DEVELOPMENT OF MICROCONTROLLER-BASED HOME GROCERY LISTING SYSTEM USING SPEECH RECOGNITION AND BARCODE SCANNER

MOHAMAD FARIS EIZLAN BIN SUHAIMI

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

DEVELOPMENT OF MICROCONTROLLER-BASED HOME GROCERY LISTING SYSTEM USING SPEECH RECOGNITION AND BARCODE SCANNER

MOHAMAD FARIS EIZLAN BIN SUHAIMI

This report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Electronics Engineering Technology (Industrial Electronics) with Honours

Faculty of Electronics and Computer Technology and Engineering

Universiti Teknikal Malaysia Melaka

2025

UNIVERSITI TEKNIKAL MALAYSIA MELAKA
FAKULTI TEKNOLOGI DAN KEJURUTERAAN ELEKTRONIK DAN KOMPUTER

BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II

Tajuk Projek : Development of Microcontroller-Based Home Grocery Listing System using Speech Recognition and Barcode Scanner

Sesi Pengajian : 2024

Saya **MOHAMAD FARIS EIZLAN BIN SUHAIMI** mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

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

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

Disahkan oleh:

.... (TANDATANGAN PENULIS)
Alamat Tetap:

.... (COP DAN TANDATANGAN PENYELIA)
Ts. Dr. Mohd Syafiq bin Mispan
Ketua Program Ijazah Sarjana Muda Teknologi Kejuruteraan Elektronik (Elektronik Industri)
Fakulti Teknologi dan Kejuruteraan Elektronik dan Komputer (FTKEK)
Universiti Teknikal Malaysia Melaka (UTeM)
Jalan Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia

Tarikh: 23rd January, 2025

Tarikh: 23rd January, 2025

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

DECLARATION

I declare that this report entitled "Development of Microcontroller-Based Home Grocery Listing System using Speech Recognition and Barcode Scanner" is the result of my own research except for quotes as cited in the references.

Signature	:	....
Student Name	:	Mohamad Faris Eizlan bin Suhaimi
Date	:	23rd January, 2025

APPROVAL

I declare that I have read this thesis and in my opinion, this thesis is sufficient in terms of scope and quality for the award of Bachelor of Electronics Engineering Technology (Industrial Electronics) with Honours.

Signature	:	....
Supervisor Name	:	Ts. Dr. Mohd Syafiq Bin Mispan
Date	:	23rd January, 2025

Signature	:	....
Co-Supervisor Name (if any)	:	NOT APPLICABLE
Date	:	....

DEDICATION

This project is dedicated to my beloved father, Suhaimi bin Abd. Rahman and mother, Salleha binti Ahmad and my family, whose unwavering support and belief in my potential have been the guiding lights on this academic journey. Their love and sacrifices have not only shaped my character but have also laid the foundation for every page written and every discovery made. To them, I owe an immeasurable debt of gratitude.

ABSTRACT

Home grocery buying and listing is a crucial aspect of household management, ensuring that the kitchen is stocked with all the necessary items. The classic pen-and-paper grocery list is ineffective since it is time-consuming and prone to human error (i.e., omitting items in a grocery list). Therefore, in this study, we proposed a microcontroller-based home grocery listing system using speech recognition and a barcode scanner. The proposed system is implemented using ESP32-S3, speech recognition, and barcode scanning module. The system receives user input through four methods: speech recognition and user entry via a mobile phone, as well as barcode scanning and user entry via a hardware keyboard. The data captured through these methods is stored in memory and subsequently transmitted via a WiFi connection to the mobile application of the home grocery listing system. The mobile application is developed using MIT App Inventor. This system gives a new satisfying experience to the users and a convenient way for them to make a home grocery list.

ABSTRAK

Pembelian dan penyediaan senarai barangan dapur merupakan aspek penting dalam pengurusan rumah tangga untuk memastikan dapur sentiasa dilengkapi dengan barangan keperluan. Penggunaan kaedah tradisional seperti senarai barang menggunakan kertas dan pen kurang berkesan kerana memerlukan masa yang lama dan terdedah kepada kesilapan manusia (contohnya, tertinggal barangan dalam senarai). Oleh itu, dalam kajian ini, kami mencadangkan sistem penyediaan senarai barangan dapur berasaskan mikropengawal menggunakan pengenalan pertuturan dan pengimbas kod bar. Sistem yang dicadangkan ini dilaksanakan menggunakan ESP32-S3, pengenalan pertuturan, dan modul pengimbasan kod bar. Sistem ini menerima input pengguna melalui empat kaedah: pengenalan pertuturan dan kemasukan data melalui telefon mudah alih, serta pengimbasan kod bar dan kemasukan data melalui papan kekunci perkakasan. Data yang diperoleh melalui kaedah ini disimpan dalam memori dan dihantar melalui sambungan WiFi ke aplikasi mudah alih sistem senarai barangan dapur. Aplikasi mudah alih ini dibangunkan menggunakan MIT App Inventor. Sistem ini memberikan pengalaman yang lebih memuaskan kepada pengguna serta cara yang lebih mudah dan efisien untuk menyediakan senarai barangan dapur di rumah.

ACKNOWLEDGEMENT

In the name of Allah SWT, I express my gratitude to the Almighty for granting me the capacity to carry out and complete this project. First and foremost, I would like to thank my family for their unwavering support and prayers for me over the past 14 weeks while I worked to submit my report for this bachelor's degree project. I am very grateful to my loving parents, Suhaimi bin Abd. Rahman and Salleha binti Ahmad, for their words of support and push for perseverance. Asyraf, Najwa, and Danish, my brothers, and sisters, have never left my side and are very dear to me.

I am also would like to express my gratitude to my supervisor, Ts. Dr. Mohd Syafiq Bin Mispan for his precious guidance, words of wisdom, and patience throughout this bachelor's degree project.

I am also indebted to Universiti Teknikal Malaysia Melaka (UTeM) for the financial support through the RM200 claim which enabled me to accomplish the project. Not forgetting my colleague, Mohamad Hakim bin Mujib for their willingness to share his thoughts and ideas regarding the project.

Finally, I would like to thank all the staffs at the Universiti Teknikal Malaysia Melaka (UTeM), fellow colleagues and classmates, the Faculty members, as well as other individuals who are not listed here for being co-operative and helpful.

TABLE OF CONTENTS

	PAGE
DECLARATION	
APPROVAL	
DEDICATION	
ABSTRACT	**i**
ABSTRAK	**ii**
ACKNOWLEDGMENTS	**iii**
TABLE OF CONTENTS	**iv**
LIST OF TABLES	**vi**
LIST OF FIGURES	**vii**
LIST OF ABBREVIATIONS	**ix**
LIST OF APPENDICES	**x**
CHAPTER 1 INTRODUCTION	**1**
1.1 Grocery Listing and Management Behaviour	1
1.2 Problem Statement	3
1.3 Project Objectives	3
1.4 Scope of Work	4
1.5 Report Outline	4
CHAPTER 2 LITERATURE REVIEW	**6**
2.1 Barcode	6
2.1.1 Types of Barcode Technology	7
2.1.2 Benefits of Using Barcode	10
2.2 Speech Recognition vs Voice Recognition	11
2.2.1 Algorithm: Speech to Text Conversion	12
2.2.2 Language Modeling	13
2.3 Known Techniques of a Home Grocery Listing System	14
2.4 Summary	23
CHAPTER 3 METHODOLOGY	**24**
3.1 Description of Methodology - Sustainable Development	24
3.2 Hardware Design: Barcode Scanner	25
3.3 Mobile Application Development	27

3.4 Integration of Hardware and Mobile Application 27
3.5 Gantt Chart 29
3.6 Summary 31

CHAPTER 4 RESULTS AND DISCUSSION 32
4.1 Home Grocery Listing Mobile Application Design 32
4.2 Speech Recognition Configuration 35
4.3 Nutrition Facts App Design 37
4.4 Barcode Scanner, LCD, and Keyboard Configuration 40
4.5 Integration of Hardware and Mobile Application 42
4.6 Functionality Verification 44
4.6.1 Data Entry of Grocery Items Without Barcode 44
4.6.2 Data Entry of Grocery Items With Barcode 46
4.6.3 Voice Recognition Data Entry using Mobile App 49
4.7 Comparison with Previous Techniques 51
4.8 Summary 52

CHAPTER 5 CONCLUSION AND RECOMMENDATIONS 53
5.1 Conclusion 53
5.2 Future Works 54
5.3 Project Commercialization 54

REFERENCES 56

APPENDICES 59

LIST OF TABLES

TABLE	TITLE	PAGE
Table 2.1	Summary of the previously proposed techniques	21
Table 4.1	Connectivity of barcode scanner to ESP32-S3.	41
Table 4.2	Connectivity of LCD to ESP32-S3.	41
Table 4.3	Connectivity of USB host to ESP32-S3.	42
Table 4.4	Data analysis for manual mode testing	45
Table 4.5	Analysis of data entry for grocery items with barcode.	48
Table 4.6	Analysis of voice recognition data entry.	50
Table 4.7	Efficiency comparison with the previous techniques.	52

LIST OF FIGURES

FIGURE	TITLE	PAGE
Figure 1.1	Forgetfullnes ratio during grocery shopping.	2
Figure 1.2	Grocery listing method.	2
Figure 2.1	EAN barcode.	8
Figure 2.2	UPC barcode.	8
Figure 2.3	QR code.	9
Figure 2.4	Top-level block diagram of the grocery level monitoring system [1].	14
Figure 2.5	SIMS prototype.	15
Figure 2.6	Top-level block diagram of the smart inventory management system [2].	16
Figure 2.7	Load cell based inventory management system [3].	17
Figure 2.8	Grocery level measurement using ultrasonic sensor.	18
Figure 2.9	Concept of AR application in grocery management.	19
Figure 2.10	Four phases using RAD approach.	20
Figure 3.1	Top-level block diagram of the home grocery listing system.	25
Figure 3.2	Design steps of home grocery listing hardware system.	26
Figure 3.3	Major hardware components in developing the grocery listing system.	26
Figure 3.4	Design steps of a mobile home grocery listing application.	28
Figure 3.5	Design steps of hardware and mobile application integration.	29
Figure 3.6	Time-line for PSM 1	30
Figure 3.7	Time-line for PSM 2	31
Figure 4.1	Main user interface of home grocery listing mobile application.	33
Figure 4.2	Block-based coding of creating “Item:” and “Add” features.	34
Figure 4.3	Block-based coding of clearing items using TinyDB in the grocery listing application.	35
Figure 4.4	Block-based coding design for speech recognition.	36
Figure 4.5	Speech recognition function.	36
Figure 4.6	User interface of nutrition facts mobile application.	37
Figure 4.7	Nutrition facts screen showing item details after user input.	38
Figure 4.8	Block-based coding design for handling data retrieval and updating the interface	39
Figure 4.9	Schematic Diagram for Grocery Listing System Hardware.	40
Figure 4.10	Home grocery listing system prototype.	43
Figure 4.11	Implementation of Web1 component to retrieve and display server data.	44
Figure 4.12	Screens displayed on the 16x2 LCD during Manual Mode operation.	46
Figure 4.13	Screens displayed on the 16x2 LCD during Manual Mode operation.	47
Figure 4.14	Registered items displayed with updated quantities.	47
Figure 4.15	Corresponding list in the mobile application.	49

Figure 4.16 Corresponding list in the mobile application. 51

LIST OF ABBREVIATIONS

1D	-	One-dimensional.
2D	-	Two-dimensional.
ADC	-	Analogue-to-digital Converter.
AR	-	Augmented Reality.
ASR	-	Automatic Speech Recognition.
EAN	-	European Article Number
HMMs	-	Hidden Markov Models
LCD	-	Liquid Crystal Display.
LDR	-	Light Dependent Resistor.
LED	-	Light Emitting Diode.
LSTM	-	Long short-term memory.
QR	-	Quick Response
RAD	-	Rapid Application Development.
RNN	-	Recurrent neural networks.
SIMS	-	Smart Inventory Management System.
UPC	-	Universal Product Code.

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
Appendix A	Full code of grocery listing system	59
Appendix B	Block codes in MIT App Inventor	73

CHAPTER 1

INTRODUCTION

This chapter describes the project background, problem statement, objectives, and scope of work in developing a microcontroller-based home grocery listing system using speech recognition and barcode scanner.

1.1 Grocery Listing and Management Behaviour

Choosing and buying groceries is frequently assisted by a list to make sure nothing is forgotten. Grocery listing and inventory management have become focal points of study due to the dynamic shifts in consumer behavior. Unplanned grocery shopping may necessitate multiple journeys to the supermarket and could take a lot of time. Many people frequently forget what they meant to buy. Figure 1.1 depicts the survey outcomes conducted in [4] which show that 40.6% of shoppers forget about what they have planned to buy and some wind up being unhappy with their purchases [4]. The findings show that improper decision-making and management behavior on grocery lists have a detrimental effect on the consumer.

To overcome unplanned grocery shopping, people tend to create grocery lists and employ various methods such as paper lists, memorization, and smartphone applications [4]. Each approach has its advantages and drawbacks. Most people still use a piece of paper as a grocery listing method as depicted in Figure 1.2. Grocery listing using paper offers a significant visual reminder but can be easily lost or forgotten. Memorize is a convenient and

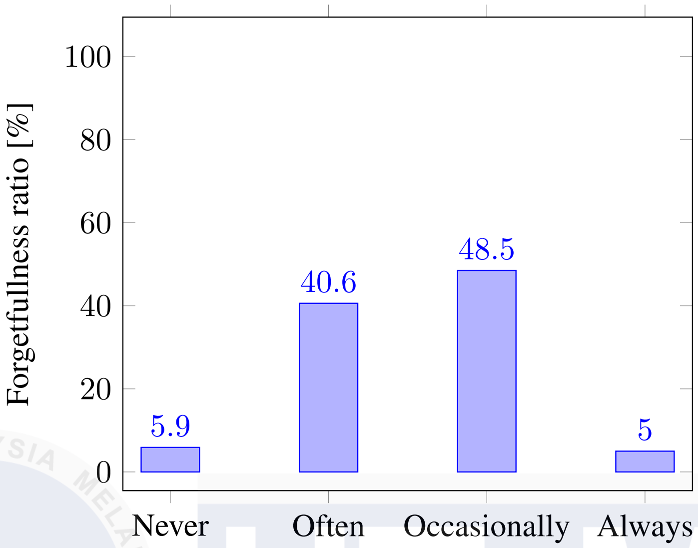


Figure 1.1: Forgetfullnes ratio during grocery shopping.

easiest grocery listing method but prone to forgetting items. Smartphone applications enable efficient grocery listing management, recommendations, and location-based features but require device access. Therefore, the choice depends on individual preferences and lifestyle needs.

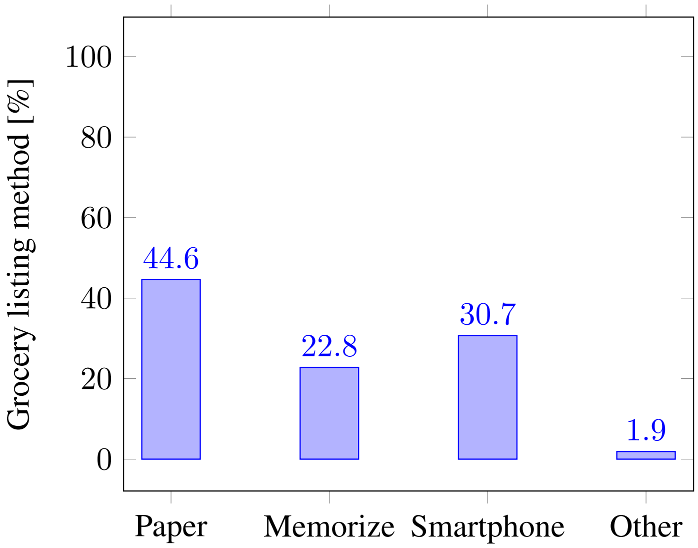


Figure 1.2: Grocery listing method.

1.2 Problem Statement

When planning for grocery items shopping, consumers always face difficulties in choosing an efficient method to do grocery items listing or management. With the current situation of rising grocery prices, an efficient method of grocery listing is demanded such that consumers can have better grocery budgeting. An inefficient method of grocery listing leads to unnecessary buying or missing items during grocery shopping which can cause stress and frustration. Consumers require a systematic and efficient grocery listing system that helps them buy only the necessary items according to the budget of the particular month.

Several techniques have been proposed in the past to develop the grocery listing system which mainly focuses on using sensors [4, 5, 2, 6, 3], smart-phone and IoT applications [7, 8, 1, 9, 10]. Nevertheless, the previous techniques experience major drawbacks such as sensor inaccuracies, high cost due to hardware complexity, and inefficient techniques. To overcome the aforementioned issues, therefore, in this project, we proposed a home grocery listing system that is developed based on a barcode scanner and speech recognition techniques. The grocery listing system is also equipped with a mobile application developed using the MIT app inventor. Users can simply check the listed groceries via smartphone.

1.3 Project Objectives

The main aim of this project is to propose a microcontroller-based home grocery listing system using speech recognition and a barcode scanner. Specifically, the objectives are as follows:

1. To develop the hardware of the home grocery listing system using barcode scanner and ESP32-S3.

2. To design a mobile application for listing the grocery using MIT App Inventor including the speech recognition feature.

3. To integrate the hardware and mobile application of the home grocery listing system and verify their functionality.

1.4 Scope of Work

The project scope of developing a microcontroller-based home grocery listing system using speech recognition and barcode scanner is as follows:

1. The project focuses on implementing a barcode scanning feature and speech recognition feature to enable users to add grocery items to their lists.

2. The project should incorporate a speech recognition that uses the English language to interpret and process user speech for adding items to the grocery list.

3. MIT App Inventor is used to develop the mobile application.

1.5 Report Outline

Chapter 2 provides an overview of the previously proposed techniques for the home grocery listing system. The advantages and disadvantages of the previously proposed techniques are also discussed in this chapter.

Chapter 3 describes the methodology and design steps to develop a home grocery listing system. Each design step to achieve the objective is illustrated in the flow chart diagram and it is thoroughly discussed in this chapter. Besides, the top-level block diagram of the developed system and the Gantt charts are also presented in this chapter.

Chapter 4 discusses the implementation and validation of the proposed home grocery listing system. The system is developed based on the speech recognition and barcode scanner. The performance of the proposed system is analyzed and presented in this chapter.

Chapter 5 concludes the project findings in this report. Suggestions for future work directions and potential commercialization are also provided.

CHAPTER 2

LITERATURE REVIEW

This chapter provides a broad overview of the project related to the topic in this report. Besides, the relevant literature is critically discussed and presented later in this chapter.

2.1 Barcode

A barcode is a machine-readable representation of text and numbers, consisting of bars and spaces. These stripes are commonly found on product packaging in convenience shops, supermarkets, and other retail establishments. Barcodes are made up of gaps and bars that can be read by an optical barcode scanner and have different widths. Currently, barcodes are the most widely used method for automatically communicating package and object information. Both the academic and business worlds have shown a great deal of interest in barcode identification and analysis. Despite its initial proposal being made more than seven decades ago, barcode technology is actively and extensively utilized today [11].

Barcodes provide an encrypted data visualization method. Conventional one dimensional (1D) barcodes use parallel line widths and spacing to express data. Two-dimensional (2D) barcodes offer an improved form of barcode known as matrix codes, which are represented in various forms. The distinctive patterns of 2D barcodes allow them to store data on both vertical and horizontal axes, offering greater data storage capacity than that of 1D barcodes.

2.1.1 Types of Barcode Technology

Today, 1D or linear barcodes are still the most common types of barcodes. They are typically used on retail items and product packaging. Traditional 1D barcodes use parallel lines to encode data. While linear barcodes have many benefits, they also have several fundamental drawbacks. These include limited data storage capacity and crucial encoding problems related to barcode breakage and distortion. Some examples of popular 1D barcodes include the Universal Product Code (UPC), extensively utilized in North America, and the European Article Number (EAN), which is more commonly found in Europe and other parts of the world [12]. However, the increasing demand for enhanced data storage, versatility, and improved scanning capabilities is driving a noticeable shift from 1D barcodes to 2D barcodes.

Figure 2.1 depicts the EAN barcode which widely recognized in Europe and the most commonly used barcode in Malaysia [13]. According to [14], this type of barcode is part of the Global Standard 1 system, which is widely used for retail products globally and has the capacity to encode up to 13 digits. EAN-13 barcodes are recognized for their 13-digit format, which includes a country code, manufacturer code, and product code. Malaysian products typically use the country code prefix ”955” within the EAN-13 barcode. This system ensures compatibility with international retail and logistics systems. The widespread use of EAN-13 barcodes helps streamline product identification and inventory management in Malaysian retail environments.

Figure 2.1: EAN barcode.

A similar barcode used primarily in North America is the UPC, also known as the Global Standards 1. UPC is a widely used barcode symbology that helps track trade items in stores worldwide . These codes consist of bars and spaces arranged in a specific pattern, representing a particular product as illustrated in Figure 2.2. The UPC format typically has 12 digits and is used in countries like the United States, Canada, the United Kingdom, Australia, and New Zealand [12]. Despite newer technologies, UPCs remain popular due to their simplicity and cost-effectiveness.

Figure 2.2: UPC barcode.

According to [15], 2D barcodes are essential part of modern data encoding and tracking systems. Unlike traditional 1D barcodes, which encode data in a linear format, 2D barcodes use patterns of squares, hexagons, dots, and other shapes to represent information. These 2D barcodes can hold significantly more data while still appearing physically smaller. Quick Response (QR) code as depicted in Figure 2.3 is the most widely used 2D barcodes globally [15]. These matrix barcodes can store various data types, including alphanumeric and binary data. They are extensively used on mobile devices. A single QR code can accommodate 1,817 Chinese characters, 4,296 Latin letters, or 7,089 integers. QR codes have four error correction levels [16], which means that even if damaged, they can often still be read correctly.

Figure 2.3: QR code.

In Malaysia, the most commonly used barcode for products is the EAN-13 barcode. This type of barcode is part of the GS1 system, which is widely used for retail products globally. EAN-13 barcodes are recognized for their 13-digit format, which includes a country code, manufacturer code, and product code. Malaysian products typically use the country code prefix “955” within the EAN-13 barcode. This system ensures compatibility with international retail and logistics systems. The widespread use of EAN-13 barcodes helps streamline product identification and inventory management in Malaysian retail environments.

2.1.2 Benefits of Using Barcode

Barcodes offer significant advantages in terms of efficiency and accuracy [17]. When products are labeled with barcodes, they can be quickly scanned at checkout counters, inventory management systems, and warehouses, reducing the time spent manually entering data and minimizing human error. Barcodes also enable real-time tracking of inventory levels, ensuring that businesses can maintain optimal stock levels without overstocking or running out of essential items. Whether in retail, logistics, or healthcare, barcodes enhance operational efficiency by automating data capture and reducing the risk of mistakes.

Implementing barcode systems can lead to cost savings for businesses by automating inventory management and reducing labor costs associated with manual data entry and inventory tracking [11]. Barcodes also facilitate better decision-making by providing accurate and up-to-date information. Businesses can analyze sales trends, monitor stock levels, and identify slow-moving items more effectively. Additionally, barcode data can integrate seamlessly with other systems, such as point-of-sale terminals, supply chain management software, and customer relationship management tools. Overall, the adoption of barcodes improves data accessibility, enhances decision-making, and contributes to overall cost efficiency.

2.2 Speech Recognition vs Voice Recognition

Voice recognition technology is a form of dynamic technological development that utilizes biometric systems from the human voice. Voice recognition technology goes beyond transcribing spoken words; it focuses on analyzing vocal traits to verify individual identities. Each person has a distinct voiceprint, which encompasses factors such as pitch, tone, accent, and speech patterns. Voice recognition systems learn and recognize these individual profiles. It can also be developed and applied to support the automation of household appliances, such as automatically cooling the room, opening doors, turning on lights, and activating electronic devices [18].

Computers can now translate spoken language into written text thanks to a technical miracle called speech recognition, sometimes referred to as automated speech recognition (ASR) [19]. It entails the steps of audio input analysis, spoken word extraction, and written word transformation. Sophisticated algorithms and linguistic models are employed by speech recognition systems to attain precise transciption. Though speech and voice recognition operate in distinct ways, they are tightly connected to offer a wide range of cross-functional features that enhance our everyday lives and open up new avenues for development.

The main difference between speech recognition and voice recognition is their purpose. Speech recognition is focused on transcribing spoken words, while voice recognition aims to identify and authenticate a person based on their vocal traits. Voice recognition systems prioritize identifying individuals based on their voiceprint, while speech recognition systems are mainly concerned with accurately transcribing spoken words. Voice recognition technology relies on complex algorithms to assess and compare sound parameters for verification, while speech recognition technology requires strong language models and extensive training to effectively convert speech into written text [20].

2.2.1 Algorithm: Speech to Text Conversion

Speech-to-text conversion, or the capacity to turn spoken words into written text, is becoming a crucial feature of many modern applications, from interactive voice response systems and virtual assistants to accessibility supports and transcription services. In addition to improving user experience by allowing hands-free control, this technology increases accessibility to digital material for people with physical or hearing disabilities. ASR, a complex system that combines acoustic modeling, feature extraction, and language processing to reliably transcribe spoken words into text, lies at the heart of this transformational technology [21]. With improvements in accuracy and efficiency, particularly in deep learning, machine learning has led to a considerable evolution in ASR systems.

The speech recognition system receives audio input through a microphone or other audio equipment. The audio input undergoes pre-processing to minimize background noise, enhance clarity, and standardize the audio signal. Using acoustic modeling techniques, the system interprets and analyzes the audio input. This involves breaking down speech into smaller segments called phonemes and matching each phoneme to a representation in language. The output of speech recognition is written text, which facilitates text-based analysis, data input, and transcription [21].

2.2.2 Language Modeling

One of the crucial elements of voice recognition systems is language modeling, which predicts the likelihood of a sequence of words. This process assigns probabilities to word sequences, helping the system understand and generate human language, and enabling voice recognition software to tell apart context and structure.

Language models come in two main types: traditional statistical models and modern deep learning models. Statistical models such as Hidden Markov Models (HMMs) and n-grams rely on the frequency of word sequences in a given dataset to predict the next word. While these models are computationally efficient and straightforward, they often struggle with long-range dependencies and words not present in the training set.

On the other hand, deep learning models, particularly those based on neural networks, have revolutionized the field of language modeling. Techniques like transformers, long short-term memory (LSTM) networks, and recurrent neural networks (RNNs) enable the understanding of complex linguistic structures and the capture of long-term relationships [22]. These models, leveraging vast amounts of data and processing power, significantly enhance the accuracy and flexibility of voice recognition systems by learning the complex patterns of human language.

2.3 Known Techniques of a Home Grocery Listing System

Several techniques of home grocery listing systems have been proposed in the past. Kaur *et al.*, [1] proposed IoT-based grocery level monitoring using a load cell sensor. Multiple nodes in a load sensor are constructed to relay the weight of various grocery items. The collected data is sent remotely by uploading the grocery level via the internet. Figure 2.4 depicts the top-level block diagram of the proposed grocery level monitoring system in [1]. Similar study was also found in [7] which uses a similar load cell to detect the weight of grocery items. Nevertheless, both studies have a major drawback in which only applicable to specific types of containers and could falter when other plastic or metal containers are used.

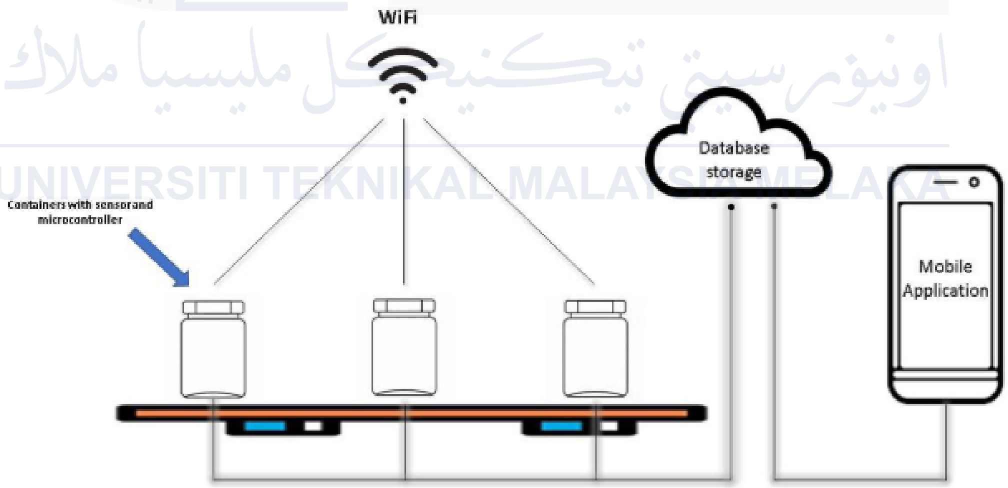


Figure 2.4: Top-level block diagram of the grocery level monitoring system [1].

In a study, Rezwan *et al.*, [5] proposed an IoT-based smart inventory management system (SIMS) for the kitchen using weight sensors, light dependent resistor (LDR), light-emitting diode (LED), Arduino Mega, and NodeMCU ESP8266 integrated with a website and mobile application. SIMS streamlines kitchen inventory management through automated reordering and real-time tracking, accessible via a website and mobile application. Users can monitor inventory, place orders, track order history, and receive notifications, enhancing convenience and efficiency in grocery shopping and inventory management. Figure 2.5 illustrates the SIMS prototype featuring smart compartments with sensors to monitor grocery levels, enabling users to generate lists, track inventory, and receive notifications for low stock. While the system is designed to be efficient and accessible, the system may face technical challenges related to sensor accuracy.

Figure 2.5: SIMS prototype.

In a recent study, Kiruthika *et al.*, [2] proposed a technique of using an IoT-based smart inventory management system using HX1711 load cell amplifier, a 24-bit analogue to digital converter (ADC), and the NodeMCU ESP9266. The load cells are employed to measure the weight of the grocery container, providing accurate data for monitoring inventory levels. The cloud service ThingSpeak is used to send load cell data, enabling users to receive notifications on their smartphones through the Blynk application when inventory levels drop below a specified threshold. Figure 2.6 depicts the top-level block diagram of the proposed inventory management system in [2]. A similar study was also found in [3], which uses a similar load cell to detect the weight of grocery items. Figure 2.7 depicts the prototype of the proposed system in [3]. Nevertheless, both of the load cell-based inventory management systems above suffer similar limitations in the capacity of the load cells, as the force applied to load cells over their maximum capacity may cause damage to load cells.

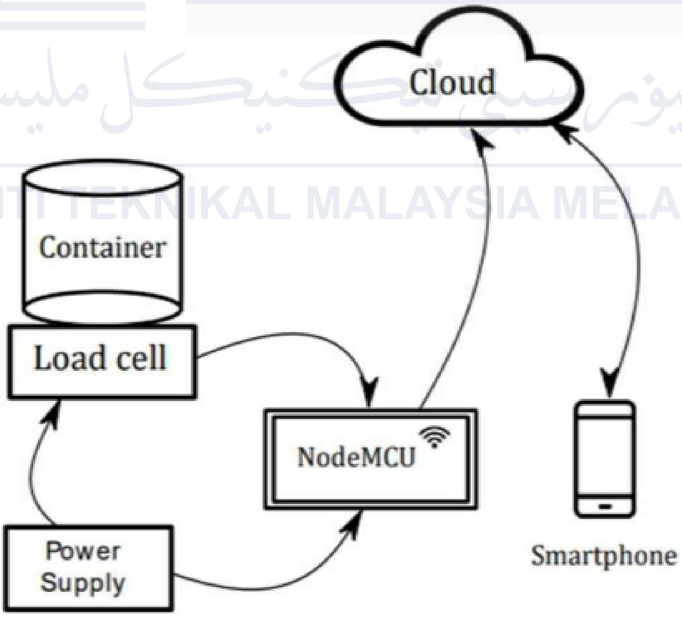


Figure 2.6: Top-level block diagram of the smart inventory management system [2].

Figure 2.7: Load cell based inventory management system [3].

Elsewhere, a technique of using an ultrasonic sensor for smart grocery level management system is proposed in [6]. Each container has an ultrasonic installed in the lid to determine the level of grocery. NodeMCU ESP8266 is used to measure and process the sensed data. Subsequently, the data is kept on a cloud platform and linked to ThinkSpeak as depicted in Figure 2.8, in which the users can monitor the data online. A similar study is found in [9] which uses a similar ultrasonic sensor to detect the grocery level in a container. A temperature sensor is also used to monitor the temperature in the grocery container. Nevertheless, both techniques above suffer a limitation in sensor accuracy in detecting the accurate level of remaining grocery in the container.

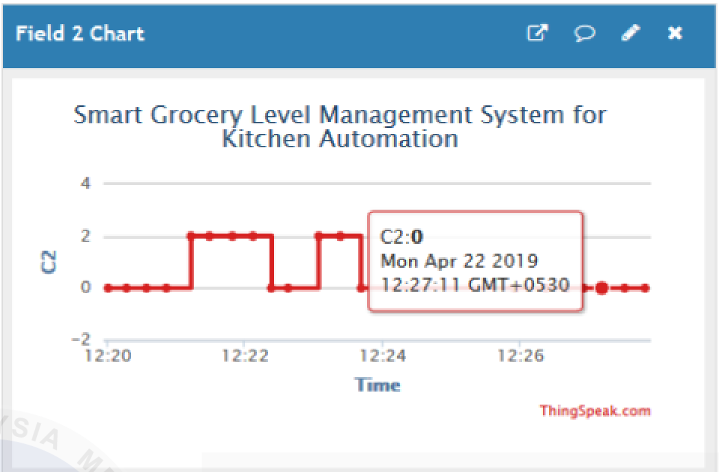


Figure 2.8: Grocery level measurement using ultrasonic sensor.

Firoz *et al.*, [4] proposed a grocery list management application using smartphone and augmented reality (AR). The system employs a two-component approach: a mobile application for users and a web-based application for supermarkets, facilitating grocery item maintenance and tracking. The mobile application is developed using a cross-platform framework for Android accessibility with a cloud database storage and AWS-hosted MySQL integration for data management. The proposed solution utilizes AR to map supermarket items, assigning each a unique AR tag for easy identification and location within the store, triggered by a QR code for indoor navigation assistance as shown in Figure 2.9. The AR application requires camera and AR devices, hence it is costly and not suitable for everyone.

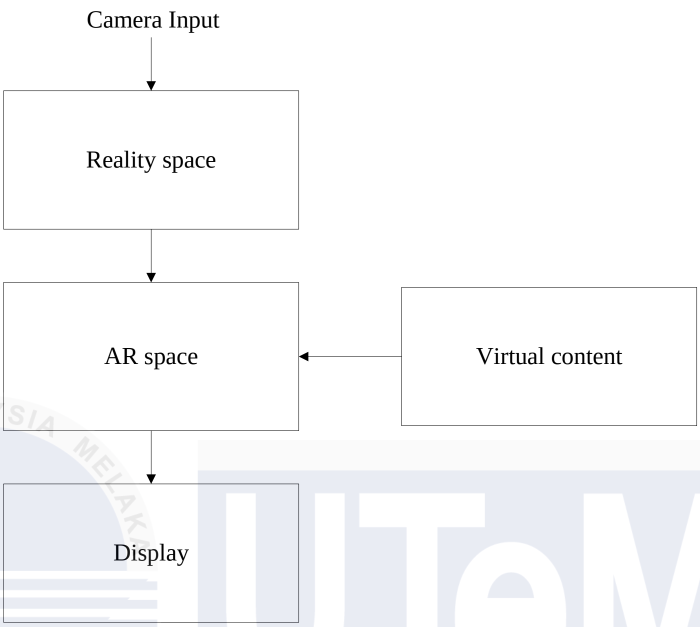


Figure 2.9: Concept of AR application in grocery management.

Elsewhere, a technique for using mobile technology and data mining techniques is proposed in [8]. The developed application known as 'Smart Shopping List' addresses the traditional methods of grocery list creation, helps users find the ideal supermarket to purchase most of their items in one place by using geolocation services, and utilizes the Ariori algorithm to recommend items to the user based on pattern matching recognition. Nevertheless, the smart shopping list system suffers from a drawback in location data accuracy, which may lead to less effective recommendations. In a study, Katuk *et al.*, [10] proposed a grocery listing system using mobile application known as SMART LIST. SMART LIST is developed using rapid application development (RAD) method. RAD is an adaptive software development approach involving prototyping to gather system requirements for applications. RAD methodology consists of four main phases: requirements planning, user design, construction, and cutover as depicted in Figure 2.10. Nevertheless, the proposed technique suffer a drawback that is the maintanance could be complex and costly.

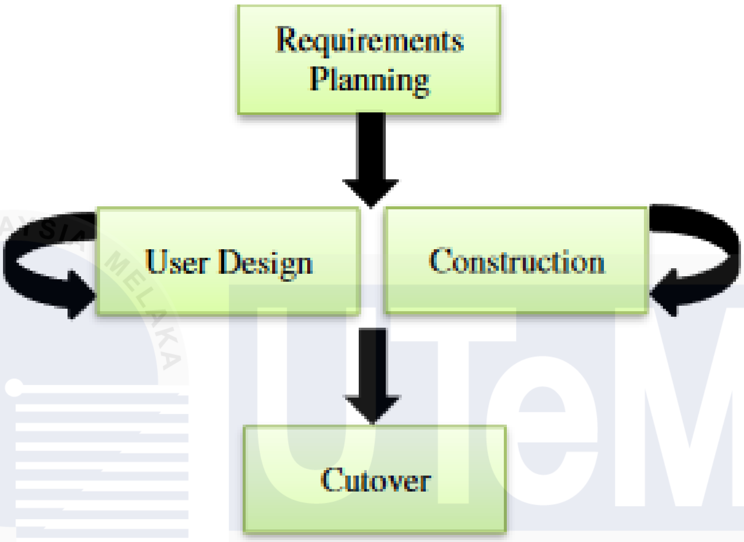


Figure 2.10: Four phases using RAD approach.

Based on all the above, common limitations of the previously proposed techniques are related to sensor accuracy and high setup cost. Therefore, in our study, we propose a microcontroller-based home grocery listing system using speech recognition and barcode scanner. All the previous techniques as discussed above are summarized in Table 2.1.

Table 2.1: Summary of the previously proposed techniques

Authors	Proposed Technique	Advantage(s)	Disadvantage(s)
Firoz *et al.*, [4]	Grocery list management application using smartphone and augmented reality.	• Efficient grocery planning. • Mobile convenience that is accessible anywhere, anytime. • Improve people's lifestyle.	• Initial setup are very costly. • Requires regular updates to maintain functionality and security.
Desai *et al.*, [7]	Developed an IoT-based prototype for monitoring grocery levels in homes and supermarkets.	• Enabling timely restocking and inventory management. • The acquired data can be remotely accessed over the internet.	• The system may be specific to certain types of containers and could falter when other plastic or metal containers are used.
Jayawilal *et al.*, [8]	Grocery list-creation process using mobile technology and data mining techniques.	• Integrates with nearby stores to make buying easier.	• Initial setup are very costly. • Accuracy of location data may be challenging.
Rezwan *et al.*, [5]	IoT-based smart inventory management system for kitchen using weight sensors, LDR, LED, Arduino Mega and NodeMCU ESP8266 Wi-Fi module with website and app.	• Efficient grocery planning. • The system can automatically order new items when the quantity of a particular item gets low.	• Initial setup are very costly. • The system may face technical challenges related to sensor accuracy.
Kiruthika *et al.*, [2]	IoT-based smart inventory management system using HX711 load cell amplifier, a 24-bit ADC and NodeMCU ESP8266.	• Real-time tracking of inventory, allowing for continuous monitoring of the stocks in the kitchen grocery container. • Cost-effective.	• Limitations in the capacity of the load cells.
Kaur *et al.*, [1]	Smart grocery monitoring system to automate the monitoring and reordering of grocery items in the kitchen.	• Automates the process of monitoring grocery items, saving time and effort for users.	• Initial setup are very costly.

Continued from previous page . . .

Authors	Proposed Technique	Advantage(s)	Disadvantage(s)
Patil *et al.*, [9]	Developed a smart grocery management system using IoT.	• Reduces the need for continuous monitoring of grocery at home. • Efficient grocery planning.	• Initial setup are very costly.
Katuk *et al.*, [10]	Developed and managing grocery lists using mobile application.	• The system can generate a grocery list based on the user's purchase history, budget, and meal plans, ensuring that the list is tailored to the user's specific needs. • Efficient and organized for users.	• Maintenance could be complex and costly.
Sakthisudhan *et al.*, [6]	Smart kitchen automation and grocery management system using ultrasonic sensor and IoT.	• Reduces food waste by optimizing grocery usage. • Real-time monitoring ensures efficient grocery management.	• The system may face technical challenges related to sensor accuracy.
Singh *et al.*, [3]	Developed a portable IoT-based grocery tracking system using ESP8266 Wi-Fi module.	• Automatically track the weight and quantity of grocery items, eliminating the need for manual tracking and creating shopping lists.	• Limitations in the capacity of the load cells.

2.4 Summary

In this chapter, an in-depth analysis is conducted on the research related to the project as well as the techniques previously proposed for a home grocery listing system. The primary objective is to examine and evaluate various approaches to identify the most effective solution for planning and executing the overall development of this project. The comprehensive research findings will be instrumental in ensuring that all the necessary components are required for the successful development and implementation of the home grocery listing system.

CHAPTER 3

METHODOLOGY

This chapter describes the methodology and design steps to develop a microcontroller-based home grocery listing system using speech recognition and a barcode scanner.

3.1 Description of Methodology - Sustainable Development

Our main goal is to develop a microcontroller-based home grocery listing system using speech recognition and a barcode scanner with a focus on sustainable development. Figure 3.1 depicts the top level of the proposed system. The ESP32-S3 microcontroller acts as the main brain in this system, which utilizes barcode scanning or user entry via a hardware keyboard to process grocery items with barcodes and either speech recognition or user entry via a mobile phone to list grocery items without barcodes. Moreover, the user can monitor the listed grocery items by using a mobile application and LCD. To design the proposed system as shown in Figure 3.1, this project is divided into three main parts which are developing the hardware of a home grocery listing system using barcode scanner and ESP32-S3, designing a mobile application using MIT App Inventor including speech recognition, and integrate the hardware and mobile application of home grocery listing system. The above three main parts are elaborated in the next sections.

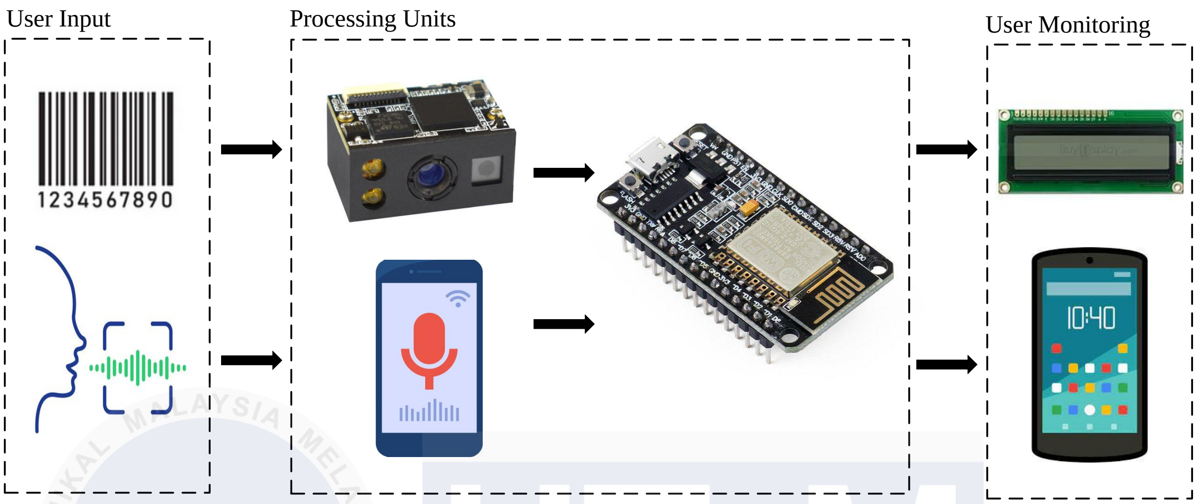


Figure 3.1: Top-level block diagram of the home grocery listing system.

3.2 Hardware Design: Barcode Scanner

Figure 3.2 illustrates the design steps to develop the hardware of the home grocery listing system. In our study, ESP32-S3 is used as a microcontroller which connected to a barcode scanner module, and a 16x2 LCD. Figure 3.3 depicts the aforementioned hardware components. The barcode scanner is used in our study to provide an alternative method for listing grocery items. Users can directly scan their grocery items' barcodes using the barcode scanner. The barcode scanner adds redundancy and improves accuracy, ensuring that no grocery item goes unnoticed. This design process harmoniously combines microcontroller versatility, and a practical barcode solution, resulting in an efficient and reliable home grocery listing system.

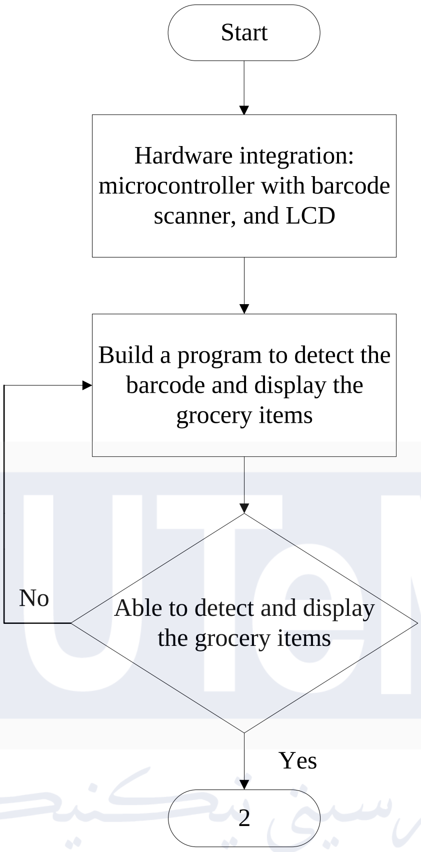


Figure 3.2: Design steps of home grocery listing hardware system.

(a) ESP32-S3 **(b)** Barcode scanner Module **(c)** 16x2 LCD

Figure 3.3: Major hardware components in developing the grocery listing system.

3.3 Mobile Application Development

Figure 3.4 illustrates the design steps to develop a mobile home grocery listing application using MIT App Inventor. It begins with designing the login page interface, ensuring users can securely access the application. Next, the main menu page is designed to display the grocery list which also includes a "Clear Items" buttons for easy items removal. An additional interface is then created to show the total amount of grocery items added to enhance the user experience. The final step involves evaluating if the mobile application's menu functions as expected; if not, designers are directed back to make necessary adjustments until it meets expectations.

Speech recognition is implemented using the Speech Recognizer feature in the MIT App Inventor. This software processes verbal grocery lists, converting spoken words into text. The system uses Automatic Speech Recognition (ASR) to analyze audio signals and match them to predefined patterns. Fine-tuning parameters like ambient noise and speech clarity enhances recognition accuracy, ensuring the system accurately captures grocery lists.

3.4 Integration of Hardware and Mobile Application

Figure 3.5 depicts the design steps of integrating the mobile application with the hardware of the home grocery listing system. Application programming interface (API) is used to link the mobile application and hardware. API allows secure communication between the mobile application and hardware like microcontroller and barcode scanner. The mobile application frequently updates its data from the hardware system. It can periodically

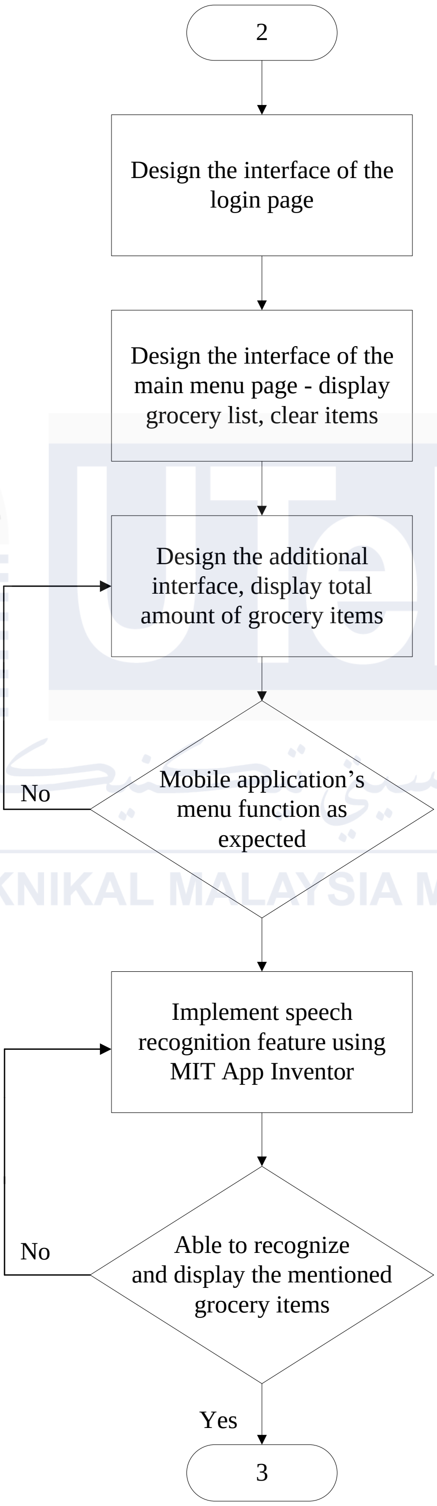


Figure 3.4: Design steps of a mobile home grocery listing application.

poll the hardware for changes. This ensures that users always have an accurate view of their grocery inventory. In the end, the design process approach combines smart application development, seamless API integration, real-time data sync, and user-friendly interactions to create a reliable home grocery listing system.

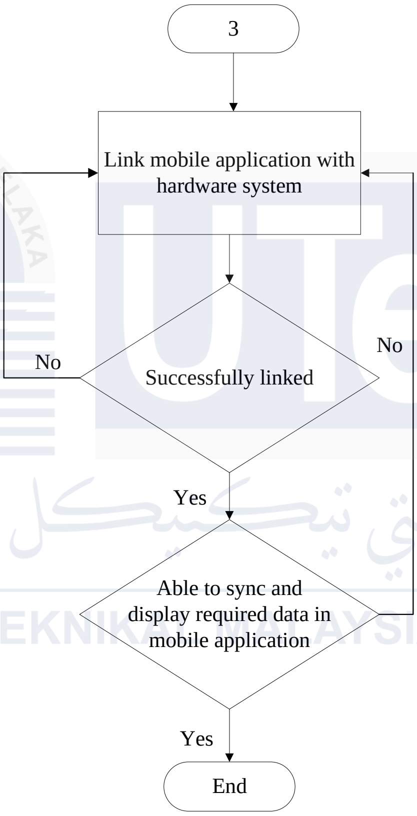


Figure 3.5: Design steps of hardware and mobile application integration.

3.5 Gantt Chart

In Projek Sarjana Muda 1 (PSM 1), the focus was on the literature review, definition of the objective, definition of the problem statement, definition of the scope of work, definition of methodology, and preliminary results. Figure 3.6 depicts the timeline of project implementation in PSM 1. The literature review mainly focuses on barcode scanning technology, speech recognition techniques, previous techniques of home grocery listing systems, and mobile application development using MIT App Inventor.

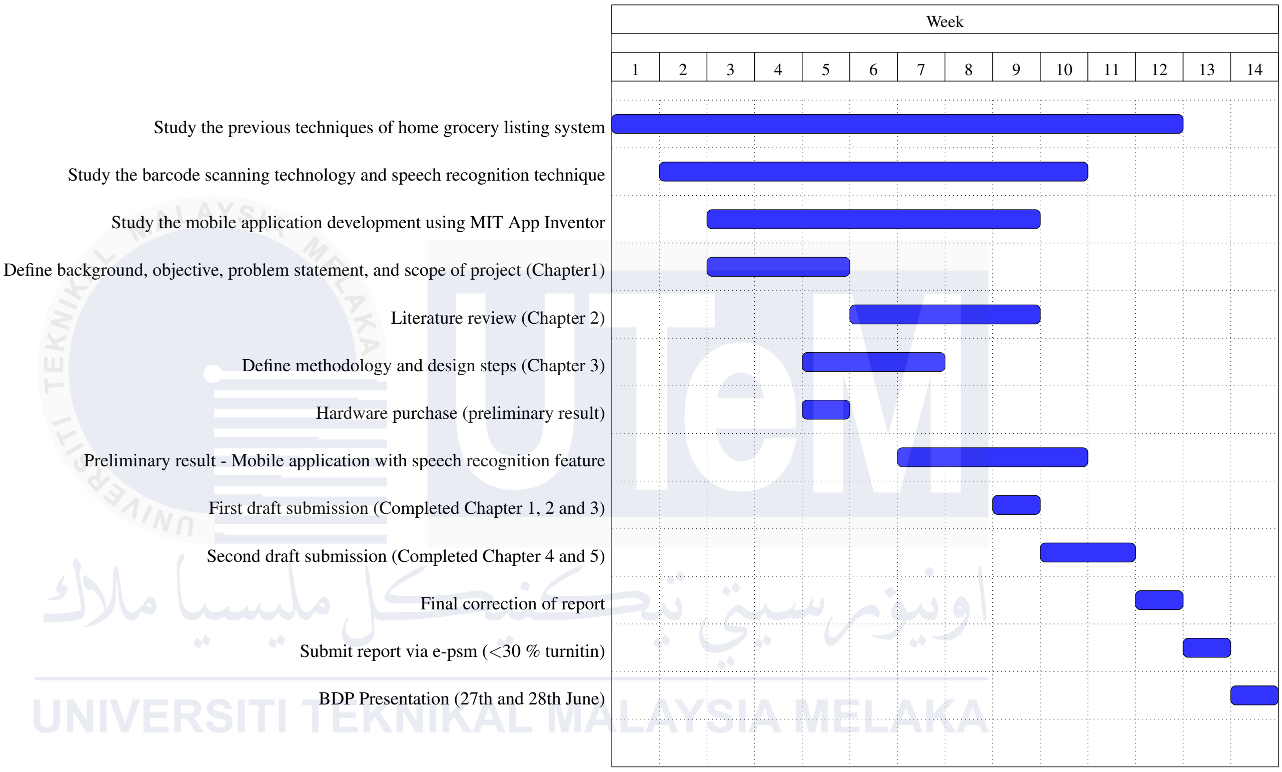


Figure 3.6: Time-line for PSM 1

In PSM 2, the focus is on hardware integration, mobile application development, testing, and validation of the proposed system. Figure 3.7 depicts the time-line for project implementation in PSM 2.

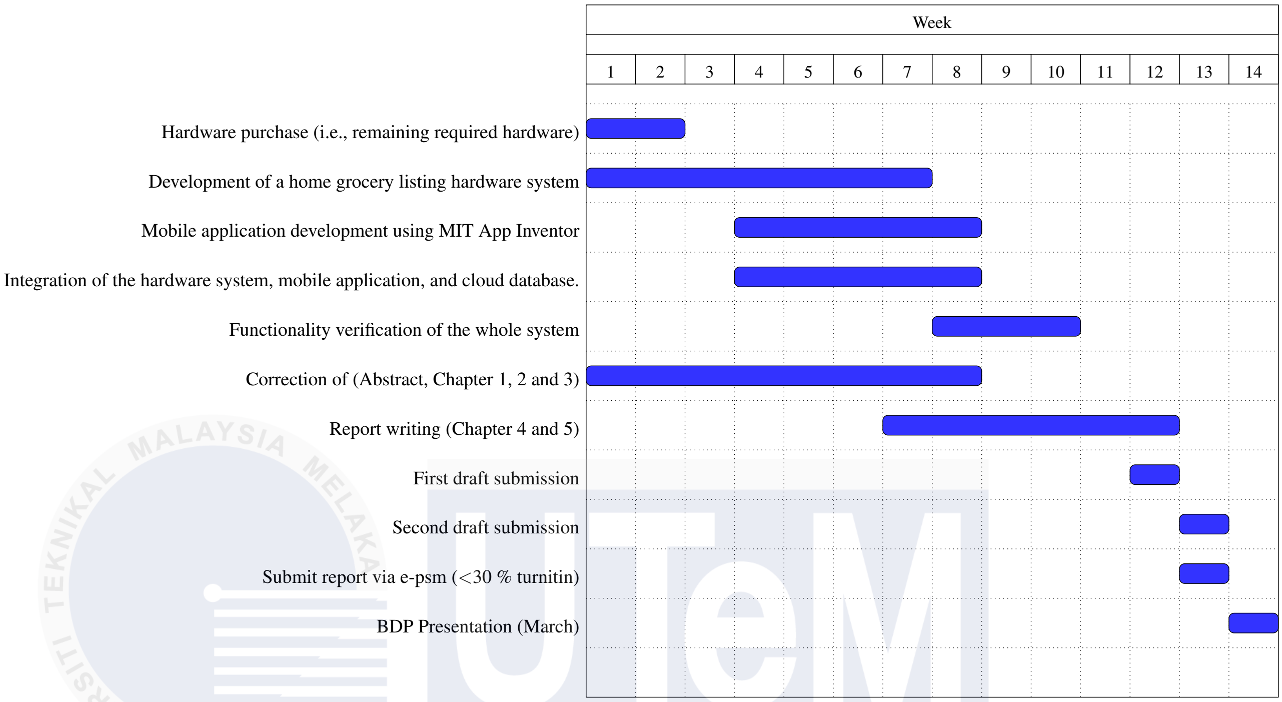


Figure 3.7: Time-line for PSM 2

3.6 Summary

This chapter describes the design steps to develop a home grocery listing system. Hardware such as ESP32-S3, barcode scanner module, and 16x2 LCD are used to develop the home grocery listing system. The system allows users to easily input and track their grocery items. Moreover, the mobile application is also developed using MIT App Inventor, in which users can display the grocery list, and recent items, and show the total amount of grocery items and speech recognition feature, providing a convinient and hands-free experience. This integrated approach ensures a seamless experience for users as they manage their grocery shopping needs.

CHAPTER 4

RESULTS AND DISCUSSION

This chapter presents the results and analysis on the development of a microcontroller-based home grocery listing system using speech recognition and a barcode scanner.

4.1 Home Grocery Listing Mobile Application Design

As mentioned in Section 3.3, MIT App Inventor is used to design the mobile application of home grocery listing system. Figure 4.1 depicts the user interface of the developed grocery listing mobile application which prioritizes simplicity and functionality. The arrangement of the primary page makes it simple to navigate and quickly access key functionality. User may add grocery items by key-in the grocery names in the "Item:" area and press "Add" button. User also can remove the grocery from the list by pressing "Clear items". Moreover, a microphone icon is provided to allow voice input, accommodating users who require or prefer this way of adding grocery to the list.

Figure 4.1: Main user interface of home grocery listing mobile application.

Figure 4.2 depicts the block-based coding of creating “Item:” and “Add” features. An empty list called “item” is created as a global variable to store the grocery items. When the “addItemBtn” button is clicked, the application retrieves the text from the “itemTxt” input field and adds it to the global items list. This updated list is then stored in the TinyDB database with the tag “itemsinDB”. After that, the “ListView1” element, which displays the list of items, is updated to show the current state of the global items list, and the “itemTxt” input field is cleared.

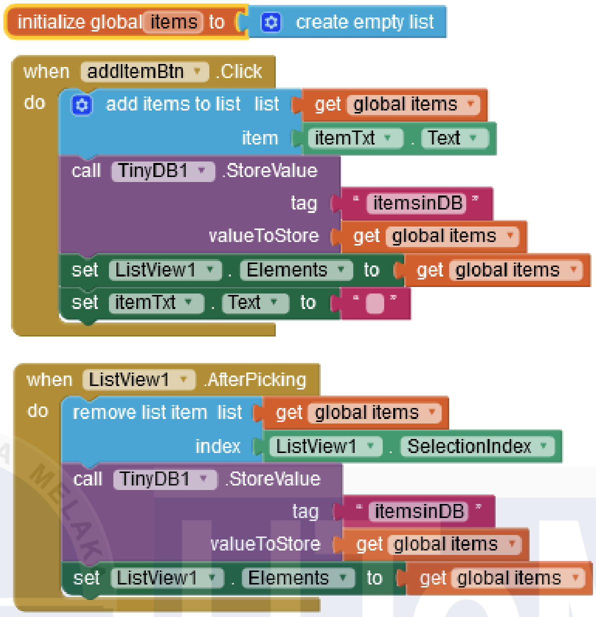


Figure 4.2: Block-based coding of creating "Item:" and "Add" features.

To remove an item, when a user selects an item from the "ListView1", the "After-Picking" event is triggered, which removes the selected item from the global items list based on its index. The updated list is then saved back to the TinyDB database under the same tag "itemsinDB", and the "ListView1" element is refreshed to display the updated list. This feature ensures that the grocery list is constantly updated and securely stored, providing a smooth user experience.

Users can remove a grocery item by pressing the "Clear Items" button. The "Clear Items" function was created using a series of block-based coding in the MIT App Inventor, as shown in Figure 4.3. When the "Clear Items" button is clicked, the code first sets the "global items" variable to an empty list. Then, it calls the "ClearAll" function of the "TinyDB1" component, which is a database. Finally, it updates the "Elements" property of the "ListView1" component to display the updated "global items" list, effectively clearing the displayed list of items. This design ensures that the application is not only easy to use and visually appealing but also functional. Action buttons are brightly coloured to guide the user's attention to important features and the layout is kept simple and uncomplicated to avoid confusion or overwhelming graphics.

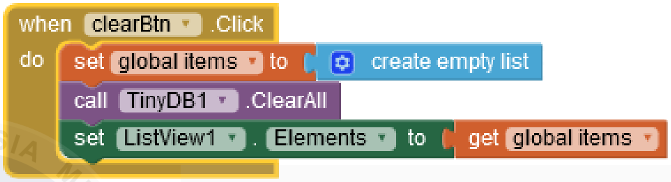


Figure 4.3: Block-based coding of clearing items using TinyDB in the grocery listing application.

4.2 Speech Recognition Configuration

User may add grocery into the list by using speech recognition feature. Figure 4.4 depicts the block-based coding design to enable speech recognition feature in mobile application. When the user clicks the "voice" button, the application requests voice input from the user using the "SpeechRecognizer1" component. User must say the word "add" followed by the grocery name. "SpeechRecognizer1" is coded to receives the voice input and able to distinguish the word "add" and grocery name. For example, when the user says "add apple", "SpeechRecognizer1" recognizes "apple" and add it to the grocery list. Figure 4.5 depicts the speech recognition functionality in the mobile application view. The speech recognition feature has been fully tested and validated to evaluate its accuracy, efficiency, and reliability. This included testing with different voices, slang, gender, and other variables.

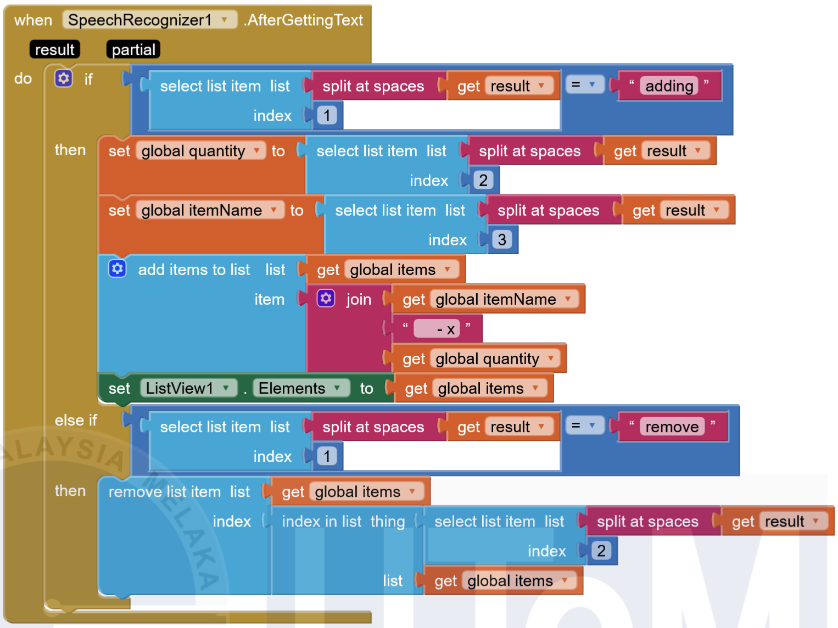


Figure 4.4: Block-based coding design for speech recognition.

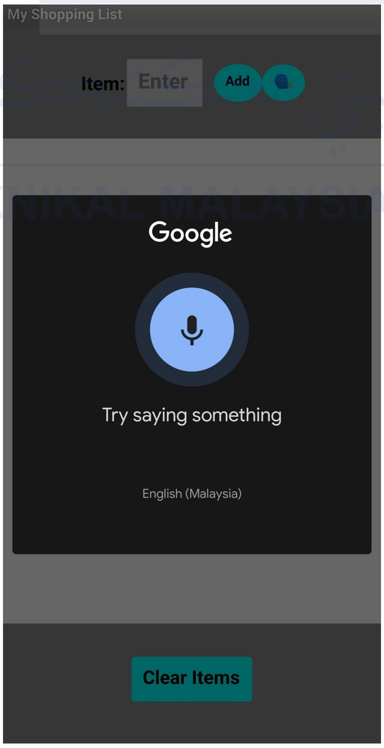


Figure 4.5: Speech recognition function.

4.3 Nutrition Facts App Design

The ‘Nutrition Facts’ is an additional feature in home grocery listing mobile application, designed to provide additional functionality on a separate screen as shown in Figure 4.6. This new feature allows users to search and view nutritional details of grocery items added to their list. After creating their grocery list on the main screen, users can navigate to the Nutrition Facts screen to check specific information about items such as serving size, calories, protein, fat, carbohydrates, fiber, sugars, and sodium.

Figure 4.6: User interface of nutrition facts mobile application.

The interface for this feature includes a text box where users can enter the name of an item and a “Search” button to find its nutrition details. Once the user inputs an item name, such as “Milo”, and presses the “Search” button, the app fetches and displays the nutritional details dynamically, as shown in Figure 4.7 . The results include values for each nutritional attribute, displayed in a structured format, ensuring clarity for the user. Additionally, the "Previous Screen" button allows users to return to the main Grocery Listing screen seamlessly.

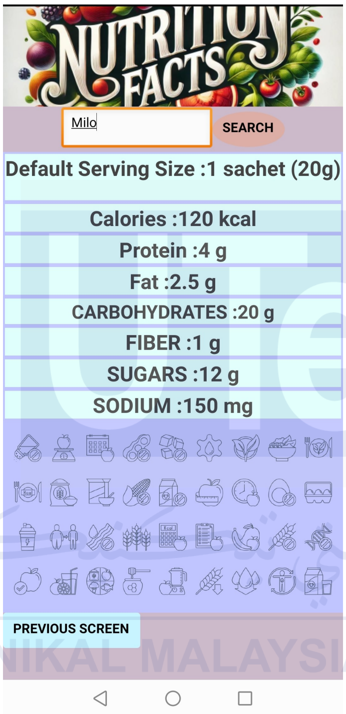


Figure 4.7: Nutrition facts screen showing item details after user input.

For data management, this extension uses Google Sheets as a backend database to store and retrieve nutritional data. When users search for an item, the app sends a request to Google Sheets using the Web component in MIT App Inventor. The nutritional details are fetched and displayed dynamically on the screen. Figure 4.8 depicts the block setup to handles the logic for retrieving data and updating the interface. The blocks use the Web component to send a request to a Google Sheets script URL, retrieve the relevant nutritional information in JSON format, and parse it. Each key-value pair (e.g., "Calories", "Protein") is then mapped to the corresponding labels on the screen. If the searched item is not found in the database, a “Not found” message is displayed to the user. This block configuration ensures seamless integration of the user interface with the database, enabling users to view accurate and updated nutritional information.

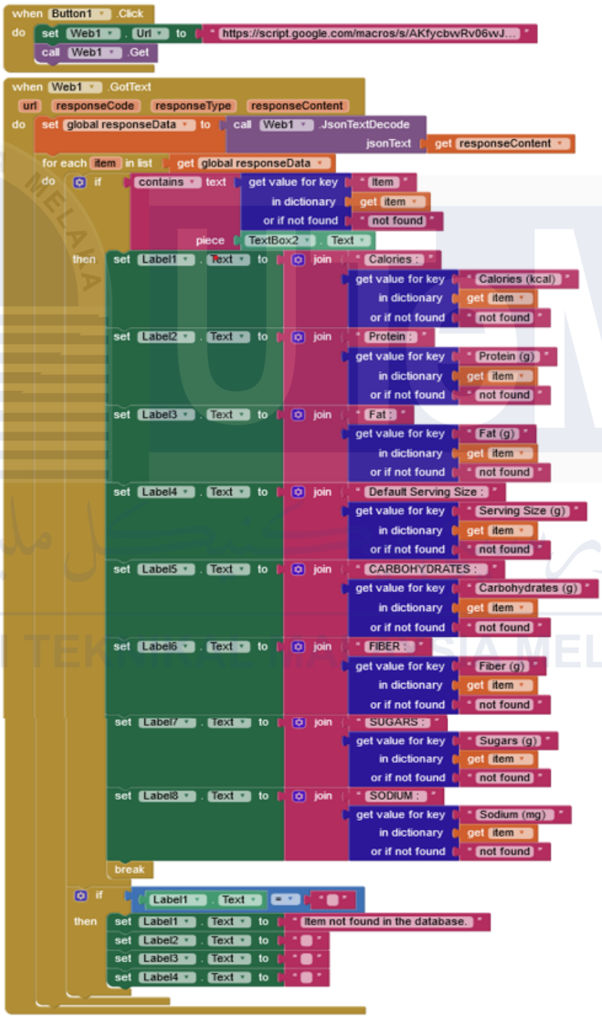


Figure 4.8: Block-based coding design for handling data retrieval and updating the interface

4.4 Barcode Scanner, LCD, and Keyboard Configuration

Figure 4.9 depicts the hardware which was designed using Fritzing software to effectively illustrate the connections and layout. Key components include the ESP32 microcontroller, barcode scanner, LCD display, and keyboard.

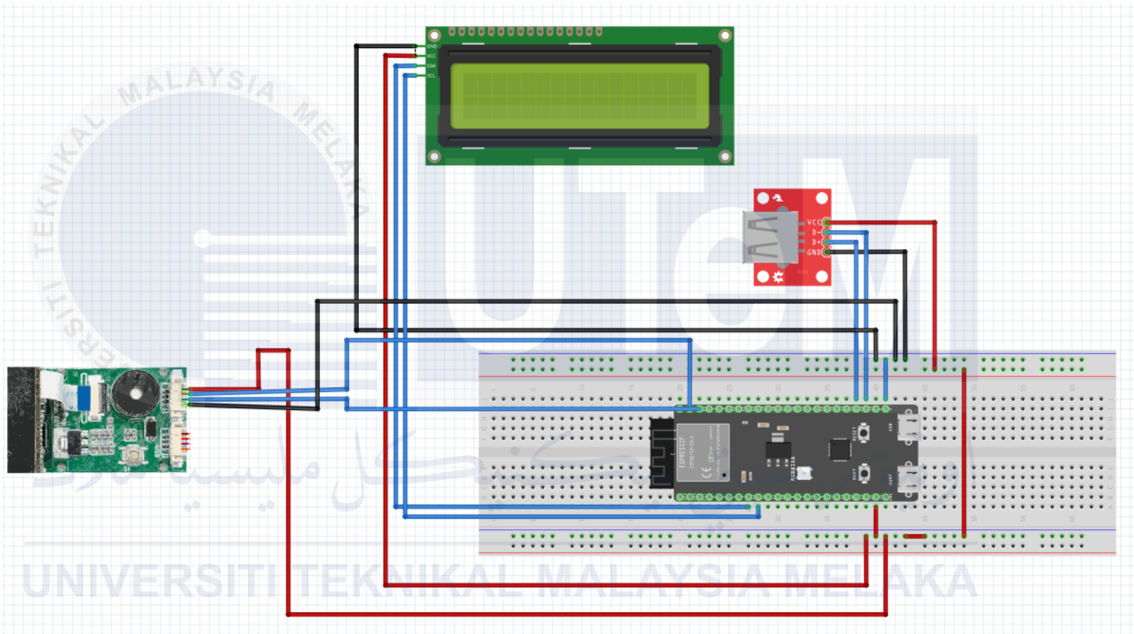


Figure 4.9: Schematic Diagram for Grocery Listing System Hardware.

The connectivity of the components in the system is structured to ensure functionality, reliability, and efficient communication. The barcode scanner is linked to the ESP32-S3 microcontroller via a UART interface, which facilitates quick and precise data transmission. This connection, as detailed in the Table 4.1, involves connecting the RX pin of the barcode scanner to GPIO 16 and the TX pin to GPIO 15 on the ESP32-S3. The LCD display is connected using the I2C protocol, simplifying the wiring and ensuring stable communication. As shown in the Table 4.2, the SDA pin of the LCD is connected to the default I2C SDA pin

(GPIO 8) on the ESP32-S3, while the SCL pin is connected to the default I2C SCL pin (GPIO 9). Additionally, the LCD is powered using a 5V supply, and the ground connection ensures electrical stability.

Barcode Scanner Pin	ESP32-S3 Pin
RX	GPIO 16
TX	GPIO 15

Table 4.1: Connectivity of barcode scanner to ESP32-S3.

LCD Pin	ESP32-S3 Pin
SDA	GPIO 8
SCL	GPIO 9
Ground	Ground
5V	5V

Table 4.2: Connectivity of LCD to ESP32-S3.

The USB host interface is integrated into the system to allow connections with external devices, offering flexibility for both manual and automatic data entry. According to the Table 4.3, the USB D+ and D- pins are connected to GPIO 20 and GPIO 19 on the ESP32-S3, respectively. The USB host also utilizes the 5V power supply and shares the common ground to enhance electrical stability and minimize interference. The entire system is powered by a common 5V supply, with all components grounded to improve electrical stability and reduce interference. The connections were designed with a focus on functionality and reliability, guaranteeing smooth communication between all hardware elements while maintaining operational efficiency.

USB Host Pin	ESP32-S3 Pin
D+	GPIO 20
D-	GPIO 19
Ground	Ground
5V	5V

Table 4.3: Connectivity of USB host to ESP32-S3.

4.5 Integration of Hardware and Mobile Application

The mobile application was integrated with the hardware system through the ESP32's web server capabilities over a Wi-Fi network. This arrangement allowed for real-time communication and data synchronization between the physical components and the mobile interface. The prototype integrates hardware components with a mobile application to create an efficient grocery management system, as shown in Figure 4.10. The system includes a barcode scanner, a 20x4 LCD display, and a USB keyboard connected to an ESP32 microcontroller. The ESP32 acts as a web server and main controller, enabling communication between the hardware and the mobile app via a Wi-Fi network. This integration created an efficient grocery listing solution, allowing users to operate the system easily from their smartphones.

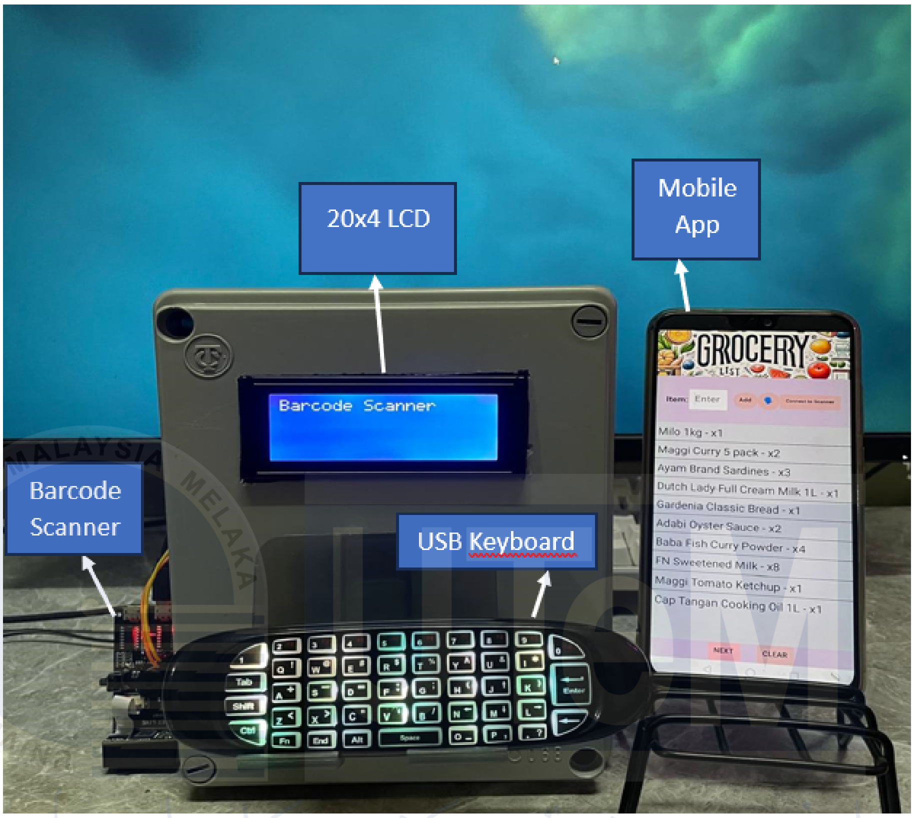


Figure 4.10: Home grocery listing system prototype.

When a product's barcode is scanned or entered manually, the ESP32 processed the information and displayed it on the LCD. At the same time, the system transmits this data to the mobile application via the HTTP protocol. The mobile apps serves as a client, enabling users to view the registered items, check the item's quantities, and interact with the system remotely. The mobile app utilizes the Web1 component to retrieve data from the designated server as shown in the Figure 4.11. When the ConnectButton is pressed, the Web1.Url is assigned with the URL's value of http://192.168.1.183/, and a GET request is initiated through the Web1.Get method. Once a response is received in the Web1.GotText event, the response code is evaluated. If the code is ”200” (indicating a successful request), the response content is divided by commas and populated into the ListView1 component. If the request success, a notification appears via Notifier1.ShowMessageDialog, stating "Data fetched successfully!" with the title "Success" and an "OK" button. This illustrates the process of retrieving and displaying data from a web server in a user-friendly list format.

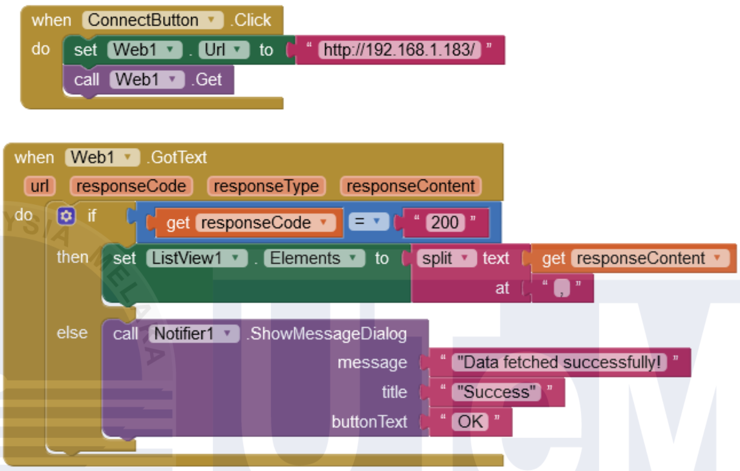


Figure 4.11: Implementation of Web1 component to retrieve and display server data.

4.6 Functionality Verification

This section describes the functionality verification of the developed hardware and mobile application of home grocery listing system.

4.6.1 Data Entry of Grocery Items Without Barcode

The system was evaluated for scenarios where no available barcode for the grocery items. The menu options feature displayed on the LCD is triggered by pressing the "Tab" button on the USB keyboard, providing users with easy access to system controls. The menu includes options such as Manual for manually entering product details, Clear to reset the current grocery list, On and Off to control the system's power state, Exit to return to the main menu or terminate the session, and Stats, which displays useful information such as Wi-Fi connection details, the number of items registered, and the memory usage. The manual data entry provides convenience experience to the user as not all grocery items come with the barcode, e.g., vegetables, meat, poultry, and fish. Table 4.4 below summarizes the process of manual data entry, from activating manual mode to entering the product details, displaying the information on the LCD, and synchronizing it with the mobile application. This functionality enhances user interaction by offering control and monitoring options in a simple and efficient manner.

Test Case	Action	System Response	Outcome
1	User select option ”1” to activate manual mode in Menu Option	System prompts the user to enter product information	Manual mode activated successfully.
2	User enters product information (name and quantity)	System displays entered data on the LCD and updates the webserver	Product information accurately recorded and displayed.
3	Data entered manually for a product	Data displayed on LCD and synchronized with the mobile application	Successful synchronization between hardware and app.

Table 4.4: Data analysis for manual mode testing

In the Menu Options, users can switch to Manual Mode by selecting option “1” on the menu displayed on the LCD screen. Upon entering Manual Mode, the system prompts the user to input the name of the item via the connected keyboard. The name is displayed on the LCD screen as it is being typed. Once the item name is provided, the system asks for the quantity of the item. After entering the quantity, the system processes the new item by storing its details, including the item name and quantity, in the internal database. The details of the registered item are then displayed on the LCD screen, confirming successful entry. Figure 4.12 depicts interface of Option Mode and functionality flow of Manual Mode, including the process of entering item details via the keyboard, displaying prompts and feedback on the

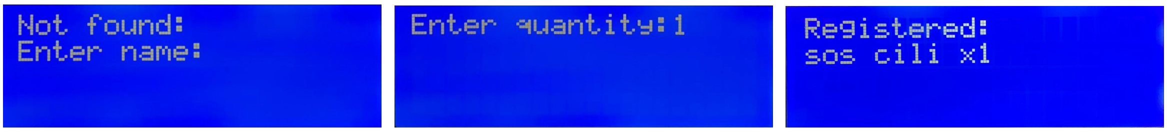
LCD screen, and registering the item in the internal database. The system functions effectively and aligns with the design specifications, as demonstrated by the successful activation, data entry, and synchronization processes detailed in Table 4.4.

(a) Menu options for system modes and operations. **(b)** manual mode activated. **(c)** Product's name entry. **(d)** Quantity entry. **(e)** Product registration successful.

Figure 4.12: Screens displayed on the 16x2 LCD during Manual Mode operation.

4.6.2 Data Entry of Grocery Items With Barcode

For the first-time entry of an item that is not yet registered in the system, scanning the barcode triggers a prompt on the LCD screen, requesting the user to input the product name and quantity manually. The system displays a clear message, guiding the user through the input process step by step. This ensures that the item details are recorded accurately. Once the user enters the product name and quantity using the connected keyboard, the system validates the data and saves the item details in both the internal memory and the webserver for future reference. This dual storage mechanism ensures data redundancy and synchronization across platforms, allowing the item to be accessed seamlessly in subsequent sessions. The entire process, from the prompt to the confirmation message displayed on the LCD screen, is illustrated in Figure 4.13, which highlights the efficient workflow for registering a new item in the system.



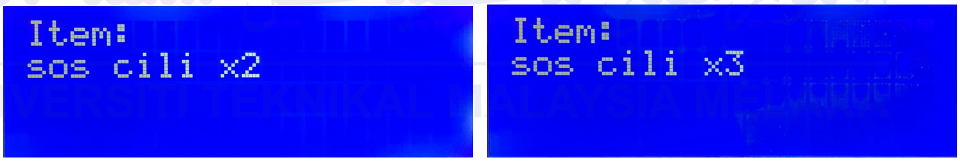
(a) System prompts for product name.

(b) Product name and quantity entry.

(c) Item registration confirmed.

Figure 4.13: Screens displayed on the 16x2 LCD during Manual Mode operation.

In contrast, items that already stored in the system provided immediate product information on the LCD and automatically adjusted the quantity after scanning using barcode scanner. This streamlined process enhances efficiency and minimizes manual input for frequently scanned items. The corresponding system behavior is illustrated in Figure 4.14, which depicts the automatic detection of stored items and the quantity entry prompt.



(a) Registered item displayed with quantity 2.

(b) Registered item with quantity updated to 3.

Figure 4.14: Registered items displayed with updated quantities.

For testing purposes, we configured the system with five new items and five pre-registered items stored in both the internal memory and the webserver. The list of these items is detailed in Table 4.5. During the testing process, each item was scanned, processed, and appropriately handled by the system according to its registration status. The system prompted for manual input of product details for new items, while pre-registered items were automatically detected and updated with the entered quantity. All ten items were successfully scanned and logged into the system, with their data reliably transmitted to the mobile application. This demonstrates that the system is capable of handling both new and pre-registered items with consistency and accuracy. The successful synchronization of this data with the mobile application is visually represented in Figure 4.15, which displays the complete list of items as seen in the app. This confirms the system's ability to ensure seamless integration between hardware and software components

Barcode ID	Reg.	Product Name	Qty.	System Action	Outcome
9556724810021	Yes	Milo 1kg	1	Displayed details on LCD, updated quantity automatically.	Successful scan; inventory updated.
9557932010584	Yes	Maggi Curry 5-pack	2	Displayed details on LCD, updated quantity automatically.	Successful scan; inventory updated.
9551028334217	Yes	Ayam Brand Sardines	3	Displayed details on LCD, updated quantity automatically.	Successful scan; inventory updated.
9557124832123	Yes	Dutch Lady Full Cream Milk 1L	1	Displayed details on LCD, updated quantity automatically.	Successful scan; inventory updated.
9556103456712	Yes	Gardenia Classic Bread	1	Displayed details on LCD, updated quantity automatically.	Successful scan; inventory updated.
9558943127645	No	Adabi Oyster Sauce	2	Prompted user to input product name and quantity.	Product added to database; inventory updated.
9556782910314	No	Baba's Fish Curry Powder	4	Prompted user to input product name and quantity.	Product added to database; inventory updated.
9558896723419	No	F&N Sweetened Milk	8	Prompted user to input product name and quantity.	Product added to database; inventory updated.
9557819231406	No	Maggi Tomato Ketchup	1	Prompted user to input product name and quantity.	Product added to database; inventory updated.
9556128347601	No	Cap Tangan Cooking Oil 1L	1	Prompted user to input product name and quantity.	Product added to database; inventory updated.

Table 4.5: Analysis of data entry for grocery items with barcode.

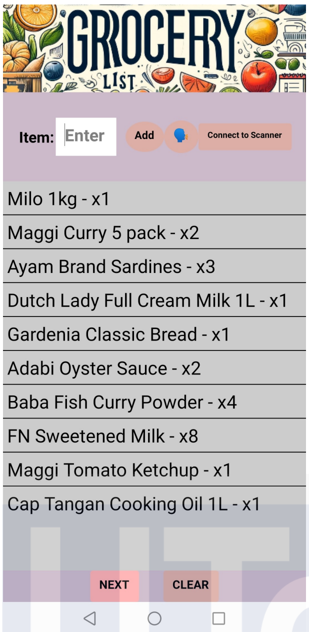


Figure 4.15: Corresponding list in the mobile application.

4.6.3 Voice Recognition Data Entry using Mobile App

Users can utilize the voice recognition feature within the mobile application to list grocery items conveniently. This feature allows users to speak product names and quantities directly through the app. To test the reliability of this feature, five individuals were selected as user samples. Various speech variations, including differences in speed, clarity, and pitch, were tested to evaluate the robustness of the system. The products used during the testing process are detailed in Table 4.6, showcasing a diverse range of grocery items. The results highlighted the system's ability to recognize and process most inputs accurately, even with variations in speech patterns. This testing also identified areas for improvement, such as enhancing recognition for low-pitched voices or very rapid speech

User ID	Product Name	Quantity	Speech Speed / Clarity	System Action	Outcome
User 1	Prawn	5	Normal	Processed voice command, logged details	Successful; correct product and quantity recorded
User 2	Cabbage	3	Fast	Processed voice command, logged details	Successful; correct product and quantity recorded
User 3	Soy Sauce	2	Slow	Processed voice command, logged details	Successful; correct product and quantity recorded
User 4	Lipbalm	1	Normal	Processed voice command, logged details	Successful; correct product and quantity recorded
User 5	Cucumber	3	Clear	Processed voice command, logged details	Successful; correct product and quantity recorded

Table 4.6: Analysis of voice recognition data entry.

The testing results indicate that the system effectively processes voice commands and accurately records both product names and quantities, even with varied speech characteristics. As shown in Table 4.6, all five users successfully interacted with the system, regardless of differences in speech speed or clarity, demonstrating the reliability of the voice recognition feature. The system consistently logged correct product details and quantities without errors, confirming its robustness in handling diverse voice inputs. Additionally, the grocery list corresponding to these results was successfully synchronized with the mobile app, as shown in Figure 4.16, illustrating the complete grocery list displayed in the app.

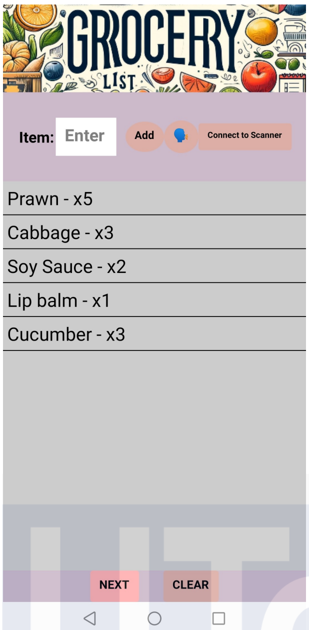


Figure 4.16: Corresponding list in the mobile application.

4.7 Comparison with Previous Techniques

The proposed home grocery listing system improves user input efficiency compared with the previous methods. Table 4.7 lists the relevant techniques that have been proposed in the past. All the previous techniques have low efficiency in which users have a limited methods of user data entry. In contrast, this project stands out by incorporating four methods of user data entry; barcode scanning or manual entry using keyboard through a hardware system and voice recognition or manual entry through mobile application. These options allow users greater flexibility and ensure the system can handle a wide range of grocery items efficiently. This variety of input methods reduces errors, improves usability, and makes the system adaptable for different scenarios. With four methods of user data entry, this project achieves a very efficient rating, setting it apart as the most effective solution in comparison to previous techniques. By using cost-effective hardware like the ESP32-S3 and creating a mobile application through MIT App Inventor, the proposed system achieves capabilities that improved accessibility and affordability. The proposed system combines the strengths of previous techniques while addressing their drawbacks, offering an accurate, user-friendly, and cost-efficient solution for home grocery listing.

System Design	Number of User Inputs	Efficiency Level
Kaur *et al.*, [1]	1	Low efficiency
Rezwan *et al.*, [5]	2	Moderate efficiency
Firoz *et al.*, [4]	2	Moderate efficiency
Katuk *et al.*, [10]	2	Moderate efficiency
Patil *et al.*, [9]	2	Moderate efficiency
Our Work	4	Very efficient

Table 4.7: Efficiency comparison with the previous techniques.

4.8 Summary

The grocery listing system has been developed which consists of hardware and a mobile app. User can perform data entry by using hardware (i.e., barcode scanning or manual entry using keyboard) or mobile app (i.e., voice recognition or manual entry). The mobile app is also comes with a feature to check or obtain nutrition information. This feature helps the user to obtain nutrition information and encourage users towards a healthy lifestyle. Thorough testing and evaluation have been conducted to ensure the system's functionality and reliability.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The project has been divided into three main objectives as described in Section 1.3. The accomplishment of each objective is described and concluded in this chapter.

The first objective has been achieved. The hardware of the home grocery listing system was developed by integrating the ESP32-S3, a barcode scanner module, and an LCD, allowing efficient data input and display for managing grocery items.

The second objective was achieved by designing and developing a mobile application using MIT App Inventor, which enables users to list grocery items. Additionally, the application integrates a speech recognition feature, allowing users to input product details hands-free. Furthermore, a nutrition facts screen feature was added to the app, enabling users to search and view detailed nutritional information for specific items, such as serving size, calories, protein, and more, enhancing the app's functionality and user experience.

The third objective has been achieved. The hardware of the home grocery listing system was successfully integrated with the mobile application, enabling seamless functionality. Data from the hardware through barcode scanning or manual data entry, is transmitted to the mobile application via the ESP32-S3's web server capabilities over a Wi-Fi network. This allows users to view and manage their grocery lists in real-time, ensuring an efficient and user-friendly experience.

5.2 Future Works

The development of a home grocery listing system holds immense potential for further enhancements and innovations. As we look ahead, several key areas can be explored to elevate the system's functionality and user experience.

- Investigate the adoption of MatrixScan technology which allows simultaneous scanning of multiple barcodes in a single sequence. This advancement can significantly reduce the time required for inventory counts and enhance accuracy.
- Develop a system that can seamlessly translate speech recognition and product information across different languages other than English.
- Enable users to add grocery items directly from smart home devices such as smart refrigerators or voice-controlled assistants.

5.3 Project Commercialization

This project is suitable for commercialization because people do grocery shopping very often. It has the potential to revolutionize the grocery management system and contribute to commercial efficiency. All things considered, the home grocery listing system's successful commercialization depends on a comprehensive strategy that blends cutting-edge technology, user-centered design, and potential marketing techniques. We can present the system as a vital tool for contemporary homes by addressing these aspects.

REFERENCES

[1] L. Kaur, “Smart grocery monitoring system using smart sensors and IoT,” in *International Journal of Recent Development in Engineering and Technology*, 2022, pp. 1–5.

[2] J. K. Kiruthika, S. Manikandan, U. Nithya, M. Preethika, and S. Poorna, “IoT-based smart inventory management system,” in *International Conference on Computer Communication and Informatics*, 2023, pp. 1–6.

[3] D. Singh, N. Desai, and R. R. Nair, “Design of portable IoT based grocery tracking system via Wi-Fi module for home automation,” *International Research Journal of Engineering and Technology*, vol. 7, pp. 5469–5473, 2020.

[4] A. Firoz and G. Ratnayaka, “ShopLister - A grocery list management application,” in *International Conference on Image Processing and Robotics*, 2020, pp. 1–6.

[5] S. Rezwan, W. Ahmed, M. A. Mahia, and M. R. Islam, “IoT based smart inventory management system for kitchen using weight sensors, LDR, LED, Arduino Mega and NodeMCU (ESP8266) Wi-Fi module with website and app,” in *International Conference on Advances in Computing, Communication & Automation*, 2018, pp. 1–6.

[6] T. S. K. Sakthisudhan, S. Mohanraj, “A smart kitchen automation and grocery management system using IoT,” *International Journal of Recent Technology and Engineering*, vol. 8, pp. 2368–2373, 2019.

[7] H. Desai, D. Guruvayurappan, M. Merchant, S. Somaiya, and H. Mundra, “IoT based grocery monitoring system,” in *International Conference on Wireless and Optical Communications Networks*, 2017, pp. 1–4.

[8] W. A. Harsha Jayawilal and S. Premeratne, “The smart shopping list: An effective mobile solution for grocery list-creation process,” in *IEEE Malaysia International Conference on Communications*, 2017, pp. 124–129.

[9] C. S. Patil and K. N. Pawar, “Smart grocery management system using Internet of Things (IoT),” *International Journal of Research in Engineering and Technology*, vol. 5, pp. 2321–7308, 2016.

[10] Y. Y. Norliza Katuk, Tamilarasi Jayasangar, “Design and development of Smart List: a mobile app for creating and managing grocery lists,” *Baghdad Science Journal*, vol. 16, pp. 462–476, 2019.

[11] R. Wudhikarn, P. Charoenkwan, and K. Malang, “Deep learning in barcode recognition: A systematic literature review,” *Institute of Electrical and Electronics Engineers Inc*, vol. 10, pp. 8049–8072, 2022.

[12] X. G. Runze Zhou, “A new method of angle-robust multiple ID-barcodes detection,” in *IEEE International Conference on Computer and Communications*, 2016, pp. 433–438.

[13] L. G. Lim, “Implementation of EAN-13 barcode system in Malaysia: A case study,” *International Journal of Supply Chain Management*, vol. 2, pp. 42–48, 2013.

[14] L. Z. YeMin Li, "Research and application of the EAN-13 barcode recognition on Iphone," in *International Conference on Future Information Technology and Management Engineering*, 2010, pp. 92–95.

[15] B. Z. Changsheng Chen, Wenjian Huang, "PiCode: A new picture-embedding 2D barcode," *IEEE Transactions on Image Processing*, vol. 25, pp. 3444–3458, 2016.

[16] K. T. Hanna, "What is a 2D barcode and how does it work?." 2021.

[17] J. W. Lijun Mao, Yan Zhang, "Application of 2D barcode technology based on image processing," in *IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference*, 2018, pp. 2613–2616.

[18] K. Khotimah, "Validation of voice recognition in various google voice languages using voice recognition module V3 based on microcontroller," in *International Conference on Vocational Education and Electrical Engineering*, 2020, pp. 1–6.

[19] F. Bee, "Speech Recognition vs. Voice Recognition: In Depth Comparison," 2022.

[20] J. C. Awni Hannun, Carl Case, "Deep speech: Scaling up end-to-end speech recognition," *International Journal of Research in Engineering and Technology*, vol. 4, 2014.

[21] V. V. K. Gulmira K. Berdibaeva, Oleg N. Bodin, "Pre-processing voice signals for voice recognition systems," in *International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices*, 2017, pp. 242–245.

[22] Q. V. L. Ilya Sutskever, Oriol Vinyals, "Sequence to sequence learning with neural networks," *International Journal of Research in Engineering and Technology*, vol. 3, 2014.

APPENDICES

Appendix A: Full code of grocery listing system

```
#include <EspUsbHost.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
#include <Preferences.h>
#include <WiFi.h>
#include <WebServer.h>
#include <map>

// WiFi credentials
const char* ssid = "POCO";
const char* password = "1234567";

// Web server and LCD
WebServer server(80);
LiquidCrystal_I2C lcd(0x27, 20, 4);

// Custom EspUsbHost class
class MyEspUsbHost : public EspUsbHost {
public:
    MyEspUsbHost() : lcd(0x27, 20, 4) {}

    void onKeyboardKey(uint8_t ascii, uint8_t keycode, uint8_t modifier) override {
    if (' ' <= ascii && ascii <= '~' && modifier == 0) {
        // Check if '7' is pressed and trigger adjust mode
        if (ascii == '7') {  // '7' key pressed
            isAdjustingQuantity = true; // Enable adjust mode
            lcd.clear();
            lcd.setCursor(0, 0);
```

```
            lcd.print("Adjust Qty Mode");
            lcd.setCursor(0, 1);
            lcd.print("Enter new qty:");
        } else if (isMenuMode) { // Check if in menu mode
            handleMenuSelection(ascii); // Handle menu selection
        } else {
            Serial.printf("%c", ascii);
            lcd.print((char)ascii);
            inputBuffer += (char)ascii;
        }
    } else if (ascii == '\r') {
        Serial.println();
        handleInput();
        inputBuffer = "";
    } else if (keycode == 0x2A) {  // Backspace key
        if (!inputBuffer.isEmpty()) {
            inputBuffer.remove(inputBuffer.length() - 1);
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("Enter:");
            lcd.setCursor(0, 1);
            lcd.print(inputBuffer);
        }
    } else if (keycode == 0x2B) {  // Tab key for menu
        isMenuMode = true; // Set menu mode
        displayMenu();
    }
}


    void handleMenuSelection(char key) {
        lcd.clear();
        switch (key) {
```

```
        case '1':
            isMenuMode = false; // Exit menu mode
            isManualMode = true; // Set manual mode
            lcd.setCursor(0, 0);
            lcd.print("Manual Mode");
            lcd.setCursor(0, 1);
            lcd.print("Enter name:");
            break;
        case '2':
            clearWebServerItems();
            lcd.setCursor(0, 0);
            lcd.print("List Cleared");
            delay(2000);
            break;
        case '3':
            lcd.backlight();
            lcd.setCursor(0, 0);
            lcd.print("System: ON");
            delay(2000);
            break;
        case '4':
            lcd.noBacklight();
            lcd.setCursor(0, 0);
            lcd.print("System: OFF");
            delay(2000);
            break;
        case '5':
            isMenuMode = false; // Exit menu mode
            lcd.setCursor(0, 0);
            lcd.print("Exiting Menu");
            delay(2000);
            lcd.clear(); // Clear LCD after displaying the message
            lcd.setCursor(0, 0);
```

```
            lcd.print("Barcode Scanner"); // Return to normal mode
            break;
        case '6': // Check WiFi
            isMenuMode = false; // Exit menu mode
            displayWiFiStatus();
            showStats();
            break;
        default:
            lcd.setCursor(0, 0);
            lcd.print("Invalid Option");
            delay(2000);
            break;
    }
    if (!isManualMode && !isMenuMode) {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Barcode Scanner");
    }
}

void showStats() {
    lcd.clear();

    // Number of registered items
    int registeredItems = barcodeItems.size();
    lcd.setCursor(0, 0);
    lcd.print("Items Registered: " + String(registeredItems));

    // Free memory
    int freeMemory = ESP.getFreeHeap();
    lcd.setCursor(0, 1);
    lcd.print("Memory: " + String(freeMemory) + " Bytes");

```

```
    delay(4000); // Display stats for 4 seconds
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Barcode Scanner");
}

void displayWiFiStatus() {
    lcd.clear();
    if (WiFi.status() == WL_CONNECTED) {
        lcd.setCursor(0, 0);
        lcd.print("WiFi Connected");
        lcd.setCursor(0, 1);
        lcd.print("IP: " + WiFi.localIP().toString());
        lcd.setCursor(0, 2);
        lcd.print("Signal: " + String(WiFi.RSSI()) + " dBm");
    } else {
        lcd.setCursor(0, 0);
        lcd.print("WiFi Not Connected");
    }
    delay(4000); // Display the status for 4 seconds
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Barcode Scanner");
}

void handleInput() {
    if (inputBuffer.equalsIgnoreCase("manual")) {
        isManualMode = true;
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Manual Mode");
        lcd.setCursor(0, 1);
        lcd.print("Enter name:");
```

```
            return;
        }

        if (inputBuffer.equalsIgnoreCase("clearlist")) {
            clearWebServerItems();
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("List Cleared");
            delay(2000);
            lcd.clear();
            return;
        }

        if (inputBuffer.equalsIgnoreCase("on.")) {
            lcd.backlight();
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("System: ON");
            delay(2000);
            lcd.clear();
            return;
        }

        if (isAdjustingQuantity && currentStep == 0) {
        int newQuantity = inputBuffer.toInt();
        adjustQuantity(currentBarcode, newQuantity);
        isAdjustingQuantity = false; // Disable quantity adjustment mode
        currentStep = 0; // Reset step
        return;
        }

        if (isRegisteringItem || isManualMode) {
            if (currentStep == 0) {
```

```
             currentItemName = inputBuffer;
             currentStep = 1;
             lcd.clear();
             lcd.setCursor(0, 0);
             lcd.print("Enter quantity:");
         } else if (currentStep == 1) {
             currentQuantity = inputBuffer.toInt();
             processNewItem(currentItemName, currentQuantity);
             currentStep = 0;
         }
     }
 }

void processScannedBarcode(const String& barcode) {
 if (barcodeItems.find(barcode) != barcodeItems.end()) {
     currentBarcode = barcode; // Set the current barcode
     barcodeItems[barcode].second++;

     webServerItems[barcode] = barcodeItems[barcode];

     lcd.clear();
     lcd.setCursor(0, 0);
     lcd.print("Item:");
     lcd.setCursor(0, 1);
     lcd.print(barcodeItems[barcode].first + " x" + String(barcodeItems[barcode].second));
 } else {
     currentBarcode = barcode; // Set current barcode for manual registration
     isRegisteringItem = true;
     lcd.clear();
     lcd.setCursor(0, 0);
     lcd.print("Not found:");
     lcd.setCursor(0, 1);
     lcd.print("Enter name:");
```

```
    }
}


    void processNewItem(const String& name, int quantity) {
        if (name.isEmpty() || quantity <= 0) {
            Serial.println("Invalid item or quantity.");
            return;
        }

        // Handle manual or scanned items
        if (isRegisteringItem && !currentBarcode.isEmpty()) {
            barcodeItems[currentBarcode] = {name, quantity};
            webServerItems[currentBarcode] = {name, quantity};
            saveBarcodeItem(currentBarcode, name, quantity);
        } else if (isManualMode) {
            String manualBarcode = "manual_" + String(manualCounter++);
            barcodeItems[manualBarcode] = {name, quantity};
            webServerItems[manualBarcode] = {name, quantity};
            saveBarcodeItem(manualBarcode, name, quantity);
        }

        // Display the registered item
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Registered: ");
        lcd.setCursor(0, 1);
        lcd.print(name + " x" + String(quantity));

        currentBarcode = "";
        isRegisteringItem = false;
        isManualMode = false;
    }
```

```
void saveBarcodeItem(const String& barcode, const String& name, int quantity) {
    preferences.putString(barcode.c_str(), name);
    preferences.putInt((barcode + "_qty").c_str(), quantity);

    String barcodeList = preferences.getString("barcodeList", "");
    if (barcodeList.indexOf(barcode) == -1) {
        barcodeList += (barcodeList.length() > 0 ? "," : "") + barcode;
        preferences.putString("barcodeList", barcodeList);
    }

    webServerItems[barcode] = {name, quantity}; // Save to webServerItems
}

void loadBarcodeItems() {
    String barcodeList = preferences.getString("barcodeList", "");
    if (barcodeList.isEmpty()) return;

    int start = 0, end = barcodeList.indexOf(',');
    while (end != -1) {
        String barcode = barcodeList.substring(start, end);
        String name = preferences.getString(barcode.c_str(), "");
        int quantity = preferences.getInt((barcode + "_qty").c_str(), 0);
        barcodeItems[barcode] = {name, quantity};
        webServerItems[barcode] = {name, quantity};
        start = end + 1;
        end = barcodeList.indexOf(',', start);
    }

    String barcode = barcodeList.substring(start);
    if (!barcode.isEmpty()) {
        String name = preferences.getString(barcode.c_str(), "");
        int quantity = preferences.getInt((barcode + "_qty").c_str(), 0);
```

```
            barcodeItems[barcode] = {name, quantity};
            webServerItems[barcode] = {name, quantity};
        }
    }
void adjustQuantity(const String& barcode, int newQuantity) {
    // Debug output for tracking
    Serial.println("Adjusting quantity...");
    Serial.println("Barcode: " + barcode);
    Serial.println("New Quantity: " + String(newQuantity));

    // Check if the barcode exists
    if (!barcode.isEmpty() && barcodeItems.find(barcode) != barcodeItems.end()) {
        if (newQuantity > 0) {
            // Update the quantity
            barcodeItems[barcode].second = newQuantity;
            webServerItems[barcode].second = newQuantity;

            // Save the new quantity to preferences
            preferences.putInt((barcode + "_qty").c_str(), newQuantity);

            // Display success message on LCD
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("Updated:");
            lcd.setCursor(0, 1);
            lcd.print(barcodeItems[barcode].first + " x" + String(newQuantity));
            delay(2000);
        } else {
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("Invalid Qty");
            delay(2000);
        }
```

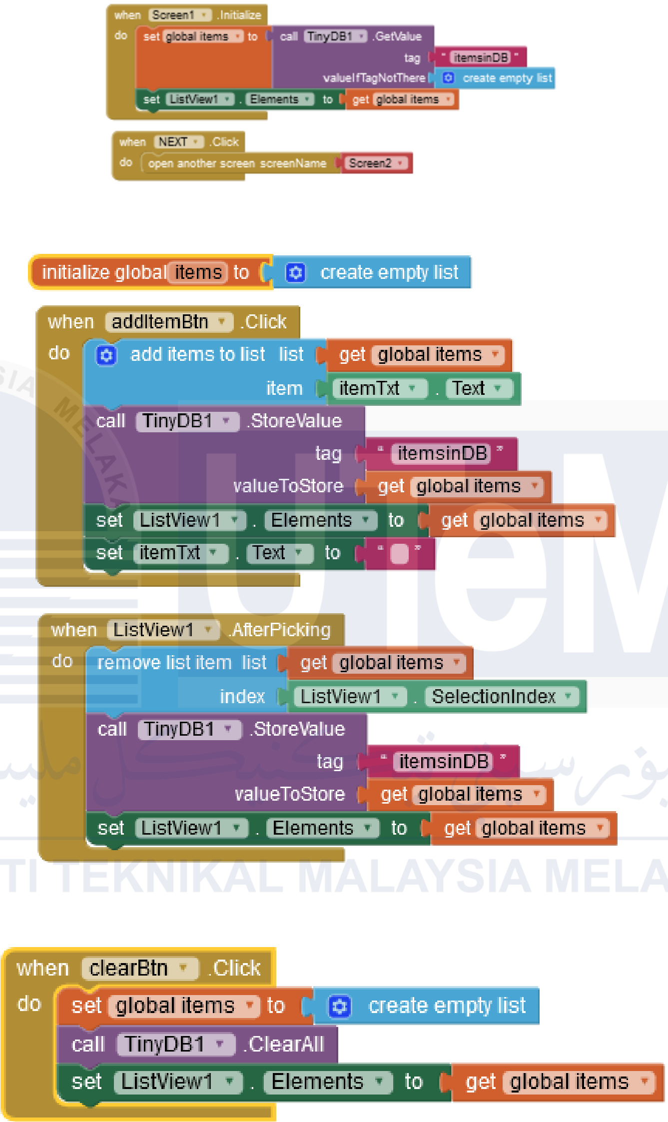
```
    } else {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Item Not Found");
        delay(2000);
    }

    // Return to default display
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Barcode Scanner");
}

    void clearWebServerItems() {
        webServerItems.clear();
        preferences.clear(); // Clear stored preferences
        Serial.println("Webserver list cleared.");
    }

    void displayMenu() {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Menu Options:");
        lcd.setCursor(0, 1);
        lcd.print("1.Manual    2.Clear");
        lcd.setCursor(0, 2);
        lcd.print("3.On        4.Off");
        lcd.setCursor(0, 3);
        lcd.print("5.Exit      6.Stats");
    }

    static String currentBarcode;
    static std::map<String, std::pair<String, int>> barcodeItems;
```

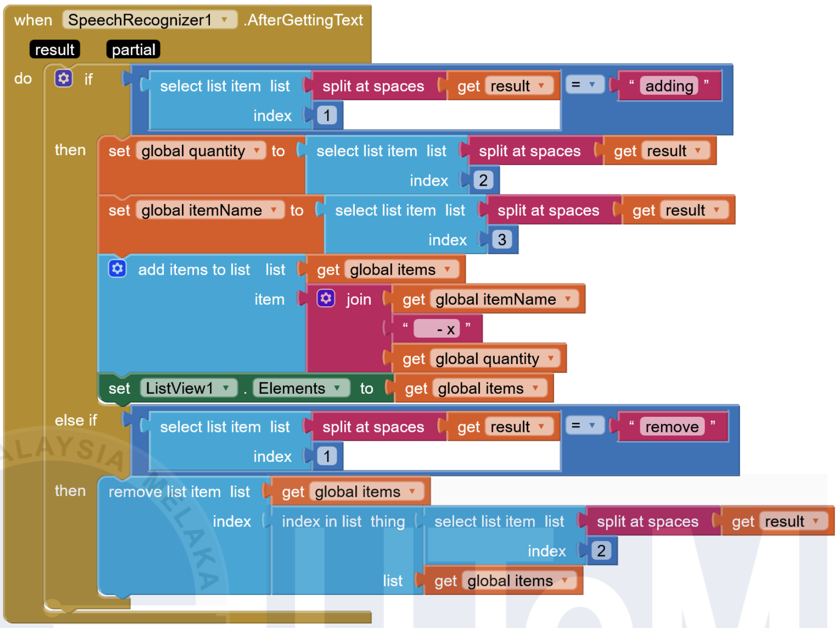
```
    static std::map<String, std::pair<String, int>> webServerItems;
    LiquidCrystal_I2C lcd;
    String inputBuffer = "";
    String currentItemName = "";
    int currentQuantity = 0;
    int currentStep = 0;
    static Preferences preferences;
    static bool isManualMode;
    static bool isMenuMode;
    static bool isRegisteringItem;
    static bool isAdjustingQuantity; // Tracks if in quantity adjustment mode
    static int manualCounter;
};

String MyEspUsbHost::currentBarcode = "";
std::map<String, std::pair<String, int>> MyEspUsbHost::barcodeItems;
std::map<String, std::pair<String, int>> MyEspUsbHost::webServerItems;
Preferences MyEspUsbHost::preferences;
bool MyEspUsbHost::isManualMode = false;
bool MyEspUsbHost::isMenuMode = false;
bool MyEspUsbHost::isRegisteringItem = false;
bool MyEspUsbHost::isAdjustingQuantity = false;
int MyEspUsbHost::manualCounter = 1;

MyEspUsbHost usbHost;
HardwareSerial ScannerSerial(2);

void setup() {
    Serial.begin(115200);
    ScannerSerial.begin(9600, SERIAL_8N1, 15, 16);

    usbHost.begin();
    usbHost.lcd.init();
```

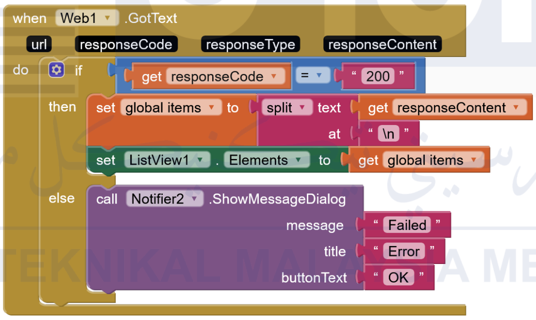
```
usbHost.lcd.backlight();
usbHost.lcd.clear();
usbHost.lcd.print("Barcode Scanner");

MyEspUsbHost::preferences.begin("barcode-db", false);
usbHost.loadBarcodeItems(); // Load stored items

usbHost.lcd.clear();
usbHost.lcd.setCursor(0, 0);
usbHost.lcd.print("Connecting to WiFi..");

WiFi.begin(ssid, password);
unsigned long startAttemptTime = millis();
const unsigned long connectionTimeout = 10000; // 10 seconds timeout

while (WiFi.status() != WL_CONNECTED && millis() - startAttemptTime < connectionTimeout) {
    delay(500);
    Serial.println("Attempting to connect..");
}

if (WiFi.status() == WL_CONNECTED) {
    usbHost.lcd.setCursor(0, 1);
    usbHost.lcd.print("WiFi Connected");
    delay(2000);

    // Start the web server
    server.on("/", []() {
String response = "";
for (const auto& item : MyEspUsbHost::webServerItems) {
    response += item.second.first + " - x" + String(item.second.second) + "\n";
}
if (response.isEmpty()) {
    response = "No items in the system.";
```

```
    }
    server.send(200, "text/plain", response);
});
        server.on("/clearlist", []() {
            usbHost.clearWebServerItems();
            server.send(200, "text/plain", "Webserver list cleared");
        });

        server.begin();
        Serial.println("Web server started");
    } else {
        usbHost.lcd.clear();
        usbHost.lcd.setCursor(0, 0);
        usbHost.lcd.print("WiFi Not Connected");
        delay(2000);
    }

    usbHost.lcd.clear();
    usbHost.lcd.setCursor(0, 0);
    usbHost.lcd.print("Barcode Scanner");
}

void loop() {
    usbHost.task();
    server.handleClient();

    if (ScannerSerial.available()) {
        String scannedBarcode = ScannerSerial.readString();
        scannedBarcode.trim();
        usbHost.processScannedBarcode(scannedBarcode);
    }
}
```

Appendix B: Block codes in MIT App Inventor




when SpeechRecognizer1 .AfterGettingText
result partial
do if select list item list split at spaces get result = "adding"
index 1
then set global quantity to select list item list split at spaces get result
index 2
set global itemName to select list item list split at spaces get result
index 3
add items to list list get global items
item join get global itemName
" - x "
get global quantity
set ListView1 . Elements to get global items
else if select list item list split at spaces get result = "remove"
index 1
then remove list item list get global items
index index in list thing select list item list split at spaces get result
index 2
list get global items


when Web1 .GotText
url responseCode responseType responseContent
do if get responseCode = "200"
then set global items to split text get responseContent
at "\n"
set ListView1 . Elements to get global items
else call Notifier2 .ShowMessageDialog
message "Failed"
title "Error"
buttonText "OK"


when Button1 .Click
do set Web1 . Url to "https://script.google.com/macros/s/AKfycbwRv06wJ..."
call Web1 .Get


when Web1 .GotText
url responseCode responseType responseContent
do set global responseData to call Web1 .JsonTextDecode
jsonText get responseContent
for each item in list get global responseData
do if contains text get value for key "Item"
in dictionary get item
or if not found "not found"
piece TextBox2 . Text
then set Label1 . Text to join "Calories : "
get value for key "Calories (kcal)"
in dictionary get item
or if not found "not found"
set Label2 . Text to join "Protein : "
get value for key "Protein (g)"
in dictionary get item
or if not found "not found"
set Label3 . Text to join "Fat : "
get value for key "Fat (g)"
in dictionary get item
or if not found "not found"
set Label4 . Text to join "Default Serving Size : "
get value for key "Serving Size (g)"
in dictionary get item
or if not found "not found"
set Label5 . Text to join "CARBOHYDRATES : "
get value for key "Carbohydrates (g)"
in dictionary get item
or if not found "not found"
set Label6 . Text to join "FIBER : "
get value for key "Fiber (g)"
in dictionary get item
or if not found "not found"
set Label7 . Text to join "SUGARS : "
get value for key "Sugars (g)"
in dictionary get item
or if not found "not found"
set Label8 . Text to join "SODIUM : "
get value for key "Sodium (mg)"
in dictionary get item
or if not found "not found"
break
if Label1 . Text = " "
then set Label1 . Text to "Item not found in the database."
set Label2 . Text to " "
set Label3 . Text to " "
set Label4 . Text to " "