Faculty of Electrical and Electronic Engineering Technology

DEVELOPMENT OF VOICE COMMAND GROCERY SHOPPING LIST MAKER BASED ON ARDUINO PLATFORM

NUR ELMIERA BINTI ISMAIL

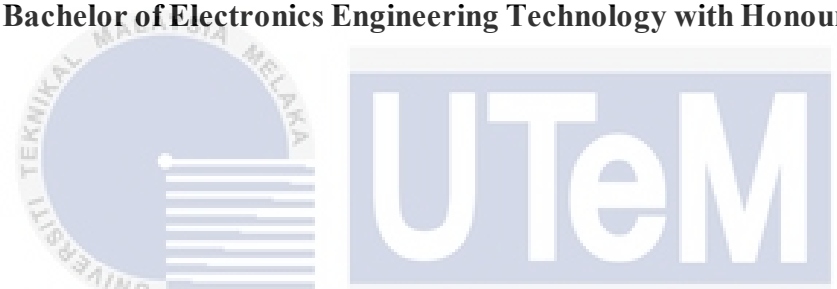
Bachelor of Electronic Engineering Technology with Honours

2021

DEVELOPMENT OF VOICE COMMAND GROCERY SHOPPING LIST MAKER BASED ON ARDUINO PLATFORM

NUR ELMIERA BINTI ISMAIL

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



Faculty of Electrical and Electronic Engineering Technology

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2021

DECLARATION

I declare that this project report entitled "Development of Voice Command Grocery Shopping List Maker based on Arduino Platform" 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	:	ELMIERA
Student Name	:	NUR ELMIERA BINTI ISMAIL
Date	:	8/2/2022

APPROVAL

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

Signature : 

Supervisor Name : TS. DR. MOHD SYAFIQ BIN MISPAN

Date : 8/2/2022

DEDICATION

To my beloved parents,
Ismail bin Omar and Julia bt Ismail who always there with me and instilled in me the virtues of perseverance and relentlessly encouraged me to strive for excellent in completing this report.

To my siblings that always generates and giving idea for me to complete this report, I would like to say thank you for always support and help me with their full of love that make me feel motivated and always in high spirits to finish my report.

To my great supervisor TS. DR. Mohd Syafiq bin Mispan, thank you for the guidance and encouragement for me to make sure my report and project is done well and always keep reminds me to complete my task and always motivate me with some brilliant idea and positive vibes also never lets any sadness dominate into my heart.

ABSTRACT

In today's modern world, due to a lack of time to manage household needs, people are always seeking a convenient and effective way of doing their daily jobs. One of the daily activities in people's lives that could be improved is going to the grocery store. It has become a habit for people to provide for their household needs in an orderly and complete manner. As a result, the goal of this project is to create a system that can use voice to list groceries and display them on a small I2C 16x2 LCD in the kitchen area. This application will help the user to list the groceries through voice recognition and takes a short time to list the items needed. The project that is used consists of the programmable voice recognition module, the small LCD display, and the microcontroller Arduino Uno board with ESP8266 (ESP-01). The list of groceries is linked to the grocery app (i.e., developed by the previous PSM students). Whenever the user enters new data into the list via the voice command system, the list in the Grocer App is updated. So, this mobile application is easy to use and shortens the time for the user. Moreover, the user can add, remove, or overwrite the list by using the grocery store's app.

ABSTRAK

Dalam dunia moden hari ini, kerana kekurangan masa untuk menguruskan hal rumah, orang selalu mencari cara yang mudah dan berkesan untuk melakukan pekerjaan harian. Salah satu perkara harian dalam kehidupan orang yang memerlukan penambahbaikan adalah ketika melakukan runcit. Sudah menjadi kebiasaan bagi orang untuk menyediakan barang runcit rumah dengan teratur dan lengkap. Oleh itu, objektif projek ini adalah untuk merancang sistem yang boleh menggunakan suara untuk menyenaraikan runcit dan paparan pada monitor I2C 16x2 LCD kecil di kawasan dapur. Aplikasi ini akan membantu pengguna menyenaraikan barang runcit melalui pengecaman suara dan mengambil masa yang singkat untuk menyenaraikan item yang diperlukan. Projek yang digunakan terdiri daripada modul pengecaman suara boleh atur cara, paparan LCD kecil I2C 16x2, dan papan mikropengawal Arduino Uno dengan ESP8266 (ESP-01). Senarai barang runcit telah berjaya dihubungkan ke Aplikasi Grocer (iaitu, yang dikembangkan oleh pelajar PSM sebelumnya). Setiap kali pengguna memasukkan data baru dalam senarai melalui sistem perintah suara, daftar di Aplikasi Grocer telah diperbaharui. Jadi, aplikasi mudah alih ini senang digunakan dan memendekkan masa untuk pengguna. Lebih-lebih lagi, pengguna dapat menambah, membuang atau menukar senarai dengan menggunakan Aplikasi Grocer.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my supervisor, Ts. Dr. Mohd Syafiq bin Mispan for his precious guidance, words of wisdom and patient throughout this project. Their invaluable support and guidance of relevant knowledge, insightful remarks and ideas throughout the research and thesis work has made a significant contribution to the project's success.

My highest sentimental appreciation and gratitude to my beloved parents, Mr. Ismail bin Omar and Mrs. Julia binti Ismail, as well as my siblings, for their love, prayers, encouragement, understanding, support and contributions throughout my entire life. And thank you to everyone who has helped and contributed to this study effort, whether directly or indirectly. Your generosity means a lot to me and will never be forgotten. Thank you kindly.

My best wishes go to all my university colleagues for their moral support, information sharing, and encouragement. Thank you once again for your friendship and memorable experiences.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	-	Application Programming Interface
GPIO	-	General Purpose Input/Output
HTML	-	Hypertext Markup Language
iOS	-	iPhone Operating System
IoT	-	Internet
JSON File	-	JavaScript Object Notation
LCD	-	Liquid-crystal display
OS	-	Operating System
PIR sensor	-	Passive infrared
PWM	-	Pulse-width modulation
TCP/IP	-	The Internet protocol suite

CHAPTER 1

INTRODUCTION

1.1 Background

In today's world, a portable computer, such as a smartphone, has become an essential part of our daily lives. Android is the operating system for these cellphones. Today, the three most widely used smartphone operating systems are Apple's iOS, Google's Android, and Microsoft's Windows. Each of these operating systems has a unique mix of advantages and disadvantages. iOS, Android, and Windows all have 1.4 million, 1.5 million, and 0.3 million applications installed, respectively, according to Statista. (Statista,2015). The cell phone has fundamentally altered people's lives. Nowadays, people interact less using SMS, preferring to interact using instant messaging services that need an Internet connection, such as WhatsApp or Telegram. Apart from that, cellphones have GPS, which enables users to navigate via navigation applications.

Today's internet businesses prioritise not just their websites, but also their mobile platforms. Mobile users have overtaken device users for the first time, according to Mobile Marketing Statistics 2015. Additionally, online analytics company Flurry reports that 80 percent of mobile advertising time is spent on apps rather than browsers. This is why internet firms must build mobile applications in order to remain competitive. Through push notifications, a client may get the most up-to-date information through a mobile application on their smartphone.

While grocery shopping might be intimidating, virtually everyone will have to do it at some time. Through the use of the current online retail buying mobile application, a new voice command capability for smartphones has been developed to aid customers with online grocery shopping and to quickly list the things required. Each suggested application comes with a unique set of disadvantages. In conclusion, this project will provide a new menu of voice command mobile applications that will make it simple to list grocery goods utilising the kitchen's voice command system.

1.2 Problem Statement

Voice commands combined with technology are becoming more prevalent in everyday life. As a result of voice search's growth and interest among internet users, eCommerce enterprises must include voice search optimization in their marketing efforts. As an example, consider online grocery buying on the Android platform.

As a result, the current online retail purchasing mobile application must be enhanced to make it simpler for customers to shop for groceries online from the comfort of their homes. Voice commands should be integrated into the current online retail purchasing mobile application system to facilitate the listing of grocery goods. Additionally, by utilising the Grocer App's voice command, users may save time by avoiding the need to travel to the living room if they forget to bring their mobile device with them while listing grocery goods in the kitchen.

Additionally, there are certain complications when a customer must recall kitchen supplies that have run out and wants to purchase them impulsively at home. The answer to this issue is to design a mobile application that supports voice commands and is capable of taking orders for things in real time while also reducing the usage of paper for grocery lists. Hence, voice commands on the grocer's app may assist.

1.3 Project Objective

The objective of this project is to list the groceries through voice commands and take a short time to list the items needed. Specifically, the objectives are as follows:

a) To design a system that uses voice commands to list groceries and display them on a small LCD.

b) To execute the voice command to be displayed directly on the small LCD and Grocer apps at the same time.

c) To evaluate the accuracy of the voice command of the system.

1.4 Scope of Project

The scope of this project are as follows:

a) This system is used for voice command grocery shopping list makers, and it includes some new feature that make it easier to use and save the user time.

b) This application will help the user to list the groceries through voice recognition and takes a short time to list the items needed.

c) The voice recognition module is considered for any 7 voice commands in the library that could be imported into recognizer. It means 7 commands/words are effective at the same time.

d) It requires a Wi-Fi connection to work.

e) When a user adds new data by voice command, the Grocer App updates the list.

f) All data collected by the Grocer App will be stored in the Firebase database.

g) It is an application that is based on the Android platform

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter aims to explore the project's relation to other research in order to do better research and minimize the excessive recurrence of the study's issue areas. All information was gathered from credible sources such as journal papers, books, conference proceedings, and websites. A voice command shopping list maker application on a Arduino based platformis one that allows the user to list groceries using speech recognition and takes a little time to list the things required. This chapter's content contains an explanation of the function and a comparison of the components.

2.2 Microcontroller

A microcontroller (MCU, abbreviation for microcontroller unit) is a kind of microcomputer that is based on a single metal-oxide-semiconductor (MOS) integrated circuit (IC). A microcontroller consists of one or more central processing units (CPUs), memory, and programmable input/output peripherals (Moskowitz, Sanford L, 2016). Additionally, on-chip programme memory in the form of ferroelectric RAM, NOR flash, or an OTP ROM is prevalent, providing a tiny amount of RAM.

In modern use, a microcontroller is similar to, but less complicated than, a system on a chip (SoC). A microcontroller may be one of the components of a system-on-chip (SoC), although it is commonly combined with advanced peripherals such as a graphics processing unit (GPU), a wireless module, or one or more coprocessors (Moskowitz, Sanford L, 2016).

For my project, I used the Arduino Uno microcontroller for research. As a result, the Methodology section discusses the usage of hardware and the kind of microcontroller in this project in more depth.

Figure 2.1 Example of Microcontroller

2.2.1 Comparison between Microcontroller

NodeMCU is an open source development board based on the ESP8266 microcontroller. The ESP-8266 module is a wireless microcontroller board that may be programmed. The ESP8266 Wi-Fi board is a system-on-chip (SOC) with an integrated TCP/IP protocol stack that can connect any secondary microcontroller to a Wi-Fi network. Because the ESP8266 board is capable of hosting an application or offloading all Wi-Fi networking tasks to another application processor, it is well suited for usage as a sensing node capable of sensing data from numerous wirelessly linked IoT sensor nodes and delivering it to a central server.

Figure 2.2 Example of NodeMCU ESP8266

While Arduino is an 8-bit microcontroller development board equipped with a USB programming port for connecting to a computer and other connections for interfacing to other devices such as sensors, motors, speakers, and diodes. It has both input and output pins, with the inputs being either digital (0–13) or analogue (A0–A5), while the output pinsare all digital (0–13). The Arduino board design, as well as the integrated development environment that includes a cross-compiler, a debugger, and a serial monitor for controllingthe inputs and outputs, is open source. Arduino may be powered through a USB cable connected to a computer, a 9V battery, or an external power source.

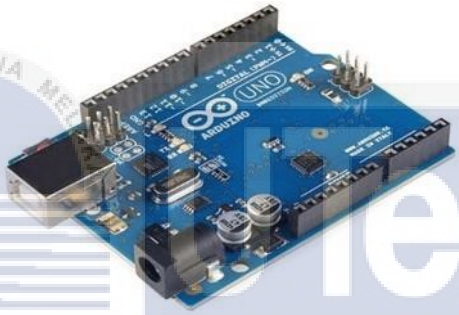


Figure 2.3 Example of Arduino Uno

Table 2.1: This is the comparison between NodeMCU ESP8266 and Arduino Uno.

	NodeMCU ESP8266	Arduino Uno
Advantages	• NodeMCU is one of the easiest to use, since it already has the necessary processing capability to run its applications, and it still has a direct connection to Wi-Fi (D. Bento, 2018) • NodeMCU makes direct references to its libraries, eliminating the need for extra libraries; the method of connection and use is as simple as selecting the device type on the platform. (D. Bento, 2018)	• The Arduino Uno is an excellent learning platform for embedded programming but does not have an integrated Wi-Fi module. (Müller, Mohammed and Kimball, 2015) • It is easy to use, programme and integrate Arduino controllers into electrical applications. (Müller, Mohammed and Kimball, 2015)
	• During the research papers, detect a few references on the NodeMCU, due to the	• Has limited capabilities using the standard libraries. (Müller, Mohammed and Kimball, 2015)

Disadvantages	fact that it is a new product, recently debuted in the market. (D. Bento, 2018) • The number of pins on the NodeMCU ESP8266 is significantly less than the number of ports on the Arduino Uno. (D. Bento, 2018)	• It is based on an 8-bit microprocessor. The Arduino Uno seems to have little computer power. (Müller, Mohammed and Kimball, 2015)

2.3 Mobile Application

A mobile app is a computer programme or software application, commonly called a mobile application or an app, intended to operate on a mobile device, such as a phone, a tablet or a wristwatch. Although applications such as email, calendars, and contact databases were initially developed to aid in productivity, public demand for apps has resulted in rapid development in other areas, such as mobile gaming, factory automation, GPS and location-based services, order-taking, and ticket purchases, resulting in the availability of millions of applications. Mobile applications are an evolution of desktop programmes for use on desktop computers and web applications for use on mobile web browsers rather than directly on the mobile device. (R. Islam and M. Mazumder, 2010).

2.3.1 Mobile Application Development

There are several classification schemes for mobile apps. A frequent technique is to distinguish between native mobile applications, web-based applications, cross-platform applications, and hybrid applications. Native applications are those that are specifically created for a particular mobile platform. As a consequence, an Apple iPhone app will not run on an Android smartphone (Khandeparkar, Gupta and B.Sindhya, 2015). As a consequence, the vast majority of businesses develop apps for a variety of platforms. Professionals build native applications using best-in-class user interface components. This results in increased speed and stability, as well as a more favourable user experience (Khandeparkar, Gupta and B.Sindhya, 2015).

A web application is developed using standard web technologies such as HTML, CSS, and JavaScript. In contrast to offline use, an internet connection is often required for proper action or access to all the features. The cloud stores the vast majority, if not all, of user data (Delia et al., 2015). However, since online methods or apps employ client-server interaction, the response time is slower and the technique is less appealing than the native approach since it is not installed on the devices (Delia et al., 2015).

The hybrid pattern is similar to the web approach in that it makes use of web technologies such as HTML, JavaScript, and CSS but is not browser-based. They instead run in the web container of the device, which enables increased access to device-specific data capabilities given through application programming interfaces (Delia et al., 2015). Finally, it leverages a procedure known as cross-compilation for cross-platform approaches. The source code will be transformed into native binaries, and the cross-compiler will create platform-specific executable code (Abrahim, 2016). These are intended to facilitate the use of web and native technologies across a range of platforms. Additionally, these applications are easier and quicker to develop. It refers to the use of a single codebase that runs across a

variety of mobile operating systems. Despite these advantages, hybrid applications struggle to execute well. Apps commonly fail to maintain a consistent look and feel across several mobile operating systems (Abrahim, 2016).

2.3.2 Mobile Operating System

Now that technology has advanced to its pinnacle, each smartphone has its own operating system built by the manufacturer to ensure device compatibility and support. A mobile operating system (Mobile OS) is a software framework that allows other programmes known as application programmes to run on mobile devices such as personal digital assistants (PDAs), tablets, cellular phones, and smartphones (CMER, (2014). It is known as a mobile operating system. There are various mobile operating systems available today. The two most common operating systems seen on cellphones are Android OS and iOS. These two mobile operating systems use different approaches.

The Android OS for mobile devices is developed by the Open Handset Alliance, which is headed by Google. Google released the Android operating system in November 2007. The bulk of the Android core code is open source and published under the Apache License, but the bulk of the software on the Android devices (such as the Play Store, Google Search, Google Play Services, Google Music, and so on) is proprietary and licensed (NCSU, 2014). Because each Android application runs in its own process with its own Unix User Identifier (UID) and permissions, apps cannot read or write each other's data or code in general. At this stage, the Android system seeks authorisation from the user; a technique for delivering permission dynamically at runtime is not feasible but would improve security transparency (Renner T., (2014).

Figure 2.4 Google Android

iOS (formerly known as iPhone OS) is an Apple Inc. mobile operating system that is exclusive to Apple devices (Gartner) (2017). Apple's iPhone, iPad, iPod Touch, and Apple TV all run on this operating system. It is proprietary and closed source, based on the open source Darwin core operating system. iOS promotes a new kind of user interaction, especially direct manipulation, for small-screen, limited-input devices. iOS is built on Mac OS X and shares the Darwin core with that operating system, which is an open source POSIX-compliant UNIX operating system (NCSU 2014). This verifies that the programme was not modified and enables the runtime to detect if an application has become untrustworthy since it was last used. In contrast to Android applications, iOS applications may only be signed with an official certification (Apple2) (2014).

Figure 2.5 iOS for iPhone

2.4 Mobile Database

The Firebase Realtime Database (Firebase) is a cloud-based NoSQL database that enables real-time data synchronisation and storage between users. To design serverless applications, users cooperate across several devices. Firebase, a Google product launched in 2017, syncs data in real time through Android, iOS, and JavaScript SDKs, enabling expressive searches that grow in size in proportion to the result set. Users who do not need real-time data synchronisation may profit from the one-time reading feature.

To simplify things, the Android application utilises Firebase to store and send data. Because Firebase is free (though it may charge a very tiny yearly cost if the number of users exceeds the free limit), and it supports real-time database systems, it is well suited for them (Sarkar, Gayen and Bilgaiyan, 2018). Fees are quite costly in comparison to other local servers, and performance is totally reliant on the fees paid to these servers. In comparison to the local server, the Firebase server does not discriminate on the basis of price or performance. Firebase is distinguished by a variety of qualities that distinguish it from other servers. Firebase Storage, Firebase Databases, Firebase Authentication, and Firebase Functions are just a few of the advanced features. As seen in Figure 2.6, the Firebase Realtime Database is a cloud-based database management system.

Figure 2.6 a Firebase is Realtime Database

MySQL is the world's most popular open-source database, founded by Oracle. It is compatible with FreeBSD, Linux, OS X, Solaris, and Windows server operating systems. It

is a relational database management system (RDMS) founded on the structured query language (SQL). Unlike Firebase, MySQL is a free and open-source database management system. MySQL may be installed in the cloud or on-premises by users.

MySQL is a widely used database for use in web applications and is a critical component of the well-known LAMP open-source web application software stack (and other "AMP" stacks). MySQL may be developed and installed using source code, but it is most often installed as a binary package unless special customizations are required. In the majority of Linux distributions, the package management system automatically downloads and installs MySQL, although further configuration is often required to modify security and optimization settings (Kazan, F. A., Terziolu, H., and Aaçayak, A. C., 2015).

Figure 2.7 MySQL is a database system used on the web.

2.5 Voice Recognition Mobile Application

While voice recognition is a technology that makes use of desirable equipment and a service that can be handled by speech, this issue impacts a large number of other people as well. In other words, the user talks to devices that employ speech recognition programmes to recognise what is said. The spoken words will be transformed into a digital signal, which will transform sound waves to a sequence of numbers that will then be adjusted to the code-specific code used to identify the word (Kwang B. Lee, Roger A. Grice, 2018). The rapid spread of cellular connectivity has increased the need for speech recognition systems. Speech applications that include speech interfaces, speech recognition, and voice conversation

management may aid users in maintaining focus on their current job without demanding additional effort from their hands or eyes (Kwang B. Lee, Roger A. Grice, 2018). As a result, I researched this voice command application in more depth in this area by doing research on previously researched relevant articles.

2.6 Previously Proposed of Voice Command Android Mobile Application

From the previous study, (Siagian and Hutauruk, 2018) proposed Voice controller mobile android application. The study's goal was to create an interactive android application that also integrated certain capabilities that are already available on android-based smartphones, such as attending to a line calling function and an outgoing calling feature, using voice commands that could be used on a tablet or any android-based phone.Speech recognition contains a data set to translate between user descriptions and the labels in the database, which contains entries on the speech recognition (sounds and patterns) that make up the sound. The system application includes data training, list commands, testing, and help. It accepts voice commands and performs operations in accordance with them. People with limited reading skills who are sighted can also utilise this programme if they are engaged in activities that prohibit them from reading.

The system's goal was to create the needed application using the Android Software Development Kit (SDK) to create a range of tools and APIs. It is based on the Linux kernel, and the Android operating system makes use of C / C + + libraries. As a result, having four data voice classifications in this class enables access to the speech recognition service for basic features such as (e.g., line calls, outgoing call features) (see Figure 2.8). The database includes many forms of speech recognition (sounds and patterns) that comprise the sound. This program is appropriate for the blind because it allows the child blind (CB) to communicate and move independently.

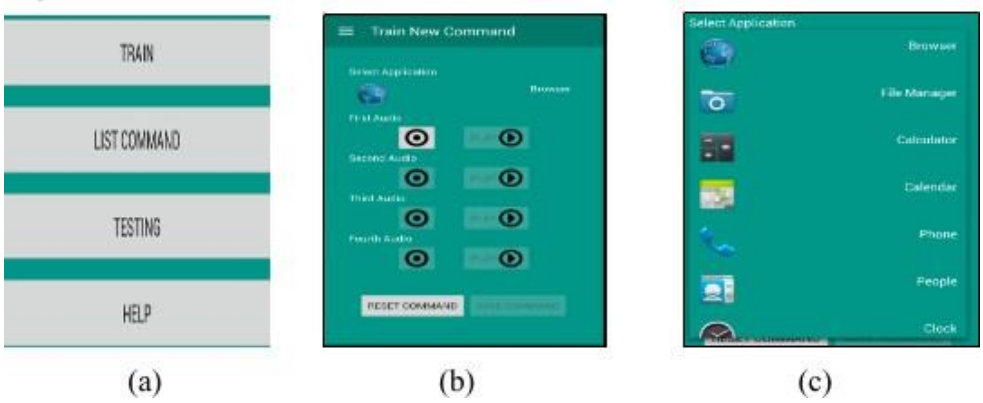


Figure 2.8 Train's menu of fast CB apps, such as a command list, data testing, and assistance.

Elsewhere, from previous study, (Madiba, Owolawi and Mapayi, 2019) proposed Wi-Fi Enabled Speech Automated Guided Vehicle using Android and NodeMCU. The project aimed to analyse the reaction of the vehicle to the supplied spoken instruction for varied directional movement, as well as obstacle recognition and collision avoidance. This paper creates an autonomously guided vehicle employing Arduino Uno as the system's core intelligence and an ultrasonic sensor for obstacle detection and avoidance. It communicates through Wi-Fi utilising voice instructions from an Android smartphone. As a result, the study produced very promising results, with accuracy, sensitivity, and precision rates of 100 percent for directional movement, sensitivity and accuracy rates of 70 percent and 85 percent for obstacle detection in the right direction, and sensitivity and accuracy rates of 80 percent and 90 percent for obstacle detection in the left direction, with precision rates of 100 percent.

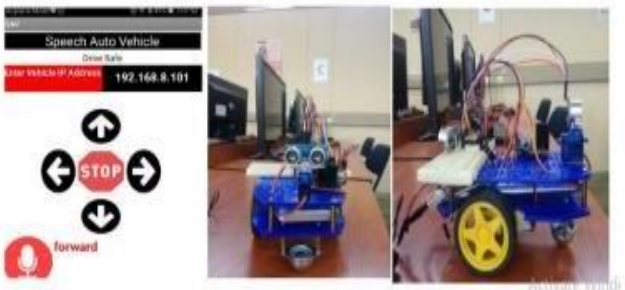


Figure 2.9 Voice command and Directional Control Interfaces With Prototype Design of the Proposed Automated Guided Vehicle

In another study, from a previous study, (Afandi and Sarno, 2020), the proposed paper is Android Application for Advanced Security System based on Voice Recognition, Biometric Authentication, and the Internet of Things. The goal of this study is to have complete control over your home from any of its corners. Home security has long been a key issue for people all over the world. By using the smart home application, the user will receive immediate notification of any intruder (i.e. giving off a burglar alarm). The concept of home automation and security via the Internet of Things is accomplished in this article by utilising a low-cost microcontroller-based Arduino board, an Android device, and Firebase services. Arduino is mostly used for prototyping. It accesses the Internet through a WI-FI hotspot. The Arduino may easily be set to connect to the internet and programmed. To obtain the needed data, we use a PIR sensor and a Flame Sensor on the hardware. As a result, it has played an important and necessary role in the fields of home automation and security, making our lives easier with the smart features inherited from automation.

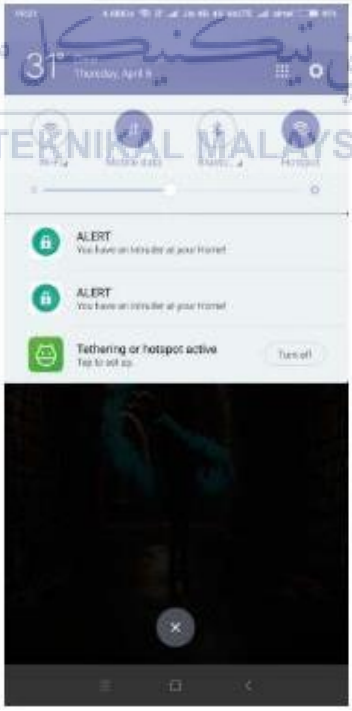


Figure 2.10 Android Application Showing Notification.

According to (Mulhern, N. McCaffrey, N. Beretta, 2013), a proposed paper designing android applications using voice controlled commands: For hands-free interaction with common household devices. This paper is intended to help people with disabilities. It enhances previously developed offline voice control of household equipment using an input-output peripheral interface control processor powered by an application on the Galaxy S Android mobile phone. The application was created to identify key word instructions and deliver signals to the appropriate pins on the input-output board.

The execution of a command was enabled via the smart phone's interface with a television remote. The transfer station displayed contains the majority of the hardware components, which are the Android Galaxy S Smart Phone, the Direct Television remote and the Generic Television. The Eclipse software development kit (SDK) is used as the computer programming environment for system development. As a result, even in the presence of ambient noise, high-quality speech recognition and command execution may be performed at greater distances. This project will continue to provide a socially-current method of wireless voice control to ease everyday domestic tasks. The gadget also has the potential for major use in assistive technology domains, boosting users' domestic freedom.

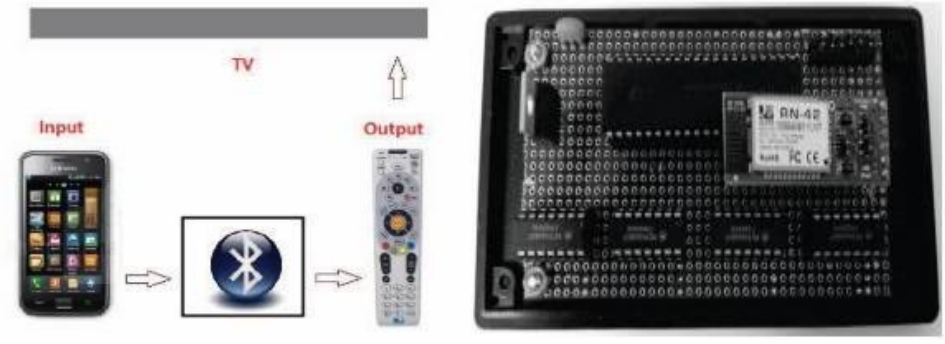


Figure 2.11 Android Galaxy S Smartphone (Left), Bluetooth Transfer Station (Middle), Direct TV Remote (Right), Generic Television (Top), Bluetooth Transfer Station (Right)

Elsewhere, Home Automation System Based Mobile Application is proposed by (Suesaowaluk, 2020) which work includes the development of a mobile application that serves as an end-user interface, allowing users to remotely control a household appliance by performing two separate tasks: first, turning on the appliance and second, turning it off. Voice command and graphical user interface through button pushing are two types of user interface. Effective work serves the needs of the family and the elderly first and foremost, and it may also assist disabled individuals by allowing them to live independently of their family members. Second, regular citizens may construct a system for their homes by expanding the fundamental communication equipment they already own at an affordable cost.

The components utilised in this system include ESP8266 Wi-Fi module, relay switches, and four Chanel input ports, along with a Graphical User Interface (GUI) made by Blynk, a smartphone application, and voice control using the Google Assistant. As the result, users with the ability to control and manage appliances located within their homes. Additionally, homeowners who choose to turn on and off certain electrical appliances may do it using a command rather than having direct access to the equipment. This technique is sensible, and it benefits the aged, the disabled, and the general public.

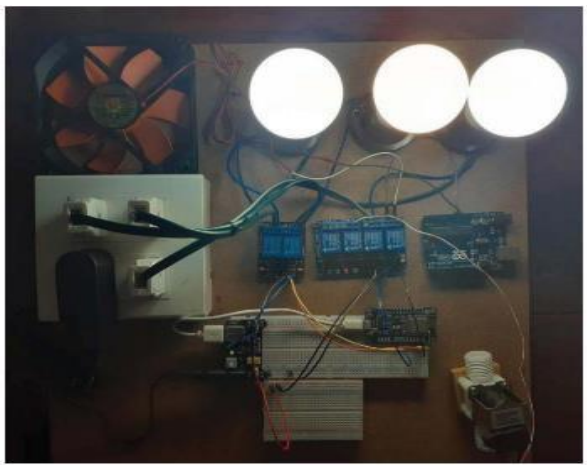


Figure 2.12 Experiment with a suite of mobile applications based on a home automation system.

2.7 Summary of the previous voice command projects

Table 2.2 Summary of the previous voice command projects.

No.	Author	Title	Components	Method	Advantages	Disadvantages
1	(Khotimah, Khusnul Santoso, Agus Budi Ma'Arif, Miftahul Azhiimah, Alfiantin Noor Suprianto, Bambang Sumbawati, Meini Sondang Rijanto, Tri, 2020)	Validation of Voice Recognition in Various Google Voice Languages using Voice Recognition Module V3 Based on Microcontroller	1. The Voice Recognition Module V3 is used to match voice commands 2. Arduino Uno Atmega328, LED as an actuator 3. LCD 16 x 2 4. mobile phone (google voice) as a voice source 5. Arduino IDE software	This Voice Recognition Module (V3) employs user voice instructions. Voice instructions are utilised as a code to turn on and off the lights. Voice commands are used as examples, e.g. the phrases "lights on" to turn on the lights, and "lights off" to turn them off. The command would be	The clarity with which voice instructions are spoken has an effect on their success rate.	The further the distance between the microphone and the Google voice speaker is, the lower successful voice commands are.

			6. Voice Recognition Module V3's voice command database	formed using 9 (nine) different languages that are picked randomly and based on Google Voice. Indonesian, English, Chinese, Japanese, Korean, Swedish, Italian, Latin, German.		
2	(Isyanto, Haris Arifin, Ajib Setyo Suryanegara, Muhammad, 2020)	Design and Implementation of IoT-Based Smart Home Voice Commands for disabled people	1. Arduino Uno Rev 3 2. Modem Wireless 3. Relay 4. Power Supply Unit (PSU)	Users do not have to move to activate or deactivate electrical devices. When the pronunciation is accurate, the Google Assistant programme will receive voice instructions.	The design is helps those with disabilities engage with their environment.	There are significant obstacles to creating devices employing Wireless Sensor Networks.

		using Google Assistant	5. Output socket to control electrical equipment. 6. Android smartphone with Google Assistant	Without the need to type text messages, voice commands are more convenient to use.		
3	(Lee, Kwang B. Grice, Roger A., 2006)	The design and development of user interfaces for voice application in mobile devices	1. Wireless Network 2. Personal digital assistants 3. Mobile computing 4. Mobile devices 5. Voice applications 6. Voice-based interfaces 7. Dictionary-and-text file database	Real-world mobile device applications are built using a commercial speech engine and then usability testing is used to improve their performance. In addition, the system utilises voice interfaces and speech recognition algorithms developed in	Be completely, naturally, and effortless to use Without little training, most humans can use speech apps to perform many tasks, and so this large range of applications	Until now, commercial mobile devices have not yet been applied in this manner. Since the application's prototype is limited to a single tiny domain, it may not respond to voice

				earlier and in-progress work to build the mobile application.	will be useful in a smart city environment.	instructions and natural speech from the actual world. The environment of the field can change, impairing speech accuracy. The application would be less advanced if we didn't use noise control.
4	(Aktar, Nasrin Jaharr, Israt Lala, Bijoya, 2019)	Voice Recognition based intelligent Wheelchair and	1. Voice recognition module, V3 2. GPS module 3. Mobile application	A wheelchair is fitted with a GPS module, and with this feature, the patient is able to control the	The Firebase-supported system is quick and inexpensive and also	Didn't use noise control.

		GPS Tracking System	4. Arduino with Wi-Fi module 5. Firebase 6. IR Sensor 7. Handicapped people with wheelchair	wheelchair with voice instructions. To identify the patient's voice, the voice module V3 is used. This kit translates voice commands into hexadecimal numbers and then transfers that data to the wheelchair's Wi-Fi module. The Wi-Fi module is in contact with the motor driver IC, which tells it to move the wheels in the specified direction.	much more user-friendly than the typical GSM-based navigation systems.	

5	(Suesaowaluk, Poonphon, 2020)	Home Automation System Based Mobile Application	1. Arduino Uno 2. Graphical User Interface (GUI) by Blynk App 3. Voice Control Google Assistant 4. Relay 2 and 4 Chanel 5. Smartphone application	This system can be managed remotely via the internet as well as locally via the Wi-Fi system. The system is divided into two modules: a mobile user interface that enables command of the system, and a GUI created from button pushing for both voice control and a graphical user interface. Module two is an experimental set of home appliance-linked devices and development tools.	Benefits first, elderly families, elderly people, and disabled people can empower themselves so that they are the least burden to their family members. Secondly, the general public can construct their own systems by expanding basic communication equipment to the necessary capacity.	The number of various equipment, such as air conditioners and additional light bulbs, would be expanded to match the demands of growing consumption.

2.8 Summary

The research papers on the existing voice command projects provided us with some ideas for our creation. Additionally, this chapter highlights the technology and critical components utilised in the development of this product's hardware and software. According to the literature analysis, each approach utilised in the study has distinct benefits and limitations that make it appropriate for the project.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter aims to cover all the details of the whole system from the beginning until the end, and the flow of steps involved in the development of this "Voice Command Grocery Shopping List Maker based on the Arduino Platform". Other than that, this chapter also discusses all the details about all the equipment used in this project. In this chapter, we also implemented a project methodology or the way this project will develop.

3.2 Methodology

The waterfall model was used to create this project. Each subsequent phase in the creation cycle will begin where the previous one ended. Once a phase's work has been approved, the phase comes to a conclusion and the following phase starts.

The benefits of the waterfall approach are that it allows for the identification of requirements well before programming begins. The waterfall also has the advantage of limiting the number of changes that must be made as the project progresses. Additionally, the Waterfall paradigm is straightforward to implement and comprehend.

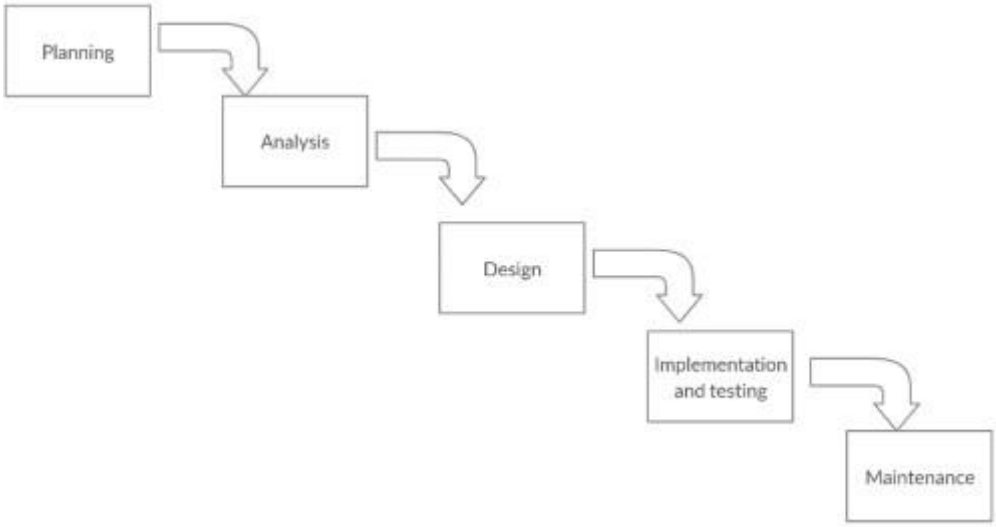


Figure 3.1 Waterfall Model

3.2.1 Planning Phase

The first step in implementing this strategy is to brainstorm ideas and discuss the project title with the supervisor. When providing a project title, the overview of the intended project is often included. A Gantt chart will be prepared throughout this phase to guarantee that all planned activities are carried out as scheduled.

3.2.2 Analysis Phase

This step collects all the application criteria. The problem statement, aim, and scope of the project have all been defined. The majority of precise information is gained via observation. During this step, a literature evaluation of the present research article is also undertaken to ascertain the application's added value.

3.2.3 Design Phase

The framework and configuration specifications were developed with an emphasis on functionality evaluation after the identification of project needs. The design phase provides an

overview of the application. A few diagrams are necessary, including a context diagram, a data flow diagram, and a user interface diagram.

3.2.4 Implementation and Testing Phase

This stage is often referred to as development, building, or coding. After the design is ready, begin developing the interface's code. Using Android Studio, an application will be built throughout this process.

Once the implementation is complete, the application is tested. Any inaccuracies or omissions can be located here. It will perform all the testing operations necessary to guarantee that the application operates as anticipated and satisfies the requirements.

The next step is to connect the hardware with the mobile application that has been built.

3.2.5 Maintenance Phase

System maintenance is the last phase of the waterfall paradigm. During this step, it will guarantee that the application is up and operating in the appropriate environment.

3.3 Whole Planning

In this, Projek Sarjana Muda (PSM) or Bachelor Degree Project (BDP) has been split into two sections, which are PSM 1 and PSM 2. All PSM students need to find a lecturer as their supervisor in PSM 1 and propose their project or select the project given by their supervisor. A meeting is arranged to discuss the project between the supervisor and the student. Then, to identify the hardware components and software that will be developed in this project, research on the project needs to be done.

3.4 Project Overview

In this section, a flowchart will be presented to illustrate the progress of this project while making a thesis and constructing a system to make this project work to achieve the desired results. In addition, this section also presents the system design of the application.

3.4.1 Developing the System Application

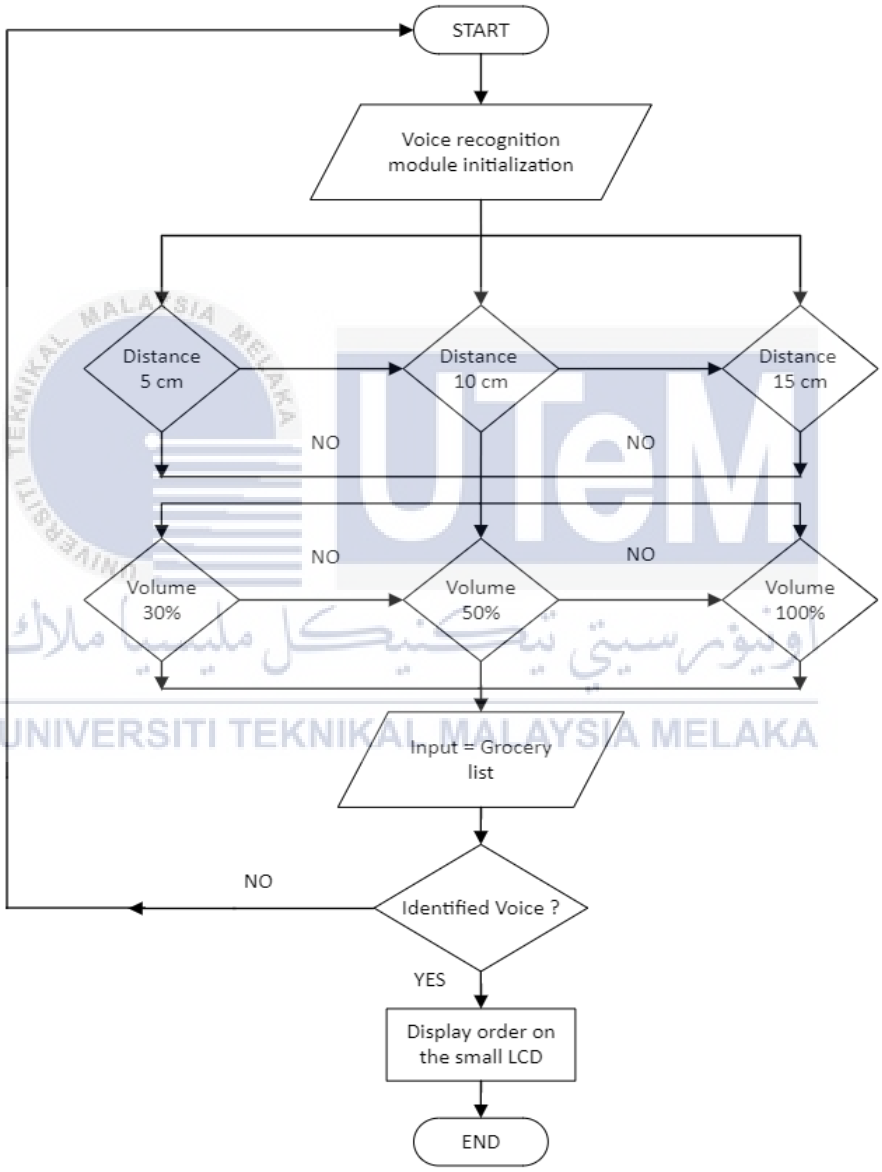


Figure 3.2 Flowchart of Objective 1

According to figure 3.2, the distance and volume of the voice affect the recognition rate of voice commands. The more distant you are from the voice module's microphone, the

more successful the voice commands are. Consequently, the shorter the distance between the speech module microphone, a high number of voice commands will be successful.

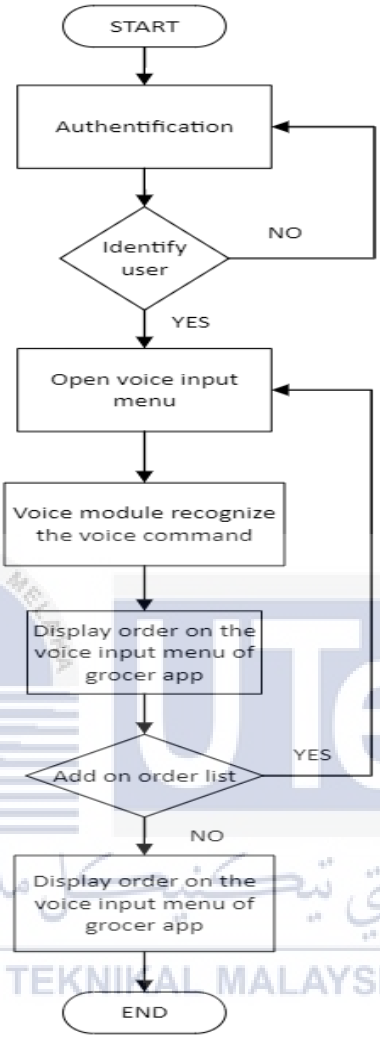


Figure 3.3 Flowchart of overall system

Before using the application, users need to enter self -authentication first. Once the application system has confirmed the user's identity as a valid user, the user can open the main menu list in this application. If the system is unable to identify the user's identity verification, the user has to register as a new user. When the voice module identifies a voice command, the Grocer application will display the order that has been placed and the order will be checked out. If it is successful, the order will be displayed. If there is an addition to the order, the user can return to the main menu list of voice input button of this application.

3.5 System Design

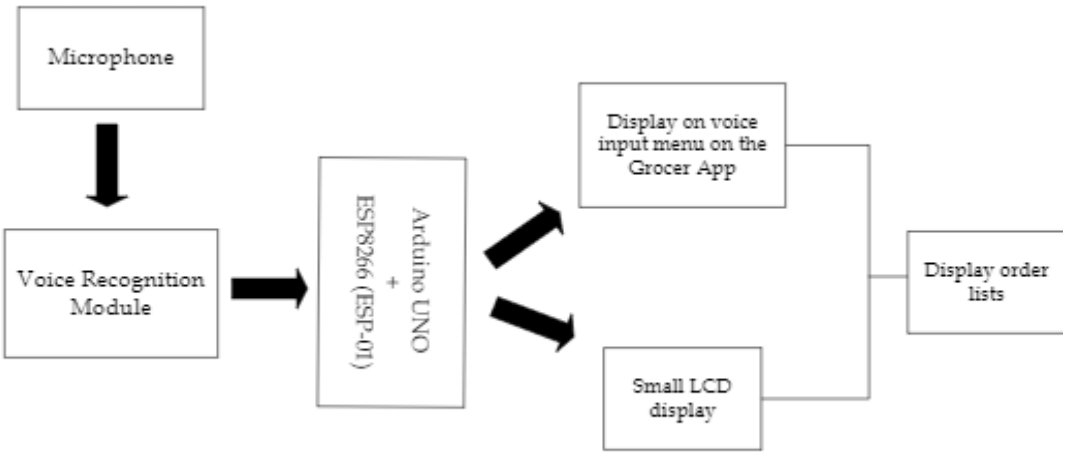


Figure 3.4 Block diagram of voice command application based on Arduino Uno with ESP8266 (ESP-01)

For this system design, the voice module recognizes the voice command by using microphone. The system will check to see if the grocery lists are displayed on the kitchen's small LCD. Otherwise, the voice command is not correctly received and cannot be displayed on an small LCD in the kitchen area for the user, but it will continue.

Following successful verification by the programmable voice module, the user's voice input is added to the Arduino UNO with ESP8266 (ESP-01), allowing the temporary memory of grocery lists to be displayed on a small LCD in the kitchen while also being displayed on the Grocer App's voice input menu. The ESP8266 (ESP-01) WiFi Module is used in this project to connect the hardware and the Android application. When a new piece of data is entered into the list by the user using the voice command system, the list in the Grocer App will be updated accordingly. Specifically, the data is kept in the Firebase database. If the result is acceptable, the lists are displayed in the Grocer app on the user's mobile phone and on a small LCD at the kitchen; otherwise, the Grocer app does not take the command order from the voice module and nothing

happens. To list the items, the user must speak into the microphone several times until the system recognises the voice and allows user to list the things.

3.6 Requirement Analysis

3.6.1 Software Requirement

- **Android Studio**

Among the platforms for building mobile applications is Android Studio, which supports the Java and Kotlin programming languages, among other things. Furthermore, Android Studio enables users to build applications by leveraging APIs and libraries, which makes the development process more straightforward and user-friendly.

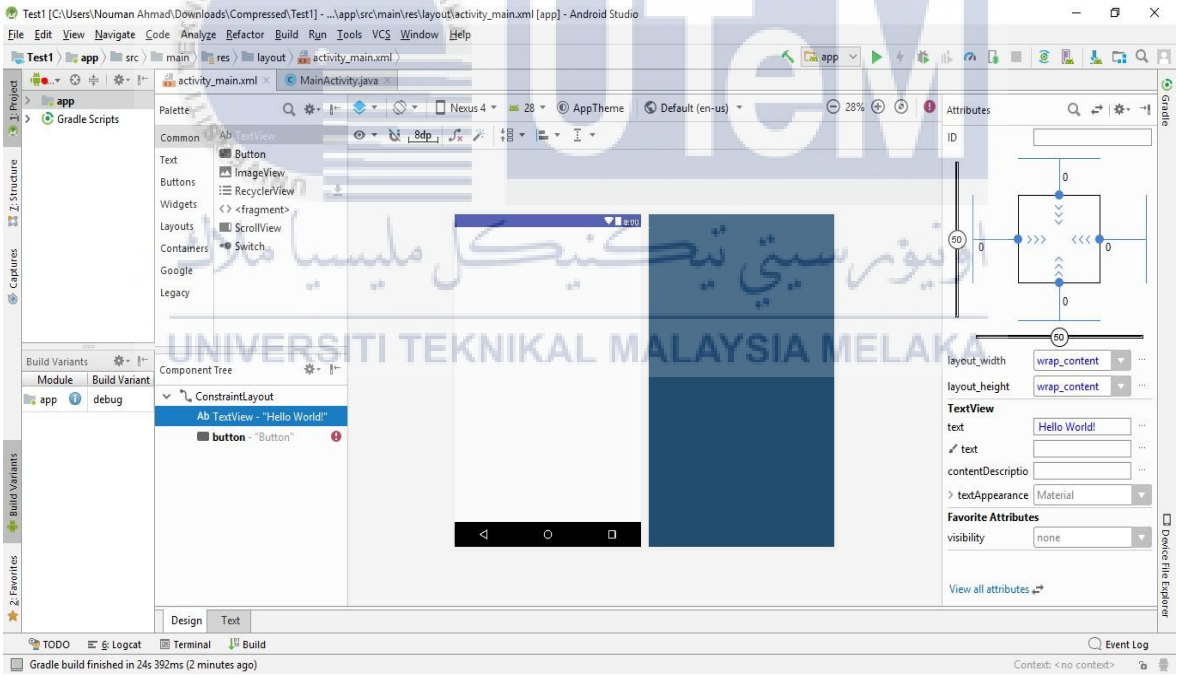


Figure 3.5 Interface of Mobile Application based Android Studio

- **Firebase Database**

To simplify things, the Android application uses Firebase to store and send data. Because Firebase is free (though it may charge a very tiny yearly cost if the number of users exceeds the free limit), and it supports real-time database systems, it is well suited for them (Sarkar, Gayen and Bilgaiyan, 2018). Fees are quite costly in comparison to other local servers, and performance is totally reliant on the fees paid to these servers. In comparison to the local server, the Firebase server does not discriminate on the basis of price or performance. Firebase is distinguished by a variety of qualities that distinguish it from other servers. Firebase Storage, Firebase Databases, Firebase Authentication, and Firebase Functions are just a few of the advanced features.

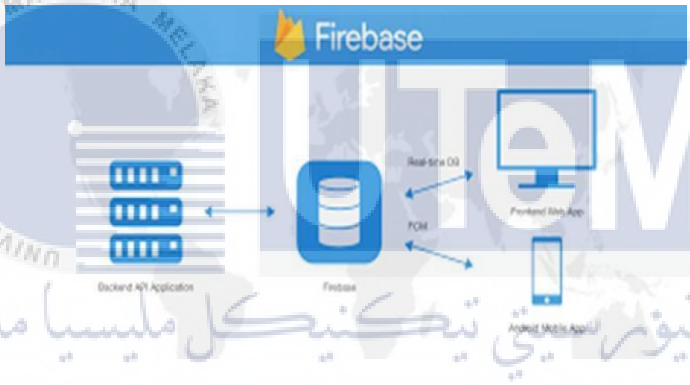


Figure 3.6 a Firebase is Realtime Database.

3.6.2 Hardware Requirement

- **Microcontroller hardware requirements**

Arduino is an 8-bit microcontroller development board equipped with a USB programming port for connecting to a computer and other connections for interfacing to other devices such as sensors, motors, speakers, and diodes. It has both input and output pins, with the inputs being either digital (0–13) or analogue (A0–A5), while the output pins are all digital (0–13). The Arduino board design, as well as the integrated development environment that includes a cross-compiler, a debugger, and a serial monitor for controllingthe inputs and

outputs, is open source. Arduino may be powered through a USB cable connected to a computer, a 9V battery, or an external power source.

It contains 14 digital I/O pins (of which 6 are PWM outputs), 6 analogue inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It comes with everything needed to support the microcontroller; simply connect it to a computer through USB or power it using an AC-to-DC adapter or battery to get started. The FTDI USB-to-serial driver chip is absent from the Uno, as is the case with all previous boards. Instead, it uses an Atmega8U2 that has been coded as a USB-to-serial converter.

Figure 3.7 Arduino Uno with board

- **Voice Recognition Module**

The voice recognition module is a compact, easy-to-use spoken recognition board that is tiny and simple to operate. It is a speaker-dependent module that is capable of recognising up to 80 different voice commands at a time. Any sound may be trained to function as a command. It is necessary for users to train the module before any voice command may be recognised. Voice instructions are placed together in a large group, similar to how books are arranged in a library. Any of the seven voice commands in the library might be loaded into the recognizer programme. This signifies that seven commands areactive at the same time.

There are two ways to operate this board: the Serial Port (which has complete functionality) and the General Input Pins (part of the function). The General Output Pins onthe

board were capable of generating a variety of different waves when a corresponding voice command was received and recognised.

Figure 3.8 Components of the Voice module

- **I2C 16x2 LCD**

Liquid crystal display technology is designed to filter out light. In order to construct an LCD, two pieces of polarised glass (also known as a substrate) are bonded together with a liquid crystal material sandwiched in between. A backlight emits light that flows through the first substrate and into the second substrate. Electrical currents enable the liquid crystal molecules to align at the same time, allowing varying amounts of light to flow through to the second substrate and form the colours and images that you see on the screen

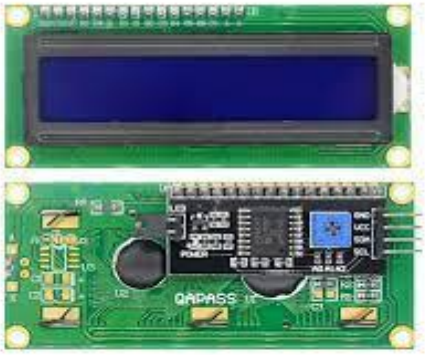


Figure 3.9 I2C 16x2 LCD

- **ESP-01 ESP8266 Serial WiFi Wireless Transceiver Module**

Any microcontroller can connect to the WiFi network using a self-contained system on chip (SOC) with an integrated TCP/IP protocol stack. When combined with another application processor, the ESP8266 can host an application or offload all Wi-Fi networking functions to that processor. Its on-board processing and storage capabilities are adequate for integration with sensors and other application-specific devices via its GPIOs, requiring minimal work beforehand and minimal loading during runtime.

Figure 3.10 ESP8266 (ESP-01) WiFi Module

- **Breadboard**

As a result of the fact that it does not require soldering, the solderless breadboard is reusable. This makes it simple to utilise for temporary prototypes and circuit design experiments, as well as for other purposes. As a result, solderless breadboards are becoming increasingly popular among students and those involved in technology education. This characteristic was not present in earlier breadboard kinds. Unlike stripboards (Veroboards) and other prototyping printed circuit boards, which are used to construct semi-permanent soldered prototypes or one-offs, printed circuit boards used for prototyping cannot be easily reused. Breadboards can be used to prototype a wide range of electronic systems, ranging from simple analogue and digital circuits to entire central processing units and everything in between (CPUs).

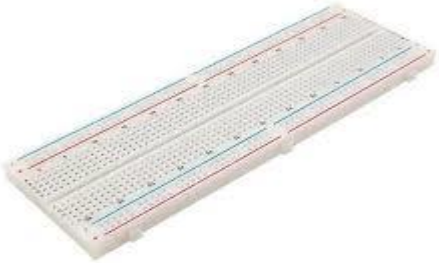


Figure 3.11 Breadboard

- **LED**

A light-emitting diode (LED) is a type of semiconductor light source that generates light when a current is applied to it. During the recombination of electrons in the semiconductor with electron holes, energy in the form of photons is released. The amount of energy required for electrons to pass through the band gap of a semiconductor determines the hue of the light (which corresponds to the energy of the photons). White light can be produced by combining numerous semiconductors or by depositing a coating of light-emitting phosphor on the surface of a semiconductor chip.

Figure 3.12 LED

- **Jumper Wires**

In their most basic form, jumper wires are just wires with connector pins at either end, which allows them to be used to connect two places to each other without the need for soldering. Jumper wires are generally used in conjunction with breadboards and other prototyping tools to make it simple to change the configuration of a circuit as needed. It's a rather straightforward process. Jumper wires, in fact, are among the most fundamental of all electrical components.

Figure 3.13 Jumper Wires Male to Male

3.7 Gantt Chart

Table 2.3 Gantt Chart

No.	Tasks	W1	W2	W3	W4	W5	W6	W7	Mid Semester Break	W8	W9	W10	W11	W12	W13	W14	W15	W16
1	Develop voice recognition mudule																	
2	Train voice recognition module																	
3	Integrate Firebase database into Grocer's App																	
4	Functional verification of Grocer's App and voice recognition module																	
5	Develop the communication between voice recognition module and Grocer's App																	
6	Troubleshoot both the hardware and the software																	
7	First draft report submission and correction of the report. Turnitin < 30%																	
8	Submit report to panel and supervisor with turnitin < 30%																	
9	BDP 2 presentation and Q&A session (12/1/2022 & 13/1/2022)																	
10	Upload the completed softcopy report to ePSM (Before 31/1/2022)																	

3.8 Summary

This chapter covered all of the stages involved in the creation of a voice command grocery shopping list maker based on Arduino Uno. The mobile application and the hardware is expected to be successfully built in accordance with the stages described above, as well as to include certain additional capabilities.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

This chapter presents all the details of the whole system from the beginning until the end and the flow of steps involved in the development of this "Voice Command Grocery Shopping List Maker based on the Arduino Platform". Other than that, this chapter also discusses all the details about all the equipment used in this project. In this chapter, we also implemented a project methodology, or the way this project is developed.

4.2 Project Graphical User Interfaces

This part will go over the overall flow of the applications. This app will continue on to the result pages till the end of the app. The Graphical User Interface (GUI) for this application will be created using Android Studio and the Java programming language.

4.2.1 Main Menu Interface of Grocer Application

Figure 4.1 Main Menu Interface of Grocer Application

Figure 4.1 depicts the main menu page. A number of new buttons have been added, including the Tesco button, Econsave button, Mydin button, Giant button, and a voice command menu for the application's main menu (the voice input button menu), which I designed. It is necessary to use Android Studio in order to develop the application. In orderto manage the voice command, the user first has to check for authentication in order to prove that the user is actually logged into the database as a login device owner. The Firebase API has supplied developers with a feature to simply utilise it.

Here, the voice recognition employs the simplest smartphone Android 1.5 (Cupcake) API. Speech Recognition functions on the basis of analogue sound impulses translated into a digital data form and ends with a text/paragraph. The text/section can also be used as a feedback parameter in

text-to-speech (TTS), so that the user knows that the voice instruction code is well received. Press the button voice and give the command speech.If the received text matches the code, the user is provided with a feedback parameter in a voice confirming that the code has been correctly received, and then the system reads the user's voice instructions and converts them into text form. I built and implemented speech user interfaces such as voice commands in this Grocer App.

After tapping on each mall, the user will be redirected to a different page. There is also a “voice input” and "logout" button, which may be found in the menu.

4.2.2 Voice Input Ordering List

Figure 4.2 Voice input lists on voice input menu

When the user talks into the microphone after tapping the voice input page, the output order list (product list) will be displayed on the screen on Android Studio interface. The product list option allows users to browse through a list of available products. The option to return to the main menu is represented by the button "back to main menu". It will store the product information in the Firebase database.

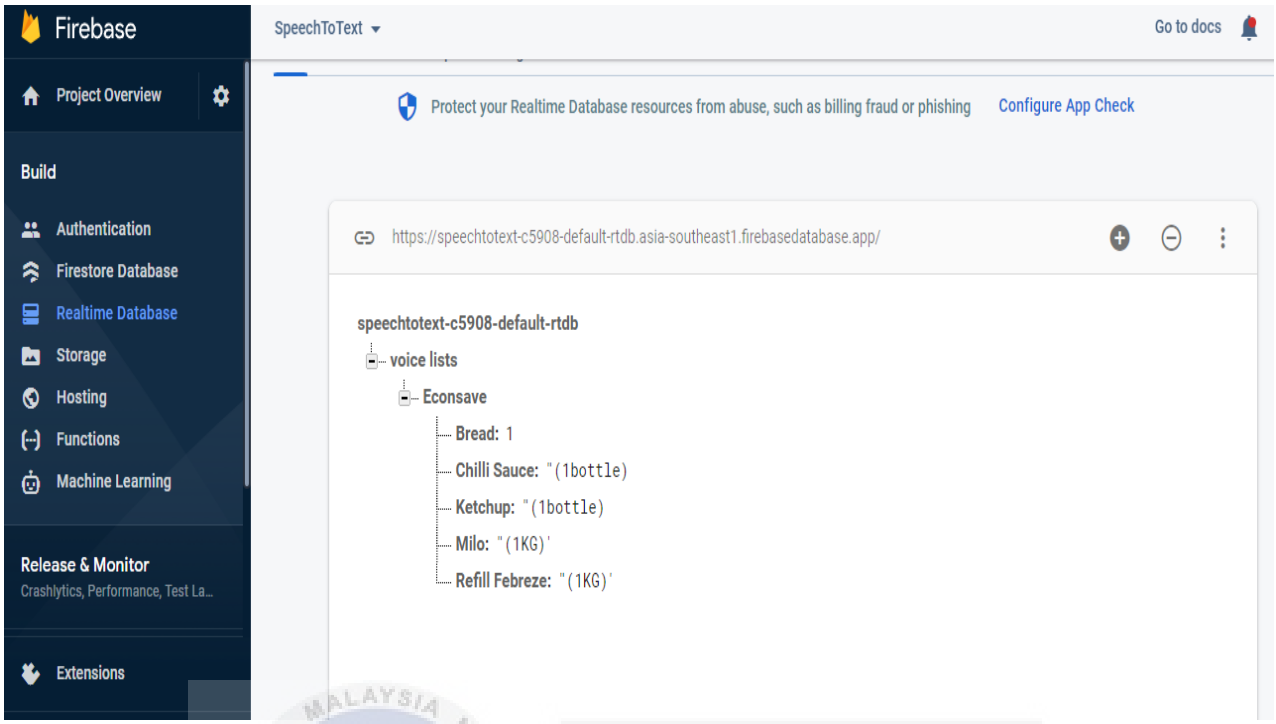


Figure 4.3 Firebase Database stored the output data from the hardware

The Figure 4.3 above show the database that were created using Firebase. This database were created when the user add the data by speaks on microphone and then the voice recognition module sent the input to the microcontroller with ESP8266 (ESP-01) WiFi Module. The user's voice input is added to the Arduino UNO with ESP8266 (ESP-01), allowing the temporary memory of grocery lists to be displayed on a small LCD in the kitchen while also being displayed on the Grocer App's voice input menu. The ESP8266 (ESP-01) WiFi Module is used in this project to connect the hardware and the Android application.

When a new piece of data is entered into the list by the user using the voice command system, the list in the Grocer App will be updated accordingly. Specifically, the data is kept in the Firebase database. If the result is acceptable, the lists are displayed in the Grocer app on the user's mobile phone and on a small LCD in the kitchen; otherwise, the Grocer app does not take the command

order from the voice module and nothing happens. To list the items, the user must speak into the microphone several times until the system recognises the voice and allows user to list the things.

This database uses the recycler view function in Android Studio. The RecyclerView function acts as a list view to display all the lists that have been saved. The RecyclerView was chosen instead of the ListView because the RecyclerView can reuse the same pattern as the database that we have created. Every single item that has been added into the database has its own id that has been automatically created by the system.

4.3 Analysis of the Hardware Implementation

Figure 4.4 Hardware Design

The ultimate result of this study's testing is the percentage of each word's pronunciation that was successful in this project. When the distance between the microphone and the user's voice is taken by 5 cm, the success rate of voice commands utilising the Voice Recognition Module V3 increases.

Table 2.4 Rate of Voice Command Success At 5 cm Distance

		The distance of 5 cm		
No.	**Command**	***30%***	***50%***	***100%***
1.	Econsave	90%	100%	100%
2.	Refill Febreze (1KG)	80%	90%	100%
3.	Ketchup (1bottle)	90%	100%	100%
4.	Chilli Sauce (1bottle)	90%	100%	100%
5.	Bread	100%	100%	100%
6.	Milo (1KG)	90%	100%	100%

From Table 2.4 above, when the distance between the microphone and the user's voice is 5 cm with a full volume (100%), it can be seen that the success rate of voice commands with a full volume (100%) approaches 100% for the voice command at a distance of 5 cm (command). Except for the command Refill Febreze, which had a 90% success rate when delivered at half volume (50%), other voice commands delivered at half volume (50%) had 100% success rates. Because the word "refill" is difficult to pronounce correctly in English, the company has taken this step.

Based on Table 2.4 above, it is known that the success rate of voice instructions is inversely correlated to the distance between the microphone and the user's voice. The greater the distance between the microphone and the user's voice, the less likely it will be that voice instructions will be successfully executed by the system. Consequently, the closer the distance between the microphone and the user's voice is, the higher the chance that voice commands will be successfully executed by the user.

Table 2.5 Rate of Voice Command Success At 10 cm Distance

		The distance of 10 cm		
No.	**Command**	***30%***	***50%***	***100%***
1.	Econsave	0%	100%	100%
2.	Refill Febreze (1KG)	0%	50%	100%
3.	Ketchup (1bottle)	0%	90%	100%
4.	Chilli Sauce (1bottle)	0%	90%	100%
5.	Bread	0%	100%	100%
6.	Milo (1KG)	0%	100%	100%

Table 2.5 shows the success percentage of voice commands when the distance between the microphone and the user's voice is increased by 10 cm. According to Table 2.5, the success rate of voice commands at a distance of 10 cm with full volume (100%) is 100% for all commands tested for the voice command. While voice commands with half the intensity (50%) in command had a success percentage of 100% with the commands Econsave, Bread, and Milo (1KG). 90% of Ketchup (1 bottle) with 90% of Chilli Sauce (1 bottle). The Refill Febreze (1KG) Command indicates a percentage of 50%. Again, it has done so since the word "refill" is unclear to pronounce correctly in English.

Table 2.6 Rate of Voice Command Success At 15 cm Distance

		The distance of 15 cm		
No.	**Command**	***30%***	***50%***	***100%***
1.	Econsave	0%	0%	100%
2.	Refill Febreze (1KG)	0%	0%	100%
3.	Ketchup (1bottle)	0%	0%	100%
4.	Chilli Sauce (1bottle)	0%	0%	100%
5.	Bread	0%	0%	100%
6.	Milo (1KG)	0%	0%	100%

Table 2.6 shows the results of the percentage of voice commands that were successfully recognised by the Voice Recognition Module V3 when the distance between the microphone and the user's voice was increased by 15 cm. Based on the results of Table 2.6 above, it can be concluded that the success rate of voice commands at a distance of 15 cm with full volume (100%) reaches 100% for all tested commands when sent at a distance of 15 cm with full volume (100%).

4.3 Summary

Based on results, the user's voice level on the voice module's microphone is directly proportional to the success rate of voice commands. The user's voice volume impacts the success of voice commands. The success rate of voice commands decreases as the user's voice level increases. So that powerful voice is better. The distance between the microphone and the user's voice is inversely proportional to the success rate of voice command. Voice commands work better when the microphone is closer to the user's mouth. To test the Voice Recognition Module V3, closeness is preferable.

CHAPTER 5

CONCLUSION

5.1 Introduction

This section explores the conclusions that may be drawn from this project, including whether the project's goals and objectives have been met or not. Additionally, this chapter explains the recommendations for this project's future development, which are discussed later in the chapter.

5.2 Conclusion

This project presented a voice command application that can be used in daily life by applying the use of smartphone applications. The suggested techniques and approaches operate perfectly and effectively while needing little hardware. Through the use of voice commands, I have developed an interactive Android application as well as the addition of the Voice Input button to the Grocer's App. In addition, this application helps the user to list groceries through voice recognition and it takes a short time to list the items needed in their daily lives. This project used the programmable voice recognition module, the small LCD 16x2 I2C display and the microcontroller Arduino UNO board. The list of groceries will be linked to the Grocer's App. Whenever the user enters new data into the list via the voice command system, the list in the Grocer App will be updated. All data collected by the Grocer App will be stored in the Firebase database. Moreover, the user can add, remove or overwrite the list by using the Grocer's App.

Except for at a distance of 5 cm and with a user voice volume of 30%, the Voice Recognition Module V3 performs well. For the "Refill Febreze" command, because the vowel pronunciation is unclear. The clarity of voice command pronunciation impacts the command success rate. Even at a distance, all commands are detected when the volume is raised to 100%. Success will diminish if the volume is reduced to 50% and 30% at 10 cm and 15 cm. The Microcontroller-based Voice Recognition Module V3 was used in this investigation to validate voice recognition. This study also employed a normal human voice with no noise. So the research outcomes were more reliable.

5.3 Recommendation

Based on this project, to make a further recommendation, the grocer's app may benefit from the addition of a few additional languages before reaching the main menu. Future development should focus on improving accuracy, which is especially important if the application is used to issue commands via speaking in a noisy setting. This problem, it was discovered, may be partially offset by the use of a headset / earbuds with a built-in microphone. The sensitivity of the microphone on the device is also critical to the effectiveness of the application. It is also possible to design additional language packs for languages, which would allow users in isolated areas, such as those living in rural areas, to benefit from voice controlled hardware and the android application without having to learn a new language and/or accent.

Other than that, future work can be accomplished by voice recognition and data transfer over Bluetooth, both of which are handled totally by an entirely offline system. The study and development will show the capability of modern mobile phone technology and its limitless uses.

REFERENCES

Aktar, N., Jaharr, I. and Lala, B. (2019) 'Voice Recognition based intelligent Wheelchair and GPS Tracking System', *2nd International Conference on Electrical, Computer Communication Engineering, ECCE 2019*, pp. 7–9.

Bento, D. A. C. (2018) 'IoT: Arduino Uno, Results of an experimentaland comparative survey', *International Journal of Advance Research in Computer Science and Management Studies*, 6(1), pp. 46–56.

Chauhan, R. *et al.* (2016) 'Study of implementation of Voice Controlled Wheelchair', *ICACCS 2016 - 3rd International Conference on Advanced Computing and Communication Systems: Bringing to the Table, Futuristic Technologies from Around the Globe*, pp. 44-46.

Isyanto, H., Arifin, A. S. and Suryanegara, M. (2020) 'Design and Implementation of IoT-Based Smart Home Voice Commands for disabled people using Google Assistant', *Proceeding – I CoSTA 2020: 2020 International Conference on Smart Technology and Applications: Empowering Industrial IoT by Implementing Green Technology for Sustainable Development*, pp. 89-94.

Khotimah, K. *et al.* (2020) 'Validation of Voice Recognition in Various Google Voice Languages using Voice Recognition Module V3 Based on Microcontroller', *Proceeding - 2020 3rd International Conference on Vocational Education and Electrical Engineering: Strengthening the framework of Society 5.0 through Innovations in Education, Electrical, Engineering and Informatics Engineering, ICVEE 2020*, pp. 1–6.

Lee, K. B. and Grice, R. A. (2006) 'The design and development of user interfaces for voice application in mobile devices', *IEEE International Professional Communication Conference*, pp. 308–320.

Omyonga, K. and Shibwabo, B. K. (2015) 'The application of real – time voice recognition to

control critical mobile device operations', 2(7), pp. 174–184.

Ooko, S. O. (2019) 'A Comparison of Arduino, Raspberry Pi and ESP8266 Boards', (December), pp. 12-14.

Siagian, P. and Hutauruk, S. (2018) 'Voice controller mobile android application', *IOP Conference Series: Materials Science and Engineering*, 420(1), pp. 82-85.

Suesaowaluk, P. (2020) 'Home Automation System Based Mobile Application', *2020 2nd World Symposium on Artificial Intelligence, WSAI 2020*, pp. 97–102.

Alzahrani, S. M. (2017) 'Sensing for the internet of things and its applications', *Proceedings - 2017 5th International Conference on Future Internet of Things and Cloud Workshops, Wi-Fi Cloud 2017*, 2017-Janua, pp. 88–92.

Madiba, J., Owolawi, P. A. and Mapayi, T. (2019) 'Wi-Fi Enabled Speech Automated Guided Vehicle using Android and Arduino', *Proceedings - 2019 International Multidisciplinary Information Technology and Engineering Conference, IMITEC 2019*, pp. 26–29.

Mulhern, N. *et al.* (2013) 'Designing android applications using voice controlled commands: For hands free interaction with common household devices', *Proceedings of the IEEE Annual Northeast Bioengineering Conference, NEBEC*, pp. 265–266.

Sarkar, S., Gayen, S. and Bilgaiyan, S. (2018) 'Android Based Home Security Systems Using Internet of Things (IoT) and Firebase', *Proceedings of the International Conference on Inventive Research in Computing Applications, ICIRCA 2018*, (Icirca), pp. 102–105.

APPENDICES

Source Code for Android Studio
Main Activity Java for Main Menu

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#596DDD"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#596DDD"
        android:orientation="vertical">

        <RelativeLayout
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_marginStart="102dp"
            android:layout_marginTop="32dp"
            android:layout_marginEnd="20dp">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_centerHorizontal="true"
                android:text="Welcome"
                android:textColor="#FFFFFF"
                android:textSize="50sp"/>

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_centerHorizontal="true"
                android:layout_marginTop="50dp"
                android:text="please select mall"
                android:textColor="#FFFFFF"
                android:textSize="20sp" />
```

```
    </RelativeLayout>

<GridLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:alignmentMode="alignMargins"
        android:columnCount="2"
        android:columnOrderPreserved="false"
        android:rowCount="3">

        <androidx.cardview.widget.CardView

            android:id="@+id/tesco"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_rowWeight="1"
            android:layout_columnWeight="1"
            android:layout_margin="12dp"
            app:cardCornerRadius="12dp"
            app:cardElevation="6dp">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:background="@android:color/white"
                android:gravity="center"
                android:orientation="vertical"
                android:padding="16dp">

                <ImageView
                    android:layout_width="108dp"
                    android:layout_height="85dp"
                    app:srcCompat="@drawable/tesco" />

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="12dp"
                    android:text="Tesco"
                    android:textColor="#000000"
                    android:textSize="22sp" />

            </LinearLayout>
        </androidx.cardview.widget.CardView>

        <androidx.cardview.widget.CardView
```

```
    android:id="@+id/econsave"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_margin="12dp"
    app:cardCornerRadius="12dp"
    app:cardElevation="6dp">

    <LinearLayout

        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@android:color/white"
        android:gravity="center"
        android:orientation="vertical"
        android:padding="16dp">

        <ImageView
            android:layout_width="108dp"
            android:layout_height="80dp"
            app:srcCompat="@drawable/econsave" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="12dp"
            android:text="Econsave"
            android:textColor="#000000"
            android:textSize="20sp" />
```



```
    </LinearLayout>
</androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView
    android:id="@+id/mydin"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_margin="12dp"
    app:cardCornerRadius="12dp"
    app:cardElevation="6dp">

    <LinearLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@android:color/white"
        android:gravity="center"
        android:orientation="vertical"
        android:padding="16dp">

        <ImageView
            android:layout_width="80dp"
            android:layout_height="80dp"
            app:srcCompat="@drawable/mydin" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="12dp"
            android:text="Mydin"
            android:textColor="#000000"
            android:textSize="20dp" />

    </LinearLayout>
</androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView
    android:id="@+id/giant"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_margin="12dp"
    android:background="@android:color/white"
    app:cardCornerRadius="12dp"
    app:cardElevation="6dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical"
        android:padding="16dp"
        tools:ignore="UseCompoundDrawables">

        <ImageView
            android:layout_width="80dp"
            android:layout_height="80dp"
            app:srcCompat="@drawable/giant" />
```

```
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="12dp"
            android:text="Giant"
            android:textColor="#000000"
            android:textSize="20sp" />

    </LinearLayout>
</androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView
    android:id="@+id/voice_input"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_margin="12dp"
    android:background="@android:color/white"
    app:cardCornerRadius="12dp"
    app:cardElevation="6dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical"
        android:padding="16dp"
        tools:ignore="UseCompoundDrawables">

        <ImageView
            android:layout_width="80dp"
            android:layout_height="70dp"
            app:srcCompat="@drawable/voice_input" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="12dp"
            android:text="voice input"
            android:textColor="#000000"
            android:textSize="18sp" />

    </LinearLayout>
</androidx.cardview.widget.CardView>
```

```
        <androidx.cardview.widget.CardView
            android:id="@+id/logout"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_rowWeight="1"
            android:layout_columnWeight="1"
            android:layout_margin="12dp"
            android:background="@android:color/white"
            app:cardCornerRadius="12dp"
            app:cardElevation="6dp">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:gravity="center"
                android:orientation="vertical"
                android:padding="16dp"
                tools:ignore="UseCompoundDrawables">

                <ImageView
                    android:layout_width="80dp"
                    android:layout_height="70dp"
                    app:srcCompat="@drawable/logout" />

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="12dp"
                    android:text="logout"
                    android:textColor="#000000"
                    android:textSize="19sp" />

            </LinearLayout>
        </androidx.cardview.widget.CardView>

    </GridLayout>
  </LinearLayout>

</LinearLayout>
```

```
        holder.textViewID.setText(Integer.toString(employeeModelClass.getId()));
        holder.editText_Name.setText(employeeModelClass.getName());
        holder.editText_Email.setText(employeeModelClass.getEmail());
        holder.editText_Weight.setText(employeeModelClass.getWeight());
        holder.editText_Price.setText(employeeModelClass.getPrice());

        holder.button_edit.setOnClickListener(new View.OnClickListener()
          {@Override
          public void onClick(View v) {
            String stringName =
            holder.editText_Name.getText().toString(); String
            stringEmail = holder.editText_Email.getText().toString();
            String stringWeight =
            holder.editText_Weight.getText().toString();String
            stringPrice = holder.editText_Price.getText().toString();

            databaseHelperClass.updateEmployee(new EmployeeModelClass
 (employeeModelClass.getId(),stringName,stringEmail,stringWeight,stri ngPrice));
            notifyDataSetChan
            ged(); ((Activity)
            context).finish();
            context.startActivity(((Activity) context).getIntent());
  }
});

        holder.button_delete.setOnClickListener(new
          View.OnClickListener() {@Override
          public void onClick(View v) {
            databaseHelperClass.deleteEmployee(employeeModelClass.getId());
            employee.remove(position);
            notifyDataSetChanged();
  }
});

    }

    @Override
    public int
      getItemCount() {
      return
      employee.size();
    }

    public class ViewHolder extends

      RecyclerView.ViewHolder{TextView textViewID;
      EditText
      editText_Email;
      EditText
      editText_Name;
      EditText
```

```
editText_Weight;
EditText
editText_Price;
Button
button_edit;
Button button_delete;

public ViewHolder(@NonNull View itemView) {
```

Firebase Database

```
@Override
public void onUpgrade(FirebaseDatabase db, int oldVersion, int
  newVersion) {db.execFirebase("DROP TABLE IF EXISTS " +
  TABLE_NAME);
  onCreate(db);
}

public void addEmployee(EmployeeModelClass employeeModelClass){
  ContentValues contentValues = new ContentValues();
  contentValues.put(DatabaseHelperClass.NAME,
  employeeModelClass.getName());
  contentValues.put(DatabaseHelperClass.EMAIL,
  employeeModelClass.getEmail());
  contentValues.put(DatabaseHelperClass.WEIGHT,
  employeeModelClass.getWeight());
  contentValues.put(DatabaseHelperClass.PRICE,
  employeeModelClass.getPrice()); FirebaseDatabase =
  this.getWritableDatabase();
  FirebaseDatabase.insert(DatabaseHelperClass.TABLE_NAME,
  null,contentValues);
}

public List<EmployeeModelClass>
  getEmployeeList(){String firebase =
  "select * from " + TABLE_NAME;
  firebaseDatabase =
  this.getReadableDatabase();

  List<EmployeeModelClass> storeEmployee = new
  ArrayList<>();Cursor cursor =
  firebaseDatabase.rawQuery(firebase, null);
  if
    (cursor.moveTo
    First()){do {
      int id =
      Integer.parseInt((cursor.getString(0)
      ));String name = cursor.getString(1);
      String email =
      cursor.getString(2); String
      weight =
      cursor.getString(3);String
      price = cursor.getString(4);
      storeEmployee.add(new
      EmployeeModelClass(id,name,email,weight,price));
    }
    while (cursor.moveToNext());
  }
  cursor.close();
  return storeEmployee;
```

```
}
public void updateEmployee(EmployeeModelClass employeeModelClass){
  ContentValues contentValues = new ContentValues();
  contentValues.put(DatabaseHelperClass.NAME,employeeModelClass.get
  Name());
  contentValues.put(DatabaseHelperClass.EMAIL,employeeModelClass.get
  Email()); contentValues.put(DatabaseHelperClass.WEIGHT,
  employeeModelClass.getWeight());
  contentValues.put(DatabaseHelperClass.PRICE,
  employeeModelClass.getPrice()); firebaseDatabase =
  this.getWritableDatabase();
  FirebaseDatabase.update(TABLE_NAME,contentValues,ID + "= ?" , new
  String[]
        {String.valueOf(employeeModelClass.getId())} );
}

public void deleteEmployee(int id){
```

Source Code for Voice Recognition Module V3

```
#include <SoftwareSerial.h>
#include "VoiceRecognitionV3.h"
VR myVR(2,3);    // 2:RX 3:TX, you can choose your favourite pins.

void printSeperator();
void printSignature(uint8_t *buf, int len);
void printVR(uint8_t *buf);
void printLoad(uint8_t *buf, uint8_t len);
void printTrain(uint8_t *buf, uint8_t len);
void printCheckRecognizer(uint8_t *buf);
void printUserGroup(uint8_t *buf, int len);
void printCheckRecord(uint8_t *buf, int num);
void printCheckRecordAll(uint8_t *buf, int num);
void printSigTrain(uint8_t *buf, uint8_t len);
void printSystemSettings(uint8_t *buf, int len);
void printHelp(void);

#define CMD_BUF_LEN      64+1
#define CMD_NUM     10
typedef int (*cmd_function_t)(int, int);
uint8_t cmd[CMD_BUF_LEN];
uint8_t cmd_cnt;
uint8_t *paraAddr;
int receiveCMD();
int checkCMD(int len);
int checkParaNum(int len);
int findPara(int len, int paraNum, uint8_t **addr);
int compareCMD(uint8_t *para1 , uint8_t *para2, int len);
```

```
int cmdTrain(int len, int paraNum);
int cmdLoad(int len, int paraNum);
int cmdTest(int len, int paraNum);
int cmdVR(int len, int paraNum);
int cmdClear(int len, int paraNum);
int cmdRecord(int len, int paraNum);
int cmdSigTrain(int len, int paraNum);
int cmdGetSig(int len, int paraNum);
int cmdSettings(int len, int paraNum);
int cmdHelp(int len, int paraNum);

const char cmdList[CMD_NUM][10] = {  // command list table
  {
   "train"  }
 ,
  {
   "load"  }
 ,
  {
   "clear"  }
 ,
  {
   "vr"  }
 ,
  {
   "record"  }
 ,
  {
   "sigtrain"  }
 ,
  {
   "getsig"  }
 ,
  {
   "Settings"  }
 ,
  {
   "test"  }
 ,
  {
   "help"  }
 ,
};
const char cmdLen[CMD_NUM]= {    // command length
  5,  //  {"train"},
  4,  //  {"load"},
  5,  //  {"clear"},
  2,  //  {"vr"},
  6,  //  {"record"},
```

```
  8,  //  {"sigtrain"},
  6,  //  {"getsig"},
  8,  //  {"Settings"},
  4,  //  {"test"},
  4,  //  {"help"}
};
cmd_function_t cmdFunction[CMD_NUM]={     // command handle fuction(function pointer
table)
  cmdTrain,
  cmdLoad,
  cmdClear,
  cmdVR,
  cmdRecord,
  cmdSigTrain,
  cmdGetSig,
  cmdSettings,
  cmdTest,
  cmdHelp,
};

/**********************************************************************
*/
/** temprory data */
uint8_t buf[255];
uint8_t records[7]; // save record

void setup(void)
{
  myVR.begin(9600);

  /** initialize */
  Serial.begin(115200);
  Serial.println(F("Elechouse Voice Recognition V3 Module \"train\" sample."));

  printSeperator();
  Serial.println(F("Usage:"));
  printSeperator();
  printHelp();
  printSeperator();
  cmd_cnt = 0;
}

void loop(void)
{
  int len, paraNum, paraLen, i;

  /** receive Serial command */
  len = receiveCMD();
  if(len>0){
    /** check if the received command is valid */
```

```
  if(!checkCMD(len)){

    /** check parameter number of the received command  */
    paraNum = checkParaNum(len);

    /** display the receved command back */
    Serial.write(cmd, len);

    /** find the first parameter */
    paraLen = findPara(len, 1, &paraAddr);

    /** compare the received command with command in the list */
    for(i=0; i<CMD_NUM; i++){
      /** compare command length */
      if(paraLen == cmdLen[i]){
        /** compare command content */
        if( compareCMD(paraAddr, (uint8_t *)cmdList[i], paraLen) == 0 ){
          /** call command function */
          if( cmdFunction[i](len, paraNum) != 0){
            printSeperator();
            Serial.println(F("Command Format Error!"));
            printSeperator();
          }
          break;
        }
      }
    }

    /** command is not supported*/
    if(i == CMD_NUM){
      printSeperator();
      Serial.println(F("Unkonwn command"));
      printSeperator();
    }
  }
  else{
    /** received command is invalid */
    printSeperator();
    Serial.println(F("Command format error"));
    printSeperator();
  }
}

/** try to receive recognize result */
int ret;
ret = myVR.recognize(buf, 50);
if(ret>0){
  /** voice recognized, print result */
  printVR(buf);
}
```

```
}

/**
 * @brief   receive command from Serial.
 * @param   NONE.
 * @retval  command length, if no command receive return -1.
 */
int receiveCMD()
{
  int ret;
  int len;
  unsigned long start_millis;
  start_millis = millis();
  while(1){
    ret = Serial.read();
    if(ret>0){
      start_millis = millis();
      cmd[cmd_cnt] = ret;
      if(cmd[cmd_cnt] == '\n'){
        len = cmd_cnt+1;
        cmd_cnt = 0;
        return len;
      }
      cmd_cnt++;
      if(cmd_cnt == CMD_BUF_LEN){
        cmd_cnt = 0;
        return -1;
      }
    }

    if(millis() - start_millis > 100){
      cmd_cnt = 0;
      return -1;
    }
  }
}

#include <SoftwareSerial.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "VoiceRecognitionV3.h"
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>

LiquidCrystal_I2C lcd(0x27,20,4);
VR myVR(2,3);    // 2:RX 3:TX, you can choose your favourite pins.

#define FIREBASE_HOST "latest-fyp-default-rtdb.asia-southeast1.firebasedatabase.app/"
#define FIREBASE_AUTH "PItwMXEBCrh6eDjDlWb958dslUokwtF9nEtmyLGm"
#define WIFI_SSID "HUAWEI nova 2i"
```

```
#define WIFI_PASSWORD "1234abcd"

void setup() {
  Serial.begin(9600);

  // connect to wifi.
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("connected: ");
  Serial.println(WiFi.localIP());

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}

int n = 0;

void loop() {
  // set value
  Firebase.setFloat("number", 42.0);
  // handle error
  if (Firebase.failed()) {
      Serial.print("setting /number failed:");
      Serial.println(Firebase.error());
      return;
  }
  delay(1000);

  // update value
  Firebase.setFloat("number", 43.0);
  // handle error
  if (Firebase.failed()) {
      Serial.print("setting /number failed:");
      Serial.println(Firebase.error());
      return;
  }
  delay(1000);

  // get value
  Serial.print("number: ");
  Serial.println(Firebase.getFloat("number"));
  delay(1000);

  // remove value
  Firebase.remove("number");
  delay(1000);
```

```
  // set string value
  Firebase.setString("message", "hello world");
  // handle error
  if (Firebase.failed()) {
      Serial.print("setting /message failed:");
      Serial.println(Firebase.error());
      return;
  }
  delay(1000);

  // set bool value
  Firebase.setBool("truth", false);
  // handle error
  if (Firebase.failed()) {
      Serial.print("setting /truth failed:");
      Serial.println(Firebase.error());
      return;
  }
  delay(1000);

  // append a new value to /logs
  String name = Firebase.pushInt("logs", n++);
  // handle error
  if (Firebase.failed()) {
      Serial.print("pushing /logs failed:");
      Serial.println(Firebase.error());
      return;
  }
  Serial.print("pushed: /logs/");
  Serial.println(name);
  delay(1000);
}

uint8_t records[7]; // save record
uint8_t buf[64];

int led1 = 9;
int led2 = 10;
int led3 = 11;
int led4 = 12;
int led5 = 13;

#define led1Record  (0)
#define led2Record  (1)
#define led3Record  (2)
#define led4Record  (3)
#define led5Record  (4)
#define led6Record  (5)
```

```
/**
  @brief   Print signature, if the character is invisible,
           print hexible value instead.
  @param   buf     --> command length
           len     --> number of parameters
*/
void printSignature(uint8_t *buf, int len)
{
  int i;
  for(i=0; i<len; i++){
    if(buf[i]>0x19 && buf[i]<0x7F){
      Serial.write(buf[i]);
    }
    else{
      Serial.print("[");
      Serial.print(buf[i], HEX);
      Serial.print("]");
    }
  }
}

/**
  @brief   Print signature, if the character is invisible,
           print hexible value instead.
  @param   buf  -->  VR module return value when voice is recognized.
           buf[0]  -->  Group mode(FF: None Group, 0x8n: User, 0x0n:System
           buf[1]  -->  number of record which is recognized.
           buf[2]  -->  Recognizer index(position) value of the recognized record.
           buf[3]  -->  Signature length
           buf[4]~buf[n] --> Signature
*/
void printVR(uint8_t *buf)
{
  Serial.println("VR Index\tGroup\tRecordNum\tSignature");

  Serial.print(buf[2], DEC);
  Serial.print("\t\t");

  if(buf[0] == 0xFF){
    Serial.print("NONE");
  }
  else if(buf[0]&0x80){
    Serial.print("UG ");
    Serial.print(buf[0]&(~0x80), DEC);
  }
  else{
    Serial.print("SG ");
    Serial.print(buf[0], DEC);
  }
  Serial.print("\t");
```

```
  Serial.print(buf[1], DEC);
  Serial.print("\t\t");
  if(buf[3]>0){
    printSignature(buf+4, buf[3]);
  }
  else{
    Serial.print("NONE");
  }
  Serial.println("\r\n");
}

void setup()
{
  /** initialize */
  myVR.begin(9600);
  lcd.init();
  lcd.backlight();
  Serial.begin(115200);
  Serial.println("Elechouse Voice Recognition V3 Module\r\nControl LED sample");

  //pinMode(led, OUTPUT);

  if(myVR.clear() == 0){
    Serial.println("Recognizer cleared.");
  }
  else{
    Serial.println("Not find VoiceRecognitionModule.");
    Serial.println("Please check connection and restart Arduino.");
    while(1);
  }
  if(myVR.load((uint8_t)led1Record) >= 0){
    Serial.println("led1Record loaded");
  }
  if(myVR.load((uint8_t)led2Record) >= 0){
    Serial.println("led2Record loaded");
  }
  if(myVR.load((uint8_t)led3Record) >= 0){
    Serial.println("led3Record loaded");
  }
  if(myVR.load((uint8_t)led4Record) >= 0){
    Serial.println("led4Record loaded");
  }
  if(myVR.load((uint8_t)led5Record) >= 0){
    Serial.println("led5Record loaded");
  }
  if(myVR.load((uint8_t)led6Record) >= 0){
    Serial.println("led6Record loaded");
  }
}
```

```
void loop()
{
  int ret;
  ret = myVR.recognize(buf, 50);
  if(ret>0){
    switch(buf[1]){
      case led1Record:
        digitalWrite(led1, HIGH);
        lcd.print("Econsave");
        delay(1000);
        digitalWrite(led1, LOW);
        break;
      case led2Record:
        digitalWrite(led2, HIGH);
        lcd.print("Refill Febreze");
        delay(1000);
        digitalWrite(led2, LOW);
        break;
      case led3Record:
        digitalWrite(led3, HIGH);
        lcd.print("Ketchup");
        delay(1000);
        digitalWrite(led3, LOW);
        break;
      case led4Record:
        digitalWrite(led4, HIGH);
        lcd.print("Chilli Sauce");
        delay(1000);
        digitalWrite(led4, LOW);
        break;
      case led5Record:
        digitalWrite(led5, HIGH);
        lcd.print("Bread");
        delay(1000);
        digitalWrite(led5, LOW);
        break;
      case led6Record:
        lcd.print("Milo");
        delay(1000);
        break;
      default:
        Serial.println("Record function undefined");
        break;
    }
    /** voice recognized */
    printVR(buf);
    lcd.clear();
  }
```