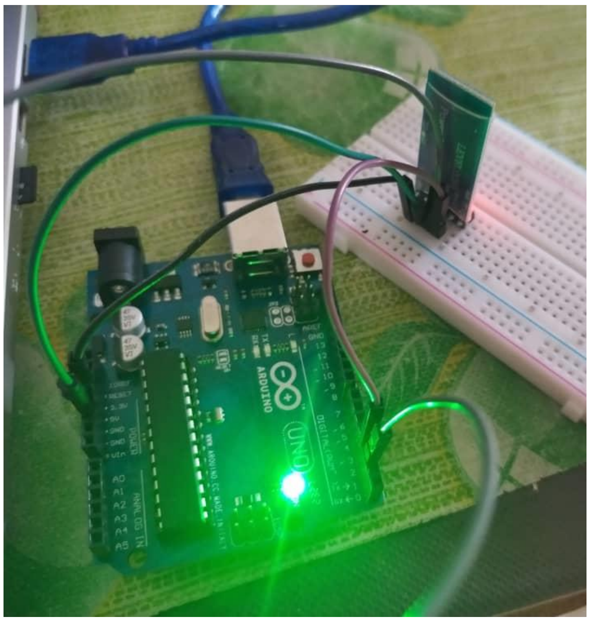
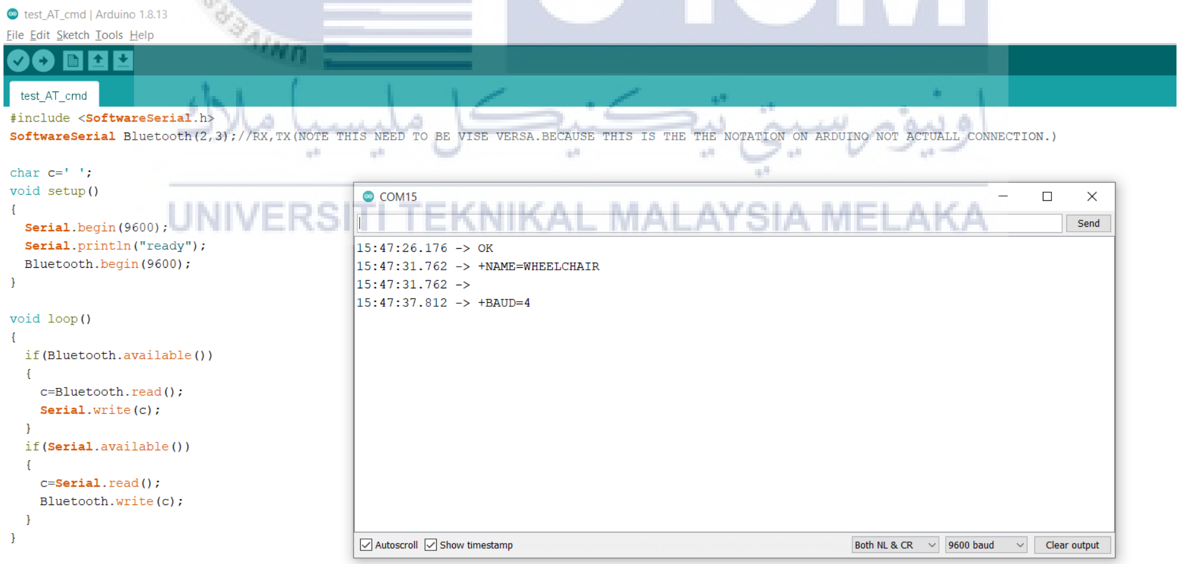


Figure 3.4.2.1.2 Arduino Connection With Hc-05

Next the code is the uploaded to the Arduino board.



```
#include <SoftwareSerial.h>
SoftwareSerial Bluetooth(2,3);//RX,TX(NOTE THIS NEED TO BE VISE VERSA.BECAUSE THIS IS THE THE NOTATION ON ARDUINO NOT ACTUALL CONNECTION.)

char c=' ';
void setup()
{
  Serial.begin(9600);
  Serial.println("ready");
  Bluetooth.begin(9600);
}

void loop()
{
  if(Bluetooth.available())
  {
    c=Bluetooth.read();
    Serial.write(c);
  }
  if(Serial.available())
  {
    c=Serial.read();
    Bluetooth.write(c);
  }
}
```

```
15:47:26.176 -> OK
15:47:31.762 -> +NAME=WHEELCHAIR
15:47:31.762 ->
15:47:37.812 -> +BAUD=4
```

Figure 3.4.2.1.3 Serial Monitor Output

The command that need to be executed in the serial monitor is:

- AT
- AT+NAME
- AT+BAUD

This is use to get the Bluetooth module information. “AT” is used to check whether the Bluetooth module is in good condition. As shown above the serial monitor show “ OK”. “AT+NAME” is use to get the Bluetooth module name and the “AT+BAUD” is use to get the baud rate of the Bluetooth module.

- AT+NAMEWHEELCHAIR

In this case the only changes done is changing the Bluetooth module name .The default name is “BT-05” , after entering the “AT+NAMEWHEELCHAIR” ,the Bluetooth module name is now “WHEELCHAIR”. The baud rate use is “9600”.

3.4.2.2 HC-05 Bluetooth module Master and slave

This setting is done to the HC-05 Bluetooth Module to allow the communication between the two nodeMCU board.

For the slave configuration the AT command used is:

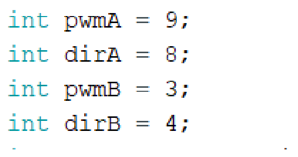
- AT+ROLE=0(To set it as slave)
- AT+UART=38400,0,0 (To fix the baud rate at 38400)
- AT+ADDR(To get the address of this HC-05, the address will be used during master configuration.)

To set the Bluetooth module to Master mode, the AT command used is :

- AT+ROLE=1(To set it as master)
- AT+UART=38400,0,0 (To fix the baud rate at 38400)
- AT+BIND=xxxx,xx,xxxxx(Input the address of the slave Bluetooth module)

3.4.2.3 Arduino coding for controlling motor

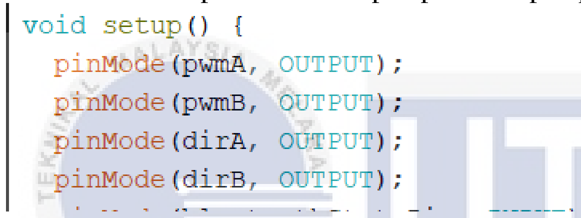
The pin of the PWM and Direction of the motor driver is declare.



```
int pwmA = 9;
int dirA = 8;
int pwmB = 3;
int dirB = 4;
```

Figure 3.4.2.3.1. Sample Of Arduino Code

The pin is the need to setup either as ouput pin or input pin.



```
void setup() {
  pinMode(pwmA, OUTPUT);
  pinMode(pwmB, OUTPUT);
  pinMode(dirA, OUTPUT);
  pinMode(dirB, OUTPUT);
```

Figure 3.4.2.3.2 Arduino Code For Setup

The “moveBasedOnCommand” is use to executed the Char receive from the bluetooth module. The Table 3.4.2.3 1 Expected Output below show the coressponding char that it will receive .

Table 3.4.2.3-1 Expected Output

CHARACTHER RECEIVE	OUTPUT
F	moveForward
B	moveBackward
R	turnRight
L	turnLeft
Y(voice char)	MoveForward
H(voice char)	moveBackward
U(voice char)	turnRight
T(voice char)	turnLeft
S	Stop

```
void moveBasedOnCommand() {
  // Process commands based on t and use the received PWM value as needed
  if (t == 'F') {
    moveForward();
  } else if (t == 'B') {
    moveBackward();
  } else if (t == 'R') {
    turnRight();
  } else if (t == 'L') {
    turnLeft();
  } else if (t == 'Y') { // Voice command: Forward
    moveForward();
  } else if (t == 'H') { // Voice command: Backward
    moveBackward();
  } else if (t == 'U') { // Voice command: Right
    turnRight();
  } else if (t == 'T') { // Voice command: Left
    turnLeft();
  } else if (t == 'S') {
    stopMotors();
  }
}

void stopMotors() {
  analogWrite(pwmA, 0);
  analogWrite(pwmB, 0);
}
```

Figure 3.4.2.3.3 Arduino code For Character Received

The PWM of is determine by the slider position on the mobile apps.

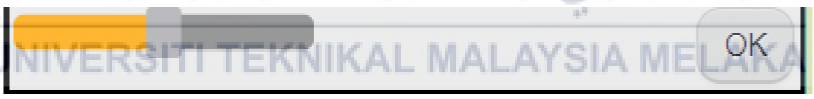
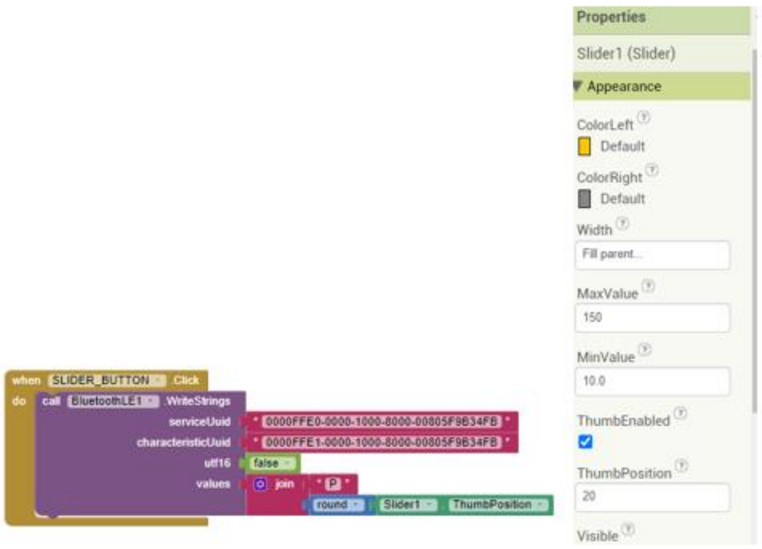


Figure 3.4.2.3.4 Slider On Mobile Apps



```
when SLIDER_BUTTON .Click
do  call BluetoothLE1 .WriteStrings
                 serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
          characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                       utf16  false
                      values  join  "P"
                                    round  Slider1 . ThumbPosition
```

Properties

Slider1 (Slider)

Appearance

ColorLeft	🟨 Default
ColorRight	⬛ Default
Width	Fill parent...
MaxValue	150
MinValue	10.0
ThumbEnabled	✓
ThumbPosition	20
Visible	

Figure 3.4.2.3.5 Mit App Code Block For Slider

The max value set on the slider properties is set at 150 and the min value is 20.When the "OK" button is click the value of the slider is send to the Arduino via Bluetooth .For example, the value send is "P90".

```
// Check if the received command is for PWM adjustment
if (t == 'P') {
  // Handle PWM value
  receivedPWMValue = bluetooth.parseInt();
  Serial.print("Received PWM value: ");
  Serial.println(receivedPWMValue);
} else {
  // Process other commands immediately
  processCommands();
}
```

Figure 3.4.2.3.6 Arduino Code For PWM Value Received

The value that is received from the mobile apps it then convert to real value of the PWM. For example the value received is “P90”. The conditional statement checks if a variable ‘t’ is equal to the character ‘P’. The ‘parseInt()’ function to read the integer value . The ‘parseInt()’ used to read and parse an integer from a stream of characters. For example , the received stream characters is ‘P90’, the function basically to extract an integer from the stream and the integer value ‘90’ is assign to the ‘receivedPWM’.

3.4.2.4 NodeMCU with master HC-05

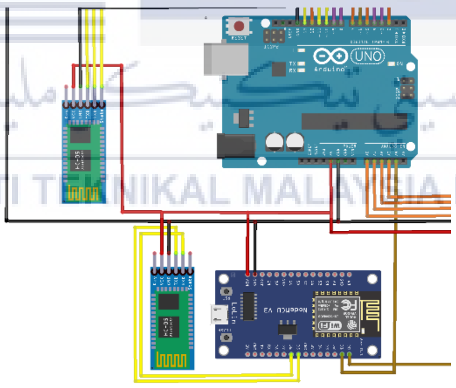
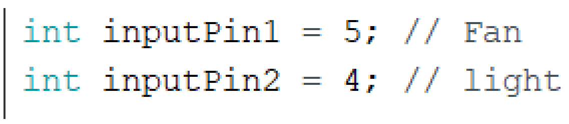


Figure 3.4.2.4.1 NodeMCU Wiring Diagram

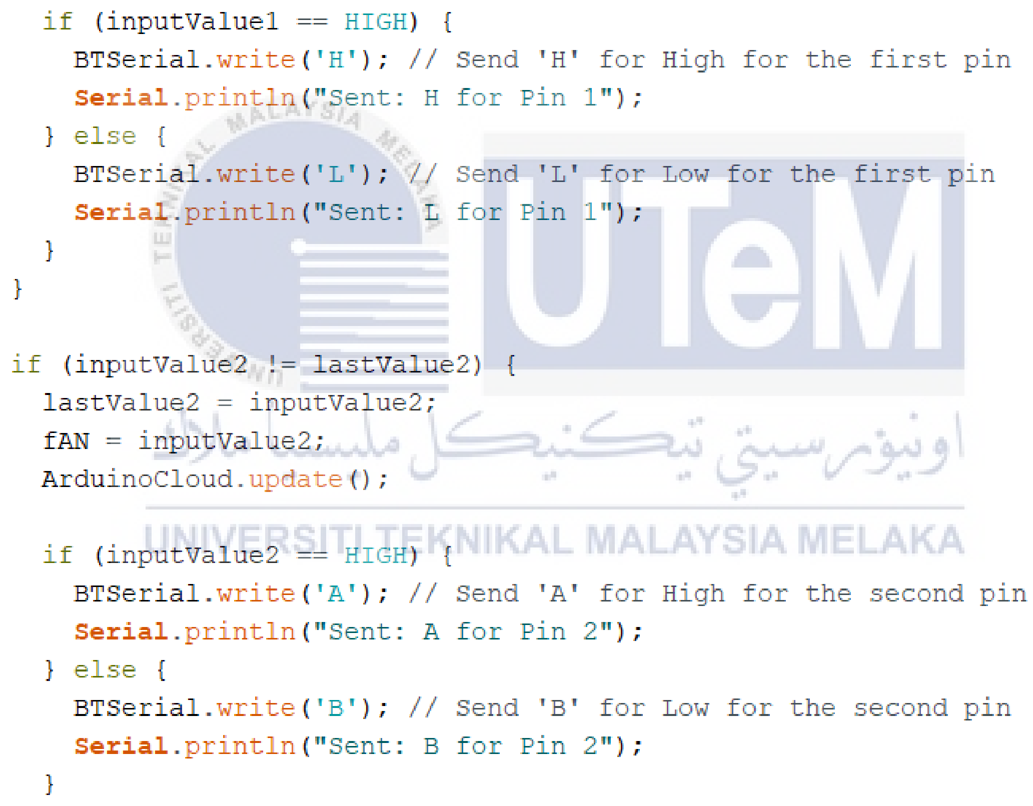
The NodeMCU that is connected with the master HC-05 is used to communicate with another NodeMCU that are connected with the slave HC-05. The NodeMCU that are connected with the slave HC-05 is used to turn on and off electrical appliance.



```
int inputPin1 = 5; // Fan
int inputPin2 = 4; // light
```

Figure 3.4.2.4.2 NodeMCU Code For Declaring Pin

The pin 4 and 5 is declare as ‘inputPin1’ and ‘inputPin2’ .This pin is used to detect ‘HIGH’ or ‘LOW’ from the Arduino pin.



```
    if (inputValue1 == HIGH) {
      BTSerial.write('H'); // Send 'H' for High for the first pin
      Serial.println("Sent: H for Pin 1");
    } else {
      BTSerial.write('L'); // Send 'L' for Low for the first pin
      Serial.println("Sent: L for Pin 1");
    }
  }

  if (inputValue2 != lastValue2) {
    lastValue2 = inputValue2;
    fAN = inputValue2;
    ArduinoCloud.update();

    if (inputValue2 == HIGH) {
      BTSerial.write('A'); // Send 'A' for High for the second pin
      Serial.println("Sent: A for Pin 2");
    } else {
      BTSerial.write('B'); // Send 'B' for Low for the second pin
      Serial.println("Sent: B for Pin 2");
    }
```

Figure 3.4.2.4.3 NoceMCU Code For Controlling Power Outlet

This code block is checking the state of ‘inputValue1’, If inputValue1 is ‘HIGH’ it sends the character 'H' over the Bluetooth serial connection using ‘BTSerial.write() ‘ to the nodeMCU that is connected with the slave HC-05. If inputValue1 is ‘LOW’ it sends the character 'L' over the Bluetooth serial connection using ‘BTSerial.write() ‘ to the nodeMCU that is connected with the slave HC-05.

Table 3.4.2.4-1 Arduino Pin 4 State and Character Send

Pin 4 State	Character Send
HIGH	H
LOW	L

Table 3.4.2.4-2 Arduino Pin 5 State and Character Send

Pin 5 State	Character Send
HIGH	A
LOW	B

3.4.2.5 NodeMCU with Slave HC-05

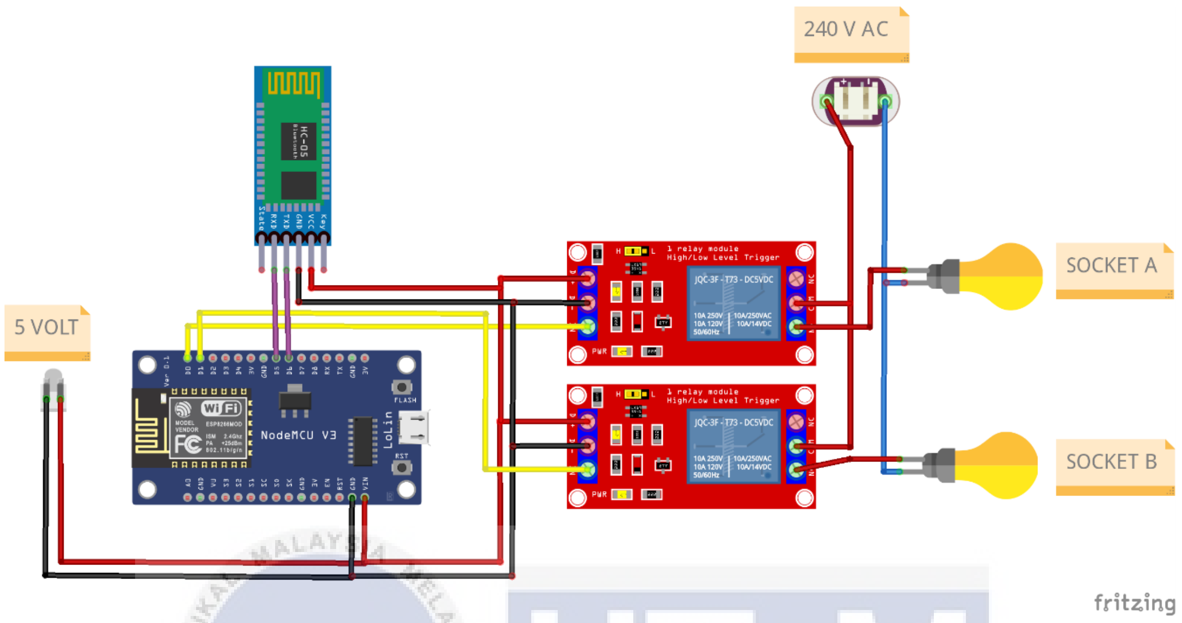
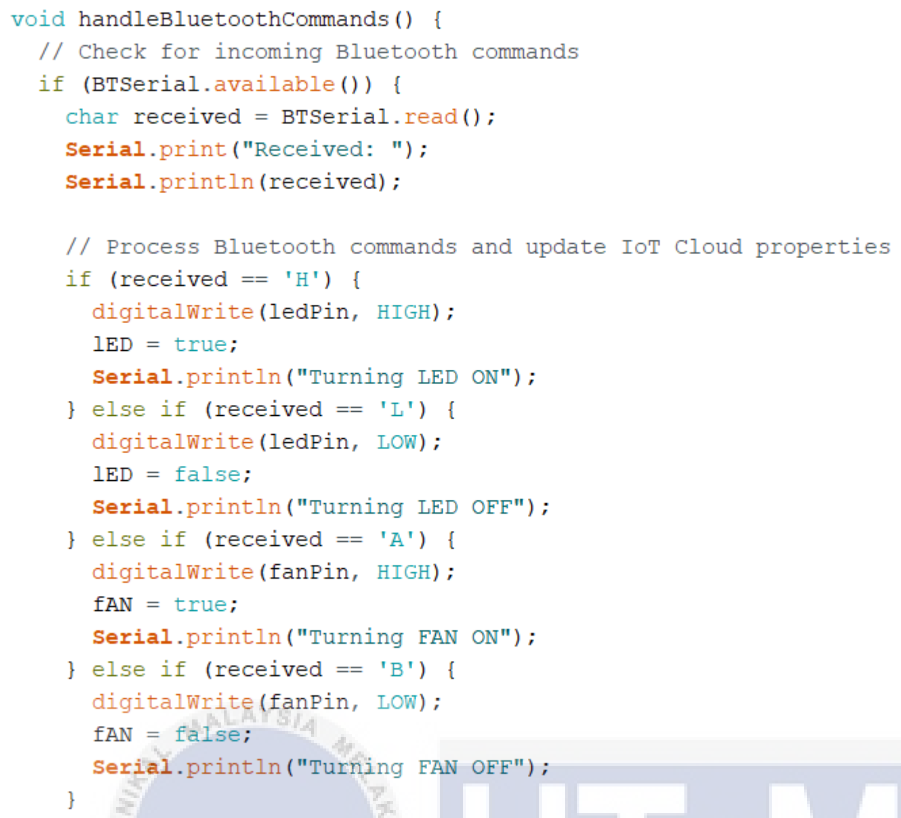


Figure 3.4.2.5.1 NodeMCU With HC-05 Slave Wiring Diagram

The second NodeMCU, connected to a slave HC-05 Bluetooth module, acts as a remote control for turning electrical appliances on and off. It communicates wirelessly with another NodeMCU connected to a master HC-05 Bluetooth module. When the second NodeMCU receives signals from the master, it interprets them as commands to control the appliances.



```
void handleBluetoothCommands() {
  // Check for incoming Bluetooth commands
  if (BTSerial.available()) {
    char received = BTSerial.read();
    Serial.print("Received: ");
    Serial.println(received);

    // Process Bluetooth commands and update IoT Cloud properties
    if (received == 'H') {
      digitalWrite(ledPin, HIGH);
      lED = true;
      Serial.println("Turning LED ON");
    } else if (received == 'L') {
      digitalWrite(ledPin, LOW);
      lED = false;
      Serial.println("Turning LED OFF");
    } else if (received == 'A') {
      digitalWrite(fanPin, HIGH);
      fAN = true;
      Serial.println("Turning FAN ON");
    } else if (received == 'B') {
      digitalWrite(fanPin, LOW);
      fAN = false;
      Serial.println("Turning FAN OFF");
    }
```

Figure 3.4.2.5.2 NodeMCU Code For Received Character

The ‘handleBluetoothCommands’ function in this Arduino code is designed to process incoming commands received over a Bluetooth connection. It first checks if there are any available characters in the Bluetooth serial buffer using ‘BTSerial.available()’. If there is data, it reads a character from the buffer using ‘BTSerial.read()’ and stores it in the variable received. The function then proceeds to interpret the received character to control the state of connected devices.

Table 3.4.2.5-1 NodeMCU With Slave HC-05 Pin 4 State

Character Received	Pin 4 State
H	HIGH
L	LOW

Table 3.4.2.5-2 NodeMCU With Slave HC-05 Pin 5 State

Character Received	Pin 5 State
A	HIGH
B	LOW

If the received character is 'H', it turns on an LED connected to the ledPin, sets a corresponding Boolean variable lED to true, and prints a message indicating that the LED is turning ON. Conversely, if the received character is 'L', it turns off the LED, sets lED to false, and prints a message indicating that the LED is turning OFF.

Similarly, if the received character is 'A', it turns on a fan connected to the fanPin, sets a Boolean variable fAN to true, and prints a message indicating that the fan is turning ON. If the character is 'B', it turns off the fan, sets fAN to false, and prints a message indicating that the fan is turning OFF.

3.5 Milestone 3 :Prototype development

The third milestone is established so that the after the hardware and software testing the component can be tested on a mini robot to allow further improvement to the system. By doing this method it can save time and resource before assembling the component to a wheelchair. Lastly is the design of the housing prototype for all the component. Repeated work on this particular aspect of the project is required in order to guarantee the highest quality end result possible for the design.

3.6 Milestone 4 : Final Product Development

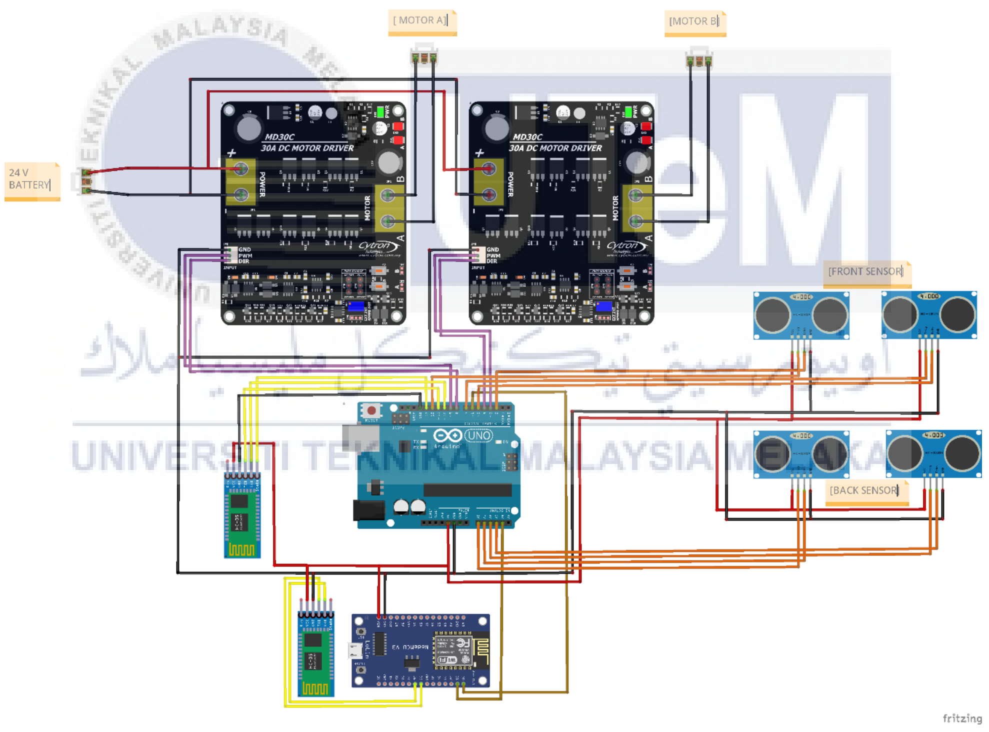


Figure 3.6.1 Full Circuit Diagram For wheechair Control

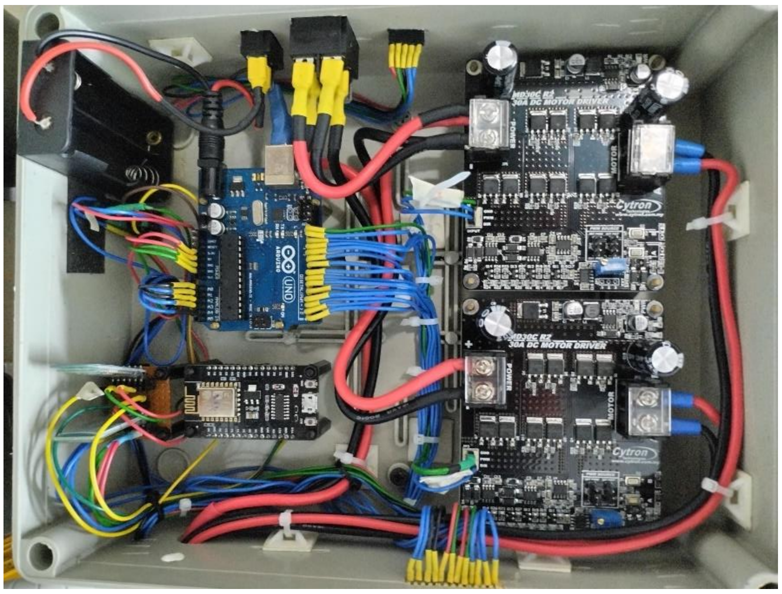


Figure 3.6.2 Layout Of The Control Box

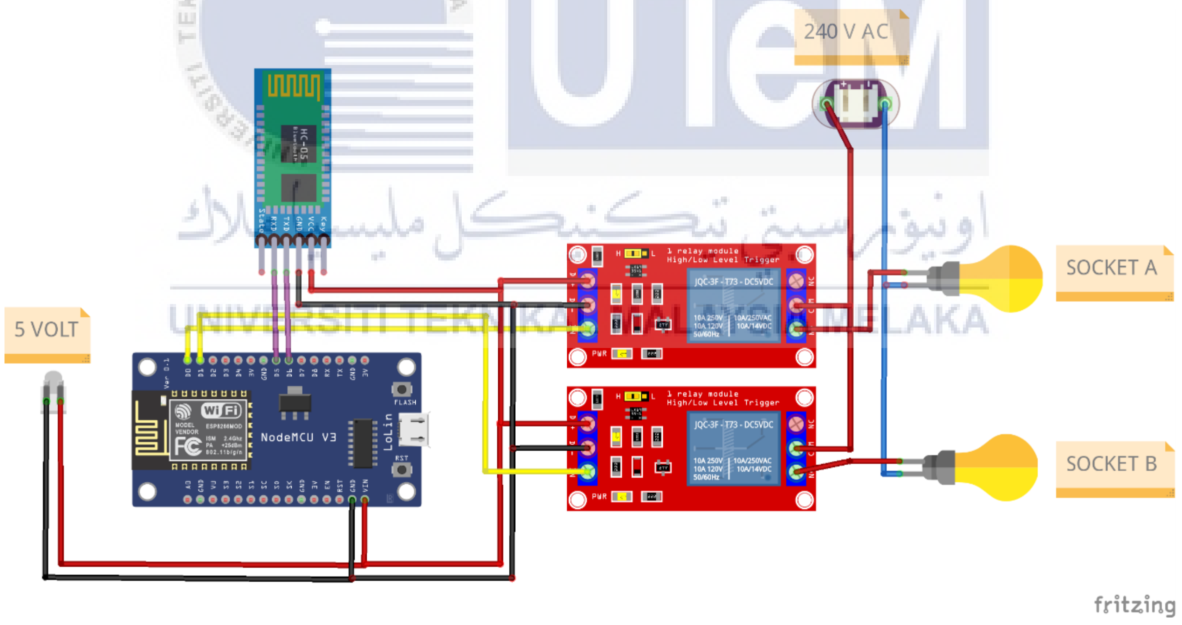


Figure 3.6.3 Wiring Diagram For Power Outlet Control

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

In this chapter, the outcomes and analysis of the Development of Wireless Control for Retrofit Wheelchair System Using Arduino-based microcontroller with IoT are presented, and the chapter also examines all of the data that is necessary to establish the effectiveness of the project. The analysis are divided into two part , software analysis and hardware analysis.

4.2 Software analysis

This subchapter contains the related graphical user interface ,Iot implementation , related coding for Arduino and nodeMCU

4.2.1 Graphical user interface (GUI)

A graphical user interface (GUI) is type of user interface that allow users to interact with a computer or other electronic device using graphical elements, such as icons ,button, or image rather than the text-based commands.

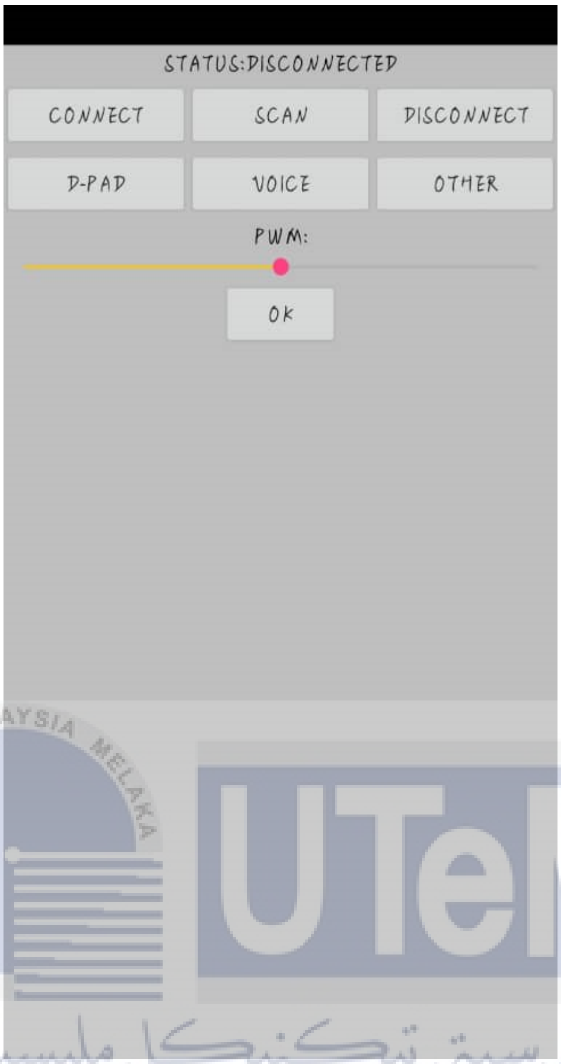


Figure 4.2.1.1 Main Layout When Apps Is Open

The following is the layout of the system's mobile app: When users activate the app, they are met with a simple and straightforward home screen. The primary navigation is clearly displayed, making it simple for users to reach crucial functionality. The layout has a user-friendly interface with a simple design to improve clarity and simplicity of usage.

Table 4.2.1-1 Mobile Apps Icon Description

Icon	Description
CONNECT	To connect to the selected Bluetooth device
SCAN	To scan available Bluetooth Device
DISCONNECT	To disconnect with the current Bluetooth device
D-PAD	To show the D-pad layout
VOICE	To show the voice layout
OTHER	To show the layout for controlling power outlet
SLIDER	To set the desire speed or pwm value

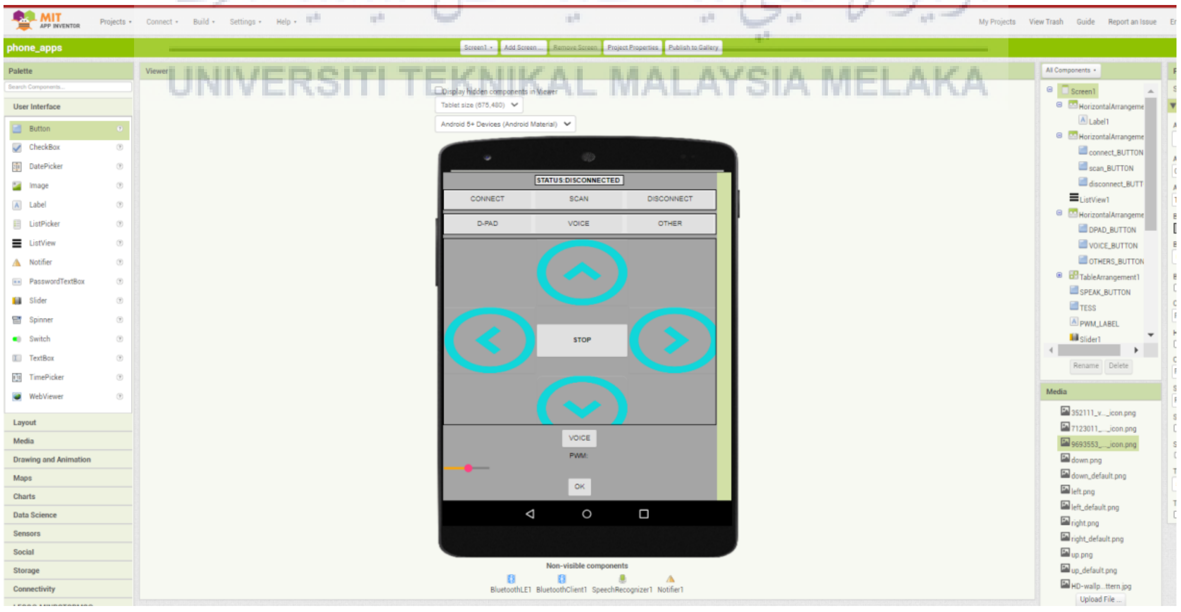


Figure 4.2.1.2 Mit App Developer Layout

4.2.2 D-pad Layout

D-pad or directional pad design is used for the layout design. The circular directional button represent the direction of the wheelchair movement.

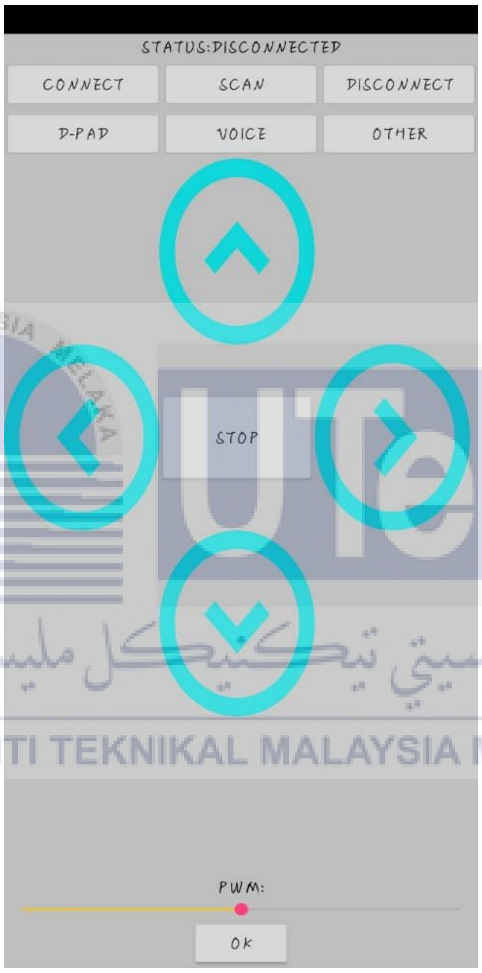
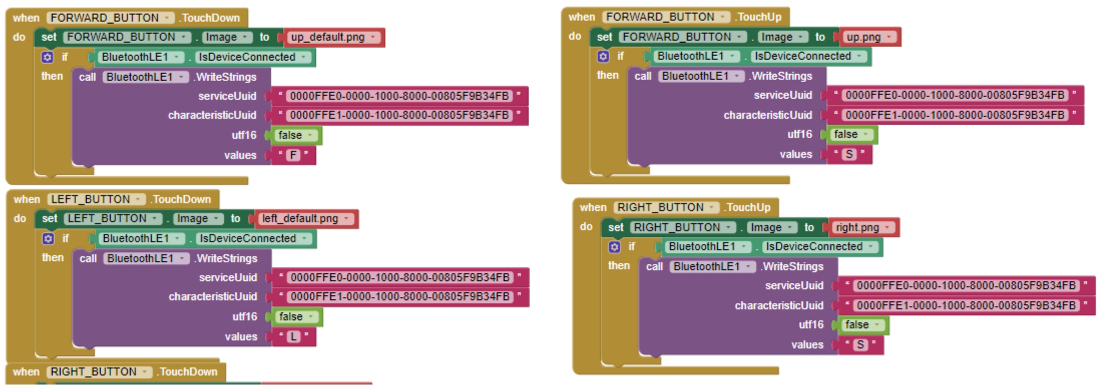


Figure 4.2.2.1 D-pad Control Layout



```
when FORWARD_BUTTON .TouchDown
do  set FORWARD_BUTTON . Image to "up_default.png"
    if   BluetoothLE1 . IsDeviceConnected
    then call BluetoothLE1 .WriteStrings
                   serviceUuid  " 0000FFE0-0000-1000-8000-00805F9B34FB "
            characteristicUuid  " 0000FFE1-0000-1000-8000-00805F9B34FB "
                         utf16  false
                        values  " F "

when LEFT_BUTTON .TouchDown
do  set LEFT_BUTTON . Image to "left_default.png"
    if   BluetoothLE1 . IsDeviceConnected
    then call BluetoothLE1 .WriteStrings
                   serviceUuid  " 0000FFE0-0000-1000-8000-00805F9B34FB "
            characteristicUuid  " 0000FFE1-0000-1000-8000-00805F9B34FB "
                         utf16  false
                        values  " L "

when RIGHT_BUTTON .TouchDown

when FORWARD_BUTTON .TouchUp
do  set FORWARD_BUTTON . Image to "up.png"
    if   BluetoothLE1 . IsDeviceConnected
    then call BluetoothLE1 .WriteStrings
                   serviceUuid  " 0000FFE0-0000-1000-8000-00805F9B34FB "
            characteristicUuid  " 0000FFE1-0000-1000-8000-00805F9B34FB "
                         utf16  false
                        values  " S "

when RIGHT_BUTTON .TouchUp
do  set RIGHT_BUTTON . Image to "right.png"
    if   BluetoothLE1 . IsDeviceConnected
    then call BluetoothLE1 .WriteStrings
                   serviceUuid  " 0000FFE0-0000-1000-8000-00805F9B34FB "
            characteristicUuid  " 0000FFE1-0000-1000-8000-00805F9B34FB "
                         utf16  false
                        values  " S "
```

Figure 4.2.2.2 Sample Of Mit App Code Block For D-pad Button

```
void moveBasedOnCommand() {
  // Process commands based on t and use the received PWM value as needed
  if (t == 'F') {
    moveForward();
  } else if (t == 'B') {
    moveBackward();
  } else if (t == 'R') {
    turnRight();
  } else if (t == 'L') {
    turnLeft();
  } else if (t == 'Y') { // Voice command: Forward
    moveForward();
  } else if (t == 'H') { // Voice command: Backward
    moveBackward();
  } else if (t == 'U') { // Voice command: Right
    turnRight();
  } else if (t == 'T') { // Voice command: Left
    turnLeft();
  } else if (t == 'S') {
    stopMotors();
  }
}
```

Figure 4.2.2.3 Arduino Code For Received Character

4.2.3 Voice layout

The voice layout uses speech to text recognition to determine the corresponding commands or inputs. The spoken words are transcribed into text . As user speak , the system process the audio input, converts it into written text , and then analyzes the text to identify the intended command.

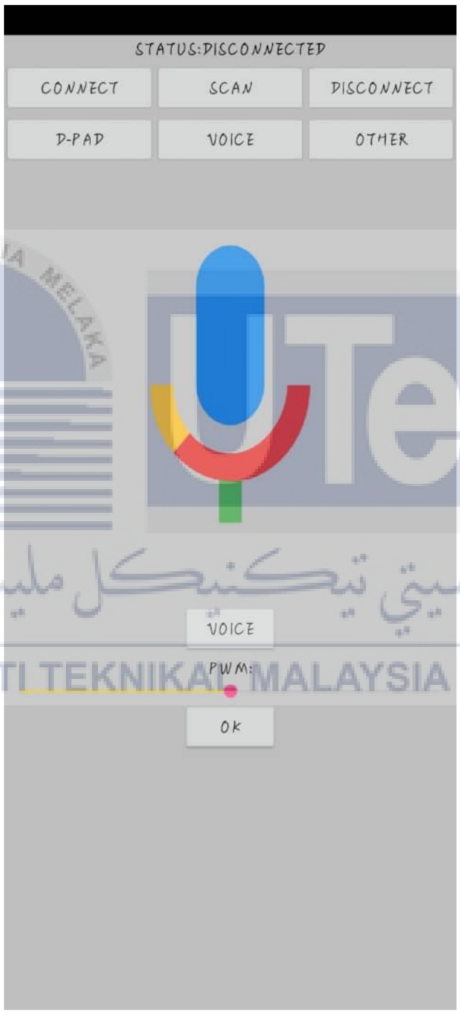
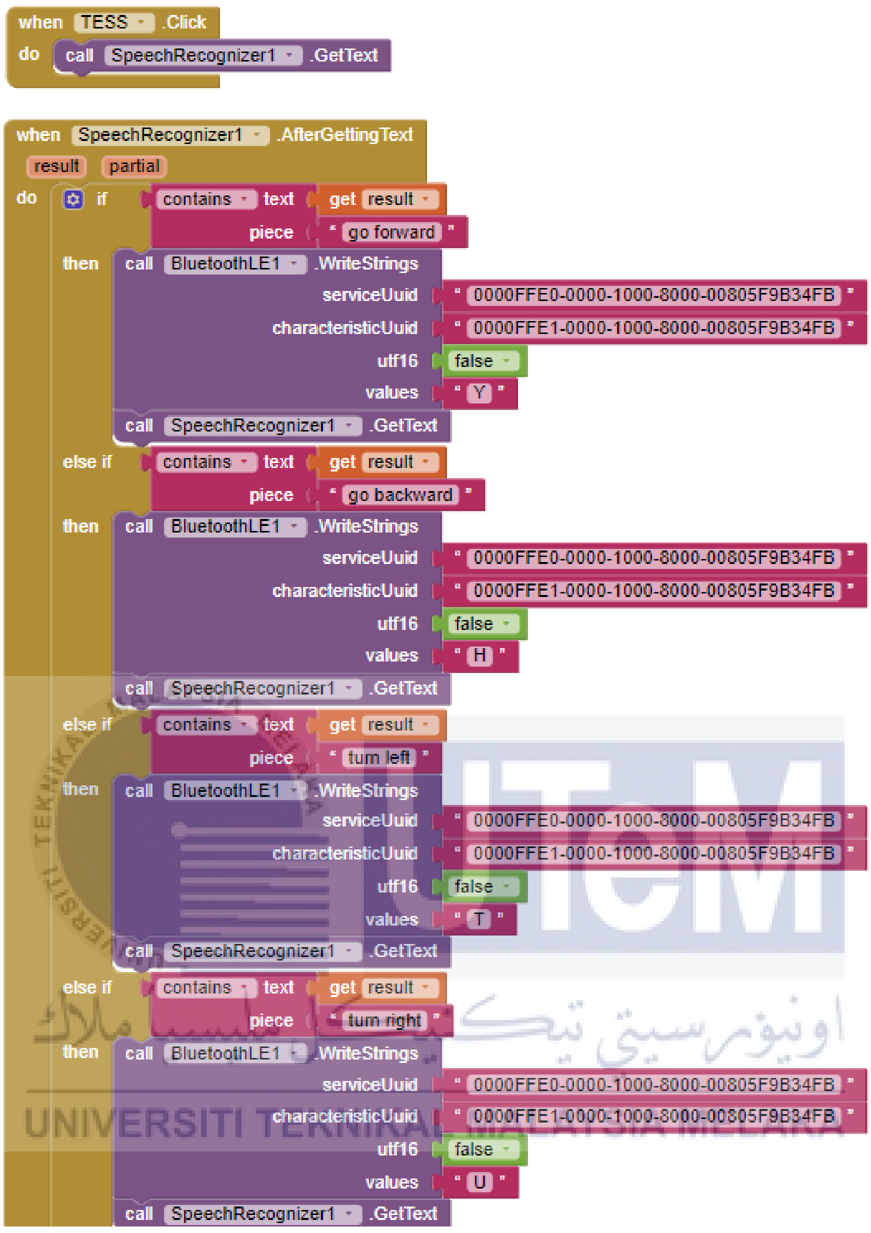


Figure 4.2.3.1 Voice Layout On Mobile App



```
when TESS .Click
do  call SpeechRecognizer1 .GetText

when SpeechRecognizer1 .AfterGettingText
  result  partial
do  if   contains text  get result
                  piece " go forward "
    then call BluetoothLE1 .WriteStrings
                        serviceUuid  " 0000FFE0-0000-1000-8000-00805F9B34FB "
                 characteristicUuid  " 0000FFE1-0000-1000-8000-00805F9B34FB "
                              utf16  false
                             values  " Y "
         call SpeechRecognizer1 .GetText
    else if contains text  get result
                  piece " go backward "
    then call BluetoothLE1 .WriteStrings
                        serviceUuid  " 0000FFE0-0000-1000-8000-00805F9B34FB "
                 characteristicUuid  " 0000FFE1-0000-1000-8000-00805F9B34FB "
                              utf16  false
                             values  " H "
         call SpeechRecognizer1 .GetText
    else if contains text  get result
                  piece " turn left "
    then call BluetoothLE1 .WriteStrings
                        serviceUuid  " 0000FFE0-0000-1000-8000-00805F9B34FB "
                 characteristicUuid  " 0000FFE1-0000-1000-8000-00805F9B34FB "
                              utf16  false
                             values  " T "
         call SpeechRecognizer1 .GetText
    else if contains text  get result
                  piece " turn right "
    then call BluetoothLE1 .WriteStrings
                        serviceUuid  " 0000FFE0-0000-1000-8000-00805F9B34FB "
                 characteristicUuid  " 0000FFE1-0000-1000-8000-00805F9B34FB "
                              utf16  false
                             values  " U "
         call SpeechRecognizer1 .GetText
```

Figure 4.2.3.2 Sample Of Mit App Code Block For Voice Command

Table 4.2.3-1 Expected Result From The Mobile Apps Control

Voice command	Character send to Arduino
Go forward	Y
Go backward	H
Turn left	T
Turn right	U
Light on	Z
Light off	X
Fan on	J
Fan off	K
Stop	S

4.2.3.1 Voice Command Analysis

Here presents the results of the voice command analysis experiment, aimed at evaluating the accuracy of voice commands. Four individuals were included in the study, each instructed to verbally express commands such as ‘go forward’ , ‘stop’ , ‘light on’ and ‘light off’. Each command were taken for five time. The experiment was designed to evaluate the system’s proficiency in recognizing and executing spoken instruction from different individuals.

Participant	Voice Command	Correct input	Wrong input
A	Go forward	5	0
B		5	0

C		4	1
D		5	0
A	Stop	5	0
B		3	2
C		5	0
D		4	1
A	Light On	5	0
B		4	1
C		3	2
D		4	1
A	Light Off	5	0
B		3	2
C		5	0
D		4	1

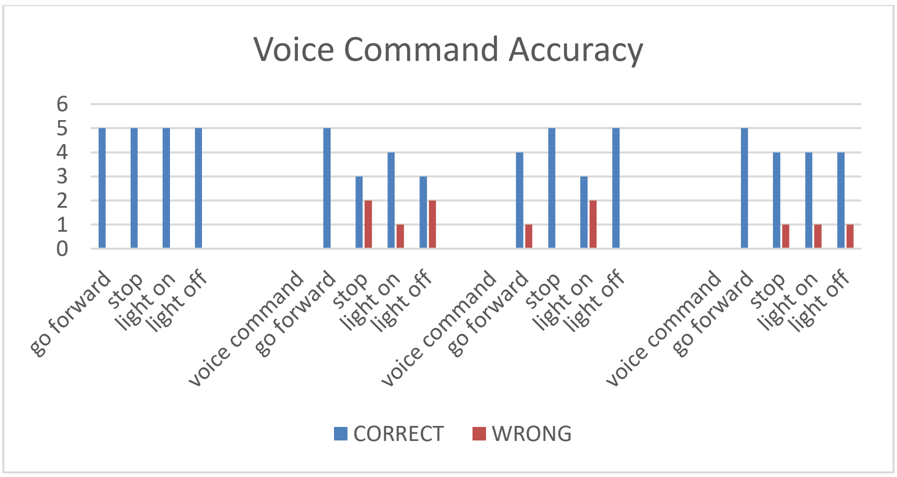


Figure 4.2.3.1.1 Graph For Voice Command Accuracy

4.2.4 Other icon layout

This show the layout for controlling the electrical outlet.

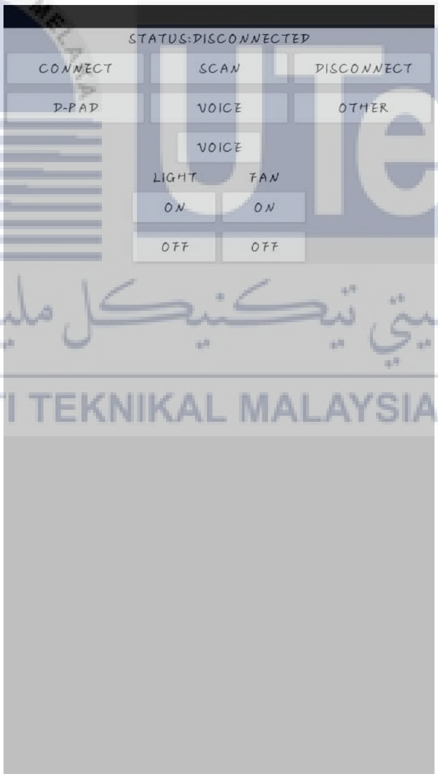
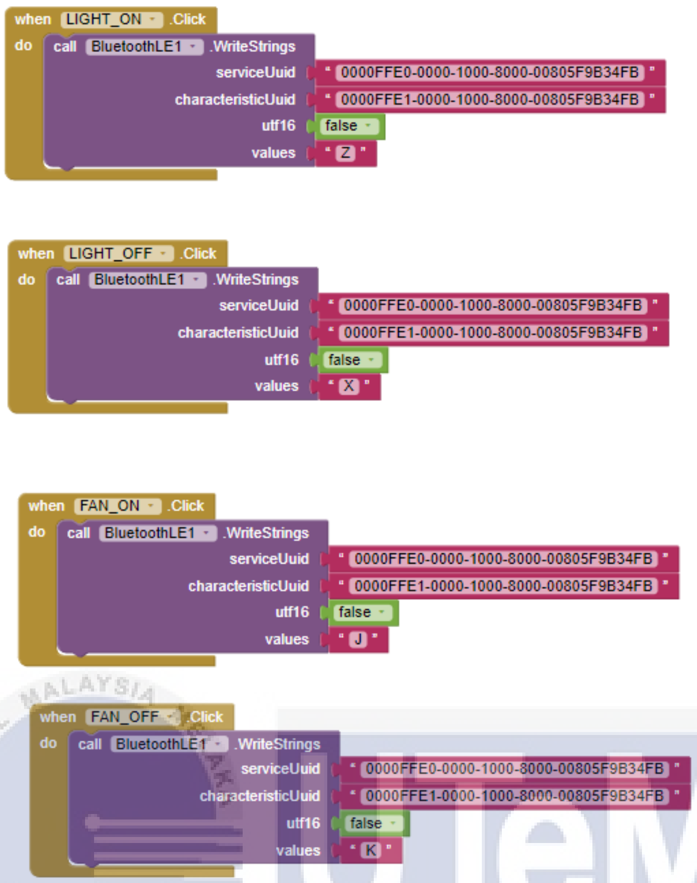


Figure 4.2.4.1 Layout For Power Outlet Control



```
when LIGHT_ON .Click
do  call BluetoothLE1 .WriteStrings
              serviceUuid  " 0000FFE0-0000-1000-8000-00805F9B34FB "
       characteristicUuid  " 0000FFE1-0000-1000-8000-00805F9B34FB "
                    utf16  false
                   values  " Z "

when LIGHT_OFF .Click
do  call BluetoothLE1 .WriteStrings
              serviceUuid  " 0000FFE0-0000-1000-8000-00805F9B34FB "
       characteristicUuid  " 0000FFE1-0000-1000-8000-00805F9B34FB "
                    utf16  false
                   values  " X "

when FAN_ON .Click
do  call BluetoothLE1 .WriteStrings
              serviceUuid  " 0000FFE0-0000-1000-8000-00805F9B34FB "
       characteristicUuid  " 0000FFE1-0000-1000-8000-00805F9B34FB "
                    utf16  false
                   values  " J "

when FAN_OFF .Click
do  call BluetoothLE1 .WriteStrings
              serviceUuid  " 0000FFE0-0000-1000-8000-00805F9B34FB "
       characteristicUuid  " 0000FFE1-0000-1000-8000-00805F9B34FB "
                    utf16  false
                   values  " K "
```

Figure 4.2.4.2 Sample Code Block Mit App For Power Outlet Control

Table 4.2.4-1 Expected Output For Light Control

Icon for light	Character send to arduino
ON	Z
OFF	X

Table 4.2.4-2 Expected Output For Fan Control

Icon for fan	Character send to arduino
ON	J
OFF	K

4.2.5 Slider icon

The slider allow the user to set the desire speed or the PWM value.

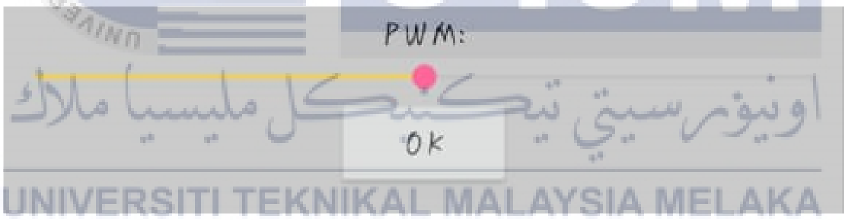
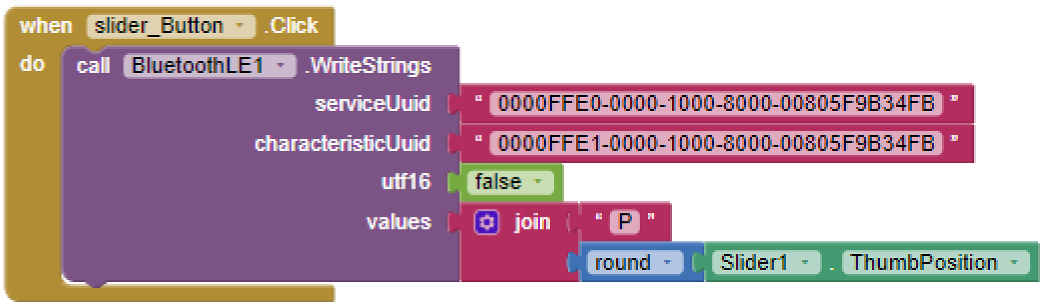
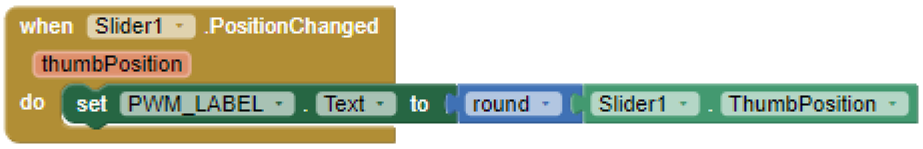
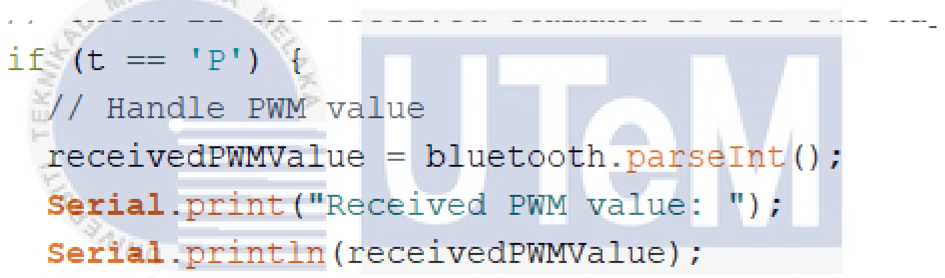


Figure 4.2.5.1 Slider Icon On Mobile App

```
when slider_Button .Click
do  call BluetoothLE1 .WriteStrings
                  serviceUuid  " 0000FFE0-0000-1000-8000-00805F9B34FB "
           characteristicUuid  " 0000FFE1-0000-1000-8000-00805F9B34FB "
                        utf16  false
                       values  join  " P "
                                     round  Slider1 . ThumbPosition

when Slider1 .PositionChanged
  thumbPosition
do  set PWM_LABEL . Text to  round  Slider1 . ThumbPosition
```

Figure 4.2.5.2 Sample Code Block Mit App For Slider



```
if (t == 'P') {
  // Handle PWM value
  receivedPWMValue = bluetooth.parseInt();
  Serial.print("Received PWM value: ");
  Serial.println(receivedPWMValue);
```

Figure 4.2.5.3 Sample Of Arduino Code For Received PWM Value

4.2.6 Arduino Cloud Dasboard

The dashboard of the Arduino cloud are used as the IOT functionality .

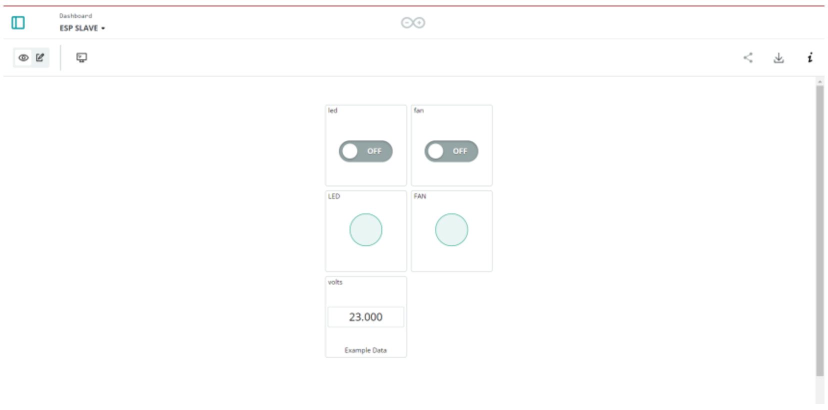
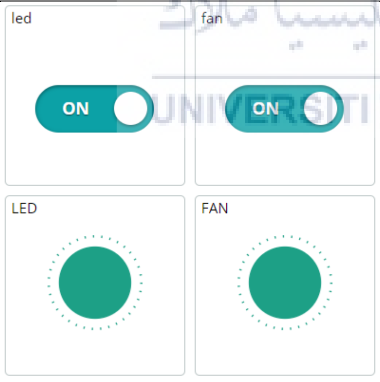
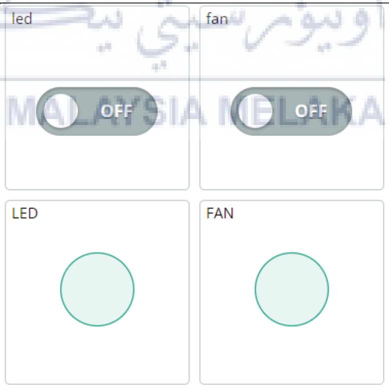


Figure 4.2.6.1 Arduino Cloud Dashboard

Table 4.2.6-1 Visual State Of The Power Outlet

On state	Off state
	

4.3 Hardware analysis

In this part , the analysis includes are the motor movement, truth table of the motor driver , battery voltage and ultrasonic analysis.

4.3.1 Movement of the motor

Based on the d-pad layout and the button that users press determine the direction of the wheelchair moving. The rotation of the motor will determine the direction whether moving forward , backward ,left or right. In order for the wheelchair to move forward, the motor orientation need to be vise versa. For example , the left motor need to turn anticlockwise and the right motor need to turn clockwise to achieve the forward movement.

Table 4.3.1-1 Button Command For The D-pad

BUTTON COMMAND	LEFT MOTOR ROTATION	RIGHT MOTOR ROTATION	CONDITION OF WHEELCHAIR
▲	ANTICLOCKWISE	CLOCKWISE	MOVE FORWARD
▼	CLOCKWISE	ANTICLOCKWISE	MOVE BACKWARD
▶	ANTICLOCKWISE	ANTICLOCWISE	TURN RIGHT
◀	CLOCKWISE	CLOCKWISE	TURN LEFT

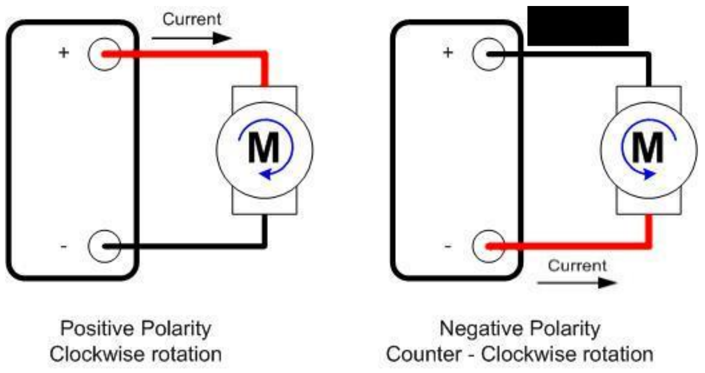


Figure 4.3.1.1 Direction Of Motor For Different Current Flow

BUTTON COMMAND	LEFT WHEEL		RIGHT WHEEL		CONDITION OF WHEELCHAIR
	PWM PIN	DIR PIN	PWM PIN	DIR PIN	
▲	1	0	1	0	MOVE FORWARD
▼	1	1	1	0	MOVE BACKWARD
▶	1	1	1	1	TURN RIGHT
◀	1	0	1	1	TURN LEFT

Figure 4.3.1.2 Truth Table For The Motor Driver

4.3.2 Battery Voltage Improvement

For the battery voltage used for this system it uses a 24 volt battery where as the previous version uses a 12 volt battery . The significant changes for this setup is that it require a lower Pulse Width Modulation (PWM) to control the wheelchair’s movement. The shift to a 24-volt battery means that a lower Pulse Width Modulation (PWM) signal is sufficient to achieve the desired motor control.

The lower PWM requirement is likely a result of the increased voltage, which delivers more power to the motors. As a higher voltage typically leads to greater motor torque and speed, a reduced PWM signal can effectively manage the motor control, providing a smoother and more controlled movement for the wheelchair.

Figure 4.3.2.1 Previous 12V Battery

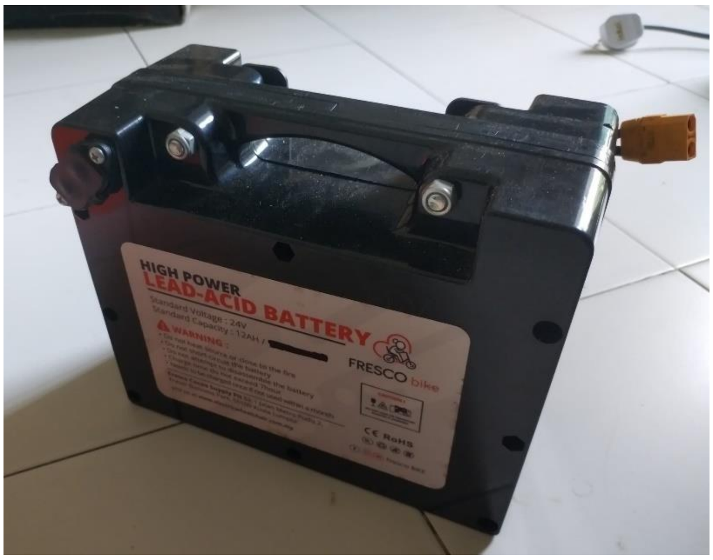


Figure 4.3.2.2 New 24 Volt Battery

The experiment that has been conducted is by using the 12 volt battery and the 24 volt battery to drive the motor of the wheelchair. For the 12 volt battery it require a 50 PWM value just to move the wheelchair on its own. Where as the 24 volt battery only require 30 PWM value just to move the wheelchair on its own. Individuals with varying weights to test the wheelchair system performance has been conducted.In the context of the wheelchair system using a 24-volt battery setup, the relationship between user weight and the associated Pulse Width Modulation (PWM) values for movement becomes apparent. Here's a breakdown:

- Person A, weighing 60 Kg, finds that a PWM value of 50 is sufficient to set the wheelchair in motion.
- For Person B, with a weight of 80 Kg, a slightly higher PWM value of 70 is needed to initiate wheelchair movement.
- Person C, weighing 90 Kg, requires the highest PWM value of 90 for the 24-volt setup to effectively move the wheelchair.

This data suggests a proportional relationship between user weight and the necessary PWM values, showcasing the system's adaptability to different loads. The incremental increase in PWM values aligns with the expected need for higher power output to accommodate heavier individuals and ensure optimal wheelchair performance.

The description of the movement refers to the acceptable operational conditions under which the wheelchair moves smoothly without encountering any issues. This implies a state where the wheelchair responds to the control inputs, demonstrating a reliable and predictable behavior without unexpected deviations or disturbances.

4.3.3 Improved Ultrasonic Sensor System Setup

The previous system relied on a single ultrasonic sensor for the obstacle detection. The current configuration , offers an improvement by combining four ultrasonic sensors. The positioning of the ultrasonic sensor provides wider coverage. The increased number of the ultrasonic sensors from one to four represents an upgraded obstacle detection system .

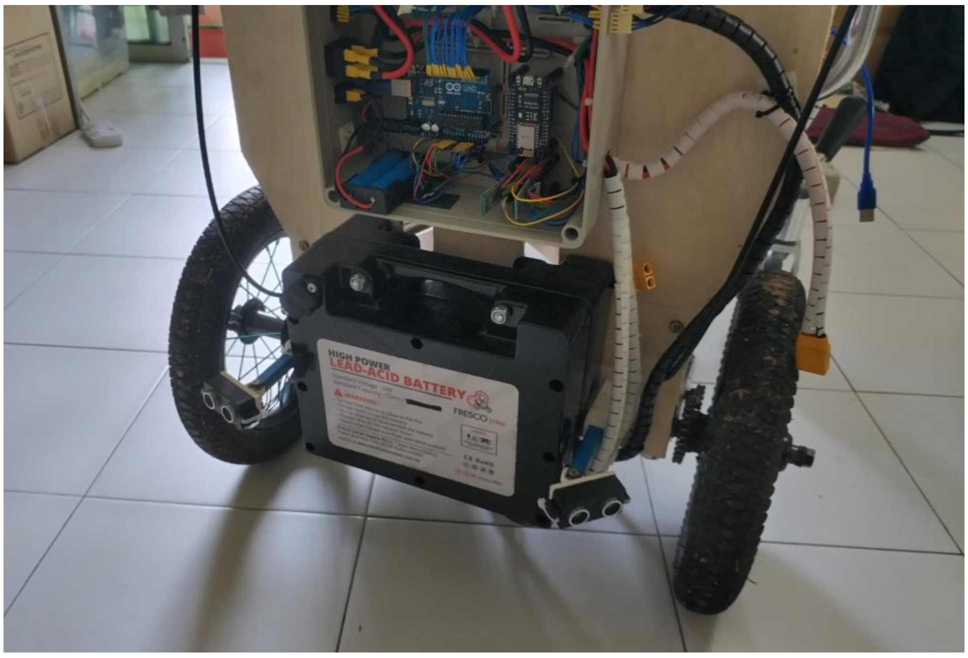


Figure 4.3.3.1 Back Sensor On The Wheelchair

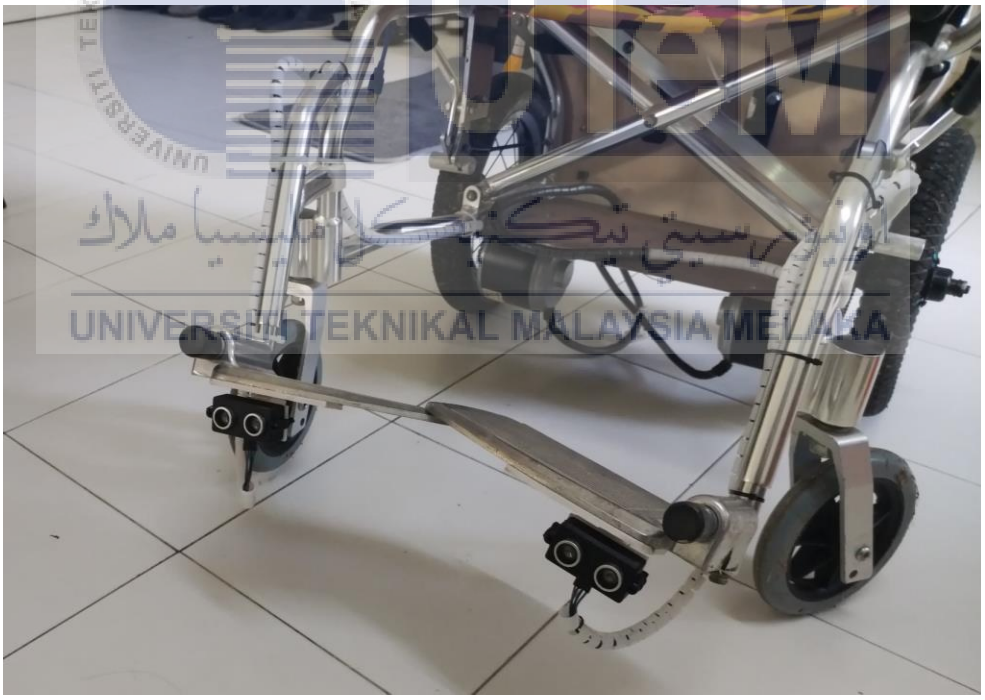


Figure 4.3.3.2 Front Sensor On The Wheelchair

4.4 Discussion

This project focuses on developing a wheelchair control system using an Android phone, with MIT App Inventor serving as the major communication platform between the phone and the wheelchair. A slider function also allows users to change the motor speed or the Pulse Width Modulation (PWM) value. The D-pad interface allows for four-way mobility, while the voice control capability allows users to command the wheelchair orally. Previously, the HC-05 Bluetooth module served as the primary link between the mobile app and the Arduino board, utilizing traditional Bluetooth technology. The AT-09 Bluetooth module, which runs on Bluetooth Low Energy (BLE), allows for decreased voltage utilization and a longer range in the improved system.

The wheelchair system has been improved in several ways, including the inclusion of four more ultrasonic sensors over the previous iteration, which used only one. This enhancement improves the obstacle detection system, resulting in a more complete knowledge of the wheelchair's surroundings. These developments add up to a more adaptable, efficient, and user-friendly wheelchair control system.

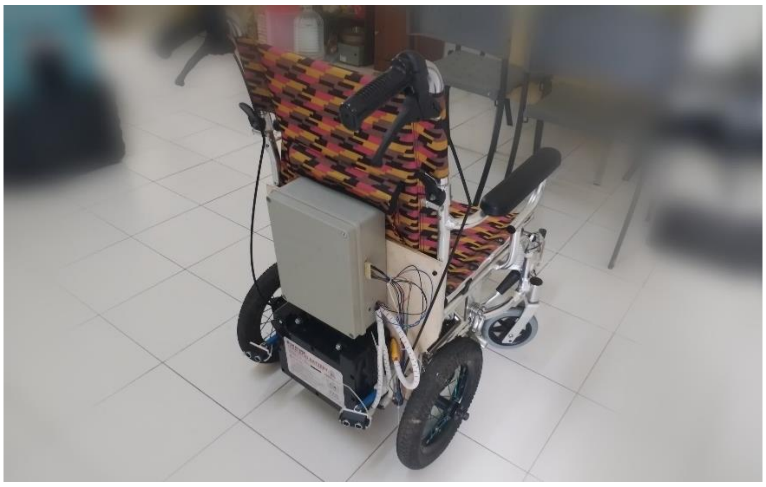


Figure 4.4.1 Completed System Wheelchair Backview

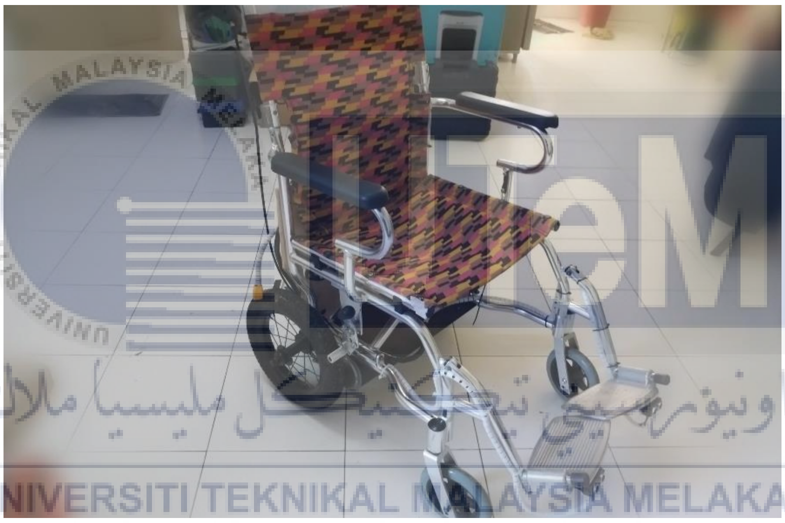


Figure 4.4.2 Completed System WheelChair Frontview

In addition to the mentioned enhancements, this project introduces further improvements, expanding the functionality and monitoring capabilities of the wheelchair control system. Users now have the ability to control a power outlet remotely, toggling it ON or OFF directly from the Android phone interface, or through voice commands. The inclusion of remote monitoring allows users to manage the power outlet through cloud services. Leveraging the

Arduino Cloud, users can not only monitor the current state of the power outlet but also remotely toggle it ON or OFF.

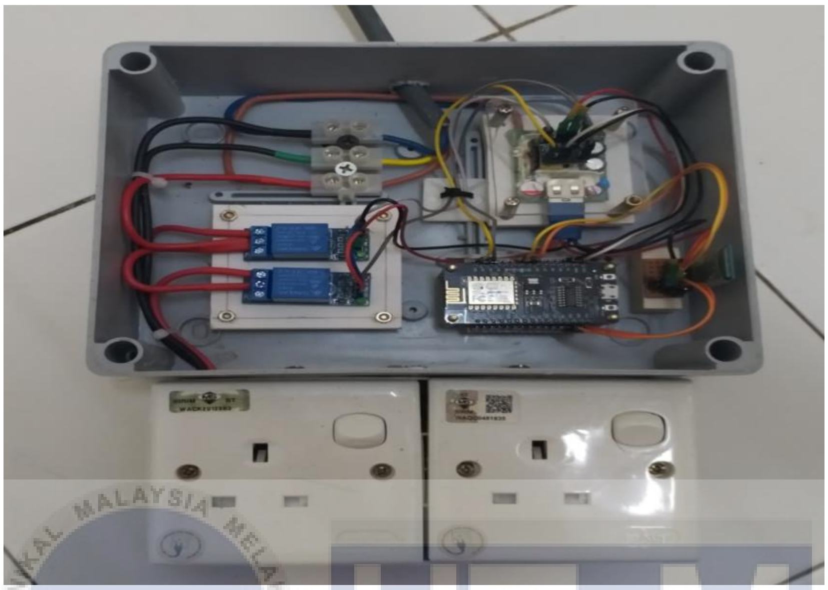


Figure 4.4.3 Completed Power Outlet Control Layout

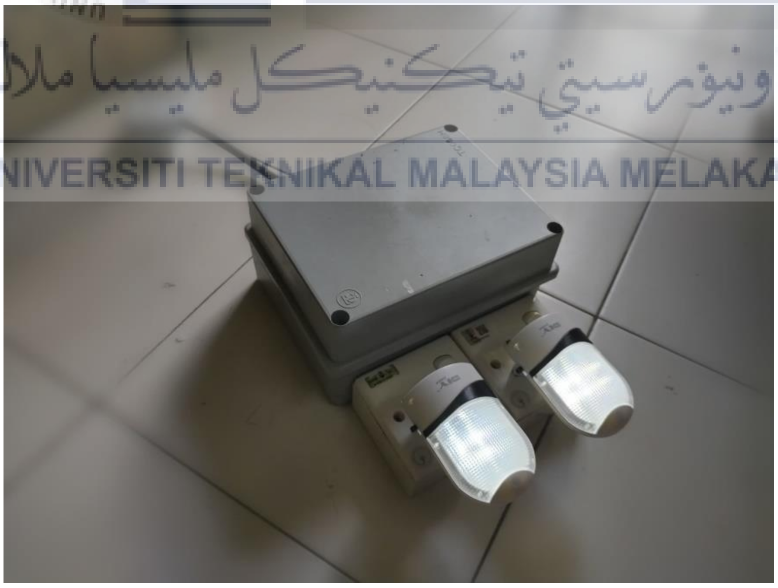


Figure 4.4.4 Both Power are turn On

Furthermore, the Arduino Cloud provides a centralized dashboard for users to observe the wheelchair's voltage status. This monitoring capability extends to both the power outlet and wheelchair through the cloud interface. The system is designed with two separate NodeMCU modules, one integrated into the wheelchair for sending data to the cloud, and the other housed in the power outlet control box. A notable upgrade is the shift in the battery voltage used to power the wheelchair's motors. The system now utilizes a 24-volt battery, doubling the power compared to the previous 12-volt configuration. This upgrade enhances the overall performance of the wheelchair, providing more power for efficient and effective movement.

In conclusion, the objectives of this project have been successfully realized through effective software and hardware implementations. The wheelchair exhibits seamless movement in response to user commands solely through the use of an Android phone. The speed control functionality, achieved through Pulse Width Modulation (PWM) signals, provides users with precise control over the wheelchair's velocity. Moreover, the project introduces wireless control for electrical appliances. For instance, a lamp can be effortlessly switched ON by a simple click of the switch button on the Android phone interface. Pressing the OFF button allow the power outlet to be turned off. This wireless control feature enhances user convenience, showcasing the project's effectiveness in achieving its intended goals

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

In conclusion, the study's objectives were met with success. The created Android system enables wheelchair control via smartphone via Bluetooth connectivity provided by MIT App Inventor software. The user-friendly control interfaces, such as the D-pad and Voice Command, allow users to effortlessly operate the wheelchair in various directions. The use of Pulse Width Modulation (PWM) signal control allows the wheelchair's speed to be adjusted.Furthermore, the Android operating system expands its ability to remotely manage domestic items like as lighting and fans. This control is made possible via a coupled Bluetooth device, which increases user convenience. The Android interface also provides real-time information on the power condition of electrical equipment. These successes are in line with the original goals, proving the effective creation of an integrated system that addresses mobility issues while also providing increased control and monitoring of domestic appliances. To deliver real-time updates on the power state of electrical appliances, the Internet of Things (IoT) framework has been seamlessly integrated into the system. The chosen platform for this integration is the Arduino Cloud. This incorporation allows for continuous monitoring and reporting of the power states of electrical appliances in real-time. This project's concepts and innovations have the potential to inspire future research endeavours and give significant insights for young engineers building solutions that fulfil commercial requirements and public interests.

5.2 Project Potential

The potential of this project it can enhance the daily live of the wheelchair users and the elderly. The wireless control system can be integrated into existing wheelchairs and enhance its potential for other type of system configuration. This project can be commercialize at healthcare institute .The target audience are those that are with lower limb disability and elderly people.

5.3 Future Works

Several areas for potential enhancements exist to improve accuracy, functionality, and accessibility for the project. Some recommendations for future research and development include:

i) Use a better braking system to stop the wheelchair
ii) Add current sensor and voltage sensor to the wheelchair system
iii) Use more advance algorithm for the obstacle detection system
iv) Use different type of sensor for the obstacle detection system
v) Use better mobile apps developer platform
vi) Improve the mechanical drive system of the wheelchair

REFERENCES

[1] A. Pajkanović [1]A. Pajkanović and B. Dokić, “Wheelchair Control by Head Motion | SJEE,” Wheelchair Control by Head Motion | SJEE, Feb. 15, 2013.

[2] Nowshin, N., Rashid, M. M., Akhtar, T., & Akhtar, N. (2018). Infrared Sensor Controlled Wheelchair for Physically Disabled People.

[3] Cerejo, R., Correia., V., & Pereira, N. (2015). EYE CONTROLLED WHEELCHAIR BASED ON ARDUINO CIRCUIT.

[4] Masato Nishimori, Takeshi Saitoh, and Ryosuke Konishi, “Voice controlled intelligent wheelchair,” SICE Annual Conference 2007, Sep. 2007, Published.

[5] Sirisha, P & Bethapudi, Prakash & Meghana, Ratna & Yamini, C & Lakshmi, E. (2020). WIRELESS SMART WHEELCHAIR.

[6] M. Islam and S. Jin, “An Overview Research on Wireless Communication Network,” Advances in Wireless Communications and Networks, vol. 5, no. 1, p. 19, 2019.

[7] Ms. Indumathy. T, 2015, An Overview of Bluetooth-Wireless Technology, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) RACMS – 2015 (Volume 3 – Issue 33)

[8] History of the Wheelchair. (2019, January 24). HISTORY.PHYSIO.

[9] Woods, B., & Watson, N. (2003). A Short History of Powered Wheelchairs. Assistive Technology, 15(2), 164–180.

[10] Amin, M. S., Rizvi, S. T. H., Malik, S., Faheem, Z. B., & Liaqat, A. (2021). Smart Wheelchair- An Implementation of Voice and Android Controlled System. 2021

International Conference on Digital Futures and Transformative Technologies (ICoDT2).

[11] Borges, B., Chandra, A., Kalantri, R., Gupta, S., Dsilva, G., & Rajguru, S. (2018). Android Controlled Wheelchair. 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC).

[12] Shawon, S. I., Bhuiyan, M. M. H., & Plateau, T. P. (2018). An Innovative Construction of Wheelchair for Handicapped Persons. International Journal of Science and Qualitative Analysis, 4(1), 13.

[13] N. Aktar, I. Jaharr and B. Lala, "Voice Recognition based intelligent Wheelchair and GPS Tracking System," 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox'sBazar, Bangladesh, 2019.

[14] Bhavsar, M., Roy, K., Kelly, J. et al. Anomaly-based intrusion detection system for IoT application. Discov Internet Things 3, 5 (2023)

[15] Viran, Shane. (2022). VOICE CONTROLLED LIGHTING SYSTEM.

[16] Abdullahi Badamasi, Yusuf. (2014). The working principle of an Arduino. 1-4.

[17] Bento, Antonio. (2018). IoT: NodeMCU 12e X Arduino Uno, Results of an experimental and comparative survey. 6. 45-56.

[18] Al Dahoud, Ali & Fezari, Mohamed. (2018). NodeMCU V3 For Fast IoT Application Development.

[19] Doshi, Aayush & Rai, Yashraj & Vakharia, Deep. (2021). Iot based Home Automation. International Journal for Research in Applied Science and Engineering Technology. 9.

[20] Ch, Rajendra Prasad. (2019). Internet of Things Based Home Monitoring and Device Control Using Esp32. International Journal of Recent Technology and Engineering. 8. 58-62.

[21] Kumar, P.K., Rao, G.J., Suguna, A.B., & Srihari, P.V. (2019). Voice Controlled Home Automation. International Journal of Scientific Research in Science, Engineering and Technology.

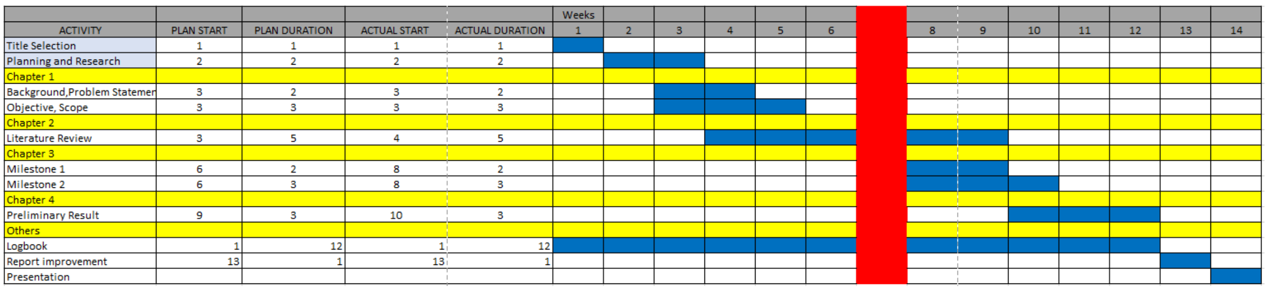
[22] Kumar, S., & Solanki, S.S. (2016). Voice and touch control home automation. 2016 3rd International Conference on Recent Advances in Information Technology (RAIT), 495-498.

[23] Hardik, S. (2014). Interfacing of AT Command based HC-05 Serial Bluetooth Module.

[24] Fenriana, I., Dwi Putra, D.S., Dermawan, B., & Kurnia, Y. (2022). Smart Home Prototype with HC–05 Bluetooth and RFID Modules, Based on Microcontroller. bit-Tech.

[25] A. Maier, A. Sharp and Y. Vagapov, "Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things," 2017 Internet Technologies and Applications (ITA), Wrexham, UK, 2017, pp. 143-148, doi: 10.1109/ITECHA.2017.8101926.

APPENDICES



					Weeks						🟥							
ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	1	2	3	4	5	6	🟥	8	9	10	11	12	13	14
Title Selection	1	1	1	1	🟦						🟥							
Planning and Research	2	2	2	2		🟦	🟦				🟥							
Chapter 1											🟥							
Background,Problem Statemer	3	2	3	2			🟦	🟦			🟥							
Objective, Scope	3	3	3	3			🟦	🟦	🟦		🟥							
Chapter 2											🟥							
Literature Review	3	5	4	5				🟦	🟦	🟦	🟥	🟦	🟦					
Chapter 3											🟥							
Milestone 1	6	2	8	2							🟥	🟦	🟦					
Milestone 2	6	3	8	3							🟥	🟦	🟦	🟦				
Chapter 4											🟥							
Preliminary Result	9	3	10	3							🟥			🟦	🟦	🟦		
Others											🟥							
Logbook	1	12	1	12	🟦	🟦	🟦	🟦	🟦	🟦	🟥	🟦	🟦	🟦	🟦	🟦		
Report improvement	13	1	13	1							🟥						🟦	
Presentation											🟥							🟦

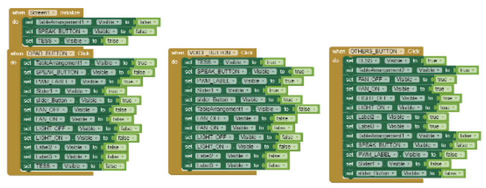
appendix 1.Gantt Chart PSM

```
#include <SoftwareSerial.h>
#include <NewPing.h>

SoftwareSerial bluetooth(10, 11); // RX, TX

char t = 'S'; // Default state is stop
unsigned long commandStartTime = 0; // Variable to store the start time of a voice-
activated command
const unsigned long commandDuration = 8000; // Duration for the voice-activated
command in milliseconds (8 seconds)
bool obstacleDetected = false; // Flag to indicate if an obstacle is detected

int pwmA = 9;
int dirA = 8;
int pwmB = 3;
int dirB = 4;
int bluetoothStatePin = 13;

int receivedPWMValue = 0; // Variable to store the received PWM value

// Define Front Ultrasonic Sensors
NewPing frontSensorA(12, 2); // trigger, echo for Front Sensor A
NewPing frontSensorB(7, 5);  // trigger, echo for Front Sensor B

// Define Back Ultrasonic Sensors
NewPing backSensorA(14, 15);  // A0, A1 as digital pins for Back Sensor A
NewPing backSensorB(16, 17);  // A2, A3 as digital pins for Back Sensor B

const int controlPin_1 = 6; // Pin for additional control light
const int controlPin_2 = 18; // Pin for additional control fan A4 as digital pin
char defaultControlState_1 = 'X'; // Default state for the additional control pin
char defaultControlState_2 = 'K'; // Default state for the additional control pin

void setup() {
  pinMode(pwmA, OUTPUT);
  pinMode(pwmB, OUTPUT);
  pinMode(dirA, OUTPUT);
  pinMode(dirB, OUTPUT);
  pinMode(bluetoothStatePin, INPUT);

  pinMode(controlPin_1, OUTPUT);
  pinMode(controlPin_2, OUTPUT);

  Serial.begin(9600);
  bluetooth.begin(9600);
}

void loop() {
  unsigned int frontDistanceA = frontSensorA.ping_cm();
  unsigned int frontDistanceB = frontSensorB.ping_cm();
  unsigned int backDistanceA = backSensorA.ping_cm();
  unsigned int backDistanceB = backSensorB.ping_cm();

  if ((frontDistanceA >= 15 && frontDistanceA <= 25) ||
      (frontDistanceB >= 15 && frontDistanceB <= 25) ||
      (backDistanceA >= 15 && backDistanceA <= 25) ||
      (backDistanceB >= 15 && backDistanceB <= 25)) {
    // Obstacle detected, stop the motors and set the obstacle flag
    stopMotors();
    obstacleDetected = true;
    // Clear the Bluetooth buffer to discard any commands received during obstacle
detection
    while (bluetooth.available()) {
      bluetooth.read();
    }
  } else {
    obstacleDetected = false;

    if (digitalRead(bluetoothStatePin) == LOW) {
      if (t != 'S') {
        stopMotors();
        t = 'S';
      }
      Serial.println("Bluetooth connection lost.");
      delay(500); // Introduce a delay to avoid rapid loop execution
    } else {
      if (!obstacleDetected && bluetooth.available()) {
        t = bluetooth.read();
        Serial.println(t);

        commandStartTime = millis();

        if (t == 'P') {
          receivedPWMValue = bluetooth.parseInt();
          Serial.print("Received PWM value: ");
          Serial.println(receivedPWMValue);
        } else {
          processCommands();
        }
      }
    }

    if (t != 'S' && millis() - commandStartTime >= commandDuration) {
      stopMotors();
      t = 'S';
    }

    if (!obstacleDetected && t != 'P') {
      processCommands();
    }
  }

  delay(50); // Introduce a small delay in the loop
}

void processCommands() {
  if (t == 'F' || t == 'B' || t == 'R' || t == 'L' || t == 'Y' || t == 'H' || t == 'U' || t == 'T' || t == 'S') {
    moveBasedOnCommand();
  } else if (t == 'Z') {
    setControlState_1('M');
  } else if (t == 'X') {
    setControlState_1('N');
  } else if (t == 'J'){
    setControlState_2('C');
  } else if (t == 'K'){
    setControlState_2('V');
  }
}

void moveBasedOnCommand() {
  if (t == 'F') {
    moveForward();
  } else if (t == 'B') {
    moveBackward();
  } else if (t == 'R') {
    turnRight();
  } else if (t == 'L') {
    turnLeft();
  } else if (t == 'Y') {
    moveForward();
  } else if (t == 'H') {
    moveBackward();
  } else if (t == 'U') {
    turnRight();
  } else if (t == 'T') {
    turnLeft();
  } else if (t == 'S') {
    stopMotors();
  }
}

void stopMotors() {
  analogWrite(pwmA, 0);
  analogWrite(pwmB, 0);
}

void moveForward() {
  analogWrite(pwmA, receivedPWMValue);
  analogWrite(pwmB, receivedPWMValue);
  digitalWrite(dirA, LOW);
  digitalWrite(dirB, LOW);
}

void moveBackward() {
  analogWrite(pwmA, receivedPWMValue);
  analogWrite(pwmB, receivedPWMValue);
  digitalWrite(dirA, HIGH);
  digitalWrite(dirB, HIGH);
}

void turnRight() {
  analogWrite(pwmA, receivedPWMValue);
  analogWrite(pwmB, receivedPWMValue);
  digitalWrite(dirA, HIGH);
  digitalWrite(dirB, LOW);
}

void turnLeft() {
  analogWrite(pwmA, receivedPWMValue);
  analogWrite(pwmB, receivedPWMValue);
  digitalWrite(dirA, LOW);
  digitalWrite(dirB, HIGH);
}

void setControlState_1(char state) {
  if (state == 'M') {
    digitalWrite(controlPin_1, HIGH);
  } else {
    digitalWrite(controlPin_1, LOW);
  }
}

void setControlState_2(char state) {
  if (state == 'C') {
    digitalWrite(controlPin_2, HIGH);
  } else {
    digitalWrite(controlPin_2, LOW);
  }
}
```

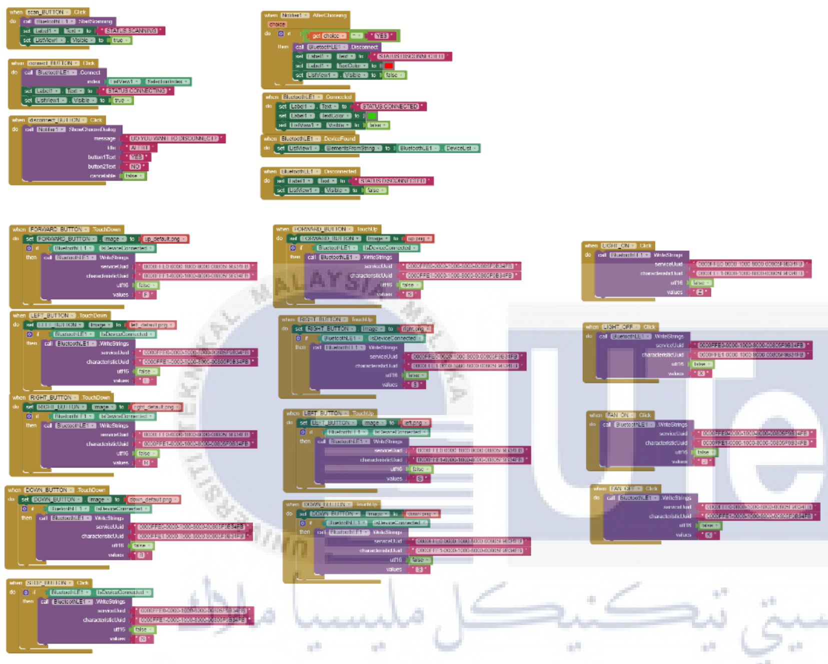
```
#include <SoftwareSerial.h>
#include "thingProperties.h"

SoftwareSerial BTSerial(14, 12); // Connect GPIO14 (D5) to TX of HC-05, GPIO12 (D6)
to RX of HC-05
int inputPin1 = 5;
int inputPin2 = 4;
int lastValue1 = LOW; // Variable to store the last read value of inputPin1
int lastValue2 = LOW; // Variable to store the last read value of inputPin2

void setup() {
  pinMode(inputPin1, INPUT);
  pinMode(inputPin2, INPUT);
  Serial.begin(9600);
  BTSerial.begin(38400);

  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  // This delay gives the chance to wait for a Serial Monitor without blocking if none is
found
  delay(1500);

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and errors
     the higher number the more granular information you'll get.
     The default is 0 (only errors).
     Maximum is 4
  */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}

void loop() {
  // Read pin values
  int inputValue1 = digitalRead(inputPin1);
  int inputValue2 = digitalRead(inputPin2);

  Serial.print("Input Value 1: ");
  Serial.println(inputValue1);

  Serial.print("Input Value 2: ");
  Serial.println(inputValue2);

  // Check and update IoT Cloud variables only when there is a change in pin values
  if (inputValue1 != lastValue1) {
    lastValue1 = inputValue1;
    lED = inputValue1;
    ArduinoCloud.update();

    if (inputValue1 == HIGH) {
      BTSerial.write('H'); // Send 'H' for High for the first pin
      Serial.println("Sent: H for Pin 1");
    } else {
      BTSerial.write('L'); // Send 'L' for Low for the first pin
      Serial.println("Sent: L for Pin 1");
    }
  }

  if (inputValue2 != lastValue2) {
    lastValue2 = inputValue2;
    fAN = inputValue2;
    ArduinoCloud.update();

    if (inputValue2 == HIGH) {
      BTSerial.write('A'); // Send 'A' for High for the second pin
      Serial.println("Sent: A for Pin 2");
    } else {
      BTSerial.write('B'); // Send 'B' for Low for the second pin
      Serial.println("Sent: B for Pin 2");
    }
  }


  delay(500); // Adjust delay as needed
}

/*
  Since LED is READ_WRITE variable, onLEDChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onLEDChange()  {
  // Set the last input pin value based on the received cloud value
  lastValue1 = lED;
}

/*
  Since FAN is READ_WRITE variable, onFANChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onFANChange()  {
  // Set the last input pin value based on the received cloud value
  lastValue2 = fAN;
}
```

```
  /*
  Sketch generated by the Arduino IoT Cloud Thing "Untitled"
  https://create.arduino.cc/cloud/things/c0dd2df2-421a-4e86-8113-23a181e1ad60

  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated when changes are made
to the Thing

  bool fAN;
  bool lED;

  Variables which are marked as READ/WRITE in the Cloud Thing will also have
functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this sketch.
*/

#include "thingProperties.h"
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(14, 12); // Connect GPIO14 (D5) to TX of HC-05, GPIO12 (D6)
to RX of HC-05
int ledPin = 5; // Pin for the LED
int fanPin = 4; // Pin for the fan

void setup() {

  pinMode(ledPin, OUTPUT);
  pinMode(fanPin, OUTPUT);
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  // This delay gives the chance to wait for a Serial Monitor without blocking if none is
found
  delay(1500);

  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and errors
     the higher number the more granular information you’ll get.
     The default is 0 (only errors).
 */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();

  BTSerial.begin(38400);

}

void loop() {
  ArduinoCloud.update();
  handleBluetoothCommands();



}

void handleBluetoothCommands() {
  // Check for incoming Bluetooth commands
  if (BTSerial.available()) {
    char received = BTSerial.read();
    Serial.print("Received: ");
    Serial.println(received);

    // Process Bluetooth commands and update IoT Cloud properties
    if (received == 'H') {
      digitalWrite(ledPin, HIGH);
      lED = true;
      Serial.println("Turning LED ON");
    } else if (received == 'L') {
      digitalWrite(ledPin, LOW);
      lED = false;
      Serial.println("Turning LED OFF");
    } else if (received == 'A') {
      digitalWrite(fanPin, HIGH);
      fAN = true;
      Serial.println("Turning FAN ON");
    } else if (received == 'B') {
      digitalWrite(fanPin, LOW);
      fAN = false;
      Serial.println("Turning FAN OFF");
    }

    // Synchronize IoT Cloud properties after processing Bluetooth comman
  }
}





/*
```

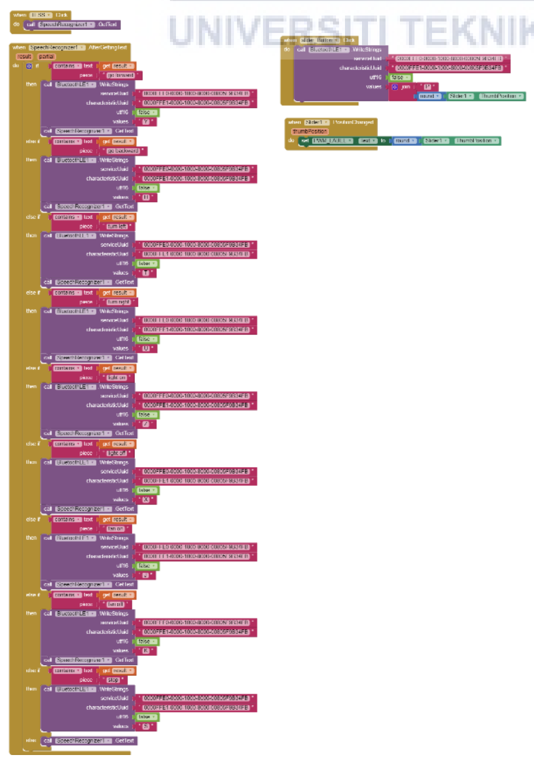
```
  Since LED is READ_WRITE variable, onLEDChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onLEDChange() {
  // Act upon LED change
  digitalWrite(ledPin, lED ? HIGH : LOW);
}

void onFANChange() {
  // Act upon FAN change
  digitalWrite(fanPin, fAN ? HIGH : LOW);
}
```



```
when Screen1 .Initialize
do  set TableArrangement1 . Visible to false
    set SPEAK_BUTTON . Visible to false
    set TESS . Visible to false

when DPAD_BUTTON .Click
do  set TableArrangement1 . Visible to true
    set SPEAK_BUTTON . Visible to false
    set PWM_LABEL . Visible to true
    set Slider1 . Visible to true
    set slider_Button . Visible to true
    set FAN_OFF . Visible to false
    set FAN_ON . Visible to false
    set LIGHT_OFF . Visible to false
    set LIGHT_ON . Visible to false
    set Label2 . Visible to false
    set Label3 . Visible to false
    set TESS . Visible to false

when VOICE_BUTTON .Click
do  set TESS . Visible to true
    set SPEAK_BUTTON . Visible to true
    set PWM_LABEL . Visible to true
    set Slider1 . Visible to true
    set slider_Button . Visible to true
    set TableArrangement1 . Visible to false
    set FAN_OFF . Visible to false
    set FAN_ON . Visible to false
    set LIGHT_OFF . Visible to false
    set LIGHT_ON . Visible to false
    set Label2 . Visible to false
    set Label3 . Visible to false

when OTHERS_BUTTON .Click
do  set TESS . Visible to true
    set TableArrangement2 . Visible to true
    set FAN_OFF . Visible to true
    set FAN_ON . Visible to true
    set LIGHT_OFF . Visible to true
    set LIGHT_ON . Visible to true
    set Label2 . Visible to true
    set Label3 . Visible to true
    set TableArrangement1 . Visible to false
    set SPEAK_BUTTON . Visible to false
    set PWM_LABEL . Visible to false
    set Slider1 . Visible to false
    set slider_Button . Visible to false
```



```
when scan_BUTTON .Click
do  call BluetoothLE1 .StartScanning
    set Label1 . Text to "STATUS:SCANNING"
    set ListView1 . Visible to true

when connect_BUTTON .Click
do  call BluetoothLE1 .Connect
                         index  ListView1 . SelectionIndex
    set Label1 . Text to "STATUS:CONNECTING"
    set ListView1 . Visible to true

when disconnect_BUTTON .Click
do  call Notifier1 .ShowChooseDialog
                         message  "DO YOU WANT TO DISCONNECT?"
                           title  "ALERT"
                     button1Text  "YES"
                     button2Text  "NO"
                     cancelable   false

when Notifier1 .AfterChoosing
 choice
do  if  get choice = "YES"
    then  call BluetoothLE1 .Disconnect
          set Label1 . Text to "STATUS:DISCONNECTED"
          set Label1 . TextColor to 🟥
          set ListView1 . Visible to false

when BluetoothLE1 .Connected
do  set Label1 . Text to "STATUS:CONNECTED"
    set Label1 . TextColor to 🟩
    set ListView1 . Visible to false

when BluetoothLE1 .DeviceFound
do  set ListView1 . ElementsFromString to BluetoothLE1 . DeviceList

when BluetoothLE1 .Disconnected
do  set Label1 . Text to "STATUS:DISCONNECTED"
    set ListView1 . Visible to false

when FORWARD_BUTTON .TouchDown
do  set FORWARD_BUTTON . Image to "up_default.png"
    if  BluetoothLE1 . IsDeviceConnected
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "F"

when LEFT_BUTTON .TouchDown
do  set LEFT_BUTTON . Image to "left_default.png"
    if  BluetoothLE1 . IsDeviceConnected
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "L"

when RIGHT_BUTTON .TouchDown
do  set RIGHT_BUTTON . Image to "right_default.png"
    if  BluetoothLE1 . IsDeviceConnected
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "R"

when DOWN_BUTTON .TouchDown
do  set DOWN_BUTTON . Image to "down_default.png"
    if  BluetoothLE1 . IsDeviceConnected
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "B"

when STOP_BUTTON .Click
do  if  BluetoothLE1 . IsDeviceConnected
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "S"

when FORWARD_BUTTON .TouchUp
do  set FORWARD_BUTTON . Image to "up.png"
    if  BluetoothLE1 . IsDeviceConnected
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "S"

when RIGHT_BUTTON .TouchUp
do  set RIGHT_BUTTON . Image to "right.png"
    if  BluetoothLE1 . IsDeviceConnected
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "S"

when LEFT_BUTTON .TouchUp
do  set LEFT_BUTTON . Image to "left.png"
    if  BluetoothLE1 . IsDeviceConnected
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "S"

when DOWN_BUTTON .TouchUp
do  set DOWN_BUTTON . Image to "down.png"
    if  BluetoothLE1 . IsDeviceConnected
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "S"

when LIGHT_ON .Click
do  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "Z"

when LIGHT_OFF .Click
do  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "X"

when FAN_ON .Click
do  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "J"

when FAN_OFF .Click
do  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "K"
```



```
when TESS .Click
do  call SpeechRecognizer1 .GetText

when SpeechRecognizer1 .AfterGettingText
 result  partial
do  if  contains text get result
                 piece "go forward"
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "F"
          call SpeechRecognizer1 .GetText
    else if  contains text get result
                      piece "go backward"
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "B"
          call SpeechRecognizer1 .GetText
    else if  contains text get result
                      piece "turn left"
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "L"
          call SpeechRecognizer1 .GetText
    else if  contains text get result
                      piece "turn right"
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "R"
          call SpeechRecognizer1 .GetText
    else if  contains text get result
                      piece "light on"
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "Z"
          call SpeechRecognizer1 .GetText
    else if  contains text get result
                      piece "light off"
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "X"
          call SpeechRecognizer1 .GetText
    else if  contains text get result
                      piece "fan on"
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "J"
          call SpeechRecognizer1 .GetText
    else if  contains text get result
                      piece "fan off"
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "K"
          call SpeechRecognizer1 .GetText
    else if  contains text get result
                      piece "stop"
    then  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  "S"
    else  call SpeechRecognizer1 .GetText

when slider_Button .Click
do  call BluetoothLE1 .WriteStrings
                      serviceUuid  "0000FFE0-0000-1000-8000-00805F9B34FB"
               characteristicUuid  "0000FFE1-0000-1000-8000-00805F9B34FB"
                            utf16  false
                           values  join "P"
                                        round Slider1 . ThumbPosition

when Slider1 .PositionChanged
 thumbPosition
do  set PWM_LABEL . Text to round Slider1 . ThumbPosition
```