

ANDROID SMART PARKING SYSTEM

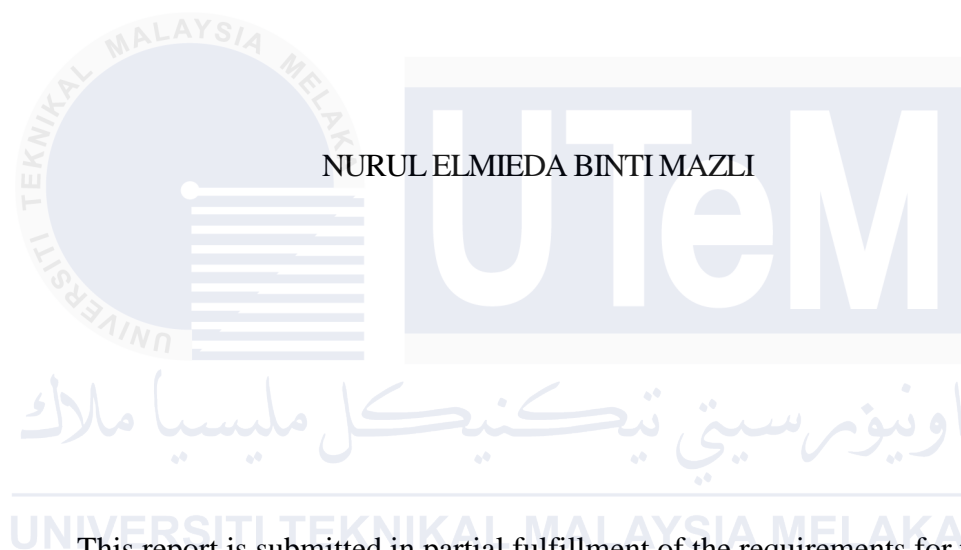


NURUL ELMIEDA BINTI MAZLI

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

ANDROID SMART PARKING SYSTEM

NURUL ELMIEDA BINTI MAZLI



This report is submitted in partial fulfillment of the requirements for the
Bachelor of Computer Science (Computer Networking) with Honours.

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY UNIVERSITI
TEKNIKAL MALAYSIA MELAKA

2024

DECLARATION

I hereby declare that this project report entitled
ANDROID SMART PARKING SYSTEM
is written by me and is my own effort and that no part has been plagiarized
without citations.

STUDENT: NURUL ELMIEDA BINTI MAZLI Date : 3 SEPTEMBER 2024

I hereby declare that I have read this project report and found

this project report is sufficient in term of the scope and quality for the award of

Bachelor of Computer Science (Computer Networking) with Honours.

SUPERVISOR : TS. ERMAN BIN HAMID

Date : 3 SEPTEMBER 2024

DEDICATION

I would like to dedicate this report to all those who have been through my ups and down in completing this project. First and foremost, I would like to thanks to my supervisor, Ts. Erman Bin Hamid for guiding me through the whole process of the project. Other than that, to my family and friends who always give me motivation and support to finish this final year project. Without them, I can not finish this project alone.



ACKNOWLEDGEMENTS

I would like to say thank you so much to my supervisor, Ts. Erman Bin Hamid and my evaluator En Nazrulazhar Bin Bahaman evaluate and gave me some feedback on my project to be improved in the future. I really appreciate all the motivation and support that they give to me.

Most importantly, thank you also to my family, supervisor and friends for their external support and for encouraging me to complete this project. Without all of them, I cannot finish this project alone. All the support that they give me has encouraged me to finish this project.

I also want to express my gratitude to my parents, who helped me finish my final year project on time by encouraging me and helping me out financially. In addition, I want to dedicate myself to my friends, classmates, adversaries, and everyone else who has helped me with my final year project by helping me stay motivated, sharing their knowledge, and being kind.

Last but not least, I wanna thank me, I wanna thank me for believing in me, I wanna thank me for doing all this hard work, I wanna thank me for having no days off, I wanna thank me for never quitting, I wanna thank me for always being a giver and trying to give more than I receive, I wanna thank me for tryna do more right than wrong. I wanna thank me for just being me at all times.

ABSTRACT

Nowadays, the number of vehicles in our country is increasing from time to time. Because of this, there are often traffic jams when cars go around looking for an empty parking space. The android smart parking system is one of the new technologies where drivers can quickly find empty parking space through an application on a mobile phone. With this technology, it can help reduce traffic congestion, waste time, and waste car fuel. For drivers, the Android application will display real-time information about parking spaces that are still empty. This means no more driving around for several minutes to find an empty parking spaces. On the other hand, users will be able to continue navigating to an empty parking spaces and can even save time and reduce frustration. This technology uses an IR sensor to be installed in the parking lot to detect whether it is occupied or empty and the data collected by this sensor will be sent to a mobile application or smartphone.

ABSTRAK

Pada masa kini, jumlah kenderaan di negara kita semakin bertambah dari semasa ke semasa. Oleh sebab itu, sering kali terjadi kesesakan lalu lintas apabila kereta berkeliling bagi mencari tempat letak kereta yang kosong. Sistem tempat letak kereta pintar android merupakan salah satu teknologi baru dimana pemandu dapat mencari tempat letak kereta yang kosong dengan pantas melalui aplikasi di telefon bimbit. Dengan teknologi ini ia dapat membantu pemandu mengurangkan kesesakan lalu lintas, membuang masa, dan membazir minyak kereta. Bagi pemandu, aplikasi Android akan memaparkan maklumat masa nyata mengenai tempat letak kereta yang masih kosong. Ini bermakna tiada lagi pemandu yang akan berkeliling selama beberapa minit bagi mencari tempat paker yang kosong. Sebaliknya, pengguna akan dapat terus mengemudi ke tempat parkir yang kosong malah dapat menjimatkan masa dan mengurangkan kekecewaan. Teknologi ini menggunakan sensor IR akan dipasang di tempat letak kereta untuk mengesan sama ada ia diduduki atau kosong dan data yang dikumpul oleh sensor ini akan dihantar ke aplikasi mudah alih atau telefon pintar.

TABLE OF CONTENT

| | |
|--|----|
| BORANG PENGESAHAN STATUS LAPORAN | 2 |
| DECLARATION | 4 |
| DEDICATION | 5 |
| ACKNOWLEDGEMENTS | 6 |
| ABSTRACT..... | 7 |
| ABSTRAK..... | 8 |
| CHAPTER 1: INTRODUCTION | 16 |
| 1.1 Introduction..... | 16 |
| 1.2 Problem statement (PS)..... | 17 |
| 1.3 Project Question (PQ)..... | 17 |
| 1.4 Project Objective (PO)..... | 18 |
| 1.5 Project Research Hypothesis..... | 18 |
| 1.6 Project Scope | 19 |
| 1.7 Project Contribution (PC)..... | 20 |
| 1.8 Conclusion | 20 |
| CHAPTER 2: LITERATURE REVIEW AND METHODOLOGY | 21 |
| 2.1 Introduction..... | 21 |
| 2.2 Research Problem | 22 |
| 2.3 Related Work/Previous Work..... | 24 |
| 2.4 Critical review of current problem and justification | 35 |
| 2.5 Proposed Solution/further project..... | 42 |
| 2.6 Conclusion | 44 |
| CHAPTER 3: METHODOLOGY | 45 |
| 3.1 Introduction..... | 45 |
| 3.2 Research process | 45 |
| 3.2.1 Data Collection | 46 |
| 3.2.2 Analysis | 46 |
| 3.2.3 Design | 47 |
| 3.2.4 Implementation | 47 |
| 3.2.5 Testing | 49 |
| 3.3 Methodology | 50 |

| | |
|---|----|
| 3.4 Conclusion | 51 |
| CHAPTER 4: ANALYSIS AND DESIGN | 52 |
| 4.1 Introduction..... | 52 |
| 4.2 Problem Analysis | 52 |
| 4.3 Requirement Analysis..... | 52 |
| 4.3.1 Data Requirement..... | 53 |
| 4.3.2 Functional Requirement..... | 53 |
| 4.4 Hardware Requirement..... | 54 |
| 4.5 Software Requirement..... | 57 |
| 4.6 High-Level Design..... | 59 |
| 4.6.1 System Architecture..... | 59 |
| 4.6.2 Interface Design..... | 60 |
| 4.7 Conclusion | 65 |
| CHAPTER 5: IMPLEMENTATION..... | 66 |
| 5.1 Introduction..... | 66 |
| 5.2 Development Environment Setup..... | 66 |
| 5.2.1 Hardware Development Setup..... | 66 |
| 5.2.1.1 Hardware Installation | 67 |
| 5.3 Software Development Environment Setup..... | 68 |
| 5.4 Software Configuration Management..... | 70 |
| 5.4.1 Configuration Environment Setup..... | 71 |
| 5.4.2 Version Control Procedure | 79 |
| 5.5 Implementation Status..... | 81 |
| 5.6 Conclusion | 82 |
| CHAPTER 6: TESTING..... | 83 |
| 6.1 Introduction..... | 83 |
| 6.2 Test Plan | 83 |
| 6.2.1 Test Organization..... | 83 |
| 6.2.2 Test Environment..... | 84 |
| 6.2.3 Test Schedule..... | 84 |
| 6.3 Test Strategy..... | 86 |
| 6.3.1 Classes Test..... | 87 |
| 6.4 Test Design..... | 87 |

| | |
|--|-----|
| 6.4.1 Test Description | 87 |
| 6.4.2 Test Data | 90 |
| 6.5 Test Result and Analysis | 93 |
| 6.5.1 Test Result on Hardware..... | 93 |
| 6.5.2 Test Result on Application | 95 |
| 6.6 Usability Test..... | 96 |
| 6.6.1 Result..... | 98 |
| 6.6.2 Interpreting Score..... | 102 |
| 6.7 Conclusion | 103 |
| CHAPTER 7: CONCLUSION..... | 104 |
| 7.1 Introduction..... | 104 |
| 7.2 Project Summarization..... | 104 |
| 7.3 Project Contribution..... | 106 |
| 7.4 Project Limitation | 106 |
| 7.5 Future Works | 107 |
| 7.6 Conclusion | 108 |
| REFERENCE..... | 109 |
| APPENDICES | 111 |
| APPENDICES A | 111 |
| APPENDICES B | 113 |

LIST OF TABLE

| | |
|---|-----|
| Table 1.1: Summary of Problem Statement..... | 17 |
| Table 1.2: Summary of Project Question..... | 17 |
| Table 1.3: Summary of the Project Objectives | 18 |
| Table 1.4: Table of Project Contribution | 20 |
| Table 2.1: Summary of Research problem | 22 |
| Table 2.2: Summary of critical review | 35 |
| Table 2.3: Comparison between the system | 42 |
| Table 5.1: Details of the pin number between the hardware | 67 |
| Table 5.2: NodeMCU ESP8266 Environment Setup..... | 69 |
| Table 5.3: Show the Development and Configuration of the ESP 8266 | 71 |
| Table 5.4: Software Implemented into Component..... | 75 |
| Table 5.5: Software Implemented into Component..... | 77 |
| Table 5.6: The implementation of status project. | 81 |
| Table 6.1: The Wi-Fi module function | 88 |
| Table 6.2: Infrared sensor function..... | 88 |
| Table 6.3: Smart Parking System function | 89 |
| Table 6.4: ESP8266 function result and analysis..... | 93 |
| Table 6.5: Sensor function result and analysis..... | 95 |
| Table 6.6: Testing on the application..... | 95 |
| Table 6.7: SUS Score Grading..... | 103 |

LIST OF FIGURES

| | |
|---|----|
| Figure 1.1: Project Research Hypothesis | 19 |
| Figure 2.1: The prototype of the research | 25 |
| Figure 2.2: The Functionalities of the System | 26 |
| Figure 2.3: Architecture of the Proposed System | 27 |
| Figure 2.4: Proposed System Block Diagram..... | 28 |
| Figure 2.5: Proposed System Architecture | 29 |
| Figure 2.6: The prototype of research with proximity sensor..... | 30 |
| Figure 2.7: The prototype of the project | 31 |
| Figure 2.7.1: The RFID Scanner and RFID Card | 31 |
| Figure 2.8: The SPIN-V Iot Sensor Prototype | 32 |
| Figure 2.9: The Hardware system of the research | 33 |
| Figure 2.10: The Hardware of the research | 34 |
| Figure 2.11 The functionality of the Android Smart Parking System | 44 |
| Figure 3.1: Flow of Research process..... | 46 |
| Figure 3.2: System Architecture | 47 |
| Figure 3.3: Hardware Connection..... | 48 |
| Figure 3.4: The functionality of Android Smart Parking System | 49 |
| Figure 3.5: The process of Rapid Application Development | 50 |
| Figure 4.1: Data Flow of the project..... | 53 |
| Figure 4.2: Block Diagram of Functional Requirement | 53 |
| Figure 4.3: NodeMCU ESP8266 | 55 |
| Figure 4.4: Infrared sensors | 55 |
| Figure 4.5: Male-to-male jumper wire..... | 56 |
| Figure 4.6: Breadboard | 56 |
| Figure 4.8: 16x2 LCD | 57 |
| Figure 4.9: IDE Interface | 58 |
| Figure 4.10: Android Studio Interface | 59 |
| Figure 4.11: Firebase database Interface | 59 |
| Figure 4.12 : System Architecture | 60 |
| Figure 4.13: Login page Interface..... | 60 |

| | |
|--|----|
| Figure 4.14: Register user Interface..... | 61 |
| Figure 4.15: Home page Interface | 61 |
| Figure 4.16: Reset Password Interface..... | 62 |
| Figure 4.17: Status parking Interface..... | 62 |
| Figure 4.18: Edit Profile Interface | 63 |
| Figure 4.19: Feedback Interface | 63 |
| Figure 4.20: Submit Feedback for General Interface..... | 64 |
| Figure 4.21: Submit Feedback for app Interface | 64 |
| Figure 4.22: Help Center Interface | 65 |
| Figure 5.2: Hardware Setup | 67 |
| Figure 5.3: The GPIO setup hardware circuit..... | 68 |
| Figure 5.4: Final Product | 68 |
| Figure 5.5: Flowchart of the development and configuration of ESP8266 | 72 |
| Figure 6.5: Website for Arduino IDE installer | 73 |
| Figure 5.7: Arduino IDE coding interface | 73 |
| Figure 5.8: Open preferences to enter library ESP8266. | 74 |
| Figure 5.9: Step-by-step to open board manager for installing the ESP8266..... | 74 |
| Figure 5.10: ESP8266 Board Package..... | 74 |
| Figure 5.11: Output of the installation..... | 75 |
| Figure 5.12: Create a project name in Firebase | 76 |
| Figure 5.13: Add the Firebase to the Android application..... | 76 |
| Figure 5.14: Create a Realtime Database..... | 77 |
| Figure 5.15: Android Studio Installer | 78 |
| Figure 5.16: Create New Project..... | 78 |
| Figure 5.17: To connect them, add the Firebase package..... | 79 |
| Figure 5.18: Show the coding that is used to connect the component Wi-Fi module and firebase cloud in the Arduino IDE..... | 80 |
| Figure 5.19: Show the coding for Infrared sensors 1,2 and 3. | 80 |
| Figure 5.20: Show the coding update into Firebase cloud..... | 81 |
| Figure 6.1: Show the prototype's role. | 85 |
| Figure 6.2: Show the functionality of the prototype. | 85 |

| | |
|--|-----|
| Figure 6.3: Show the connection to the Firebase | 85 |
| Figure 6.4: Show how to use the Smart Parking System application. | 86 |
| Figure 6.5: ESP8266 connected with laptop..... | 90 |
| Figure 6.6: The details of port uses..... | 91 |
| Figure 6.7: Setup connection between hardware | 91 |
| Figure 6.8: Android Studio's software..... | 92 |
| Figure 6.9: User information from database | 93 |
| Figure 6.10: Scope status from database..... | 93 |
| Figure 6.11: ESP8266 success to connect the internet | 94 |
| Figure 6.12: Name of the responses..... | 98 |
| Figure 6.13: Age of the responses..... | 98 |
| Figure 6.14: Answer question 1 from respondents | 99 |
| Figure 6.15: Answer question 2 from respondents | 99 |
| Figure 6.16: Answer question 3 from respondents | 99 |
| Figure 6.17: Answer question 4 from respondents | 100 |
| Figure 6.18: Answer question 5 from respondents | 100 |
| Figure 6.19: Answer question 6 from respondents | 100 |
| Figure 6.21: Answer question 7 from respondents | 101 |
| Figure 6.22: Answer question 8 from respondents | 101 |
| Figure 6.23: Answer question 9 from respondents | 101 |
| Figure 6.24: Answer question 10 from respondents | 102 |

CHAPTER 1: INTRODUCTION

1.1 Introduction

As the population increased in the metropolitan cities, the usage of vehicles increased. Finding a car parking space in most metropolitan areas, especially during the rush hours, is difficult for drivers. Indisciplinary parking may result in damage to the car. Thus there is a need to provide sufficient parking spaces coupled with plenty of slots to help the user park his car safely. The Android Smart Parking System is introduced in this to make parking easier for everyone. For drivers, our Android app will display real-time information about available parking spaces nearby. This means no more aimlessly driving around in circles. Instead, the user will be able to navigate directly to an open spot, saving time, and frustration. IR sensors will be installed in parking spaces to detect whether they are occupied or vacant and the data collected by these sensors will then be sent to a mobile app or smartphone. It also contains an LCD to show the number of vacancies the parking has and it also shows the respective slot which is empty or filled.

1.2 Problem statement (PS)

Drivers often struggle to find parking spaces in buildings, particularly during peak hours. This leads to traffic congestion within the building and can cause confusion for drivers unfamiliar with the layout, even with the assistance of parking guidance systems. Consequently, time and petrol are wasted as drivers search for parking spots.

Table 1.1: Summary of Problem Statement

| PS | Problem Statement |
|-----|---|
| PS1 | Drivers tend to waste their time and fuel in search of empty slots. |
| PS2 | Drivers parking in non-parking areas cause jams on the road which can lead to accidents. |
| PS3 | Some of the existing smart parking systems do not provide smart parking applications that show the empty parking slot to the users. |

1.3 Project Question (PQ)

Project Research Questions are used to identify the question of the existing Android smart parking system. Based on the research, we can conclude that there are a few weaknesses in the current smart parking system. The table 1.2 shows the summary of the project research question.

Table 1.2: Summary of Project Question

| PS | PQ | Project Question |
|-----|-----|--|
| PS1 | PQ1 | How user can understand the technique of smart parking IoT applications? |
| PS2 | PQ2 | How to design and develop the smart parking IoT application? |
| PS3 | PQ3 | What is the usability of a smart parking IoT application? |

1.4 Project Objective (PO)

The improvement that would intend to attain by the time the project is completed is outlined in the objective of the project. This project's problem statement and project question need to be taken into consideration before making any decisions regarding the improvement. The objectives for this project are shown below.

Table 1.3: Summary of the Project Objectives

| PS | PQ | PO | Project Objectives |
|-----|-----|-----|--|
| PS1 | PQ1 | PO1 | To design and study the problem of the existing parking system and the possibility of enhancement. |
| | | PO2 | To develop the Smart Parking IoT application that could alert the owner about the parking through the application. |
| | | PO3 | To test the smart parking IoT application's effectiveness. |

1.5 Project Research Hypothesis

A research hypothesis is a statement made by researchers to improve the outcome of their investigation. According to studies, the present smart parking system lacks features and is unsatisfactory. Some hypothesis have been proposed to improve the existing smart parking system. Figure 1.1 shows the problem with the present smart parking system as well as the hypothesis for improvement.

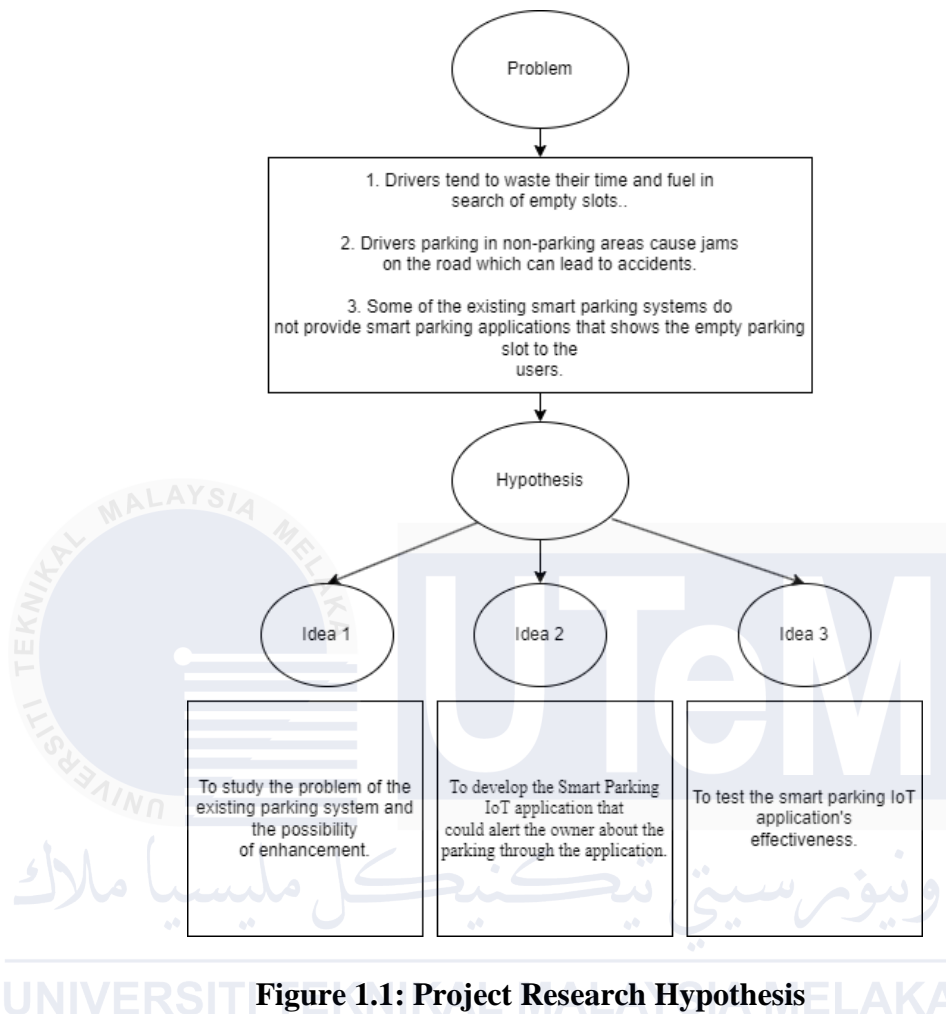


Figure 1.1: Project Research Hypothesis

1.6 Project Scope

A prototype application running on the Android operating system is the project's final output. For now, the application can only be downloaded and used on Android-powered handsets. The application is intended for people who utilize parking guidance system-equipped designated parking lots. In addition to more precise information, the application will be made to provide users with real-time information on the availability of parking spaces. The parking assistance system will still be in existence, but the application will help it along by maximizing its potential and improving its usefulness. As a result, the outcome will be more effective than it was with the first parking guidance system.

1.7 Project Contribution (PC)

The contribution of the project defines the expected production of the project. This section is comprised of the project's objectives. The contribution to the strive can be identified in the table below.

Table 1.4: Table of Project Contribution

| PS | PQ | PO | PC | Project Contribution |
|-----|-----|-----|-----|--|
| PS1 | PQ1 | PO1 | PC1 | Suggested creating an Android studio app to use a smartphone to monitor the parking system. |
| PS2 | PQ2 | PO2 | PC2 | Proposed a hardware solution that functions as a smart parking system and uses a NodeMCU ESP8266 to connect to an Android application. |

1.8 Conclusion

This chapter concludes the purpose, background, and summary of the project. To sum up, this chapter has given the background to this project and a summary of it. The problem statement subsection presents the current problems faced by modern technologies. In addition to meeting these needs, this project aims to produce an Android application that is even more effective than normal one can be more secure and user-friendly enough for everyone to enjoy. To this end, the chapter also describes the pictures and shows strategies for doing this project. This system can meet the goals laid out above for such a project. This systematic report will go through each step-by-step process it follows the sequence of phases that a project should then the objectives of the project have been attained and the android smart parking system.

CHAPTER 2: LITERATURE REVIEW AND METHODOLOGY

2.1 Introduction

This chapter discusses the problem and solution of the existing smart parking system. In order to have a better understanding of the concept and technique needed to be implemented in this project, this chapter also contains published data and materials, articles, and research findings from earlier projects that are relevant to the goal of this project.

In this chapter, we will summarize the research like an academic paper that includes current information, significant findings, and theoretical and methodological contributions to linked studies.. Then, we will discuss key topics, like current problems and ways to improve for the future. We will also talk about how the project is progressing, focusing on gathering data, checking what the current system can do, and finding its weaknesses. The project flow will also be utilised to describe extra tasks and functions. This is being done to collect information, evaluate the capabilities of the current system, and discover its shortcomings. Enhancements can be made to strengthen the project's ability to handle user complaints, starting with the weak point. By starting with what's not working well, we will make changes to better meet the user's needs. Before selecting a better method for the task, we will compare all the facts and information gathered from journals and other sources to find the best way forward.

This chapter's first part will have a look at related work. This will show how the various parts work together to form an Smart Parking System. Moreover, this chapter also compares the equipment and tools that are most suited for the project. The main goal of this project is to produce hardware and software that interacts with an Android smartphone and a NodeMCU.

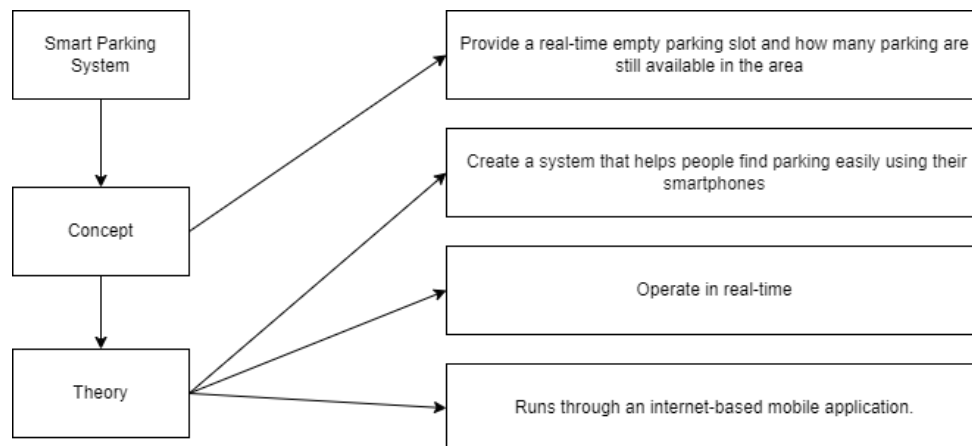
2.2 Research Problem

The research problem in a literature review report is the main issue or problem that the author hopes to solve by reviewing the results of previous research. It establishes the review's objectives and parameters and provides direction for choosing and analysing relevant research. A clear, understandable, and important research problem is required.

i. Concept

According to (Vermesan, O. and Friess, P, 2014), IoT is a platform that has ability to transmit data over a network to make things simpler in daily life activities. Study have been conducted that IoT is one of the fastest growing technology that uses physical devices on the network. It is embedded with any electronics, applications, sensors and network connectivity that allows the communication and exchange of data. The purpose of the Smart parking system is to help drivers find a parking slot easily without wasting time and fuel where it will run in real time. This system can be connected to a smartphone where the user can check the available parking slots and how much parking is still available in the area. It's all about using technology to make parking less stressful and more convenient for everyone.

Table 2.1: Summary of Research problem



ii. Theory

The Android Smart Parking project is about making parking easier using smartphones. It's like a helper that tells you where you can park and lets you reserve a spot ahead of time. With this project, you can use your phone to find parking quickly, pay for it easily, and even keep track of your parking spot. It's all about using technology to make parking less stressful and more convenient for everyone.

iii. Application

Numerous varieties of smart parking systems are being released onto the market. The most well-known and prevalent are:

1. IoT-based Smart Parking System using Android Application (Nor Bakiah Abd Warif, Mohd Izzat Syahmi Saiful Azman, Nor-Syahidatul N Ismail, Muhammad Akmal Remli, 2020)
2. Smart Car Parking System Solution for the Internet of Things in Smart Cities (Wael Alsafery, Badraddin Alturki, Stephan Reiff-Marganiec, Kamal Jambi, 2018)
3. Smart Parking System with PlacePod, LoRaWAN IoT sensors and Android app (Sofeem Nasim, Mourad Oussalah, Tarja Outila, Johannes Jutila, 2022)
4. An Efficient Car Parking Management System using raspberry-pi (G. Aravindh, M. Arun Kumar, 2020)
5. Automated Car Parking System Commanded by Android Application (Mrs. D.J. Bonde, Rohit Sunil Shende, Ketan Suresh Gaikwad, Akshay Sambhaji Kedari, Amol Uday Bhokre, 2014)

6. ITS for Smart Parking Systems, towards the creation of smart city services using IoT and cloud approaches (Luis Felipe Herrera-Quintero, Julián Vega-Alfonso, Diego Bermúdez, Luis Andres Marentes, Klaus Banse, 2019)
7. IOT Based Smart Parking System With E-Ticketing (Chinnabattuni Avinash, Gaddam Rohit, Chintakrindhi Rajesh, Aala Suresh, Sunil Chinnadurai, 2022)
8. IoT-based Smart Vehicle Presence Sensor SPIN-V for Smart Parking System (David A. Michel-Torres, Luis F. Luque-Vega, Emmanuel Lopez-Neri, Miriam A. Carlos-Mancilla, Luis E. Gonzalez-Jiménez, 2019)
9. Navigation based -Intelligent Parking Management System using Queuing theory and IOT (Divya pandey, Seema Hanchate Author , 2018)
10. An IoT Based Smart Parking System (Mehala Chandran, Nur Fadila Mahrom, Thennarasan Sabapathy, Muzammil Jusoh, Mohd Nasrun Osman, Mohd Najib Yasin, N.A.M Hambali , R.Jamaluddin, N.Ali, Yasmin Abdul Wahab, 2019)

2.3 Related Work/Previous Work

This section provides an overview of related work, previous work, and studies after the researchers conducted a comprehensive and in-depth search. It also explains and exposes the purpose of each tool, concept, and theoretical framework so that the reader can finish the project and fully understand the basic concept. Lastly, it specifies how each research project will benefit from the project/tool.

Nor Bakiah Abd Warif, Mohd Izzat Syahmi Saiful Azman, Nor-Syahidatul N Ismail, Muhammad Akmal Remli in their research on “IoT-based Smart Parking System using

Android Application (2020)”, as shown in Figure 2.1 introduced the Android-based Car Parking Monitoring System (ACPMS), an IoT-based smart parking system that uses an Android application. There are three main implementations which are a mobile application, database, and numerous hardware connections are used in the design of the Android Car Parking Monitoring System. The information about the availability of the car will be obtained by connecting an Arduino Uno to an ultrasonic sensor, and the data will be sent over a WiFi shield to a cloud database. In addition to being utilised as a means of establishing an internet connection, a Wi-Fi shield repeatedly loops to determine whether the parking lot is available. There will be periodic transfers of the data to servers. Next, all of the data that has been sent from the Arduino Uno in real-time may be retrieved and displayed via the ACPMS mobile application. Thus, by using the established ACPMS, users can get the most recent information regarding the availability of automobile parking lots.

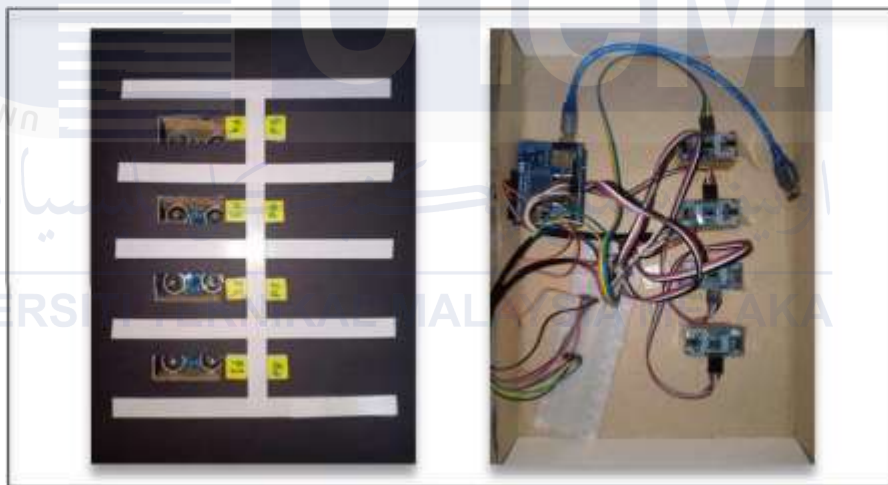


Figure 2.1: The prototype of the research

Wael Alsafery, Badraddin Alturki, Stephan Reiff-Marganiec, Kamal Jambi in their research on “Smart Car Parking System Solution for the Internet of Things in Smart Cities (2018)” proposed a Smart car parking system solution for the Internet of Things in Smart Cities. Their technology not only offered data on the quantity and closest parking spots, but it also provided information on the state of traffic congestion on the roadways. They used a machine learning technique that was based on data processing and analysis using the particular data they

had independently obtained. Before providing customers with information about traffic congestion and the closest parking spot, they employ cloud web services to gather data from fog microcontroller dispersed devices surrounding the users, and analyse, and process the data. The system needs to put in a lot of work, which is expensive, to gather the data, and the machine learning algorithm shouldn't be overly complicated to process large amounts of data quickly. Features of machine learning and deep learning are still in their infancy. As a result, the implementations call for additional infrastructure costs for hardware and software.

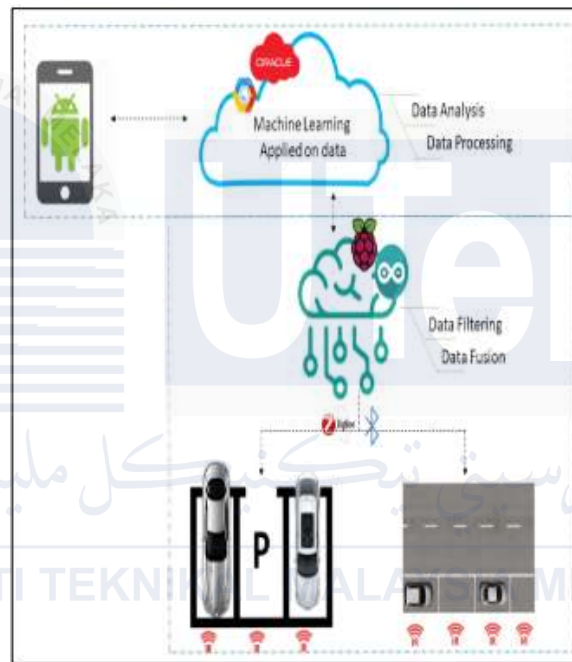


Figure 2.2: The Functionalities of the System

Sofeem Nasim, Mourad Oussalah, Tarja Outila, Johannes Jutila in their “Smart Parking System with PlacePod, LoRaWAN IoT sensors and Android app (2022)”. There are five primary parts to the system which is Placepod Sensor, Data Communication module, Cloud Message Broker, Cloud platform and, Mobile Application. A series of PNI PlacePod sensors that are buried underground have been installed for this reason. These sensors allow data packets to be transmitted to a cloud server via communication with the LoRa network. An Android application has been created on the client side to communicate with users and offer various features. This

program allows the user to view the history of each parking lot's status, estimate the availability of a certain parking lot using a straightforward linear regression-based method, and learn about the parking lot's availability and real-time parking status.

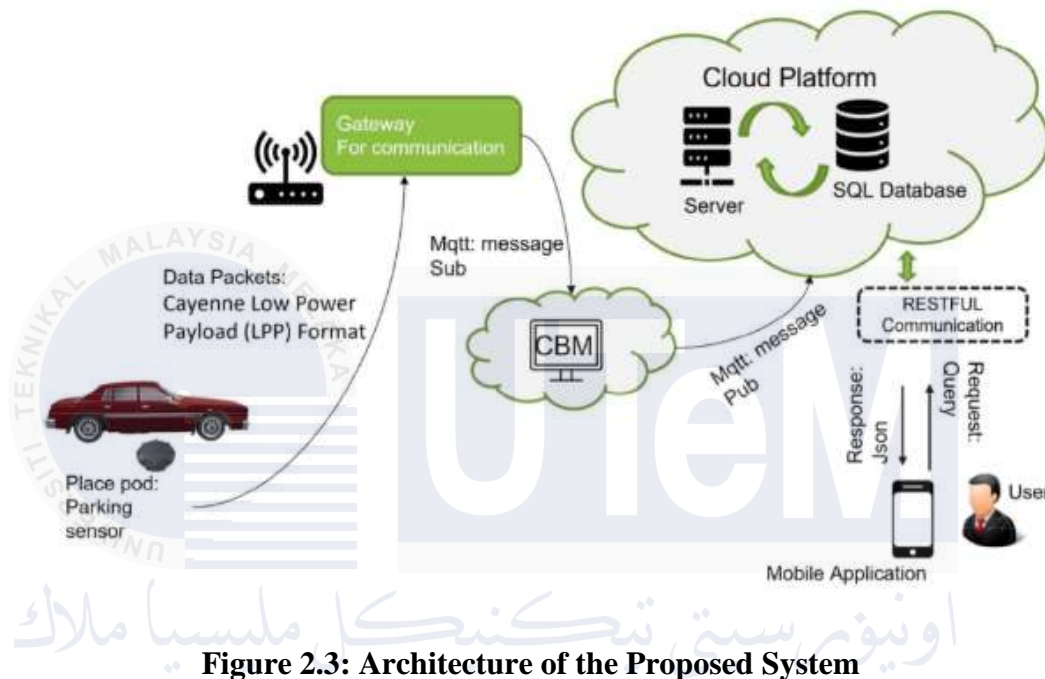


Figure 2.3: Architecture of the Proposed System

G. Aravindh, M. Arun Kumar in their research “An Efficient Car Parking Management System using Raspberry Pi (2020)” proposed a car parking management system by using Raspberry Pi that is simple, affordable, and effective solution for parking cars. With the use of an Android application, this system can keep an eye on and manage individual parking spaces. The Raspberry Pi processor receives the detecting signal from the sensor, and after processing it, data is sent to an Android smartphone via Wi-Fi by turning on a new app. Monitoring the number of unoccupied parking spaces on each floor was the primary goal of the Java platform-based programme. The technology provides a cost-effective and accurate solution, and it can direct customers to park their vehicles in available places while adhering to safety regulations. The RPI objects are linked to a variety of proximity sensors that are positioned

without making physical touch. By turning on Wi-Fi on the Android mobile, users can communicate with the RPI's detected output.

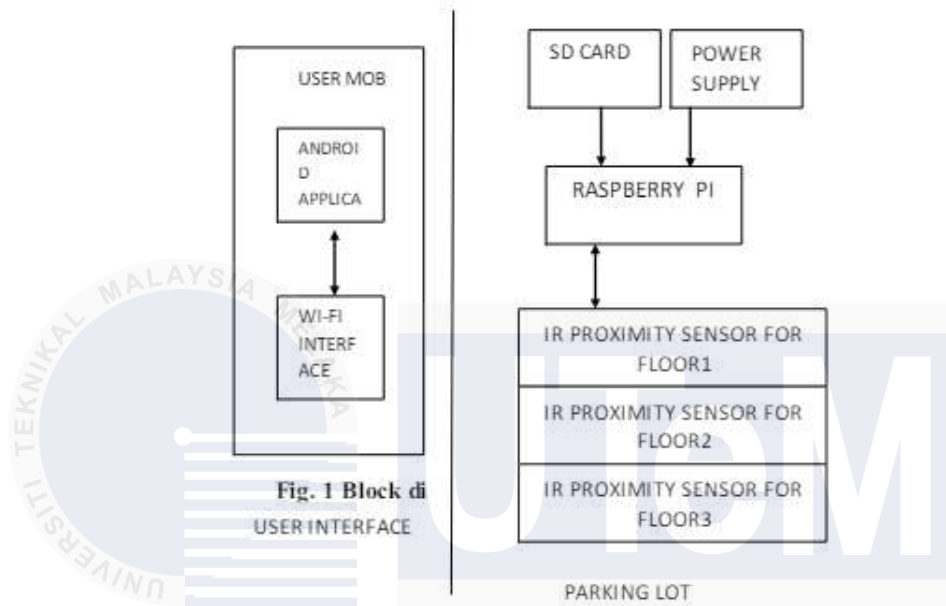


Figure 2.4: Proposed System Block Diagram

Mrs. D.J. Bonde, Rohit Sunil Shende, Ketan Suresh Gaikwad, Akshay Sambhaji Kedari, Amol Uday Bhokre in their “Automated Car Parking System Commanded by Android Application (2014)”. In this research the system propose a automated car parking system that divided into following 4 modules which is Interfacing LCD with Microcontroller, Interfacing GSM with Microcontroller, Interfacing RF Module with Microcontroller, Android Application Development. The communication between the car's microcontroller and the parking area would provide this information. The car would then follow the path to the available parking space and park when the information was collected. The LCD display's data would automatically update. Then, the driver would use the Android application to transmit an encoded "Get My Car" message. This message would be retrieved at the parking area unit, and the car in question would follow its route back to its starting point based on the data recorded.

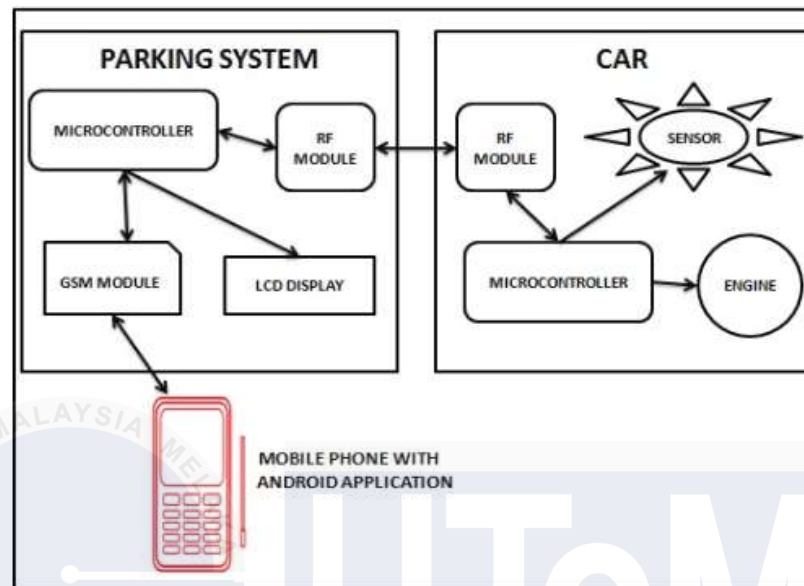


Figure 2.5: Proposed System Architecture

Luis Felipe Herrera-Quintero, Julián Vega-Alfonso, Diego Bermúdez, Luis Andres Marentes, Klaus Banse in their “ITS for Smart Parking Systems, towards the creation of smart city services using IoT and cloud approaches (2019)”. The suggested architecture is built upon established principles including the NoSQL method, cloud computing, and service-oriented architecture. Using electronic sensors that are linked to the internet as a data source, provides advantages for automation and data collection. Furthermore, a data pipeline for integration into complex machine learning or analytical systems is suggested. They also suggested two secure methods to implement smart city services which is an Internet of Things strategy to monitor parking lot activity using ISO 20922-defined application layer protocols, communications using the Message Queuing Telemetry Transport (MQTT) protocol, a lightweight messaging protocol that runs on top of the TCP/IP protocol, it also using cloud technologies from Google and Amazon to benefit from IT platform services. Lastly, an ITS-related smart city service (smart parking systems) is used as a proof of concept. Several sensor networks, including sensors and raspberries, with proximity sensors connected are used in the study. Sensors simulate the parking lot being

occupied. The device switches from an unoccupied to an occupied state when an object enters the sensor's reading range and stays there for more than sixty seconds.

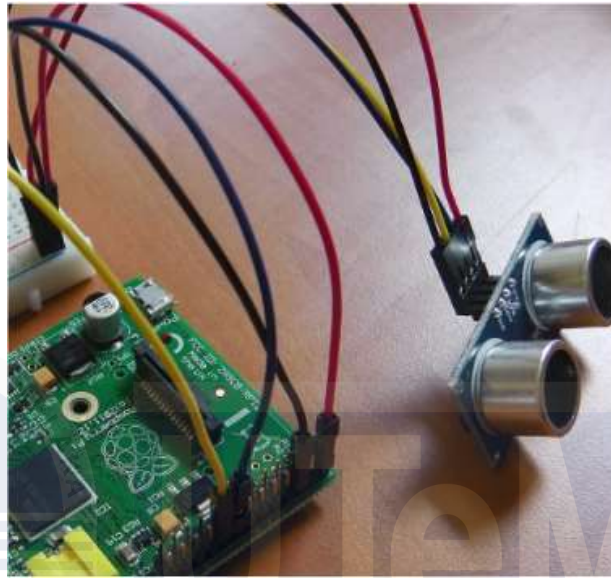


Figure 2.6: The prototype of research with proximity sensor

Chinnabattuni Avinash, Gaddam Rohit, Chintakrindhi Rajesh, Aala Suresh, and Sunil Chinnadurai in their “IoT Based Smart Parking System With E-Ticketing (2022)” as shown in Figure 2.7 is the prototype of this IoT based smart parking system with E-Ticketing. The Arduino UNO, RFID Scanner, RFID Cards, Servo Motor, IR Sensors, GSM Module, and NODE MCU are the main parts of this system. In this case, the slot availability data is transmitted via Node MCU acting as a Wi-Fi module. In Figure 2.7.1 the RFID card known as the "white card" needs to be scanned using the nearby RFID scanner in order to determine its validity. The gate won't open if the card is not valid

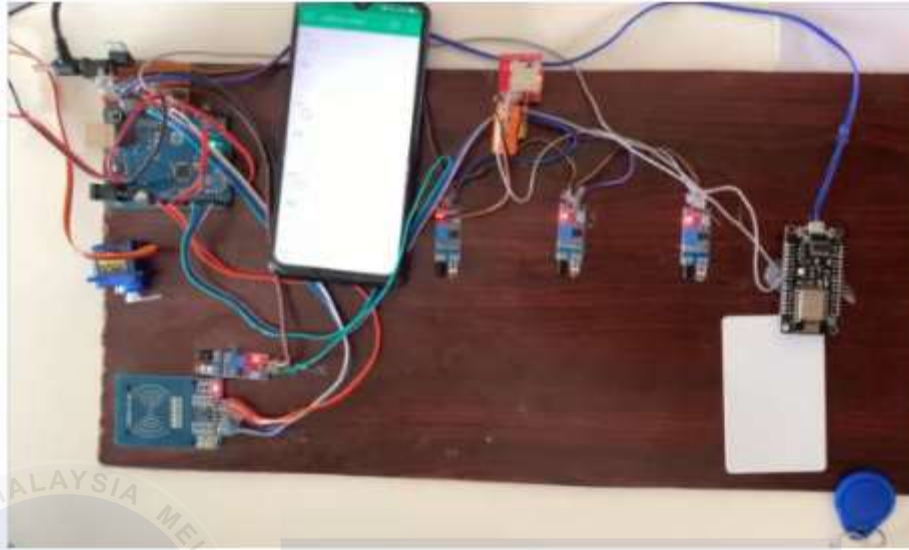


Figure 2.7: The prototype of the project



Figure 2.7.1: The RFID Scanner and RFID Card

David A. Michel-Torres, Luis F. Luque-Vega, Emmanuel Lopez-Neri, Miriam A. Carlos-Mancilla, Luis E. Gonzalez-Jiménez in their research on “IoT-based Smart Vehicle Presence Sensor SPIN-V for Smart Parking System (2019)”. As shown in Figure 2.8 the SPIN-

V Iot Sensor Prototype consist of Raspberry Pi which acts as controlling the sensor, the LED indicator, and the Pi camera, and manages the communication between the overall system and the cloud services. It also has an ultrasonic sensor HC-SR04 to detect the existence of a car in the parking lot, Pi V2 camera to identify the plate's number, an LED strip, a buzzer and a battery. The use of the SPIN-V Iot Sensor prototype is when the SPIN-V Led Indicator is off the and the user is arrives to the parking lot, so when the SPIN-V sensor detect the vehicle then it will take the plate image by the SPIN-V camera and obtain the picture car plate text internally. The text is then forwarded to the Assignment Manager for storage.



Figure 2.8: The SPIN-V Iot Sensor Prototype

Divya pandey, Seema Hanchate Author in their research on “Navigation based - Intelligent Parking Management System using Queuing theory and IOT (2018)”. In this research a smart parking plan was developed using the queuing theory approach. This approach allows the parking system to be modelled as a queue, and assessment parameters such as the wait time for users in a system or line may be predicted. Infrared sensors and wireless sensor nodes are also used with the Internet of Things idea to provide users with real-time parking lot status updates and parking lot recommendations to minimise peak-hour wait times. By utilising a multiple server single queue system and improving the current system for improved user experiences, the parking space nearest to the current location may be located. The hardware use in this project is Infrared sensor, LED, Arduino Uno, PCB and jumper wire.



Figure 2.9: The Hardware system of the research

Mehala Chandran, Nur Fadila Mahrom, Thennarasan Sabapathy, Muzammil Jusoh, Mohd Nasrun Osman, Mohd Najib Yasin, N.A.M Hambali , R.Jamaluddin, N.Ali, Yasmin Abdul Wahab, in their research on “An IoT Based Smart Parking System (2019)”. The author proposed an Android application for a parking reservation system. They suggest using an infrared sensor that is placed in the parking lot to find out whether any cars are parked. Before arriving at the place, the user can reserve the lot. As shown in Figure 2.10 is the hardware of the research’ project

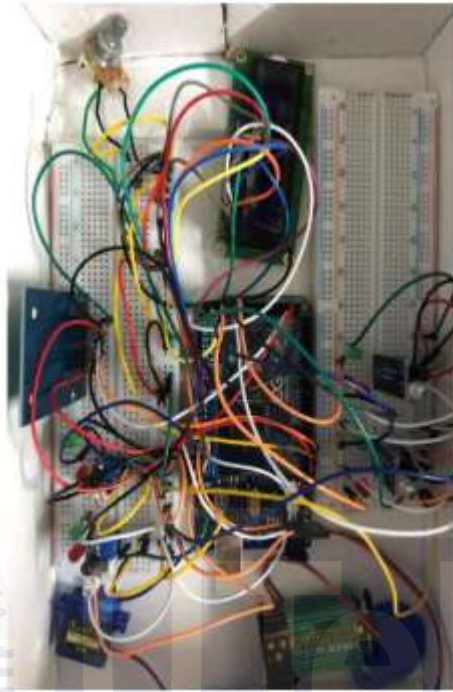


Figure 2.10: The Hardware of the research



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2.4 Critical review of current problem and justification

Table 2.2: Summary of critical review

| Research Title | Purpose | Description | Advantage |
|---|---|---|--|
| IoT-based Smart Parking System using Android Application Author: Nor Bakiah Abd Warif, Mohd Izzat Syahmi Saiful Azman, Nor-Syahidatul N Ismail, Muhammad Akmal Remli | The Android-based Car Parking Monitoring System (ACPMS) is a user-friendly smartphone application designed to make finding a specific parking spot easier. ACPMS can help a user find the closest parking lot, verify if there are any empty spots and notify the user using the mobile application | This system is a real-time application that just needs users to view the amount of parking spaces available in the shopping complex by accessing the application via their mobile device. Users will be directed via the app to the closest parking lot that is open. | Users can access the information, which is constantly updated, on their smartphone for every second. |
| Smart Car Parking System Solution for the Internet of Things in Smart Cities Author: Wael Alsaferi, | To reduce car emissions from drivers hunting for empty parking places, identify open parking spaces, prevent traffic congestion, | The proposed method gathers raw data locally, applies data fusion and filtering techniques to extract features, and minimises the amount of data | Helps users save time and maintain the safety of the user while driving. |

| | | | |
|--|---|---|---|
| Badraddin Alturki, Stephan Reiff- Marganec, Kamal Jambi | and save the user time. | transferred over the network. Subsequently, machine learning algorithms are used to process and evaluate the converted data on the cloud. | |
| Smart Parking System with PlacePod, LoRaWAN IoT sensors and Android app Author: Sofeem Nasim, Mourad Oussalah, Tarja Outila, Johannes Jutila | The system was created to provide a piece of real-time parking information on parking availability. | A series of PNI PlacePod sensors that are buried underground have been installed for this reason. These sensors allow data packets to be transmitted to a cloud server via communication with the LoRa network. An Android application has been created on the client side to communicate with users and offer various features. | Efficient parking management. can improve user experiences with valuable information about parking availability, potentially including real-time updates and navigation assistance to locate available spots. and it can reduce operational costs for parking facility owners. |
| An Efficient Car Parking | To reduce | Monitoring and handling parking | A low cost, convenient, |

| | | | |
|---|--|---|---|
| <p>Management System using raspberry-pi</p> <p>Author: G. Aravindh, M. Arun Kumar</p> | <p>the car parking problems with low-cost manner.</p> | <p>spaces for individuals with the help of android application.</p> | <p>low power consumption</p> |
| <p>Automated Car Parking System Commanded by Android Application</p> <p>Author: Mrs. D.J. Bonde, Rohit Sunil Shende, Ketan Suresh Gaikwad, Akshay Sambhaji Kedari, Amol Uday Bhokre</p> | <p>To build an effective car parking system, and autonomous system that is controlled by an Android application and to automate the cars and the entire parking lot in order to reduce the amount of human intervention in the system.</p> | <p>Automated parking is a technique that makes use of sensing devices to park and remove vehicles. An application based on Android controls entering and exiting the parking lot.</p> | <p>It can reduce the need for human intervention, save time and fuel.</p> |
| <p>ITS for Smart Parking Systems, towards the creation of smart city services using IoT and cloud approaches</p> | <p>Created a ITS smart parking systems to improve's city management and living quality standart and it also</p> | <p>To create an Internet of Things-Cloud platform with the goal of distributing pertinent data that makes it possible to</p> | <p>Reduce the search time for parking and traffic jams in the urban area of the city.</p> |

| | | | |
|---|--|--|--|
| <p>Author: Luis Felipe Herrera-Quintero, Julián Vega-Alfonso, Diego Bermúdez, Luis Andres Marentes, Klaus Banse</p> | <p>want to allows the system to be integrate into several services and can be extended to other smart services.</p> | <p>locate parking spots in Bogota, Colombia, quickly and effectively. This data is collected via ' sensors that were previously placed in various city parking lots. It then passes through a processing system that was created and implemented using a novel architecture.</p> | |
| <p>IOT Based Smart Parking System With E-Ticketing</p> <p>Author: Chinnabattuni Avinash, Gaddam Rohit, Chintakrindhi Rajesh, Aala Suresh, Sunil Chinnadurai</p> | <p>To reduce the traffic congestion when queue for paying the parking ticket and find a vacant parking area easily without any wasting time.</p> | <p>RFID cards are used in this parking system to uniquely identify each vehicle and deduct the parking fee before it enters the parking area, while an Arduino UNO serves as the processing unit. Only when there is enough money in the owner's account</p> | <p>Save user time, a Simple and quick online ticketing procedure that doesn't require human assistance, and the entire process is user friendly.</p> |

| | | | |
|--|---|--|---|
| | | will it be deducted, a notification will be sent to their phone, and the gate will open for them to park their car. | |
| <p>IoT-based Smart Vehicle Presence Sensor SPIN-V for Smart Parking System</p> <p>Author: David A. Michel-Torres, Luis F. Luque-Vega, Emmanuel Lopez-Neri, Miriam A. Carlos-Mancilla, Luis E. Gonzalez-Jiménez</p> | <p>To detect the presence or absence of a vehicle in a parking space . Can improve efficiency by providing real-time parking information, SPIN-V helps optimize parking space usage</p> | <p>The SPIN-V is made up of an LED indicator to show that a space has been reserved through a mobile application, as well as a distance sensor and camera to detect the arrival of a car into the parking slots.</p> | <p>Reducing wasted time circling and searching for parking, more convenient and frustration-free parking experience and lead to lower operational costs for parking facilities.</p> |
| <p>Navigation based - Intelligent Parking Management System using Queuing theory and IOT</p> <p>Author: Divya pandey, Seema Hanchate Author</p> | <p>For a solution to the parking issue in metropolitan areas, research and assess the current parking system and use the novel concept of queuing theory.</p> | <p>This allows users to schedule a parking space, view the current parking lot status on campus, and receive recommendations for the closest and best parking lots. Priority allotment</p> | <p>Users get access to real-time parking lot status information on campus, the ability to reserve a space, and recommendations for the closest and best parking lot.</p> |

| | | | |
|---|---|---|--|
| | | <p>is implemented during peak hours. Therefore, we model the parking system as a queue in order to obtain evaluation parameters such as the user's wait time in the system and the queue's wait time.</p> | <p>Priority-based parking lot allocation is implemented during peak hours.</p> |
| <p>An IoT Based Smart Parking System</p> <p>Author: Mehala Chandran, Nur Fadila Mahrom, Thennarasan Sabapathy, Muzammil Jusoh, Mohd Nasrun Osman, Mohd Najib Yasin, N.A.M Hambali , R.Jamaluddin, N.Ali, Yasmin Abdul Wahab</p> | <p>To provide an automated method that enables people to reserve a place with a few clicks using a specially created Android mobile application and through the system, the user can also view the amount of time that they spent parked, and costs can be calculated and communicated to them through an online application.</p> | <p>The main way that the system functions is by using sensors installed on each parking space to identify available spots and provide information. Microcontroller, which acts as a communication channel between those peripherals or devices, processes this after. At last, users may use their smartphones to check the</p> | <p>Reduces the driver's frustrations and low cost equipment needed.</p> |

| | | | |
|--|--|---|--|
| | | availability of slots in specific areas before making a reservation. | |
|--|--|---|--|



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2.5 Proposed Solution/further project

There are many types of the development and method that being used from the related work that have been studied. In summary, of the several varieties of smart parking systems, I have selected three sorts of systems to construct my own system.

There are three type of different system which is An IoT Based Smart Parking System (Mehala Chandran, Nur Fadila Mahrom, Thennarasan Sabapathy, Muzammil Jusoh, Mohd Nasrun Osman, Mohd Najib Yasin, N.A.M Hambali , R.Jamaluddin, N.Ali, Yasmin Abdul Wahab, 2019). The second one is Smart Parking System with PlacePod, LoRaWAN IoT sensors and Android app (Sufeem Nasim, Mourad Oussalah, Tarja Outila, Johannes Jutila, 2022)and the third one is IoT-based Smart Parking System using Android Application (Nor Bakiah Abd Warif, Mohd Izzat Syahmi Saiful Azman, Nor-Syahidatul N Ismail, Muhammad Akmal Remli, 2020).

From the three type of different system, all of it has a low cost and WiFi access which can helps to connect between the android application and the system. It also can provide real-time data input from the system that will be sent direct to the android application. This shows that users can access the information about the parking, on their smartphone easily. The table below contains a summary of the features that were selected from various systems.

Table 2.3: Comparison between the system

| <div> <div>Title</div> <div>Aspect</div> </div> | An IoT Based Smart Parking System | Smart Parking System with PlacePod, LoRaWAN IoT sensors and Android app | IoT-based Smart Parking System using Android Application |
|---|--|--|---|
| Cost | Reasonable | Cheap | Cheap |
| WiFi Connectivity | Yes | Yes | Yes |
| Power source type | Plugged to battery | Plugged to battery | Plugged to battery |

| Real-time data input | Yes | Yes | Yes |
|----------------------|--|--|--|
| Advantages | Allows users to reserve parking spots, reducing wasted time searching for parking and leading to a more efficient use of parking spaces. | Can improve user experiences with valuable information about parking availability, potentially including real-time updates and navigation assistance to locate available spots. and it can reduce operational costs for parking facility owners. | Users can access the information, which is constantly updated, on their smartphone for every second. |

A software system is developed in the Android Smart Parking System to reduce traffic congestion and provide real-time parking information on parking availability. Users can log in and register before start using the Android application. An ESP8266 NodeMCu is equipped with a Wi-Fi shield and infrared sensors. The shield will then instantly transmit the sensor data, including its location and availability, to a Firebase cloud. When a user connects their mobile device to the Android application, they will receive information on the available parking lot. Android application will be able to quickly present real-time parking lot availability data gathered from infrared sensors. The figure 2.11 shows the functionality of the Android Smart Parking System

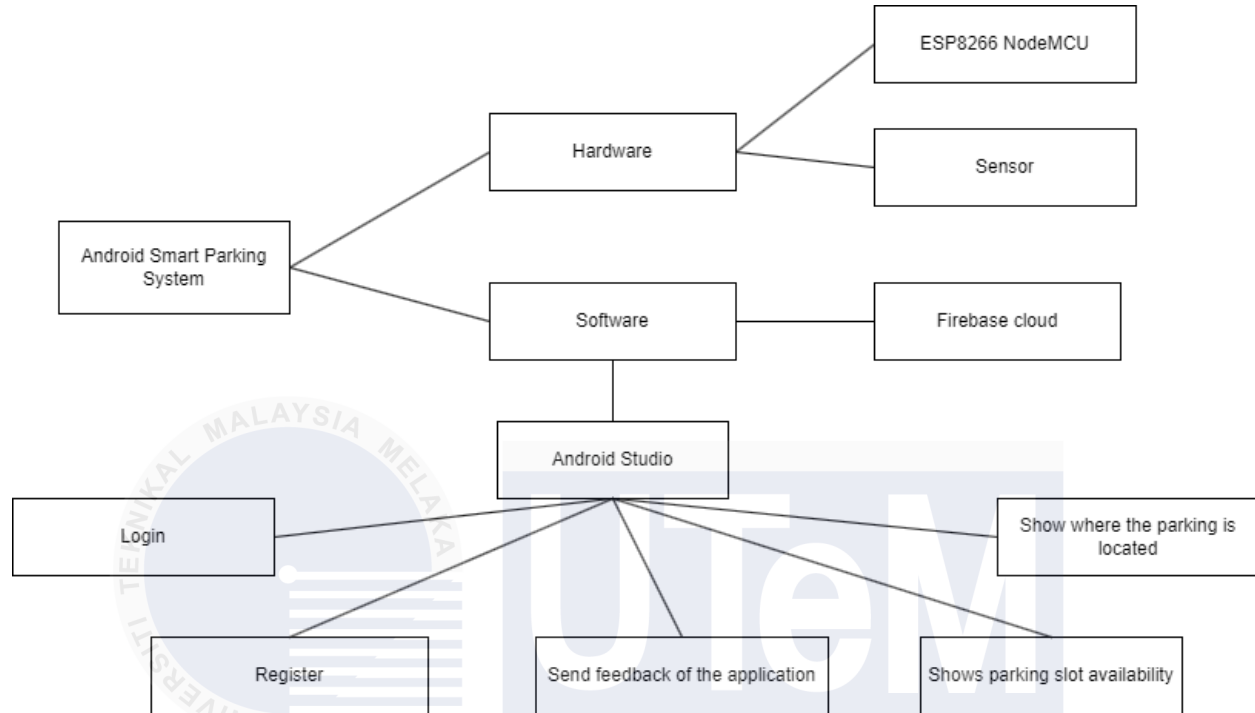


Figure 2.11 The functionality of the Android Smart Parking System

2.6 Conclusion

In conclusion, the literature study is an essential chapter that plays a critical role in developing the project concept. It aids in comprehending the system's current features and provides a clear understanding of how the system will be implemented. The study and research can facilitate a more seamless and understanding project progression. In basically, the goal of this chapter is to enhance the earlier research and enhance the problem-solving process.

CHAPTER 3: METHODOLOGY

3.1 Introduction

In this chapter, we will discuss the methodology used in this project. Methodology refers to the systematic and theoretical analysis of methods, practices, and procedures that will be used in a field of study or research. There are many types of methodology applied to different types of development such as Agile, Rapid Application Development (RAD), Waterfall, and Object Oriented Methodology. Each software development process has its own advantages and scope. The Rapid Application Development (RAD) approaches are used in this project. The Rapid Application Development (RAD) technique substitutes rapid planning for minimum preparation. It enables early project collaboration between the system developer and end users. This is because the project completed the customer's request and could be completed in the minimum amount of time.

3.2 Research process

The product's iterative development is referred to as the research process. The process comprises multiple phases, namely Data Collection, Analysis, Design, Implementation, and Testing. To complete the project, each step is crucial. The following figure illustrates the flow of the research process.

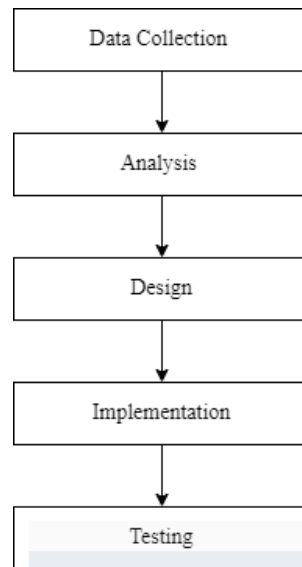


Figure 3.1: Flow of Research process

3.2.1 Data Collection

Collecting data for an Android Smart Parking System involves gathering relevant information and measurements to manage parking spaces efficiently. The first step in the data collection process for an Android Smart Parking System is to identify relevant data sources. In this project, we acquire data through multiple means to ensure comprehensive coverage and accuracy.

Firstly, occupancy sensors embedded in parking spaces detect whether a space is occupied or vacant. These sensors, which can include infrared sensors to provide real-time updates on parking availability.

Furthermore, all this data is transmitted to the user through the Android application, providing real-time information on parking availability. This approach ensures that users have a smooth and efficient parking experience.

3.2.2 Analysis

It has been shown that the current smart parking systems do not function to their maximum capacity based on the results of internet research and analysis. This is due to the fact that certain

smart parking systems do not provide consumers with any applications or interfaces. This is the cause of the situation as it stands.

Besides that, a few of the currently available smart parking systems do not support an internet connection and do not provide a smart parking system application to search an available parking spaces in the shopping mall. If the users in an urgent situation they need to spend more time in searching the available parking spaces.

3.2.3 Design

a. System Architecture Design

An Android application that is connected to the internet and designed to identify available parking spaces is the expected outcome of an Android Smart Parking system. In the parking area, the sensor that has been connected to the NodeMCU ESP8266 will display the availability of the parking spaces. Then, after being transmitted, the data will be kept safe in the Firebase cloud. The primary interface of the application will display available parking spaces, and profile of the users. The details of project design is discussed in details in chapter 4.

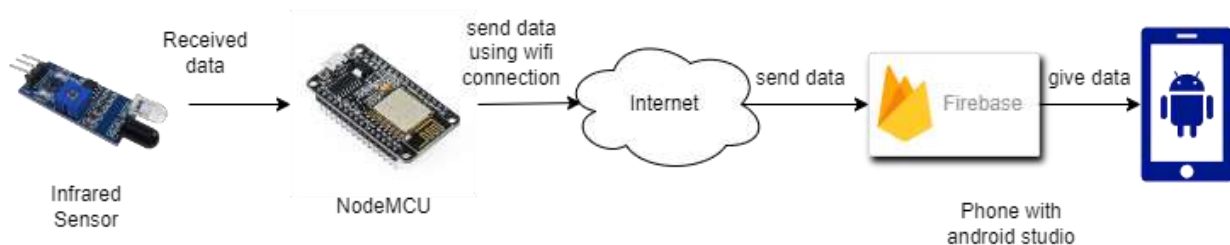


Figure 3.2: System Architecture

3.2.4 Implementation

The process of outlining how to actually build the system based on the plans and version is known as project execution. There are two steps in the implementation process; these are called

the hardware connection and the android studio application, respectively. In chapter 5, every last detail of the planned execution is discussed in detail.

a. Hardware connection

The hardware connections used in this project are shown in the above diagram. Using a male to male jumper wire, the IR sensors are connected to the NodeMCU microcontroller to enable data retrieval. It is necessary to programme the Android Studio application to ensure that the NodeMCU connection is established and operating properly. To process the data, Google Firebase must get all of the data received from the NodeMCU.

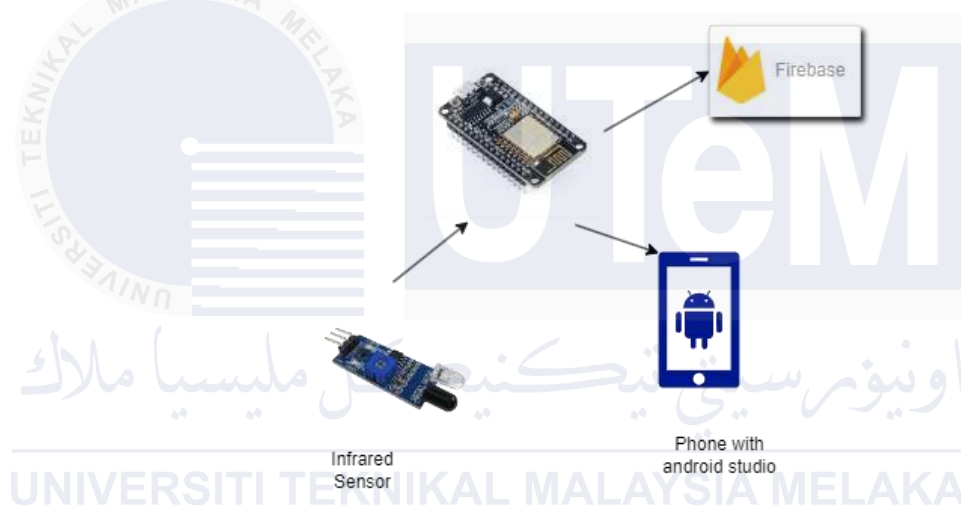


Figure 3.3: Hardware Connection

b. Android Studio application

Android studio is an application integrated development environment monitor the Android Smart Parking System. The figure below shows the functionality of the Android Smart Parking System in details.

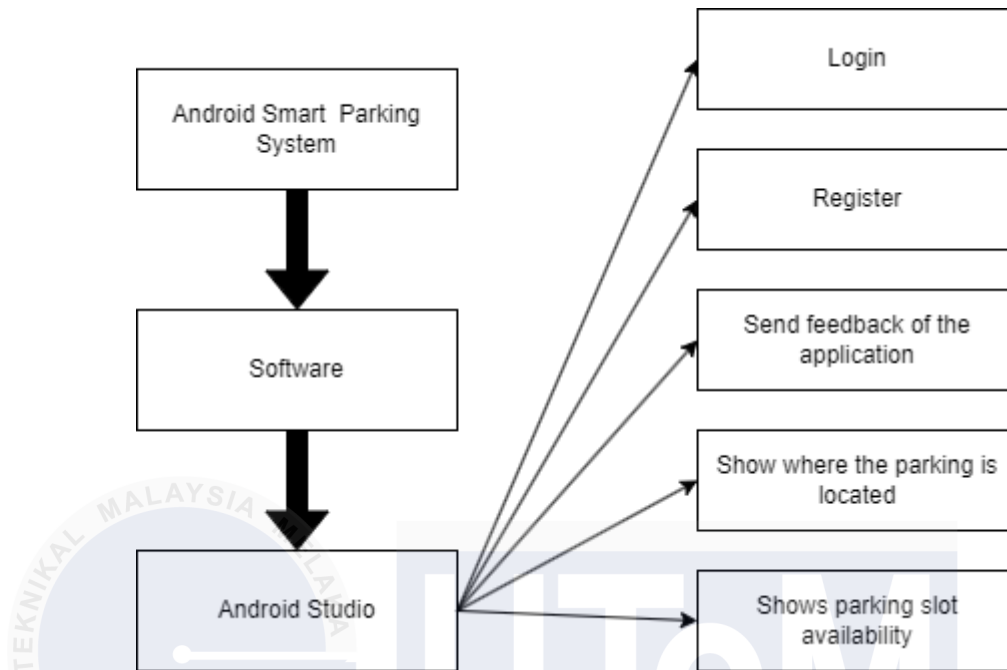


Figure 3.4: The functionality of Android Smart Parking System

Log in – Users who have already registered can log in to Android Application.

Register – This process is to create a new account for the new user. The information of the new user is saved.

Send feedback of the application – Allow users to send feedback on their parking experiences. Users can submit comments, ratings or suggestion.

Show where the parking is located – Users can see where the available parking spaces through the application.

Shows parking spaces available – User can see the real-time updates of the available of parking spaces.

3.2.5 Testing

Testing is the process of running a system application and ensuring that everything is functioning as it should. Hardware testing and software application testing are the two phases that

have been prioritised in the testing process for the Android Smart Parking system. Every step of the testing process must adhere to the proper protocol. A detailed explanation is discussed in chapter 6.

Hardware testing - Hardware testing includes sensors and the ESP8266 NodeMCU.

Software application testing – Every developed feature of the system is checked to make sure it functions correctly.

3.3 Methodology

This project uses the Rapid Application Development (RAD) methodology. This methodology is the best choice since, in contrast with the standard software development lifecycle, it is designed to generate a high-quality result in the smallest amount of time. Furthermore, the Rapid Application Development (RAD) technique can produce a product in a short amount of time and gather end-user input on their needs. RAD is a four-phase software development cycle that incorporates the components of the classic SDLC, or Standard System Development Life Cycle. The four stages include of requirement planning, construction, testing, and cutover as well as user design. The RAD process is shown in the following figure.

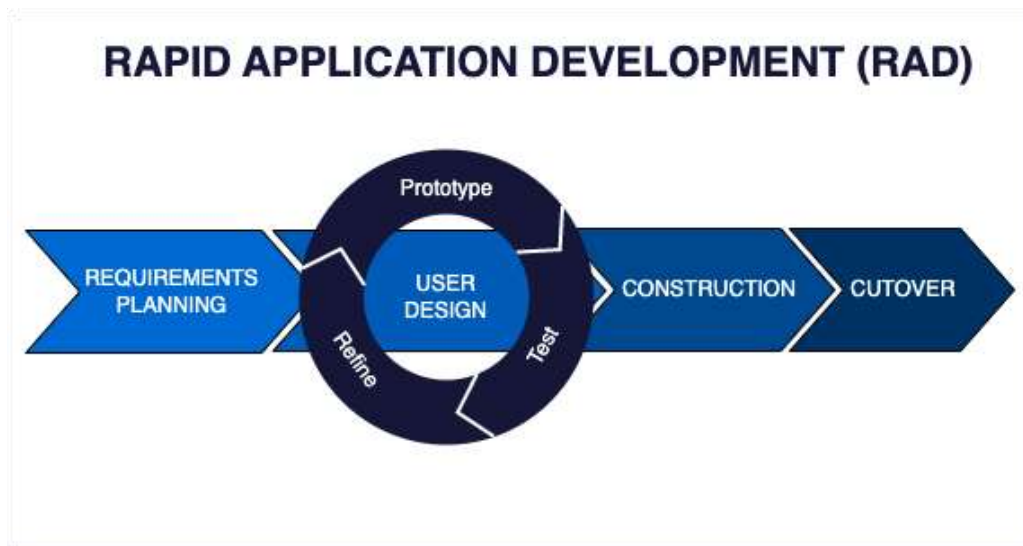
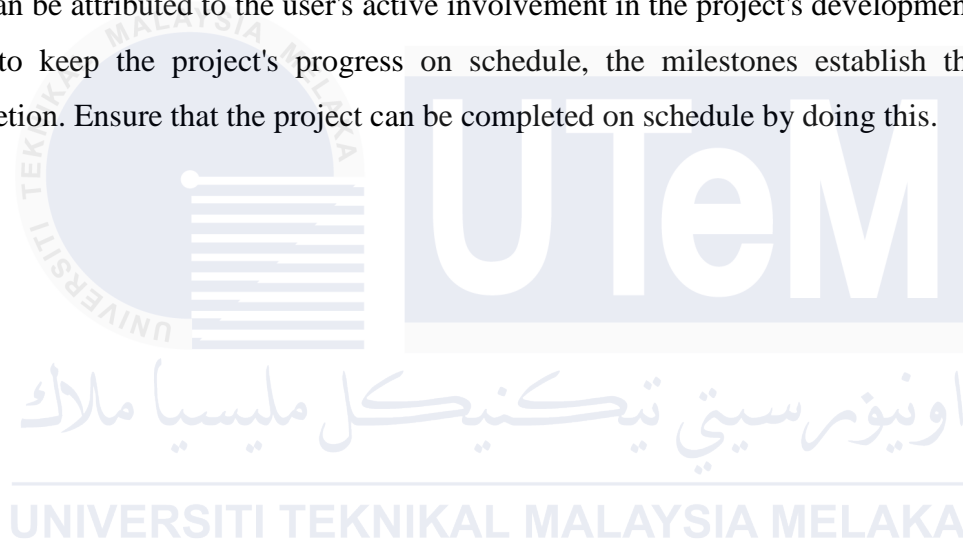


Figure 3.5: The process of Rapid Application Development

3.4 Conclusion

In summary, the project development technique is defined in this chapter, which is highly important because it may control the project's flow structure. This chapter also includes a discussion of the technique. Because the prototype model utilised for this project is simple, the developer will find it easier to comprehend and implement it. The developer can improve system development and boost user satisfaction by following the processes in the prototype approach. This can be attributed to the user's active involvement in the project's development. Moreover, in order to keep the project's progress on schedule, the milestones establish the deadline for completion. Ensure that the project can be completed on schedule by doing this.



CHAPTER 4: ANALYSIS AND DESIGN

4.1 Introduction

In this section, analysis and design are crucial for thoroughly understanding the project's goals, requirements, and proposed solutions. It's important to present clear, concise, and well-structured information to ensure a complete understanding of the project's objectives and solutions. The hardware and software necessary for this project are outlined in the requirements. Additionally, a detailed block diagram architecture and thorough analysis are essential to ensure the project's successful completion and accurate design. The design of the interface are also provided to have a clear view to be implement in the next chapter.

4.2 Problem Analysis

Current Smart Parking systems available on the market are not comprehensive, primarily because they lack applications designed to provide information directly to users. To address this gap, our project incorporates an Android Studio application as a monitoring tool. This application is designed to track environmental changes within the parking area and deliver reliable, real-time information to users, thereby enhancing the overall functionality and user experience of the Smart Parking system

4.3 Requirement Analysis

The requirements for both input and output for this project are outlined in this section. It also displays every functional need required to finish building this project.

4.3.1 Data Requirement

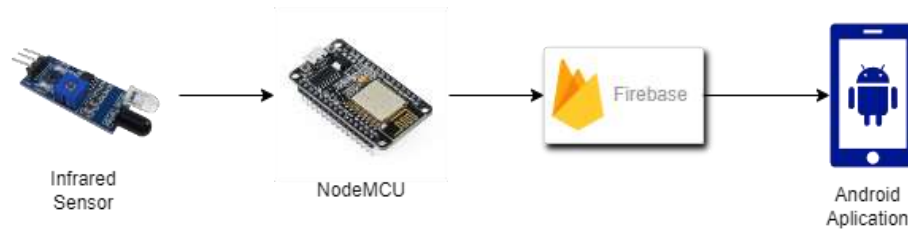


Figure 4.1: Data Flow of the project

The figure 4.1 above shows the data flow of the system. The anticipated outcome of the Smart Parking system is an internet-connected Android application designed to monitor parking availability and environmental conditions within parking areas. In regions with high parking demand, sensors connected to NodeMCU will detect available parking spaces. This data will be sent and stored in the Firebase cloud. The application's primary interface will display real-time parking availability and historical data from the previous few days, providing users with up-to-date information to assist in their parking decisions.

4.3.2 Functional Requirement

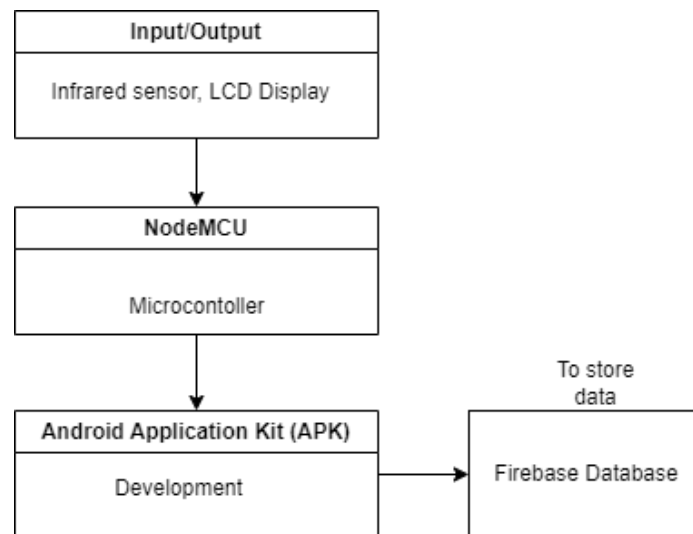


Figure 4.2: Block Diagram of Functional Requirement

This project is divided into several blocks namely microcontroller block, input/output block and development block. The figure 4.2 shows the details of block diagram for this project.

1) The Android Application Kit transmits instructions to the NodeMCU microcontroller, which then relays these commands to the infrared sensors to detect parking space availability. The control code for managing the sensors via the microcontroller is fully developed using an Integrated Development Environment (IDE), ensuring precise and efficient communication between the components.

2) Input/Output sensors will detect the availability of parking spaces using infrared technology. These sensors must be directly linked to the NodeMCU microcontroller.

3) Android Application Kit (APK): The primary operation is situated in the development block, where the Android Application Kit manages the NodeMCU microcontroller. This operation involves monitoring the NodeMCU to observe parking space availability using infrared sensors. The APK can exchange commands with the NodeMCU microcontroller, enabling two-way communication.

4.4 Hardware Requirement

1) NodeMCU

Built around the ESP8266EX microcontroller, the NodeMCU ESP8266 is a flexible development board with 10 GPIO pins, 4MB of flash memory, 80KB of RAM, and a 32-bit Tensilica L106 RISC core operating at 80 MHz. This tiny board can be programmed using the Arduino IDE, Lua, or MicroPython and runs between 2.5 and 3.6 volts. It also features integrated Wi-Fi capability for 802.11 b/g/n standards and may be used as an access point or client for wireless networks. It is well-known for its cost and robust development community. It also has a micro-USB interface for programming and power supply, and it supports a wide range of sensors and actuators for Internet of Things applications.



Figure 4.3: NodeMCU ESP8266

2) Infrared sensor

An infrared sensor is an electronic device used to measure and detect infrared radiation in its surrounding environment. There are two types of infrared sensors which are active infrared sensors and passive infrared sensors. The active infrared sensor produces and detects infrared radiation. They have two parts which is a light-emitting diode (LED) and a receiver. The infrared light from the LED bounces off of objects when they get close to the sensor, and the receiver picks this up. Because they function as proximity sensors, active infrared sensors are frequently employed in obstacle detection systems, including robotic ones.



Figure 4.4: Infrared sensors

3) Jumper wire

The use of jumper wires are to establish connections within devices such as sensors, NodeMCU, and Raspberry Pie is a straightforward method. Jumper wires come in a range of colours, lengths, and varieties. Male to male, male to female, and female to female are the three different types of connector wire. While they are all essentially the same, each type of jumper wire is ideally suited for a specific device. A male-to-male connector wire example is shown in Figure 4.5.

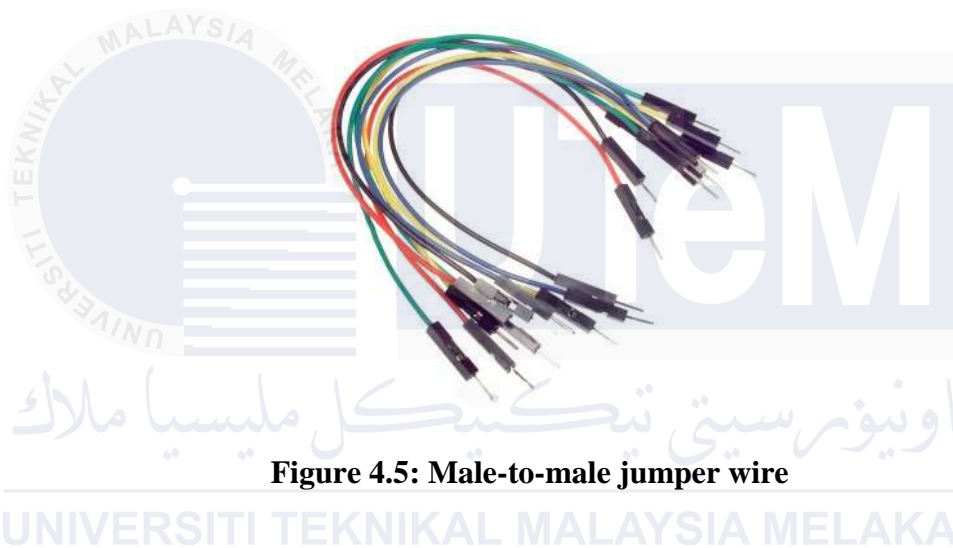


Figure 4.5: Male-to-male jumper wire

4) Breadboard

A breadboard is a board used to connect electronic components, such as wires, resistors, capacitors, and coils, to conduct various experiments and projects.

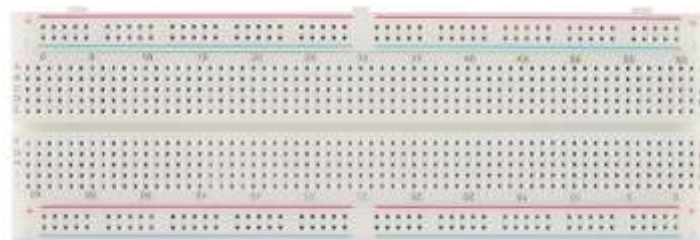


Figure 4.6: Breadboard

5) LCD Display

An LCD screen is a type of electronic display module that creates a visible image by using liquid crystal. One of the most fundamental modules used in DIY projects and circuits is the 16x2 LCD display. A display with 16 characters per line is translated into two lines by the 162. Every character on this LCD is shown as a 5 by 7 pixel matrix.



Figure 4.8: 16x2 LCD

4.5 Software Requirement

1) Arduino IDE

The NodeMCU programming application is called Arduino IDE (Integrated Development Environment). The software provides an easy-to-use interface for writing, assembling, and uploading code to Arduino microcontrollers. For the source code to work correctly, the model and port number must be selected when uploading. Windows, Linux, and Mac OS are supported for this Arduino IDE. The figure 4.9 shows the interface of IDE.

2) Android Package Kit (APK)

Figure 4.10: Android Studio Interface

3) Firebase

Firebase is a cloud-based platform that offers open-source capabilities, allowing developers to build and deploy web applications. React Native integration makes it easier to create applications for any web browser and works with both iOS and Android platforms. Firebase makes it easier to download and install apps from a phone or browser because it was created especially for mobile devices. It's also free for anyone interested in developing their own application to use Firebase. Its primary goal consists of updating data as soon as an external table is received.

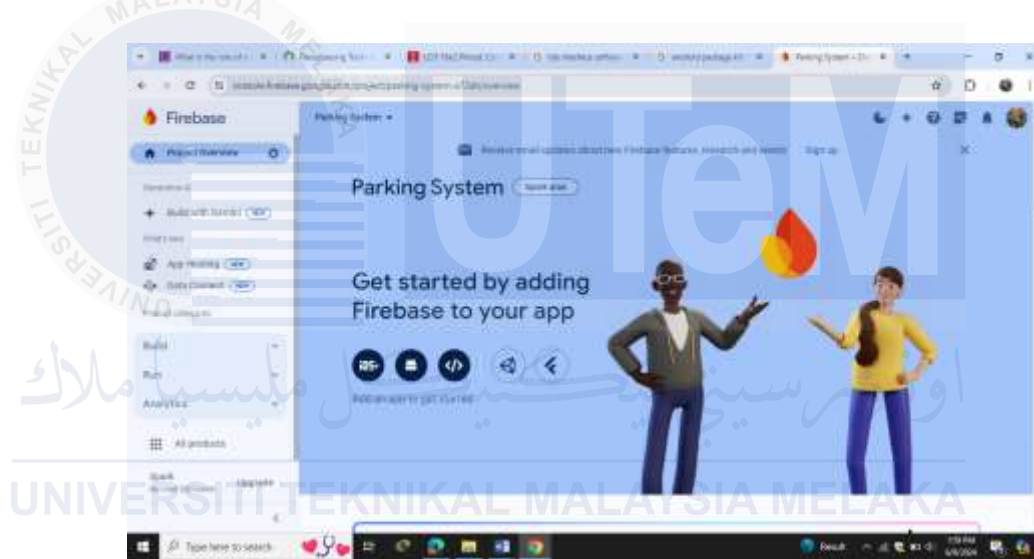


Figure 4.11: Firebase database Interface

4.6 High-Level Design

The conceptual or architectural design of the system or solution that is being developed is referred to as high-level design (HLD) in a project. It outlines the general architecture, constituent parts, interfaces, and principal features of the project. It is essential to ensuring that the project is successfully implemented in a well-organized manner.

4.6.1 System Architecture

The software architecture of the system is an Android-based APK system. The user can quickly and effectively build the system with the help of this software. The user made use of an effective and user-friendly method. System architectures are necessary for every product's development. It is the process of creating, developing, and putting into place a system that meets all needs in order to achieve the goal. Figure below show the system architectural of Smart Parking System.

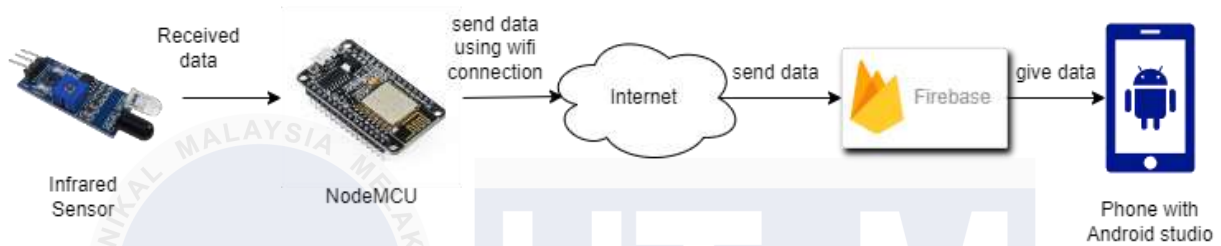


Figure 4.12 : System Architecture

4.6.2 Interface Design

A proposed user interface would be designed for the Android Smart Parking System. The below figures show an example of the proposed design.

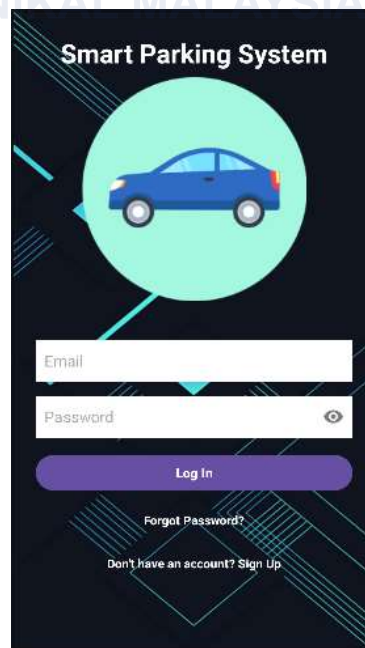
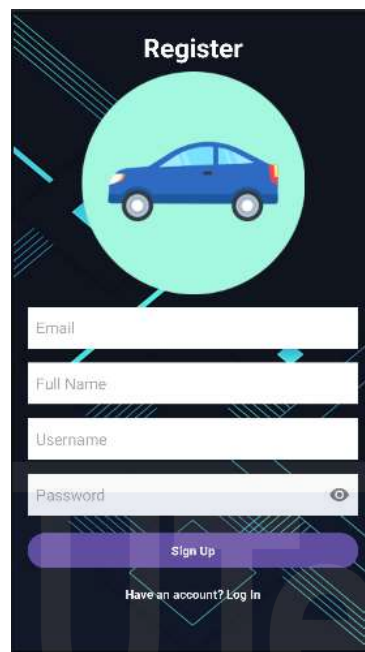
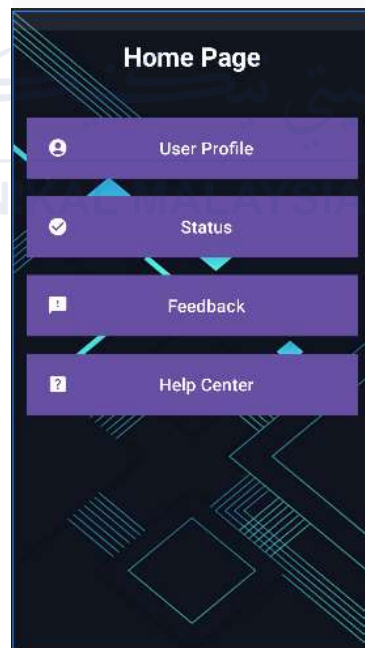


Figure 4.13: Login page Interface



The 'Register' interface features a dark blue background with a teal car icon inside a circle at the top. Below the icon are four white input fields labeled 'Email', 'Full Name', 'Username', and 'Password'. The 'Password' field includes an eye icon for toggling visibility. A teal 'Sign Up' button is positioned below the fields, and a link 'Have an account? Log In' is at the bottom.

Figure 4.14: Register user Interface



The 'Home Page' interface has a dark blue background with a teal car icon at the top. Below the icon are four teal buttons with white icons and text: 'User Profile' (person icon), 'Status' (checkmark icon), 'Feedback' (speech bubble icon), and 'Help Center' (question mark icon).

Figure 4.15: Home page Interface

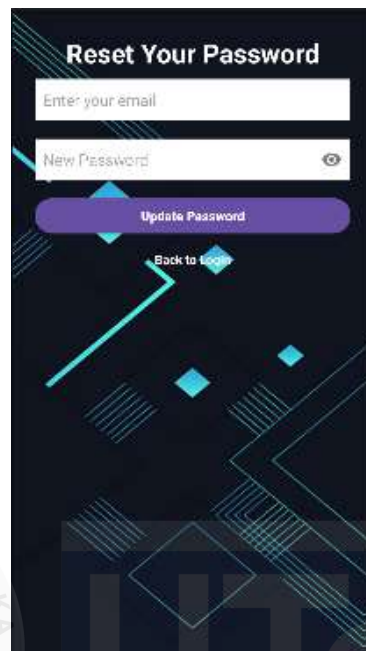


Figure 4.16: Reset Password Interface

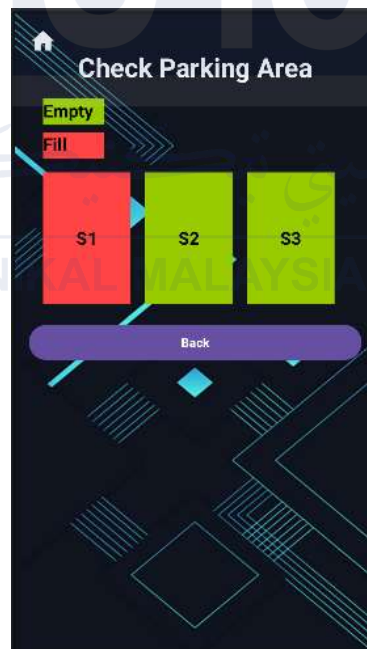


Figure 4.17: Status parking Interface

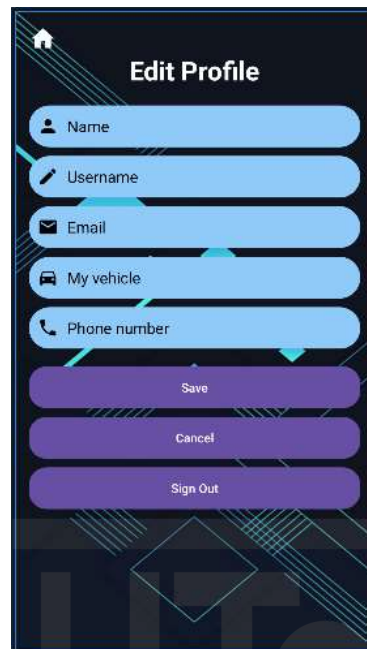


Figure 4.18: Edit Profile Interface

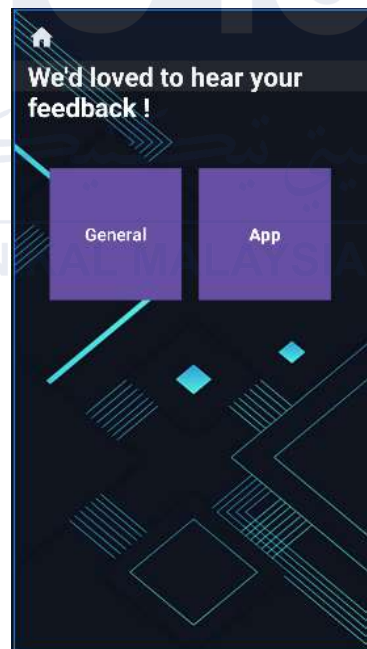


Figure 4.19: Feedback Interface

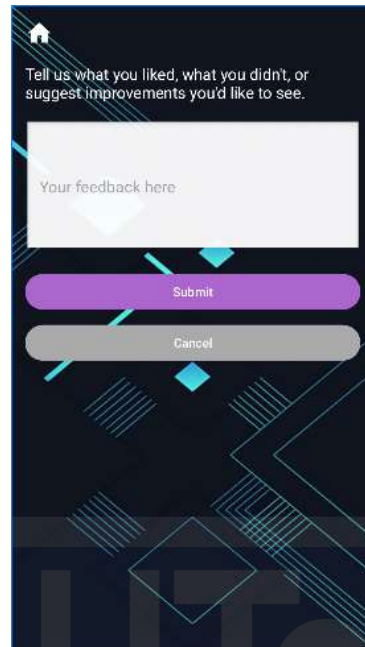


Figure 4.20: Submit Feedback for General Interface

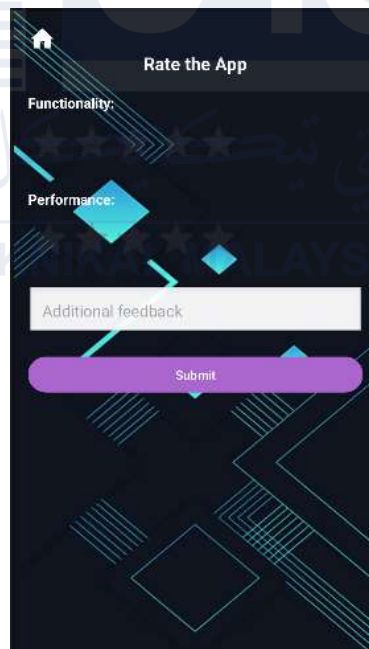


Figure 4.21: Submit Feedback for app Interface

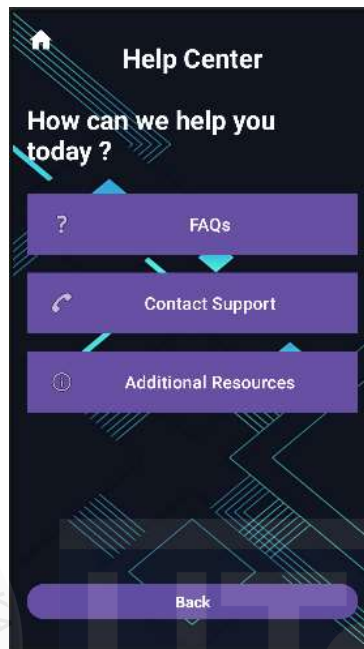


Figure 4.22: Help Center Interface

4.7 Conclusion

One of the most crucial aspects of project implementation is analysis and design. Before beginning a project, all hardware and software needs must be determined and studied. This chapter serves as the implementation's pre-preparation phase. It also incorporates the system's general flow for a better understanding before implementation. The project's implementation strategy and the expected outcomes are covered in the upcoming chapter, Implementation. Thus, the phase of analysis and design facilitates a seamless implementation process.

CHAPTER 5: IMPLEMENTATION

5.1 Introduction

To integrate the Smart Parking System with Android smartphone applications, this chapter primarily focuses on using software and hardware development. The explanation of establishing management settings in detail is provided to guarantee a smooth and effective project operation. In order to create the Smart Parking System and give it the ability to detect parking spaces that are available and monitor occupancy, the NodeMCU microcontroller was used. The project's conclusion shows that the prototype was carried out successfully. The relevant tasks here include preparing the development environment, managing software configurations, defining version control protocols, and reporting on implementation progress.

5.2 Development Environment Setup

There are hardware and software requirements for setting up the development environment for the Android Smart Parking System. The steps involved in these setup procedures are clearly described in a systematic and understandable way. Chapter 4 provides an introductory overview of the required hardware and software, and the following section goes into further detail explaining how they work together.

5.2.1 Hardware Development Setup

The hardware setup is shown in Figure 5.1 below. Three infrared sensors are connected to the NodeMCU. After being transmitted, the data will next be kept safe on the Firebase cloud.

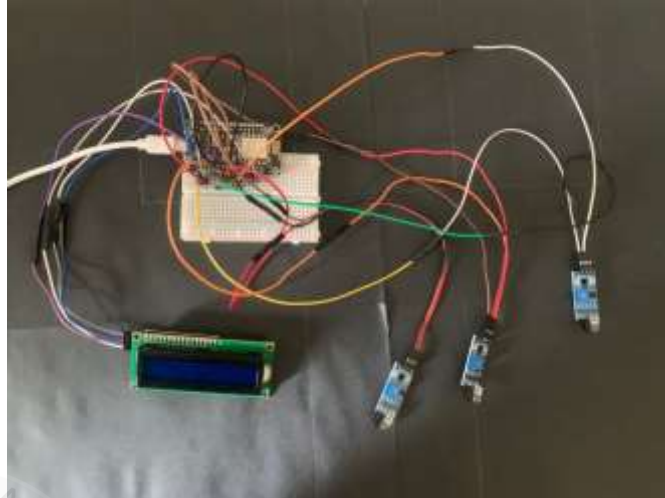


Figure 5.2: Hardware Setup

5.2.1.1 Hardware Installation

Table 5.1 shows the details of the pin number used for ESP8266 NodeMCU and the connection is established using the jumper wire. The pin number of the ESP8266 NodeMCU can be referred to the figure 5.3. Lastly, the final products are shown in Figure 5.4.

Table 5.1: Details of the pin number between the hardware

| Hardware | Pins |
|-------------------|---|
| I2C LCD Display | GND: G VCC: 3V SDA: D2 SCL: D1 |
| Infrared sensor 1 | VCC: 3V GND: G OUT: D6 |
| Infrared sensor 2 | VCC: 3V GND: G OUT: D7 |
| Infrared sensor 3 | VCC: 3V GND: G OUT: D4 |

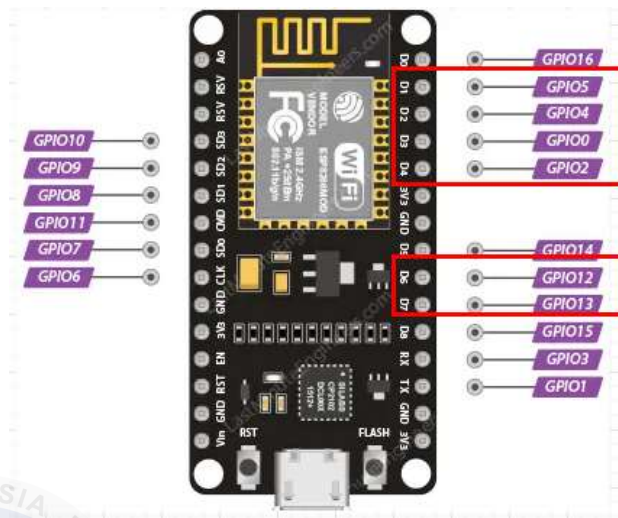


Figure 5.3: The GPIO setup hardware circuit



Figure 5.4: Final Product

5.3 Software Development Environment Setup

The requirements for the development prototype in this project are described in this subtopic. This specification is necessary to build the project's hardware and software. Additionally, these elements enable the user to take a more active role in the project's production. The environment of this project includes the following components:

i. ESP8266 NodeMCU Environment Setup

The main board needed for this project to work as designed is the NodeMCU ESP8266. A smart parking system prototype's NodeMCU environment must be configured with the correct software and hardware in order to produce a functional prototype that can detect a car in the parking lot. The following list outlines the hardware and software needed to set up a NodeMCU environment:

a. Hardware Requirements

- NodeMCU ESP8266 microcontroller
- Infrared sensor
- Jumper wire
- Breadboard
- I2C LCD display
- Power Supply

b. Software Requirements

- Arduino IDE
- Firebase cloud
- Android Studio

Based on the hardware and software requirements listed above, all of them are part of the development environment for implementing the Smart Parking System. NodeMCU uses a Wi-Fi module to connect to the user's mobile phone. The table below summarises the system configuration for the NodeMCU environment that was put up in this project.

Table 5.2: NodeMCU ESP8266 Environment Setup

| No | System Configuration | Specification |
|----|----------------------|---|
| 1 | Operating System | No real OS |
| 2 | Hardware | <ul style="list-style-type: none"> • Operating Voltage: 3.3V |

| | | |
|---|----------|--|
| | | <ul style="list-style-type: none"> • Input Voltage: 7- 12V • Flash Memory: 4 MB • 2.4 GHz Wi-Fi, supporting WPA/WPA2 • Supports standard TCP/UDP Server and Client |
| 3 | Software | Arduino Software (IDE) |

ii. Firebase Environment Setup

Firebase, a Google-developed platform, plays an important role in the Smart Parking System project. This solution efficiently stores and synchronizes data in real-time by utilizing Firebase's real-time database features. This project also goes into great detail about all of Firebase's cloud features. Most importantly, Firebase makes it possible to save modifications to sensor data in a cloud-based SQL database with ease. The most important thing is that Firebase keeps all of this data current and up to date by keeping it all in a state of continuous currency. As a result, the system's end users get access to the most recent and precise data accessible.

iii. Android Smart Parking Application Environment Setup

The program should run well to allow users to readily use smart parking applications. The user can view the available spot in the parking lot after registering or logging into the applications. This application must be well-crafted so that users can reduce their time to find an available parking spot.

5.4 Software Configuration Management

An overview of the configuration management system's design and setup for this project may be found in this subtopic. In addition, the technological tools that we use to help manage our

configuration are covered in the configuration. Any project's advancement is aided by its implementation. Code is written and assembled using the Arduino IDE, and applications for smartphones are made with Android Studio.

5.4.1 Configuration Environment Setup

Setting up a smart parking system is done step-by-step. The setup and configuration are further explained in the flowcharts below, which begin with how to make the prototype function and work and end with the project being completely operational.

i. Development and Configuration (ESP8266 and Arduino IDE)

Table 5.3: Show the Development and Configuration of the ESP 8266

| Component Name | Software Implemented |
|----------------|----------------------|
| ESP 8266 | Arduino IDE |
| | Infrared Sensor 1 |
| | Port 3 COM |
| | Infrared Sensor 2 |
| | Port 3 COM |
| | Infrared sensor 3 |
| | Port 3 COM |
| | Firebase cloud |

Table 5.3 shows the software programs that are used with the ESP8266 NodeMCU. Using port 3 COM and the Arduino IDE are necessary for the development of the code-operative prototype. The prototype would be unable to upload the code because port 3 COM is missing, making it inoperable. To connect to the ESP8266 NodeMCU, this project also includes elements like an Infrared sensor, Firebase Cloud, and Android Studio. To successfully monitor and operate the components, these connections are essential.

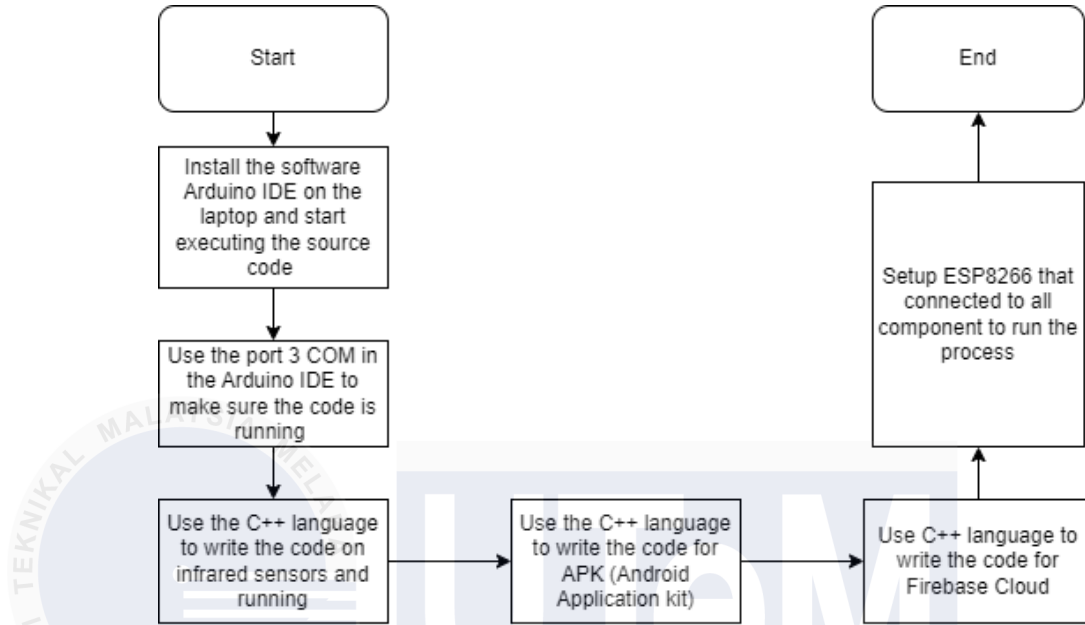


Figure 5.5: Flowchart of the development and configuration of ESP8266

Here are the steps for configuring the Arduino IDE environment, including libraries for the NodeMCU. These step-by-step directions are intended to help users do their jobs more efficiently and easily.

Step 1: Go to the Arduino website to download and launch the Arduino IDE Installer.

Downloads



Arduino IDE 2.3.2

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits

Windows MSI installer

Windows ZIP file

Linux AppImage 64 bits (X86-64)

Linux ZIP file 64 bits (X86-64)

macOS Intel, 10.15: "Catalina" or newer, 64 bits

macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

Figure 6.5: Website for Arduino IDE installer

Step 2: After the software has been installed, this is the interface that you might see to modify the code for microcontroller.



Figure 5.7: Arduino IDE coding interface

Step 3: Install ESP8266 board libraries in order to connect Arduino IDE with the microcontroller. Click File>Preferences. Then, enter http://arduino.esp8266.com/stable/package_esp8266com_index.json into the “Additional Boards Manager URLs”. Then, click the “OK” button.

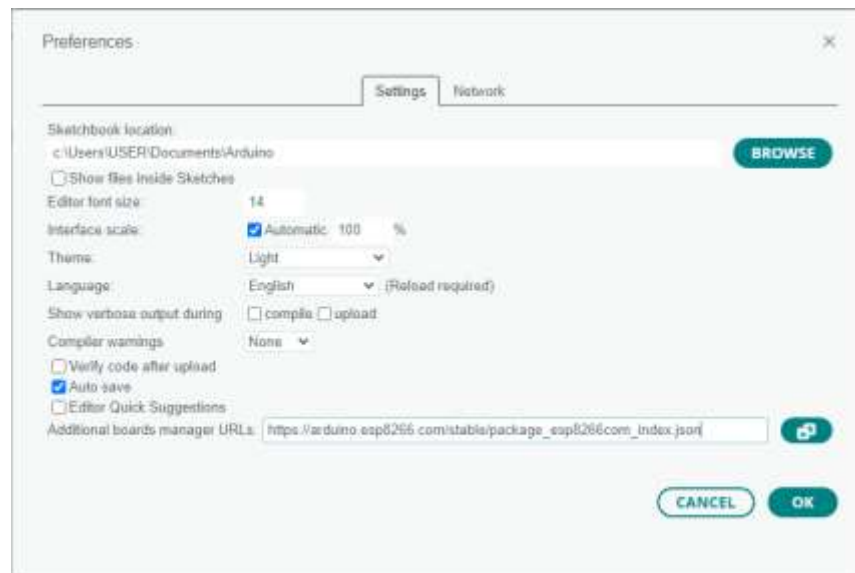


Figure 5.8: Open preferences to enter library ESP8266.

Step 4: Go to Tools > Board > Boards Manager to install the ESP8266 by ESP8266 Community.

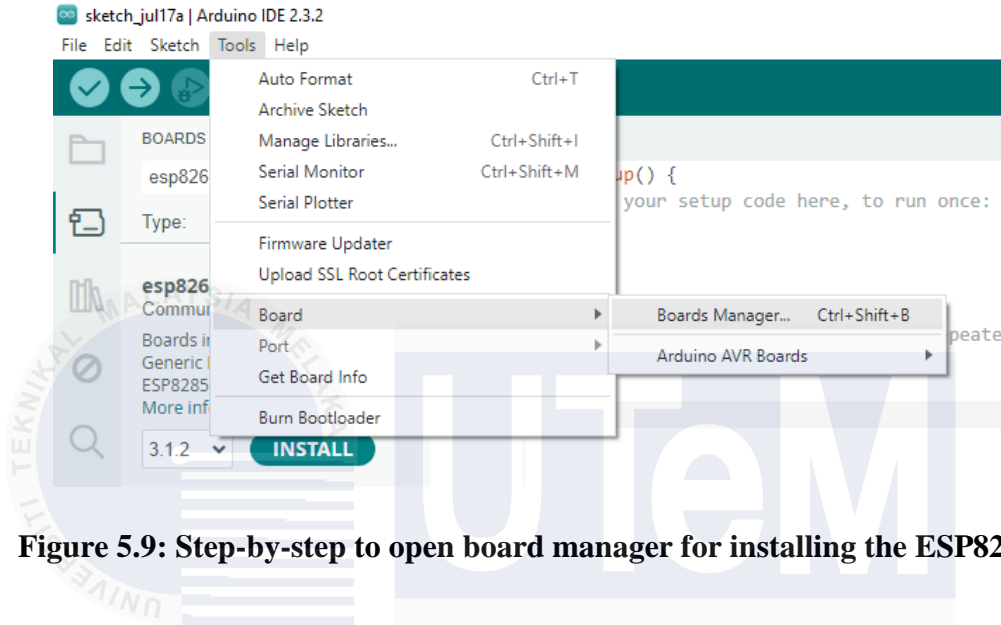


Figure 5.9: Step-by-step to open board manager for installing the ESP8266

Step 5: Search for ESP8266 and click the “ESP8266 by ESP8266 Community” the click install icon. That is everything there is to it. After a few seconds, it should be mounted.

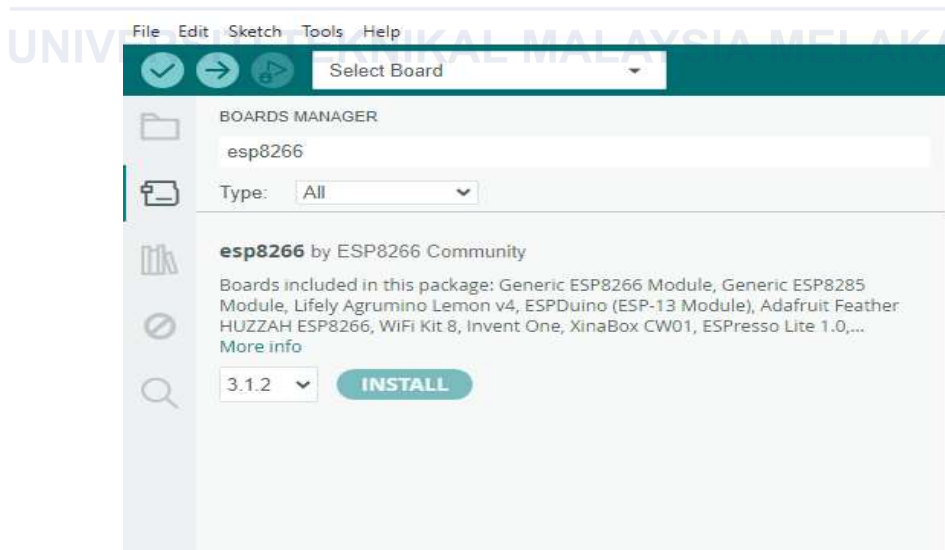


Figure 5.10: ESP8266 Board Package

Step 6 : Then, this is the output after the installation is done.

```

Output
esp8266:mkspiffs@3.1.0-gcc10.3-e5f9fec installed
Installing esp8266:mklittlefs@3.1.0-gcc10.3-e5f9fec
Configuring tool.
esp8266:mklittlefs@3.1.0-gcc10.3-e5f9fec installed
Installing esp8266:python3@3.7.2-post1
Configuring tool.
esp8266:python3@3.7.2-post1 installed
Installing platform esp8266:esp8266@3.1.2
Configuring platform.
Platform esp8266:esp8266@3.1.2 installed

```

Figure 5.11: Output of the installation

ii. Development and Configuration (Infrared Sensor)

Table 5.4: Software Implemented into Component

| Component Name | Software Implemented |
|----------------|----------------------|
| NodeMCU | Arduino IDE |
| | Port 3 COM |
| | Infrared sensor 1 |
| | Port 3 COM |
| | Infrared sensor 2 |
| | Port 3 COM |
| | Infrared sensor 3 |
| | Port 3 COM |

iii. Development and Configuration (Firebase)

The real-time database provided by Firebase is used by this Smart Parking system project to store and synchronise data instantaneously. Firebase guarantees that all data is up to date and that users are always getting the most recent information. The Firebase configuration is shown in the diagram below, which includes timestamped updates for all of the data.

Step 1: Create the project name in Firebase.

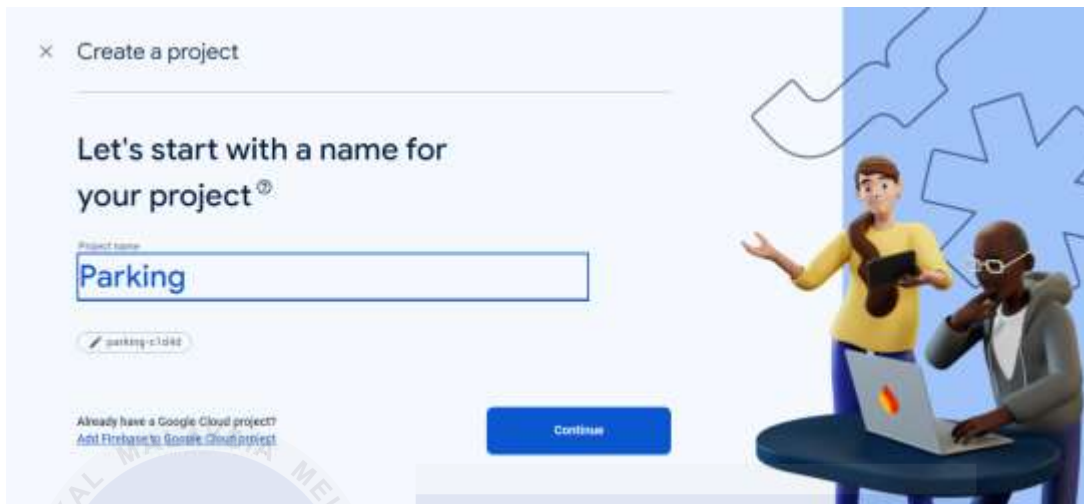


Figure 5.12: Create a project name in Firebase

Step 2: Connect app to Firebase which must add firebase to the Android app. Enter the app package name.

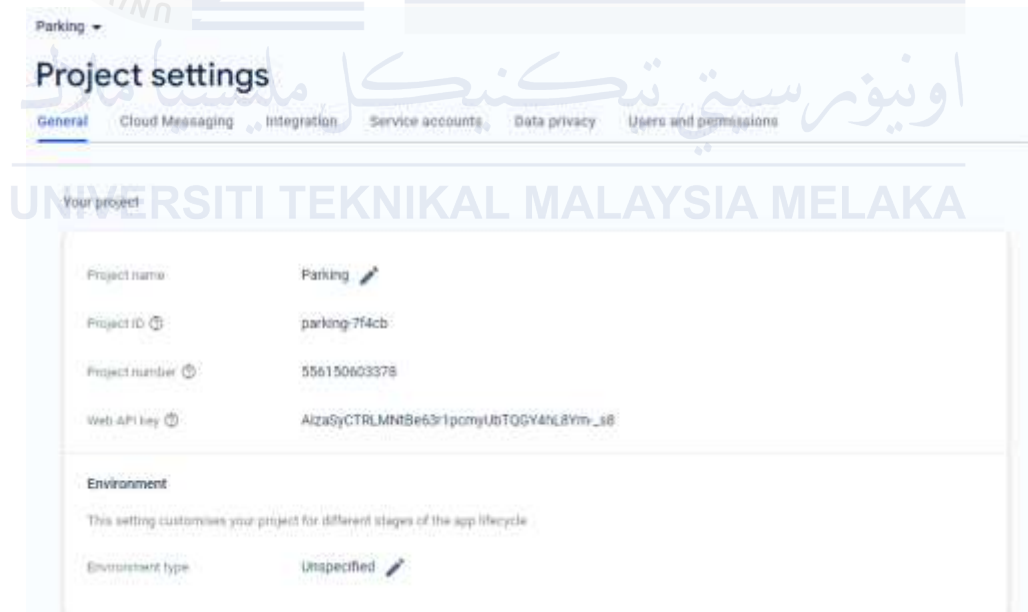


Figure 5.13: Add the Firebase to the Android application

Step 3: Choose a real-time database and create the database structure

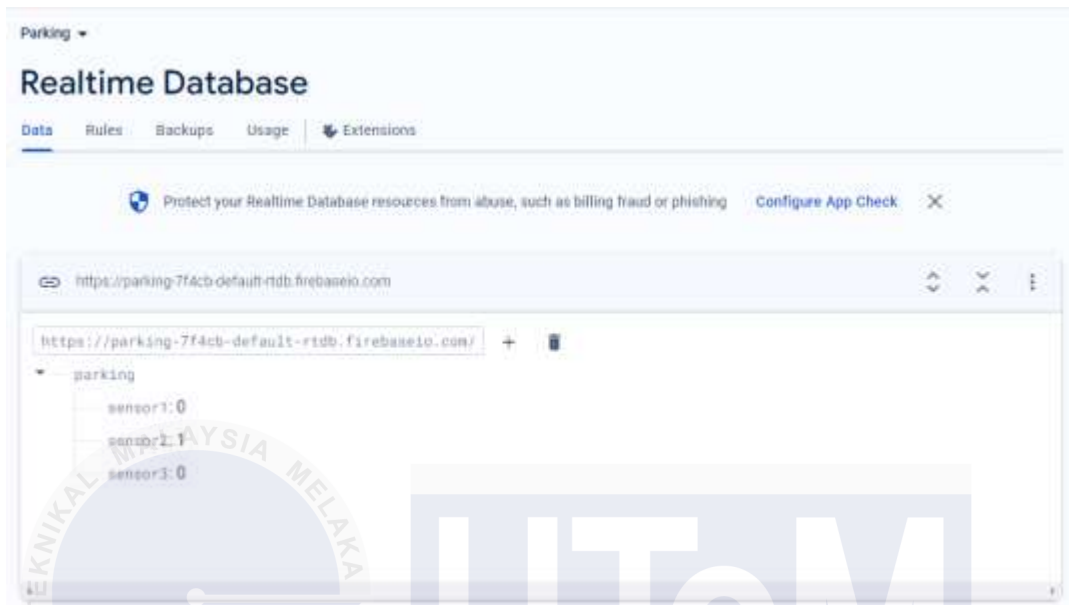


Figure 5.14: Create a Realtime Database

iv. Development and Configuration (Android Studio)

The table 5.5 shows the development and configuration of the Android Studio. The application helps to monitor the prototype by updating the available parking space.

Table 5.5: Software Implemented into Component

| Component Name | Software Implemented |
|-----------------|----------------------|
| Infrared sensor | Arduino IDE |
| | Port 3 COM |
| | Java language |

Step 1: Download Android Studio Installer.

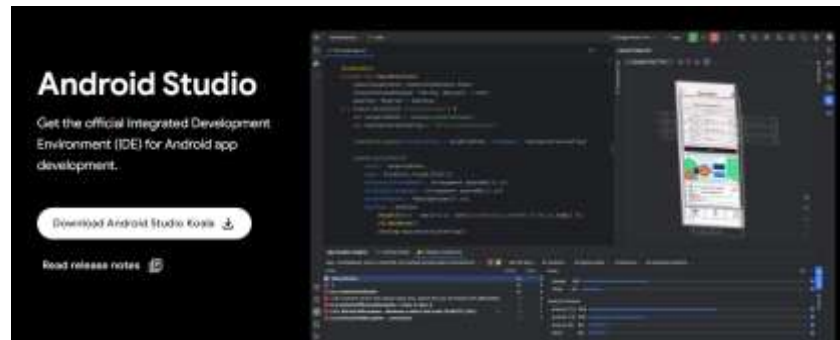


Figure 5.15: Android Studio Installer

Step 2: Create a new project to start implementing the code.

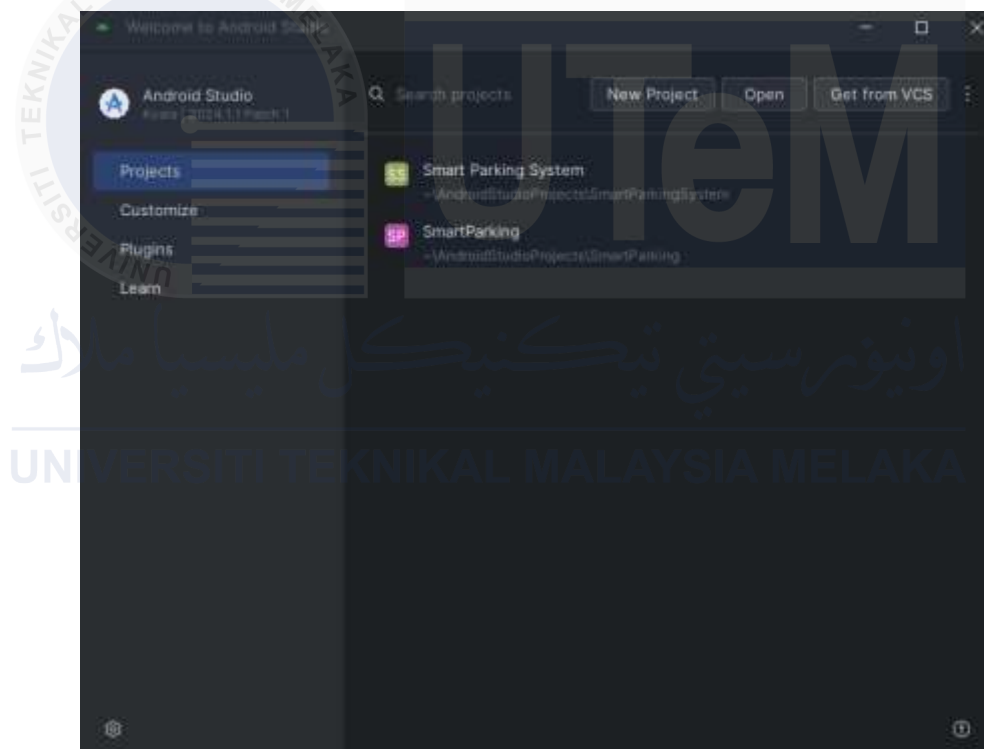


Figure 5.16: Create New Project

Step 3: After create new project, add package firebase in Android Studio to connect between both of them. Click Project->Android->Gradle Scripts->build.gradle(Module:app)

```

41 dependencies {
42
43     implementation(libs.appcompat)
44     implementation(libs.material)
45     implementation(libs.activity)
46     implementation(libs.constraintlayout)
47     implementation(libs.firebase.database)
48     implementation(platform("com.google.firebase:firebase-bom:33.1.2"))
49     implementation("com.google.firebase:firebase-analytics")
50     implementation("com.google.firebase:firebase-database") //untuk library database
51     implementation("com.google.firebase:firebase-auth")
52     implementation(libs.navigation.fragment)
53     implementation(libs.navigation.ui)
54     testImplementation(libs.junit)
55     androidTestImplementation(libs.ext.junit)
56     androidTestImplementation(libs.espresso.core)
57 }

```

Figure 5.17: To connect them, add the Firebase package.

5.4.2 Version Control Procedure

This section will show how to configure the Arduino IDE to program the ESP8266 NodeMCU board with the provided source code. The board chosen is the ESP8266 NodeMCU, which is connected to COM port 3. These steps are critical to ensuring that the source code runs correctly and smoothly. Following that, users must choose which components will use this board and port. The picture below shows the source code for connecting the Wi-Fi module to the microcontroller and saving data in the cloud using Firebase.

```

Parking.ino
1  #include <LiquidCrystal_I2C.h>
2  #include <ESP8266WiFi.h>
3  #include <FirebaseESP8266.h>
4
5  // Set the LCD number of columns and rows
6  int lcdColumns = 16;
7  int lcdRows = 2;
8
9  // Set LCD address, number of columns and rows
10 LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);
11
12 // Define the pins for the IR sensors
13 const int irSensorPin1 = D6; // First sensor pin
14 const int irSensorPin2 = D7; // Second sensor pin
15 const int irSensorPin3 = D4; // Third sensor pin
16
17 // Firebase configuration
18 #define FIREBASE_HOST "https://parking-7f4cb-default-rtdb.firebaseio.com/"
19 #define FIREBASE_AUTH "XoAzgtAEwllFYhZwbCXLNXSu2DNcCIX1K8JvYHkv"
20
21 // WiFi credentials
22 const char* ssid = "sriutama3_2.4Ghz@unifi";
23 const char* password = "Kronisu3";
24
25 // Firebase objects
26 FirebaseData firebaseData;
27 FirebaseConfig firebaseConfig;
28 FirebaseAuth firebaseAuth;

```

Figure 5.18: Show the coding that is used to connect the component Wi-Fi module and firebase cloud in the Arduino IDE.

```

Parking.ino
67 void loop() {
68   // Read the states of the IR sensors
69   int sensorState1 = digitalRead(irSensorPin1);
70   int sensorState2 = digitalRead(irSensorPin2);
71   int sensorState3 = digitalRead(irSensorPin3);
72
73   // Log the sensor states for debugging
74   Serial.print("Sensor 1: ");
75   Serial.println(sensorState1);
76   Serial.print("Sensor 2: ");
77   Serial.println(sensorState2);
78   Serial.print("Sensor 3: ");
79   Serial.println(sensorState3);
80
81   // Update Firebase and LCD only if there is a change in sensor states
82   if (sensorState1 != previousSensorState1 || sensorState2 != previousSensorState2 || sensorState3 != previousSensorState3) {
83     // Debugging statement for sensor change
84     Serial.print("Sensor 1 state changed to: ");
85     Serial.println(sensorState1);
86     Serial.print("Sensor 2 state changed to: ");
87     Serial.println(sensorState2);
88     Serial.print("Sensor 3 state changed to: ");
89     Serial.println(sensorState3);
90
91     // Build the status messages
92     String statusMessage1 = buildStatusMessage(sensorState1, 1);
93     String statusMessage2 = buildStatusMessage(sensorState2, 2);
94     String statusMessage3 = buildStatusMessage(sensorState3, 3);

```

Figure 5.19: Show the coding for Infrared sensors 1,2 and 3.


```

Parking Ino:
96 // Update Firebase
97 bool success1 = Firebase.setInt(firebaseData, "/parking/sensor1", sensorState1);
98 bool success2 = Firebase.setInt(firebaseData, "/parking/sensor2", sensorState2);
99 bool success3 = Firebase.setInt(firebaseData, "/parking/sensor3", sensorState3);
100
101 // Debugging Firebase update
102 Serial.print("Firebase update sensor 1: ");
103 Serial.println(success1 ? "Success" : "Failed");
104 Serial.print("Firebase update sensor 2: ");
105 Serial.println(success2 ? "Success" : "Failed");
106 Serial.print("Firebase update sensor 3: ");
107 Serial.println(success3 ? "Success" : "Failed");
108
109 // Display the status messages on the LCD
110 displayStatusMessages(statusMessage1, statusMessage2, statusMessage3);
111
112 // Update the previous sensor states
113 previousSensorState1 = sensorState1;
114 previousSensorState2 = sensorState2;
115 previousSensorState3 = sensorState3;
116
117
118 delay(500); // Reduce the delay to make the system more responsive
119 }
120
121 String buildStatusMessage(int sensorState, int sensorNumber) {
122   String statusMessage = "S" + String(sensorNumber) + ":";
123   statusMessage += (sensorState == LOW) ? "Full" : "Empty";

```

Figure 5.20: Show the coding update into Firebase cloud

5.5 Implementation Status

A key component of the project's duration and status is the progress of the smart parking system's implementation. Every component or module's development status is listed in detail in table 5.6 below. The name, description, amount of time needed to finish, and the date the module was successfully finished are all included in this breakdown.

Table 5.6: The implementation of status project.

| No. | Module Name | Description | Duration |
|-----|-----------------|--|----------|
| 1 | Setup Installer | Installation and setting up the software Arduino IDE with the port, Firebase Cloud and Android Studio. | 3 days |

| | | | |
|---|---------------------------|--|---------|
| 2 | Assemble Hardware | Setting up the NodeMCU ESP8266, WiFi module, Infrared sensors, wire and I2C LCD. | 6 days |
| 3 | Implement all source code | Implement code for all part which is ESP8266, infrared sensor, firebase cloud, android application | 43 days |
| 4 | Develop application | This application builds the software and install it on the smartphone to ensure that the updates are received by the user. | 30 days |
| 5 | Test the prototype | Do some tests to check if the prototype is functional. | 6 days |

5.6 Conclusion

Lastly, the implementation phase provides a detailed explanation of the creation process of the Smart Parking System project. The procedures and tasks associated with this project are completed gradually to ensure its efficient and successful operation. Additionally, during this process, the environment and configuration must be properly set up to be more effective. There is a thorough discussion of every aspect of the software and hardware utilized in the configuration. To move on to the testing and analysis phase of Chapter 6, all the necessary data must be collected and gathered before going through it.

CHAPTER 6: TESTING

6.1 Introduction

This chapter covers the project's testing phase, which aims to make sure the system and product being developed function as designed and meet all requirements. To ensure the project's success and that it is appropriate for its intended use, extensive testing is carried out on all project components as well as the product to provide the desired output. This project specifically uses a NodeMCU ESP8266, and its main focus is on the Android Smart Parking system, which shows real-time information about available parking spots to help users avoid wasting time. Testing is repeated several times until the expected results are obtained. Moreover, this chapter covers a number of subtopics, such as the test plan, test setting, test design, test outcome, and interpretation.

6.2 Test Plan

This section covers the principles of each method and product, including product testing. This portion covers the research process, including participants and methods. This test strategy helps identify potential difficulties with device operation. The project aims to ensure that the product and system strategy can effectively address the user's problem. A test strategy is crucial before users use the project's products or systems.

6.2.1 Test Organization

In this part, two people are responsible for carrying out the system's testing process. The first is the system developer, who creates the system; the second is the end user. Both testers play different roles in the testing process.

i. System Developer

The System Developer is responsible for doing thorough system testing, actively finding out any weaknesses or problems in the system. They are also responsible for ensuring that the system is stable and runs smoothly.

ii. End User

The End User plays an important role in finding system flaws, providing feedback on user interfaces and functionality, and suggesting enhancements to the system.

6.2.2 Test Environment

In this project, the prototype is tested with Wi-Fi connectivity to ensure that it can receive messages from Smart Parking. The project is powered by a 5V source connected via a USB cable. This prototype uses the NodeMCU's ESP8266 microcontroller to monitor modules that connect with it. The ESP8266 NodeMCU is connected to an Infrared sensor to detect available parking spaces. Data is stored and maintained using a Firebase-integrated website, guaranteeing that the project can give accurate and timely updates. As a result, a complete test environment was created throughout the project development period to ensure that the testing process was reliable and precise.

6.2.3 Test Schedule

This section aims to estimate project completion time by creating a test schedule. During the implementation phase, errors and problems occurred, requiring extra time to fix and check. Continuous testing is ongoing until the system is fully functional. The evaluation plan's duration is as follows. After setting up the prototype, users must upload and test the code on hardware to ensure it works properly. As a result, the prototype must function correctly and successfully.

i. Start the prototype

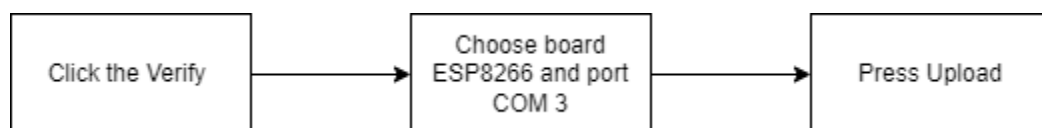


Figure 6.1: Show the prototype's role.

Figure 6.1 illustrates how to begin the prototype. Use the "verify" button in the Arduino IDE to guarantee code is error-free. To create a functional prototype, users should choose the appropriate board. In this project, we choose the ESP8266 NodeMCU board with port COM3. The prototype cannot run if users are unable to choose the correct board and port. Then, to import code into the prototype, simply press the "upload" button.

ii. Functionality of the prototype

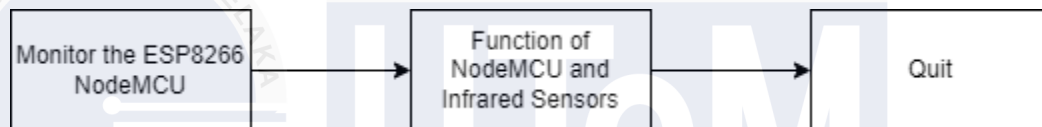


Figure 6.2: Show the functionality of the prototype.

Figure 6.2 illustrates the prototype's functioning. The NodeMCU ESP8266 is controlled by crucial components. The monitoring NodeMCU connects all components for increased usefulness. After the user enters the parking space, the infrared sensor is used to detect whether it is available or occupied. The functionality components help the user view it within the application.

iii. Connect with Firebase

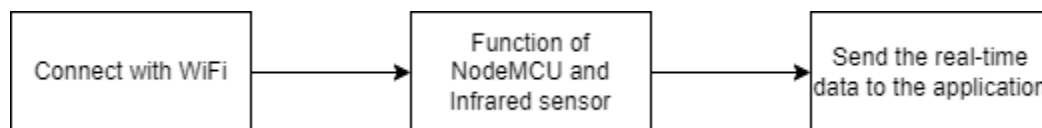


Figure 6.3: Show the connection to the Firebase

Figure 6.3 illustrates that the user needs Wi-Fi to connect to the Firebase, which sends real-time data and alerts to their smartphone. All the data get from the sensor and mobile application will be securely stored in the Firebase cloud.

- iv. Start the 'Smart Parking System' Application

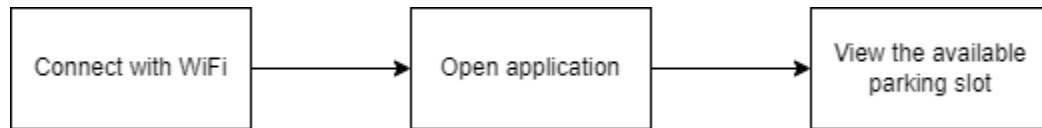


Figure 6.4: Show how to use the Smart Parking System application.

To use the Smart Parking system application shown in Figure 6.4, users must first establish an internet connection. This connection is required to activate the program and receive real-time data from the infrared sensors. Users will receive an update of the available parking slots.

6.3 Test Strategy

This section describes the project's testing strategy, which is intended to detect cars in parking. Users must install the Smart Parking system app on their Android devices in order to receive updates and view detailed information. The prototypes have undergone rigorous testing to ensure optimal performance. The testing is divided into two parts: analyzing the prototype and reviewing how information is received via the Smart parking system application. The major goal of these tests is to acquire insight into both prototype testing and information testing processes.

- i. Testing on receiving the information

The information received when there are available parking spaces and sent to the user's smartphone for this testing. However, in order to receive information, the smartphone must first be connected to the internet.

- ii. Testing the sensor's detection

To test the sensor's detection, it will require an internet connection and electricity. The infrared sensor will detect available and occupied parking spaces and update the application to alert the user.

6.3.1 Classes Test

i. Functionality test

This function is used to validate the smart parking prototype's operation in connection with the NodeMCU ESP8266, as well as to evaluate sensor performance. The process began with the ESP8266 component, which validated the sensors' capacity to detect readings and send data to the cloud. Furthermore, this data can be seen readily within applications, offering a full and real-time representation of the information.

6.4 Test Design

In the testing process, this is one of the most crucial steps when the accuracy and efficacy of the designed modules or components in the system are verified. Every component or module is examined to verify the process. Verifying that the device fulfils the requirements and operates as intended requires testing. The project can achieve its goals by improving its functionality through the completion of this study. To further fulfil system objectives, testing may help to improve system functionality. Each component and module has been carefully examined to guarantee faultless operation.

6.4.1 Test Description

The purpose of this section is to determine which aspects of the project require testing. Tables 6.1–6.3 show the test cases used in this project. This part covers all the components and modules necessary to achieve a precise and successful outcome. This prototype requires comprehensive knowledge of all project components. Users must understand project procedures and performance expectations. The table below contains information about all of the test cases.

- i. Table 6.1 shows the results of Wi-Fi module testing

Table 6.1: The Wi-Fi module function

| | |
|-------------------------|---|
| Test | Test of the Wi-Fi module's connectivity |
| Test Purpose | To test whether the Wi-Fi module has a connection with the internet. |
| Test Environment | To perform this pre-situation test, a Wi-Fi module must be configured. Follow the installation and configuration steps outlined in Section 5.4.1. |
| Test Setup | <ol style="list-style-type: none"> i. Download the Arduino IDE and Install it on the laptop. ii. Create the source code at the NodeMCU ESP8266. iii. After running the code, the Wi-Fi module functions to test the internet connection. |
| Expected Result | After running the code, the Wi-Fi module functions to test the internet connection. |
| Error Message | None |
| Result | Pass |

- ii. Table 6.2 shows the results of the Infrared sensor testing.

Table 6.2: Infrared sensor function

| | |
|-------------------------|---|
| Test | Test of the infrared sensor connectivity. |
| Test Purpose | To test the detection of a car in parking. |
| Test Environment | To run this pre situation test, the infrared sensor connection with NodeMCU ESP8266 must be set up. |

| | |
|------------------------|--|
| Test Setup | <ul style="list-style-type: none"> i. Download the Arduino IDE and Install it on the laptop. ii. Connect the infrared sensor to the NodeMCU ESP8266 using a jumper wire. iii. Create the source code for the infrared sensor. iv. After running the code, the infrared sensor is function to detect the car in the parking slot. |
| Expected Result | After running the code, the infrared sensor is function to detect the car in the parking slot. |
| Error Message | None |
| Result | Pass |

- iii. Table 6.3 shows the results of the Smart Parking System application on smartphone testing.

Table 6.3: Smart Parking System function

| | |
|-------------------------|---|
| Test | Test of the Smart Parking alert's application connectivity. |
| Test Purpose | To test which user can check updates on the mobile application. |
| Test Environment | In order to run this pre-situation test, the smartphone needs to have an Internet connection to connect with the smart parking to get information about the parking slot. |
| Test Setup | <ul style="list-style-type: none"> i. Download and install Android Studio on the laptop. ii. Create a new project and design a simple application for a smart parking system. |

| | |
|------------------------|---|
| | <ul style="list-style-type: none"> iii. Connect with the Firebase cloud. iv. Create the source code for the application and run it. v. After running the code, the user can view the details using this application. |
| Expected Result | After running the code, the user can view the information of the parking from the prototype. |
| Error Message | None |
| Result | Pass |

6.4.2 Test Data

i. NodeMCU ESP8266 connectivity test

The ESP8266 microcontroller board is connected to the laptop using a micro-USB cable as shown in Figure 6.5 below.

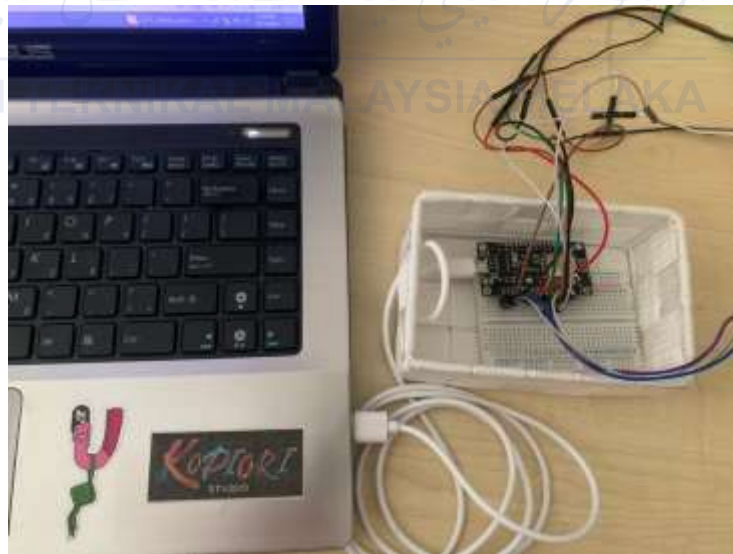


Figure 6.5: ESP8266 connected with laptop

We may check the connectivity status in the Arduino IDE to see if the connection was successfully created. To program the NodeMCU, go to the Tools menu and choose

COM 3 as the port. The diagram below shows the connection between the Arduino board and the USB port, which is used to display the code upload port. If the source code is free of errors, the console window will confirm that the source code was successfully uploaded.

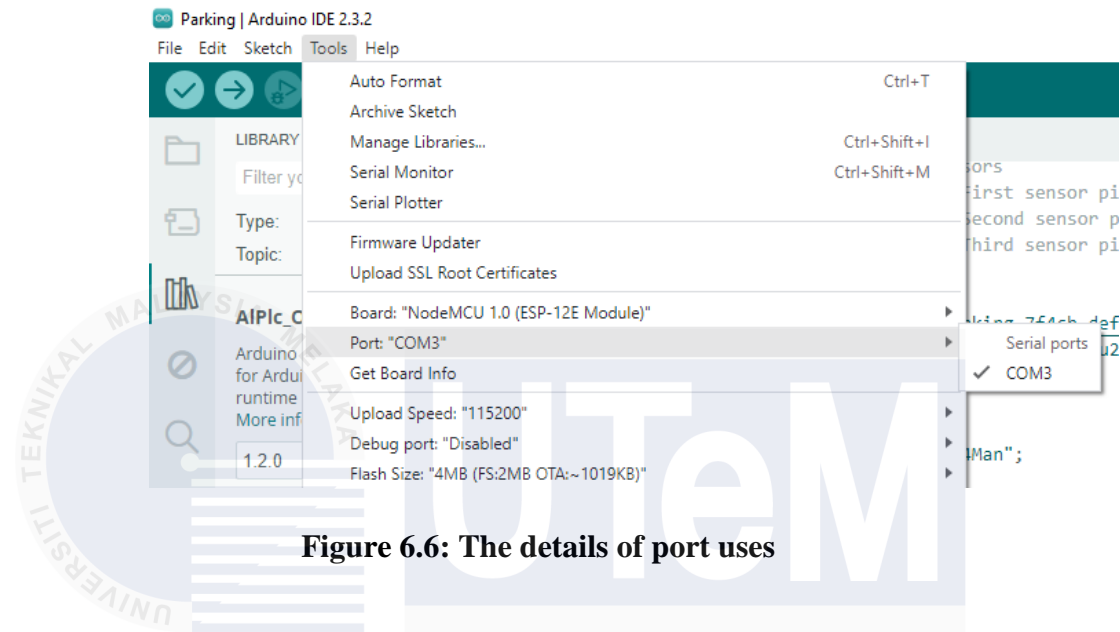


Figure 6.6: The details of port uses

ii. Component connectivity test

Figure 6.7 shows the setup connection of the ESP8266 NodeMCU and Infrared sensor component using the jumper wires to connect each other.

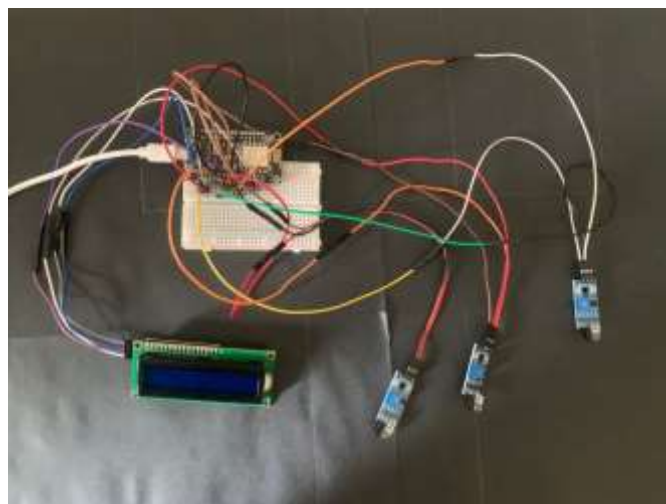


Figure 6.7: Setup connection between hardware

iii. Application connectivity test

Android Studio served as the development platform for creating the Smart parking system application in this project. This application is designed for Android smartphones and provides users with real-time updates on available parking spaces in the parking through an application.

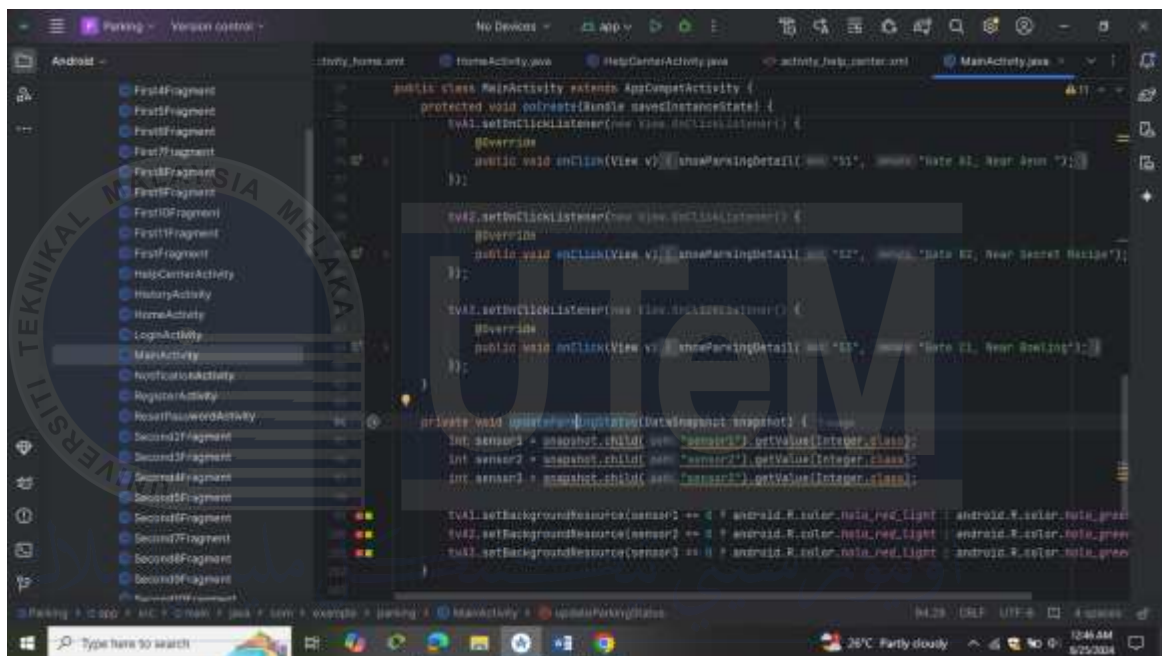


Figure 6.8: Android Studio's software

Figure 6.8 above shows the software that created the application. It also connects with Firebase to store the data in real-time.



Figure 6.9: User information from database

Figure 6.9 shows the details of user from database that has been registered to using the apps and figure 6.10 shows the scope status that has been released from the smart parking system.



Figure 6.10: Scope status from database

6.5 Test Result and Analysis

After finishing the previously specified test schedule, test strategy, and test design. This sub-topic focuses on identifying test cases, identifying testers, and determining if test findings are satisfactory or unsuccessful. The research findings for each project component are presented below.

6.5.1 Test Result on Hardware

i. NodeMCU ESP8266

Table 6.4: ESP8266 function result and analysis

| Test Case Identification | Test Identification | Result Expectation | Success/Fail |
|--------------------------|---------------------|--------------------|--------------|
| | | | |

| | | | |
|---|---|--|---------|
| 1 | Uploading the codes and click run | The microcontroller is connected the internet connection that has set in the code. | Success |
| 2 | When the codes are not uploaded, and the run button is not clicked. | The microcontroller has no internet connection. | Success |

In this project, an ESP8266 Wi-Fi module is essential for transmitting and continuously updating information on the user's smartphone. Users will not be able to receive Smart parking system updates until this module is installed, and car movements will not be detected. Configuring the ESP8266 in the Arduino IDE and assigning it a username and password are critical steps toward securing its operation.

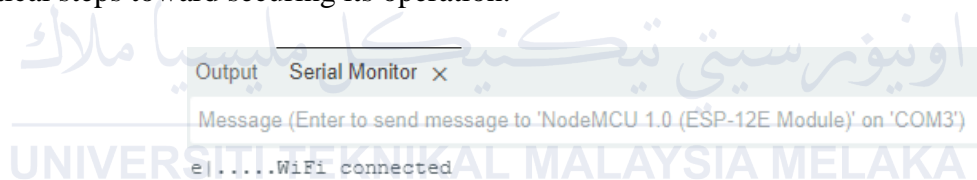


Figure 6.11: ESP8266 success to connect the internet

Figure 6.11 show how the Wi-Fi module is successful in connecting to the internet. This component helps the user to get the real-time data about the available parking spaces.

ii. Infrared Sensor

Table 6.5: Sensor function result and analysis

| Test Case Identification | Test Identification | Result Expectation | Success/Fail |
|--------------------------|---|---|--------------|
| 1 | Uploading the codes and click run | The sensors are functioning to get the reading of the detection of the car. | Success |
| 2 | When the codes are not uploaded, and the run button is not clicked. | The sensors are not functioning to get the reading of the detection of the car. | Success |

6.5.2 Test Result on Application

i. Smart Parking System

Table 6.6: Testing on the application

| Test Case Identification | Test Identification | Result Expectation | Success/Fail |
|--------------------------|---|---|--------------|
| 1 | The user can view the available parking | Users can view the available parking slots. | Success |
| 2 | The user can edit their profile | User can edit their profile | Success |
| 3 | The user can submit the feedback on the application | Users can send some feedback about the parking. | Success |
| 4 | The user receives the notification | Receives the notification about parking. | Fail |
| 5 | The user can view the history of the parking | The history can be viewed by the user. | Fail |

6.6 Usability Test

In 1986, John Brooke established the System Usability Scale (SUS) (Brooke, 1996). The effectiveness of objects and products in terms of their design and usability can be quickly and easily assessed using this scale. SUS is a practical and reliable method for evaluating perceived usability that may be used on a variety of digital products and services. Programmers and developers can use it to determine whether a design solves a certain issue successfully. Symk (2020) states that SUS is used for an overall usability evaluation in compliance with ISO 9241-11 and covers the following elements while not intended for diagnostic use :

- Effectiveness: Can users accomplish their objectives successfully?
- Efficiency: How much money, time, and effort are needed to reach those objectives?
- Satisfaction: Did the user have a good experience?

Each of the 10 questions in the SUS questionnaire has five response possibilities, ranging from "Strongly Agree" to "Strongly Disagree," for each respondent. I also customized the System Usability Scale (SUS Brooke's questions for my project while retaining their original meaning and intent. These questions are intended to be completed quickly and with minimal input, making it easy to collect user feedback throughout each testing session. The Google Forms platform was used for the survey since it can generate charts and organize real-time answer data. The questionnaire contains 10 questions that I have been modified according to smart parking system project, which are given below.

1. I think that I would like to use this smart parking system application frequently.
2. I found the smart parking system application unnecessarily complex.
3. I thought the smart parking system application was easy to use.

4. I think that I would need the support of a technical person to be able to use this smart parking system application.

5. I found the various functions in this smart parking system application were well integrated.

6. I thought there was too much inconsistency in this smart parking system application.

7. I would imagine that most people would learn to use this smart parking system application very quickly.

8. I found the smart parking system application very cumbersome to use.

9. I felt very confident using the smart parking system application.

10. I needed to learn a lot of things before I could get going with this smart parking system application.

6.6.1 Result

According to (Symk, 2020) findings in 2020, a minimum of five user feedback answers are required to acquire credible data. As a result, five people filled out this questionnaire. Each respondent participated in a usability testing session with the smart parking system device and had 1-2 minutes to complete the Google Form questionnaire. The test results are provided below.

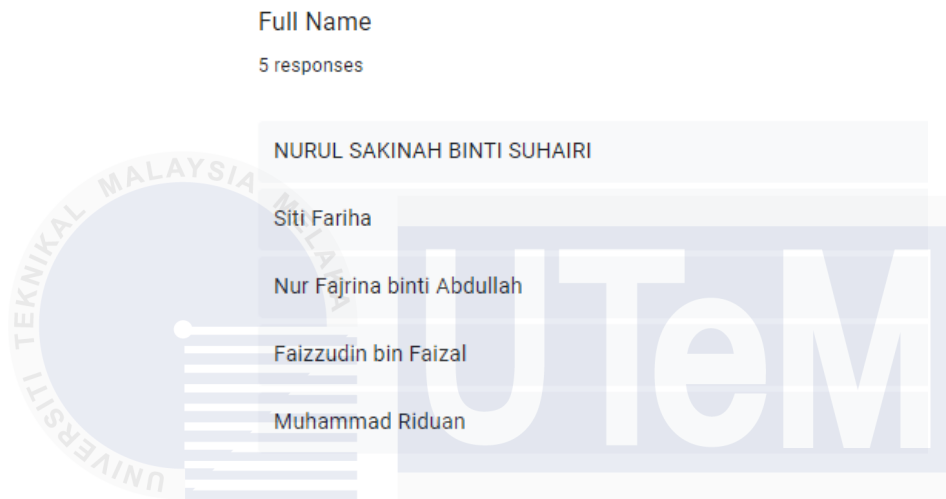


Figure 6.12: Name of the responses

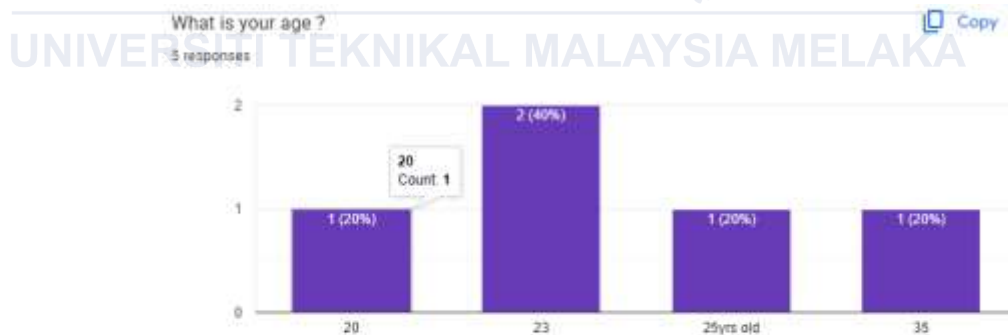


Figure 6.13: Age of the responses

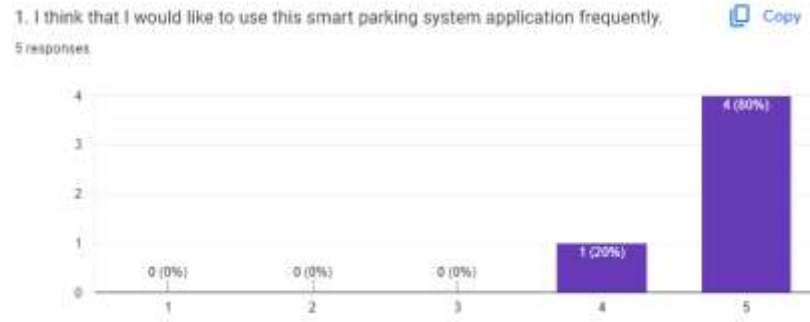


Figure 6.14: Answer question 1 from respondents

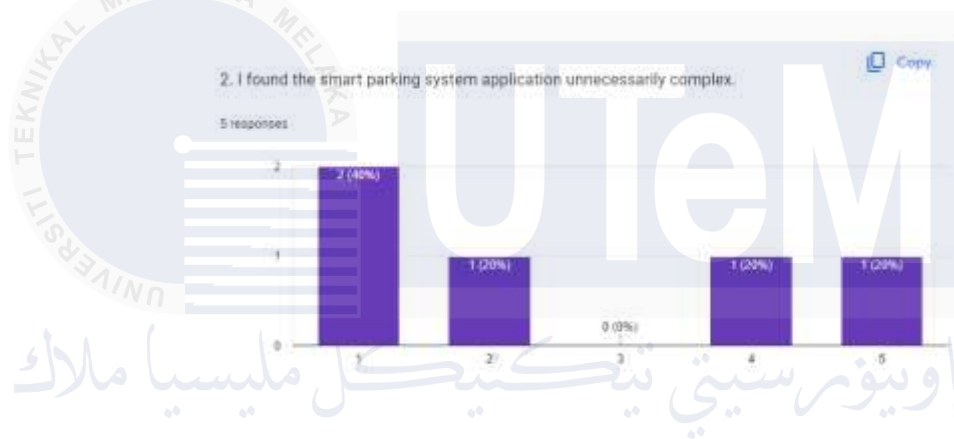


Figure 6.15: Answer question 2 from respondents

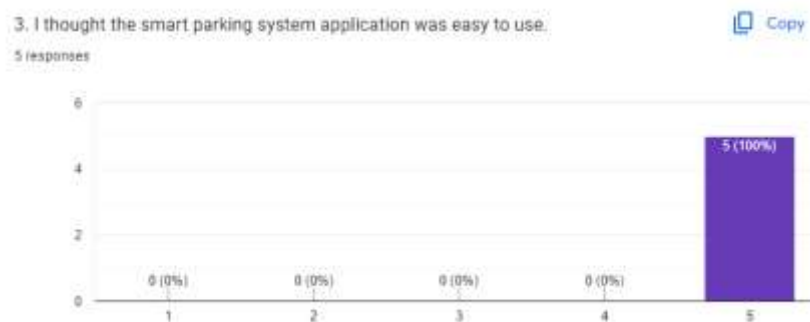


Figure 6.16: Answer question 3 from respondents

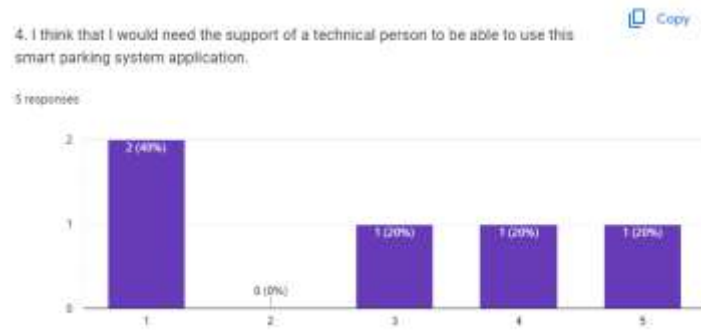


Figure 6.17: Answer question 4 from respondents

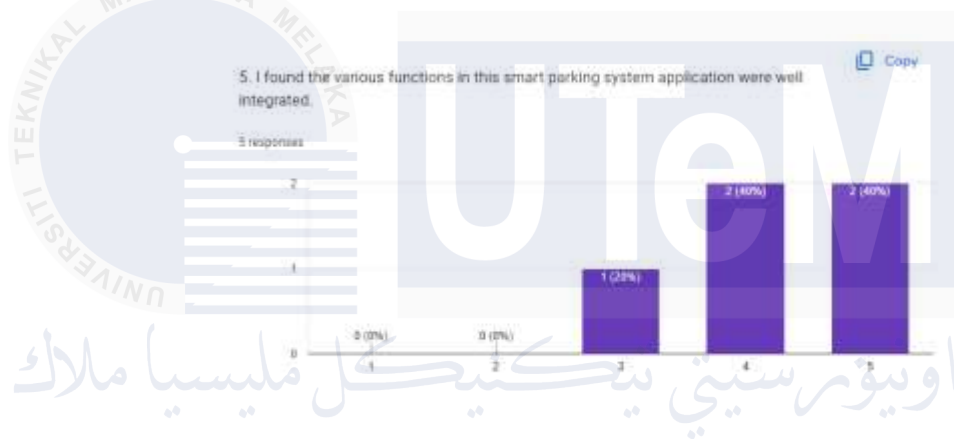


Figure 6.18: Answer question 5 from respondents



Figure 6.19: Answer question 6 from respondents

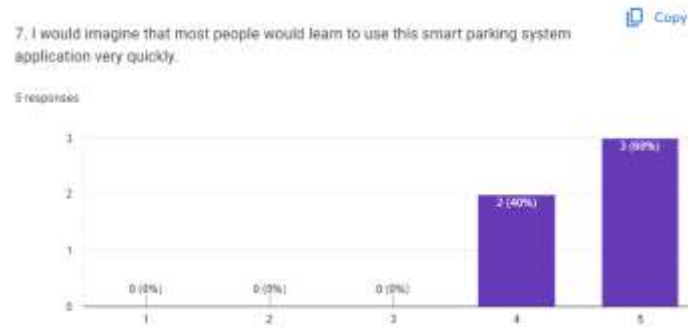


Figure 6.21: Answer question 7 from respondents

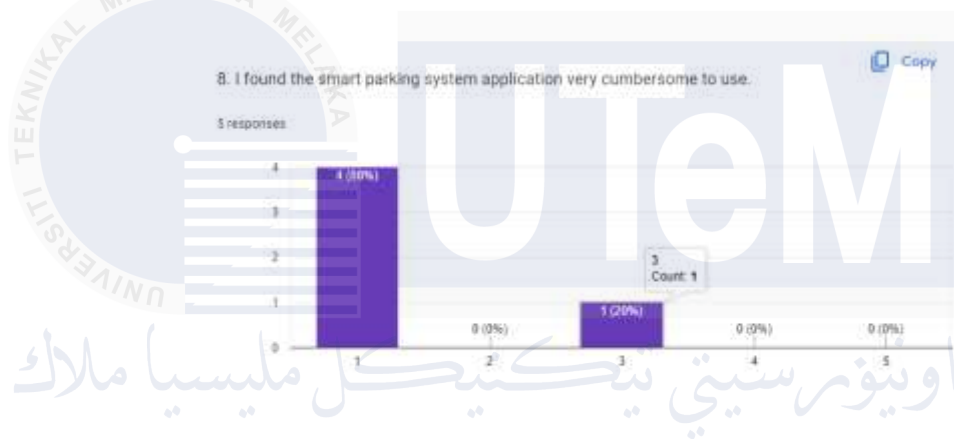


Figure 6.22: Answer question 8 from respondents



Figure 6.23: Answer question 9 from respondents

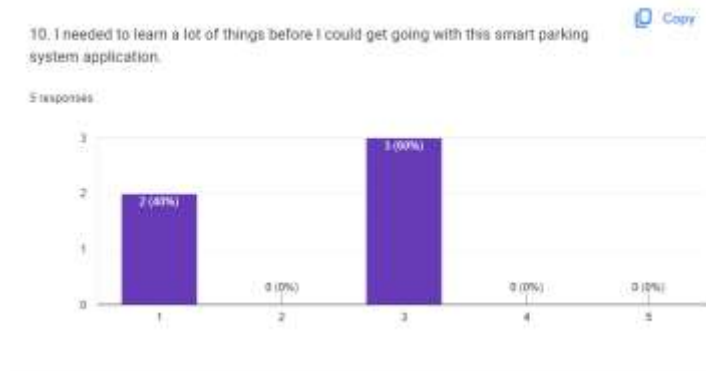


Figure 6.24: Answer question 10 from respondents

6.6.2 Interpreting Score

The SUS score will be determined using the response scale, which has a range of 1 to 5. Add the total scores for all of the odd-numbered questions (1,3,5,7,9) for 5 respondents. The odd-numbered questions express positive attitudes, while the even-numbered ones express negative views. Subtract one from each answer to an odd statement to obtain the SUS score. In this case, the total score for odd-numbered questions will be reduced by 25, as there are 5 odd-numbered questions for every 5 responders. Next, add the answers to each even-numbered question (2, 4, 6, 8, 10) and subtract them from 125 to obtain Y. Lastly, multiply the new values' total scores (X + Y) by 2.5.

$$X = \text{Total Odd Numbered Questions} - (5 \times 5) = (15+17+19+20+20) - 25 = 91 - 25 = 116$$

$$Y = (12.5 \times 10) - \text{Total Even Numbered Questions} = 125 - (7+10+14+20+20) = 125 - 71 = 54$$

$$\text{Total Score} = \frac{(x+y) \times 2.5}{500} \times 100 = \frac{(116+54) \times 2.5}{500} \times 100 = 85$$

Following the scoring tabulation process stated above produces a score out of 100, in this case 85/100. According to Sauro, a SUS score is not a percentage but can range from 0 to 100. (2019). Letter grades can be integrated into SUS scores, making the results easier to convey.

Table 6.7: SUS Score Grading

| Grade | SUS Score | Percentile | Adjective |
|-------|-------------|------------|-----------|
| A | 78.9 - 100 | 85 - 100 | Excellent |
| B | 72.6 - 78.8 | 65 - 84 | Good |
| C | 62.7 – 72.5 | 35 – 64 | OK |
| D | 51.7 – 62.6 | 15 – 34 | Poor |
| E | 0 - 51.6 | 0 - 14 | Awful |

The SUS score for the Smart Parking System falls between 78.9 and 100. This shows that the Smart Parking System gets an A which is an excellent rating.

6.7 Conclusion

In a summary, from what I have been doing from this chapter, I found that there are some goals that I cannot achieve which is in testing the notification and history. The testing failed and it did not fulfil the requirement but some of the designs of the Smart parking system application can met the goals with a clear, concise, and productive design, developing the tool prototype, and testing the prototype's effectiveness. The purpose of the feature test is to validate that the prototype functions as intended. After analyzing the data, conclusions were reached from different settings. The project summary, contribution, limitations, and the overall project progress will all be covered in the upcoming chapter.

CHAPTER 7: CONCLUSION

7.1 Introduction

This chapter covers several important topics, including a project summary, project contributions, project limitations, and future project activities. Following that, a project summary provides an overview of project accomplishments as well as information on the development and testing processes. This chapter also digs into the project's restrictions and potential enhancement opportunities, with the goal of improving dependability, efficiency, and overall effectiveness.

7.2 Project Summarization

Smart Parking System application is a project in which user can view the available parking spaces through an application. Users need to register and then log in to the application. The project is built with a NodeMCU ESP8266 and an infrared sensor. Features include strong processors, built-in Wi-Fi, and deep sleep mode. The hardware is capable of connecting to mobile phones via a Wi-Fi connection. As a result, when the mobile phone is connected to Wi-Fi, the user can use the application.

There are three objectives that need to be achieved as stated in Chapter 1 which are to study the problem of the existing parking system and the possibility of enhancement. Next, to develop the Smart Parking IoT application that could update the owner about the parking slot through the application. Also, to test the smart parking IoT application's effectiveness. The project's objective has been fulfilled from this project. The prototype device uses an ESP8266 NodeMCU microcontroller board to monitor and control all components. It uses an infrared sensor to detect cars and provide real-time updates on parking availability. Every component plays a role in ensuring that the prototype functions well. This smart parking system works effectively after the hardware and software are integrated.

i. Project Weakness

User Experiences: The lack of development in the notification and history interface is a key flaw in this project's user experience, which may result in users missing important changes and being unable to effectively manage their parking history.

Connectivity: The communication between the ESP8266 and Firebase is mainly dependent on a steady Wi-Fi connection. In places with not sufficient Wi-Fi, the system may have delays or difficulties while updating parking status.

Hardware uses: The project does not provide a camera so it cannot automatically detect the number plate that enters the parking. Other than that, Infrared sensors may be prone to inaccuracies due to surroundings such as light, dirt, or obstacles. This could result in false readings and unreliable parking status updates.

ii. Project Strength

Real-Time Parking Status Updates: The integration of infrared sensors with the ESP8266 NodeMCU and Firebase enables real-time updates on parking slot availability, providing users with accurate and timely data.

Automation and Efficiency: The technology automates the process of recognising available parking spaces, which reduces the need for manual intervention and improves parking management efficiency. The system assists users in finding parking more quickly by delivering real-time information on available spaces, minimising the time spent looking for a spot.

User-friendly interface: The Android app's home screen, which includes buttons for User Profile, Status, Feedback, and Help Centre, simplifies navigation and improves the overall user experience.

7.3 Project Contribution

The Android Smart Parking System is intended to give consumers with a seamless parking experience by delivering crucial functionality via a mobile application. This system contains a home interface that allows users to quickly manage their profiles, check the status of parking lots, submit feedback, and contact a help centre for assistance. The application's integration of these elements allows users to properly manage their parking needs, guaranteeing they can discover open spots fast and with the least effort. Although the project does not presently have a history or notification interface, it is still a useful tool for improving the parking experience. This project serves both drivers and parking facility managers by improving parking administration and providing a simple and user-friendly interface.

7.4 Project Limitation

Every project has limitations, and it's critical to be aware of them in order to successfully manage expectations and seek to improve the system. Understanding these limits is critical for successful project management and risk assessment. Mitigation measures and continuous improvement initiatives can be used to overcome some of these limits over time.

Here are potential limitations of a smart parking system project:

- i. **Lack of History Interface:** The application lacks a history interface, limiting users' ability to trace past parking activity. This may diminish the app's overall utility for users who need to review their parking history for record-keeping or personal reference purposes.
- ii. **Absence of Notification System:** Without a notification interface, the app cannot send users real-time notifications on parking availability or other crucial occurrences. This

- limitation may reduce the app's ability to keep users informed, particularly in dynamic or high-demand parking environments.
- iii. **Sensor accuracy:** The system's infrared sensors may generate false positives (showing that a parking space is occupied when it isn't) or false negatives. This can result in incorrect parking status updates, irritating customers who rely on the app for real-time data.
 - iv. **Data Connectivity:** The system relies largely on a steady internet connection to enable real-time communication between the ESP8266 NodeMCU, Firebase, and the Android app. In places with weak or inconsistent connectivity, users may encounter delays or difficulties in obtaining parking status updates, reducing the app's dependability and user happiness.

7.5 Future Works

Regarding the Android smart parking system project, the following are some possible future directions and areas of work:

- i. **Mobile Application Development:** Create a history interface so that users may track their previous parking behaviours, such as entry and leave times, parking duration, and fees. This feature would improve the user experience by giving relevant insights and records. Aside from that, include a real-time notification system to notify users of available parking spaces, reservation confirmations, and reminders to move their vehicles. This might dramatically boost user engagement and pleasure.
- ii. **Enhance Sensor Accuracy:** Upgrade the system with more advanced sensors, such as ultrasonic or camera-based sensors, to offer more accurate and dependable data on parking space availability.

- iii. **Cross-Platform Compatibility:** Create an iOS version of the app to reach consumers on Apple devices, increasing the system's accessibility and user base. Make a web-based version of the application for users who prefer to handle their parking needs from a computer or website.

7.6 Conclusion

In conclusion, the creation and deployment of the Android Smart Parking System represents a significant step forward in parking management, providing users with a more efficient and easy parking experience. This project offers real-time parking space updates and user management capabilities through the use of infrared sensors, ESP8266 NodeMCU technology, and a customised mobile application connected with Firebase. The system's strengths include the capacity to simplify the parking process, improve user accessibility, and provide necessary features like profile management, parking status checks, feedback, and a support centre.

However, while the project has made great progress, it does have limitations. The current implementation lacks sophisticated features like notification alerts and a history interface, which could improve the user experience and system utility. Furthermore, issues with sensor accuracy and data communication are challenges that must be solved to ensure the system's reliability and efficiency.

Looking ahead, there are numerous chances to develop and expand the system. Future advancements may include the incorporation of more precise sensors, the implementation of real-time notifications, and the addition of a history interface. By addressing these areas for improvement and embracing emerging technology, we may increase the system's robustness and reliability. The smart parking system can evolve to meet the increasing demands of urban mobility through continual innovation and focused on user improvements, resulting in more efficient and user-friendly parking solutions.

REFERENCE

- Abd Warif, Nor & Azman, Mohd & Ismail, Nor-Syahidatul & Remli, Muhammad. (2020). IoT-based Smart Parking System using Android Application. 1-6. <https://doi.org/10.1109/ETCCE51779.2020.9350914>
- Alsafery, W., Alturki, B., Reiff-Marganec, S., & Jambi, K. (2018). *Smart Car Parking System Solution for the Internet of Things in Smart Cities*. <https://doi.org/10.1109/cais.2018.8442004>
- Nasim, Sofeem & Oussalah, Mourad & Outila, Tarja & Jutila, Johannes. (2022). Smart Parking System with PlacePod, LoRaWAN IoT sensors and Android app. 278-284. <https://doi.org/10.1109/IRIS4793.2022.00066>
- Aravindh, G. & Kumar, M.. (2020). An Efficient Car Parking Management System using raspberry-pi. 154-158. <https://doi.org/10.1109/ICISS49785.2020.9316049>
- Bonde, Devyani & Shende, Rohit & Kedari, Akshay & Gaikwad, Ketan & Bhokre, Amol. (2014). Automated car parking system commanded by Android application. 1-4. <https://doi.org/10.1109/ICCCI.2014.6921729>
- Chandran, M., Mahrom, N. F., Sabapathy, T., Jusoh, M., Osman, M. N., Yasin, M. N., Hambali, N., Jamaluddin, R., Ali, N., & Wahab, Y. A. (2019). An IoT Based Smart Parking System. *Journal of Physics. Conference Series*, 1339(1), 012044. <https://doi.org/10.1088/1742-6596/1339/1/012044>
- Herrera-Quintero, L. F., Vega-Alfonso, J., Bermudez, D., Marentes, L. A., & Banse, K. (2019). ITS for smart parking Systems, towards the creation of smart city services using IoT and cloud approaches. *ITS For Smart Parking Systems, Towards the Creation of Smart City Services Using IoT and Cloud Approaches*. <https://doi.org/10.1109/scsp.2019.8805705>

Avinash, C., Rohit, G., Rajesh, C., Suresh, A., & Chinnadurai, S. (2022). IOT based smart parking system with E-Ticketing. *IOT Based Smart Parking System With E-Ticketing*. <https://doi.org/10.1109/icmacc54824.2022.10093659>

Michel-Torres, D. A., Luque-Vega, L. F., Lopez-Neri, E., Carlos-Mancilla, M. A., & Gonzelez-Jimenez, L. E. (2019). IoT based Smart Vehicle Presence Sensor SPIN-V for smart parking system. *IoT Based Smart Vehicle Presence Sensor SPIN-V for Smart Parking System*. <https://doi.org/10.1109/sysose.2019.8753862>

Pandey, D., & Author, S. H. (2018). Navigation based - intelligent parking Management System using queuing theory and IOT. *Navigation Based - Intelligent Parking Management System Using Queuing Theory and IOT*. <https://doi.org/10.1109/icgciot.2018.8753053>

Alkhuraiji, S. (2020). Design and implementation of an Android Smart Parking mobile Application. *TEM Journal*, 1357–1363. <https://doi.org/10.18421/tem94-06>

Instructables. (2020, February 19). *How to install Arduino IDE on Windows 10*. Instructables. <https://www.instructables.com/How-to-Install-Arduino-IDE-on-Windows-10/>

Buzdar, K. (2024, May 26). *How to install Android Studio on Windows*. Ultrahost Knowledge Base. <https://ulthost.com/knowledge-base/install-android-studio-on-windows/>

Admin. (2021, November 21). *Smart Parking System Project using Arduino and IR Sensor*. ElectroDuino. https://www.electroduino.com/smart-parking-system-project-using-arduino-and-ir-sensor/#google_vignette

APPENDICES

APPENDICES A

i. Source code of the ESP8266 NodeMCU

```
Parking.ino
1  #include <ESP8266WiFi.h>
2  #include <FirebaseESP8266.h>
3  #include <LiquidCrystal_I2C.h>
4
5  // Set the LCD number of columns and rows
6  int lcdColumns = 16;
7  int lcdRows = 2;
8
9  // Set LCD address, number of columns and rows
10 LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);
11
12 // Define the pins for the IR sensors
13 const int irSensorPin1 = D6; // First sensor pin
14 const int irSensorPin2 = D7; // Second sensor pin
15 const int irSensorPin3 = D4; // Third sensor pin
16
17 // Firebase configuration
18 #define FIREBASE_HOST "https://parking-7f4cb-default-rtdb.firebaseio.com/"
19 #define FIREBASE_AUTH "XoAzgtAEw1lFYhZwbCXLNXSu2DNCIX1K8JvYHkv"
20
21 // WiFi credentials
22 const char* ssid = "sriutama3_2.4Ghz@unifi";
23 const char* password = "Kronisu3";
24
25 // Firebase objects
26 FirebaseData firebaseData;
27 FirebaseConfig firebaseConfig;
28 FirebaseAuth firebaseAuth;
29
```

Shows the source code for the ESP8266 connect to Wi-Fi using Arduino IDE

Parking.ino

```

30 // Variables to store the previous sensor states
31 int previousSensorState1 = HIGH;
32 int previousSensorState2 = HIGH;
33 int previousSensorState3 = HIGH;
34
35 void setup() {
36     // Initialize Serial for debugging
37     Serial.begin(115200);
38
39     // Initialize LCD
40     lcd.init();
41     // Turn on LCD backlight
42     lcd.backlight();
43
44     // Initialize the IR sensor pins as inputs
45     pinMode(irSensorPin1, INPUT);
46     pinMode(irSensorPin2, INPUT);
47     pinMode(irSensorPin3, INPUT);
48
49     // Display the initial messages
50     displayInitialMessages();
51
52     // Connect to WiFi
53     WiFi.begin(ssid, password);
54     while (WiFi.status() != WL_CONNECTED) {
55         delay(500);
56         Serial.print(".");
57     }

```

Shows the sources code for initialize the LCD and IR sensor pins using Arduino IDE

```

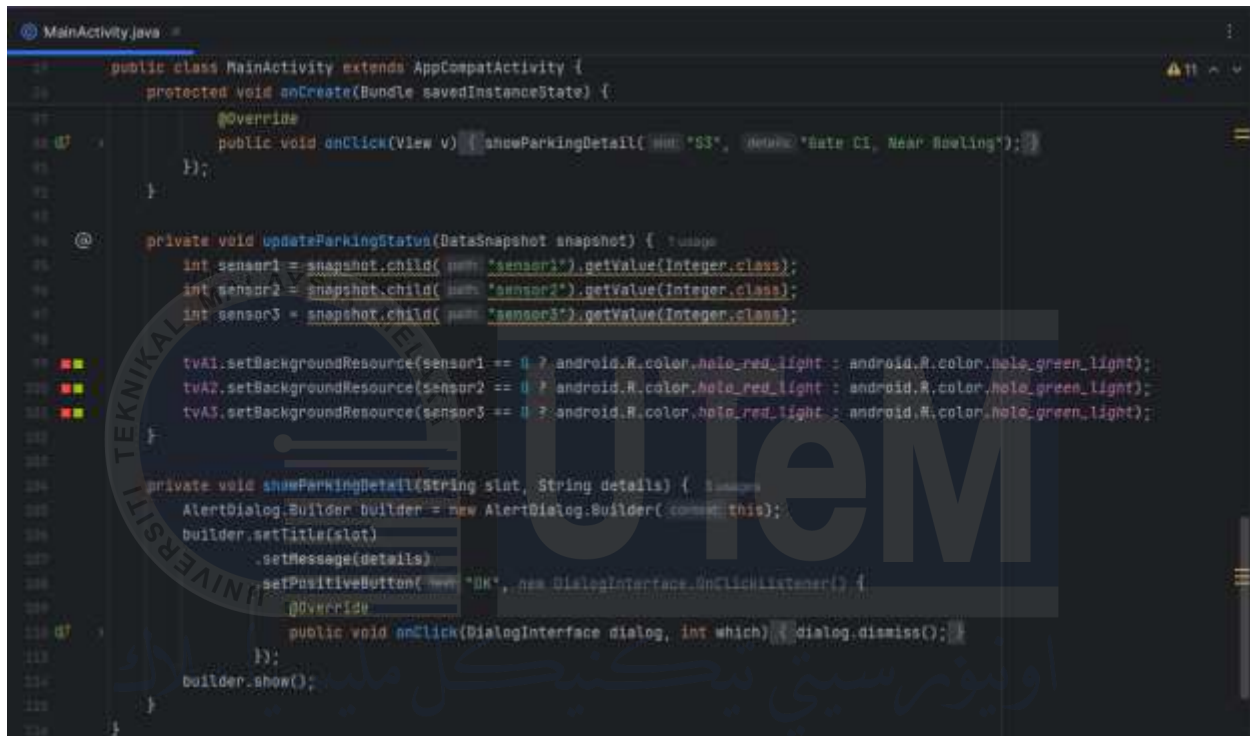
67 void loop() {
68     // Read the states of the IR sensors
69     int sensorState1 = digitalRead(irSensorPin1);
70     int sensorState2 = digitalRead(irSensorPin2);
71     int sensorState3 = digitalRead(irSensorPin3);
72
73     // Update Firebase and LCD only if there is a change in sensor states
74     if (sensorState1 != previousSensorState1 || sensorState2 != previousSensorState2 || sensorState3 != previousSensorState3) {
75
76         // Build the status messages
77         String statusMessage1 = buildStatusMessage(sensorState1, 1);
78         String statusMessage2 = buildStatusMessage(sensorState2, 2);
79         String statusMessage3 = buildStatusMessage(sensorState3, 3);
80
81         // Update Firebase
82         updateFirebase("/parking/sensor1", sensorState1);
83         updateFirebase("/parking/sensor2", sensorState2);
84         updateFirebase("/parking/sensor3", sensorState3);
85
86         // Log parking history
87         logParkingHistory(sensorState1, sensorState2, sensorState3);
88
89         // Display the status messages on the LCD
90         displayStatusMessages(statusMessage1, statusMessage2, statusMessage3);
91
92         // Send notifications for available slots
93         sendNotifications(sensorState1, sensorState2, sensorState3);
94     }

```

Shows the sources code for the infrared sensor using Arduino IDE

APPENDICES B

ii. Source code of the Android Studio



```

1: MainActivity.java
2:
3: public class MainActivity extends AppCompatActivity {
4:     protected void onCreate(Bundle savedInstanceState) {
5:
6:         // TODO: Add your own initialization here
7:
8:         // TODO: Add your own initialization here
9:
10:    }
11:
12:    @Override
13:    public void onClick(View v) { showParkingDetail( "S3", "Gate C1, Near Howling"); }
14:
15:    }
16:
17:    private void updateParkingStatus(DataSnapshot snapshot) { //usage
18:        int sensor1 = snapshot.child( "sensor1").getValue(Integer.class);
19:        int sensor2 = snapshot.child( "sensor2").getValue(Integer.class);
20:        int sensor3 = snapshot.child( "sensor3").getValue(Integer.class);
21:
22:        tvA1.setBackgroundResource(sensor1 == 0 ? android.R.color.holo_red_light : android.R.color.holo_green_light);
23:        tvA2.setBackgroundResource(sensor2 == 0 ? android.R.color.holo_red_light : android.R.color.holo_green_light);
24:        tvA3.setBackgroundResource(sensor3 == 0 ? android.R.color.holo_red_light : android.R.color.holo_green_light);
25:    }
26:
27:    private void showParkingDetail(String slot, String details) { //usage
28:        AlertDialog.Builder builder = new AlertDialog.Builder( context.this);
29:        builder.setTitle(slot)
30:        .setMessage(details)
31:        .setPositiveButton( "OK", new DialogInterface.OnClickListener() {
32:            @Override
33:            public void onClick(DialogInterface dialog, int which) { dialog.dismiss(); }
34:        });
35:        builder.show();
36:    }
37:
38:    }
39:
40:    }

```

Source code to send the real-time information of parking spaces by using Android Studio