

Shuttle IoT: UTAR Bus Management System

By

CHIN PEI SHAN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION SYSTEMS (HONOURS)

INFORMATION SYSTEMS ENGINEERING

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2024

REPORT STATUS DECLARATION FORM

Title: Shuttle IoT: UTAR Bus Management System

Academic Session: June 2024

I CHIN PEI SHAN

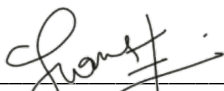
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in


Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



(Supervisor's signature)

Address:

No, 20 Laluan Bunga Kenanga 2,
Taman Bunga Kenanga,
31000 Batu Gajah, Perak.

Aun Yichiet

Supervisor's name

Date: 11/9/2024

Date: 12/09/2024

Universiti Tunku Abdul Rahman			
Form Title : Sample of Submission Sheet for FYP/Dissertation/Thesis			
Form Number: FM-IAD-004	Rev No.: 0	Effective Date: 21 JUNE 2011	Page No.: 1 of 1

FACULTY/INSTITUTE* OF INFORMATION COMMUNICATION AND TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

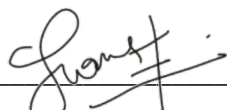
Date: 11/9/2024

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that Chin Pei Shan (ID No: 010807-08-0072) has completed this final year project/ dissertation/ thesis* entitled "Shuttle IoT: UTAR Bus Management System" under the supervision of Dr Aun Yichiet (Supervisor) from the Department of Computer and Communication Technology, Faculty of Information Communication and Technology.

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,




 (Chin Pei Shan)

*Delete whichever not applicable

DECLARATION OF ORIGINALITY

I declare that this report entitled “**Shuttle IoT: UTAR Bus Management System**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature :  _____

Name : Chin Pei Shan

Date : 11 Sep 2024

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere appreciation to my FYP supervisor, Dr Aun Yi Chiet who has given me this chance to involve in an IoT project. Dr Aun had given me a lot of suggestion to improve my hardware and software. A million thanks to you.

Not only that, I also like to express my sincere thanks to my parents for their supports and cooperative as well as continuous encouragement throughout the course.

ABSTRACT

Addressing the inefficiencies of bus wait times for UTAR students, this project introduces a real-time bus location tracking system with the function of calculate number of remaining seats, integrating IoT technologies within a Rapid Application Development (RAD) framework. The prototype includes a Raspberry Pi 4 paired with a Neo-M6 GPS HAT with Antenna and 5MP camera module. The GPS module utilizes satellite communication channel to pinpoint geographical coordinates. And the camera module utilizes WebSocket to stream video to AWS EC2 for processing. These coordinates and processing result stored within Firebase's Realtime Database for efficient data handling and retrieval. The system leverages Google Maps API, acquired through Google Cloud services, to render the positional data onto a map within a custom-built Flutter application. This setup ensures that the Raspberry Pi 4 on the bus provides real-time updates on both its location and the number of remaining seats. Consequently, UTAR students are empowered to monitor bus locations via the application in real-time, significantly optimizing their time management by diminishing periods spent waiting for transportation.

TABLE OF CONTENTS

TITLE PAGE	i
REPORT STATUS DECLARATION FORM	ii
SUBMISSION SHEET	iii
DECLARATION OF ORIGINALITY	iv
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii-ix
LIST OF FIGURES	x
LIST OF TABLES	xi
LIST OF ABBREVIATIONS	xii
CHAPTER 1 INTRODUCTION	1-4
1.1 Problem Statement and Motivation	1
1.2 Motivation	2
1.3 Research Objectives	2-3
1.4 Project Scope and Direction	3
1.5 Contributions	4
CHAPTER 2 LITERATURE REVIEW	5-13
2.1 Moovit	5-7
2.1.1 Function, Feature and Strength	5-7
2.1.2 Limitation	7
2.2 Pulse	8-10
2.2.1 Function, Feature and Strength	8-9
2.2.2 Limitation	9-10
2.3 Google Maps	11-12
2.3.1 Function, Feature and Strength	11-12
2.3.2 Limitation	12
2.4 Summary Table of Review Existing System	13

CHAPTER 3 PROPOSED METHOD/APPROACH	14-26
3.1 System Requirement	14-15
3.1.1 Hardware	14-15
3.2 System Design Diagram	16-17
3.2.1 System Architecture Diagram	16-17
3.2.2 AWS Architecture Diagram	18
3.2.3 Connection Diagram	19
3.3 Deep Learning Model	20
3.3.1 YOLOv8 Model	20-22
3.4 Mounting Locations of Hardware	22-23
3.5 Implementation Issues and Challenges	23-24
3.6 Timeline	25-26
CHAPTER 4 PRELIMINARY WORK	27-34
4.1 Setting Up	27-28
4.1.1 Hardware	27-28
4.1.2 Software	28
4.2 Whole Prototype	29
4.3 Accuracy of YOLO	30
4.4 Timeliness of YOLO Detection and Counter Update	30
4.5 Bus Tracked in Google Map	31-32
4.6 Passenger Detection in Bus	33-34
CHAPTER 5 SYSTEM IMPLEMENTATION	35-37
5.1 System Initialization	35-36
5.2 Feasibility of Bus Management System	37
CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION	38-52
6.1 Performance Evaluation	38-41
6.2 Accuracy Assessment	41-45
6.3 System Integration Testing	46-49
6.4 Case Study	49-50
6.5 Results on Questionnaire	50-52

CHAPTER 7 CONCLUSION **53**

REFERENCES **54**

APPENDIX A

Source Code and Screenshots

Survey

Weekly Report

Poster

PLAGIARISM CHECK RESULT

CHECK LISTS

LIST OF FIGURES

Figure Number	Title	Page
Figure 2-1	Logo of Moovit	5
Figure 2-2	Suggested routes	6
Figure 2-3	Arrival Updates	7
Figure 2-4	Logo of Pulse	8
Figure 2-5	Interface after searching	8
Figure 2-6	Interval time of train	9
Figure 2-7	Fail to search destination	10
Figure 2-8	Logo of Google Maps	11
Figure 3-1	Neo-M6 GPS HAT with Antenna	15
Figure 3-2	Camera Module 5MP	15
Figure 3-3	Use Case Diagram	16
Figure 3-4	System Component Diagram	17
Figure 3-5	AWS Architecture Diagram	18
Figure 3-6	Connection diagram of RPi 4 GPS Module, and Camera Module	19
Figure 3-7	Circuit Diagram	19
Figure 3-8	YOLOv8 architecture	20
Figure 3-9	Detail YOLOv8 architecture	20
Figure 3-10	Object detection using YOLOv8	21
Figure 3-11	Raspberry Pi is mounted at the upper right of the driver's seat	22
Figure 3-12	Raspberry Pi is mounted at the top of the bus	23
Figure 3-13	Gantt Chart for FYP 1	25
Figure 3-14	Gantt Chart for FYP 2	26
Figure 4-1	Whole prototype of Raspberry Pi	29
Figure 4-2	Bus tracked at bus stop 1	31
Figure 4-3	Bus tracked at bus stop 2	31
Figure 4-4	Bus tracked at bus stop 3	32

Figure 4-5	Passenger detection when getting on bus part 1	33
Figure 4-6	Passenger detection when getting on bus part 2	33
Figure 4-7	Passenger detection when getting off bus part 1	34
Figure 4-8	Passenger detection when getting off bus part 2	34
Figure 6-1	AWS EC2 instance	38
Figure 6-2	Public IP in client_web_socket.py	38
Figure 6-3	Run calculate.py	39
Figure 6-4	Frames sent to EC2	39
Figure 6-5	Output of processing	40
Figure 6-6	Updated seats number	41
Figure 6-7	Output of gps_send.py	41
Figure 6-8	Key in longitude and latitude received to search for location	42
Figure 6-9	Current location display on Android phone	42
Figure 6-10	Original video part 1	43
Figure 6-11	Original video part 2	44
Figure 6-12	Backend of video processing	45
Figure 6-13	Result of available seats	45
Figure 6-14	Run gps_send.py and client_web_socket.py	46
Figure 6-15	Output of calculate.py	47
Figure 6-16	Information on Firebase	48
Figure 6-17	Location on flutter app	48
Figure 6-18	Pie chart before students used the app	50
Figure 6-19	Pie chart after students used the app	51
Figure 6-20	Pie chart of students saving time	51
Figure A-1	Connection of Pi 4 with Neo-M6 GPS HAT with Antenna	A-1
Figure A-2	Python code to read only longitude and latitude	A-1
Figure A-3	Longitude and latitude data	A-1
Figure A-4	Python code to send data to firebase	A-2
Figure A-5	Data sent into Firebase Realtime Database	A-2
Figure A-6	Firebase Realtime Database	A-3
Figure A-7	Flutter app show updated location	A-3
Figure A-8	Compare longitude and latitude on Google Map	A-4

Figure A-9	Install GPS module on car using power bank as the power supply	A-4
Figure A-10	Location updated when new data sent into database	A-5
Figure A-11	AWS EC2 instance	A-5
Figure A-12	Source code of tracker.py	A-10
Figure A-13	Survey part 1	A-11
Figure A-14	Survey part 2	A-12

LIST OF TABLES

Table Number	Title	Page
Table 2-1	Summary of Review System	13
Table 3-1	Specifications of laptop	14
Table 3-2	Specifications of Raspberry Pi	14
Table 4-1	Function of all components	29
Table 4-2	Accuracy of YOLOv8	30
Table 4-3	Timeliness of YOLOv8 detection and counter update	30
Table 6-1	Testing Parameters	49
Table 6-2	Accuracy in Bus 1, 2, 3	49-50
Table A-1	Source Code of client_web_socket.py	A-5-A-6
Table A-2	Source code of calculate.py	A-6-A-10

LIST OF ABBREVIATIONS

<i>API</i>	Application Programming Interface
<i>GND</i>	Ground
<i>GPS</i>	Global Positioning System
<i>LED</i>	Light-emitting Diode
<i>OS</i>	Operating System
<i>RPi</i>	Raspberry Pi

CHAPTER 1

Introduction

In this chapter, the introduction, problem statement, motivation, and objective as well as contributions to develop “Shuttle IoT: UTAR Bus Management System” will be presented.

1.1 Problem Statement

In this section, three problem statements related to the current situation that UTAR students face while waiting for a bus will be discussed.

1. Inconsistent punctuality of transport arrivals.

UTAR students always faced issues about the inconsistent punctuality of bus arrivals. It is because UTAR will predefine the schedules of each bus and send the schedules to students through UTAR email. The students can only look at the predetermined schedules to know the estimated arrival time of the targeted transport. However, the transports may arrive earlier or later than predefined schedules due to unforeseen circumstances. It may cause the students to miss the bus even if they arrive at the station on time.

2. Lack of real-time tracking applications

UTAR do not have a real-time tracking application for tracking location of buses. The students do not know the current bus location and cannot estimate the arrival times of bus at bus stations, hence they can only wait until a bus arrives.

3. Lack of information on seat availability

UTAR do not have a real-time counting system for counting number of passengers in and out of the buses. As a result, it does not provide information about the availability of seats. Students often face difficulties boarding the bus because the number of students waiting at the bus station exceeds the remaining seats on the bus.

1.2 Motivation

The aim of the thesis is to propose a new application for tracking buses and calculate the remaining seats of bus using GPS sensor and camera module to address the current problems faced by students in their daily university lives. Currently, the public transport services of UTAR do not consist of an application to track the transports, they are only based on the predefined schedules of the bus, which may not be efficient enough for the students to know the route and arrival time of the transports.

The predefined and updated schedules that were planned by UTAR management will be saved in PDF files and sent to the email addresses of students. The students can only view the schedules after downloading the file. Students who do not have the habit of checking emails regularly will miss important information about UTAR public transport. Therefore, the new tracking application was developed to solve inconsistent punctuality, unclear schedules, and a lack of real-time information, which led to inconvenience and time waste.

This new application has integrated IoT into buses to enhance the overall transportation experience for students. By implementing a tracking application, it seeks to provide students with real-time information about arrival times, locations, and seat availability of buses. So, the students can plan their time efficiently.

In conclusion, the motivation stems from a desire to create a more efficient, user-friendly, and transparent transportation application that aligns with the busy lifestyles of the students. The integration of IoT technology can streamline transportation and reduce waiting times. Ultimately, the goal is to make student transportation more accessible, convenient, and timesaving.

1.3 Research Objectives

The aim of this project is to solve the problems faced by UTAR students.

- 1. To develop a location tracking module for UTAR bus and shuttle using GPS-on-Pi.**

This objective aims to track real-time location of bus and shuttle by locate a GPS sensor. The GPS sensor can receive longitude and latitude from satellites and send it back to Raspberry Pi 4 to further process.

2. To integrate GPS coordinate on Google Map for intuitive tracking.

An application is created to display current location of buses on Google Map by using Google Map Api. Students can install the app and monitor the location of buses.

3. To develop a crowd counting method to count the number of passengers in real time using Yolov8.

The camera module will attach on the Raspberry Pi to collect the information about number of passengers in and out of the bus to calculate the remaining seats in the bus.

1.4 Project Scope and Direction

At the end of the project, a UTAR Bus Management System is expected to be delivered. In this system, there will be a new monitoring model with its application to track the real-time location of UTAR buses. With this new model with application, it has two specific functions which are real-time transport location tracking and calculating the number of seat availability, which were not provided by UTAR previously. To ensure the functions working properly, the model will apply Raspberry Pi 4 which is also known as RPi 4 to analyze the data collected from Neo-M6 GPS HAT with Antenna module to collect longitude and latitude of the location. Also, a camera module will be used to detect the number of passengers entering and exiting the bus.

Since UTAR does not have a proper management system for public transport, this project involves using a RPi 4 connected with a Neo-M6 GPS HAT with an Antenna module as well as a camera module to improve this situation. The RPi 4 is suitable for this project as it has its own operating systems such as Debian, Raspbian and Linux [1]. For innovative, several programming languages such as C++, Python and Scratch can be chosen to implement in the OS so it is flexible and different kinds of functions can be added easily. For instance, there are various types of sensors can be used for RPi4 to track the bus location, but in this scenario, Neo-M6 GPS HAT with an

Antenna module is the best to implement in this bus management system since it can receive more accurate GPS data such as longitude and latitude of the location.

1.5 Contributions

The project is focused on helping UTAR students by using a tracking application on UTAR buses for a better time management. The primary contribution of the project is to provide students with real-time information about the transport location, arrival time, and number of seat availability of UTAR buses. The implementation of the tracking application contributes to the efficiency of transportation schedules. Furthermore, the improvement of the user-friendly interface of the application also ensures the students access the information more easily and clearly by looking at the interface of the application.

By integrating GPS, the system can adapt to real-time changes or updates and ensure timely arrivals. This can ease the students in boarding UTAR buses on time. The students who do not own a private transport and who prefer not to drive themselves can take bus from hostel area to campus. Although there is a predefined time schedule for the UTAR buses, the arrival of buses may not always be exactly punctual, which can lead to wasted waiting time for students. Hence, this system can address this issue since the students can open the application to check the current location of targeted transport whether near the bus stop or just setting off. This piece of real-time information can let the students in managing their time effectively instead of wasting a lot of time at station.

Additionally, the integration of the camera module in bus can calculate the real-time number of passengers in the bus and the remaining seats. This real-time updated information helps students plan their transportation. They will no longer have to deal with full buses while waiting.

In conclusion, the project stands as a meaningful contribution to the field by providing streamlined public transport services in UTAR, predictable arrival times, and transparency of public transport information. It can help the students make better decisions regarding their transportation choices.

CHAPTER 2

Literature Reviews

2.1 Moovit



Figure 2-1 Logo of Moovit

In 2012, Moovit was released as a free application and identified as a MaaS provider. MaaS stands for “Mobility as a Service,” which will provide a single and on-demand mobility service to users by integrating various kinds of transportation services. Moovit provide a wide range of transportation options, such as bus, MRT, LRT, walking, or a mix of these to satisfy the users’ needs. Main purpose of the app is to calculate the shortest path and most convenient way for the user to get to their destination by using any mode of transportation. According to the Moovit official website, this app has helped over 1.7 billion riders by using 45 languages in 3500 cities across 112 countries. The Moovit app was recognized by Google in 2016 as the “Best Local App” and as one of the Best Apps in the App Store in 2017, as well as a finalist in the 2018 “Build for Billions” category [2].

2.1.1 Function, Feature and Strength

1. Provide the most efficient routes

Moovit provides route planning to the users by using method of crowdsourced data[3]. It means that the app collects data from all users. The collected data is analyzed and then used to generate multiple suggestion routes for the users. For instance, users can set their origin and destination in the app before departing to

find the most efficient route. The approximate time needed and modes of transportation to arrive at the destination for every suggested route are displayed on the app. By considering various factors, users can choose the appropriate route to save their time.

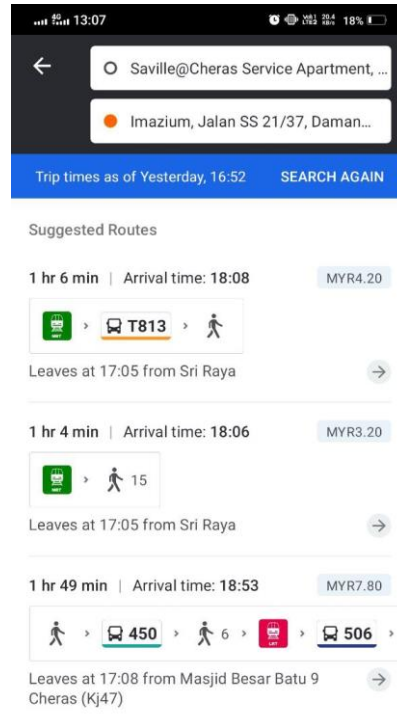


Figure 2-2 Suggested routes

2. Live location

Live location enables the users to stay updated with the location of transport. It can track the transport real-time location on the map. Therefore, this function can reduce waiting times at the station since users can predict arrival times based on the real-time location of the transport they are intending to board.

3. Arrival updates

Users can receive live notifications on transport arrival times. The app will display the arrival time of each route to ensure the users plan their routes in advance. This function can also calculate the distance from the current location to the stop or station to keep them in the loop. For instance, users can click the button with a bell icon to activate the arrival update function. Next, choose the line that needs to be updated, and then users will receive notifications for that line [4].

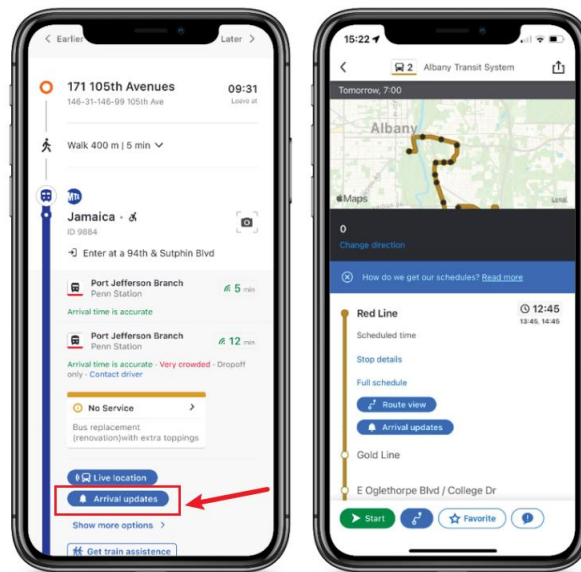


Figure 2-3 Arrival updates

2.1.2 Limitation

1. Exclusive to premium users

Several functions in Moovit are now exclusively for premium users only after the recent update. Functions such as live location is currently unavailable for normal users. The users need to subscribe as a premium user to unlock this live location function.

2. Lack of accuracy of the route and transport information

Moovit gather the transport information by crowdsourcing method led to inaccurate of the information. Crowdsourcing is define as collect the data from users through the Internet[5]. Hence, Moovit do not provide real-time updates for users causing them to miss the transport and end up wasting time.

2.2 Pulse



Figure 2-4 Logo of Pulse

Pulse is an application launched by Prasarana Malaysia Berhad. Prasarana is an entity that wholly owned by government to operates Malaysia rail services that are LRT, MRT and KL Monorail. Not only that, Prasarana also operates the stage bus service in Kuala Lumpur, Selangor, Penang and Pahang [6]. Pulse which is stand for Planning Your Lifestyle Efficiently is currently available for Rapid rail and bus services in KL, Penang and Kuantan from 2021 [7].

2.2.1 Function, Feature and Strength

1. Provide real time location

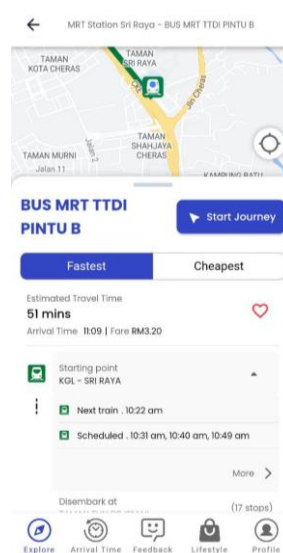


Figure 2-5 Interface after searching

Pulse enables user to perform action of start journey by clicking on the button at the right. This function aims to synchronize the user's location with the transport by providing real-time location updates even when boarding inside the

transport. Hence, the user who are no sense in direction will know their current location.

2. Fare for entire route

Pulse also will calculate the total fare for entire route after user set the origin and destination. This function can make sure the users prepare enough money before boarding transportation.

3. Interval arrival time of transport

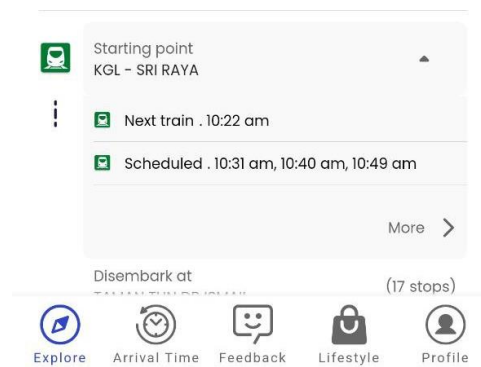


Figure 2-6 Interval time of train

After user search for the route, the screen will display the methods and transport mode to reach the destination. Each transport mode will display the interval arrival time to let the user plan their route or journey more efficiently.

2.2.2 Limitation

1. Unable to search location

The search box contains a placeholder describing that users can search by destination, bus number, and station. However, users often encounter difficulties when attempting to search for a specific destination. For instance, a user aiming to reach Imazium in Petaling Jaya from Saville Condo in Cheras, but it receives no results for this route. Successful searches are limited to inputting MRT stations or bus numbers only. This limitation causes inconvenience for users.

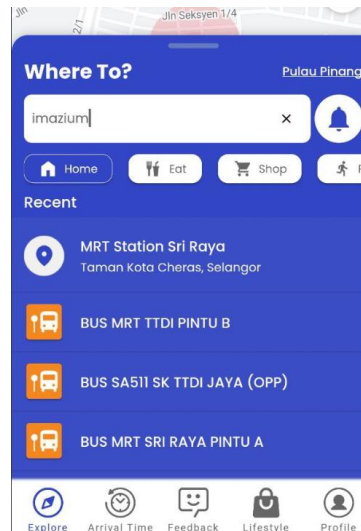


Figure 2-7 Fail to search destination

2. Bad user interface

The user interface of Pulse is bad as it takes much time for users to understand how to use the application. The search box does not have a search button for users to click, it requires users to press “Enter” on the phone keyboard for a search.

3. Unsatisfactory app performance

The app performance of Pulse is also unsatisfactory for users. The primary reason is due to prolonged response times when users interacting with the app. Users always experience delays when clicking on the search box or buttons. It causes the users to wait for a few seconds before they can input or get information.

2.3 Google Maps



Figure 2-8 Logo of Google Maps

Google Maps is a web service that launched on Feb 8, 2005 by Google company to provides longitude and latitude of a location by integrating GPS technology. Beyond traditional road maps, Google Maps provides satellite and aerial perspectives of numerous locations. Google Maps also provides street views featuring that captured from vehicles in some cities [8]. Not only that, Google Maps is also a built-in application on Android devices designed to enhance user accessibility with over 1 billion people worldwide utilizing Google Maps every month in 2020 [9].

2.3.1 Function, Feature and Strength

1. Booking online

Google Maps provides a function to ensure users make a booking through the app. For instance, the users can check the availability of hotels nearby the current location, then the screen will show several options of website to book the hotels. The users can click one of the options such as Agoda website then it will navigate to the Agoda to ensure the users look at the details of hotel and make a booking. This function makes the users can search and book the hotels conveniently.

2. Collect real-time traffic information through surveys

Google Maps will provide some questions for users currently on board to answer. The questions include accessibility, temperature, security onboard and women's section allow answering from all the public. Google Maps will collect all of these responses to help other passengers on their journey.

3. Provide best route

Google Maps also suggest several routes to the users especially for the users who plan to reach destination by public transport. It will calculate the estimated time and transport modes to reach the destination. Hence the users have multiple options to choose which is most appropriate and convenience.

2.3.2 Limitations

1. Do not include fare calculations

Google Maps provides the suggested route and transport mode to users but it do not include fare calculations. The users do not know how much it will spend on this journey or route. It is very inconvenient for them to check at another website or in front of station.

2. Lack of live location

Google Maps also do not include live location to show when the public transport will come and where the transport is located. It causes the users only can waiting cluelessly at the station until the next transport came.

3. Do not provide transit map

This app also does not provide a full transit map. Transit map is a map that contains overview of entire transit network or lines. Some users hope to view the complete network of MRT and LRT lines in a full view instead of single view. Not only that, but the users may also become difficult to identify alternative routes or interchanges lines without a full transit map. Hence, it has made them less convenient to plan routes efficiently.

2.4 Summary Table of Review Existing System

Table 2-1 Summary of Review System

	Moovit	Pulse	Google Maps
Function, Feature and Strength			
1. Provide suggested routes	/	/	/
2. Live location	/	/	
3. Arrival updates	/		
4. Fare for entire route	/	/	
5. Provide internal arrival time of transport		/	
6. Booking online			/
7. Collect data through survey			/
Limitation			
1. Exclusive premium users	/		
2. Lack of accuracy of the information	/		
3. Unsynchronized live location	/		
4. Unable to search specific location		/	
5. Bad user interface		/	
6. Unsatisfactory app performance		/	
7. Do not provide transit map			/

CHAPTER 3

Proposed Method/Approach

The project developed using methodology of Rapid Application Development which were categorized into four phases in the development, which were project planning, user design, development, and implementation.

3.1 System Requirement

3.1.1 Hardware

The hardware involved in developing the proposed application are a laptop, Raspberry Pi 4, Neo-M6 GPS HAT with Antenna, and camera module.

1. Laptop

Table 3-1 Specifications of laptop

Description	Specifications
Model	HP Laptop 15s-du3xxx
Processor	11 th Gen Intel Core i3
Operating System	Windows 11
Graphic	Intel UHD Graphics
Memory	8GB RAM
Storage	512GB of SSD

2. Raspberry Pi

Table 3-2 Specifications of Raspberry Pi

Description	Specifications
Model	Raspberry Pi 4
Operating System	Debian with Raspberry Pi Desktop
Storage	SD Card with 16GB

3. Neo-M6 GPS HAT with Antenna



Figure 3-1 Neo-M6 GPS HAT with Antenna

Neo-M6 GPS HAT with antenna is a GPS module which use NMEA format to display data received from satellites. This GPS module is equipped with the u-Blox Neo-6M chip. Additionally, 5v, TX, RX and Ground pins of GPS module must connected with 5v, RX, TX and Ground pins of Raspberry Pi respectively by using female-to-female jumper wire. Make sure the RX pin of GPS module is connected with TX pin of RPi and TX pin of GPS module is connected with RX pin of RPi.

4. Camera Module 5MP

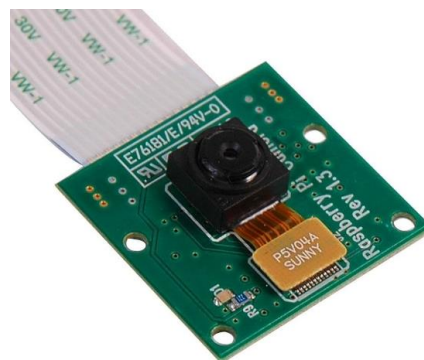


Figure 3-2 Camera Module 5MP

This camera module is compatible with Raspberry Pi Zero and all latest versions. It can capture pictures and videos with a maximum resolution of 1080p. The module features a 5-megapixel sensor and supports various imaging modes which are suitable for a wide range of applications. Its compact size and versatility make it an excellent choice for this project that requiring high-quality imaging.

3.2 System Design Diagram

3.2.1 System Architecture Diagram

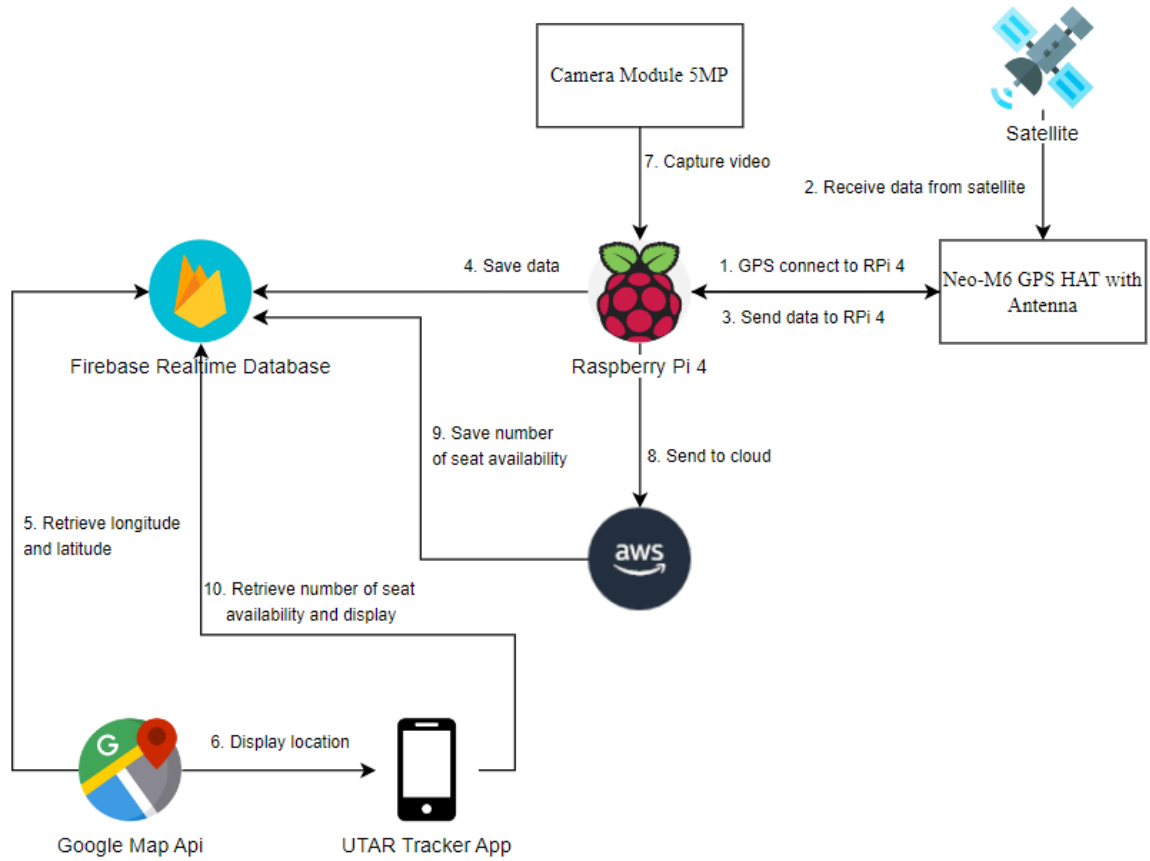


Figure 3-3 Use Case Diagram

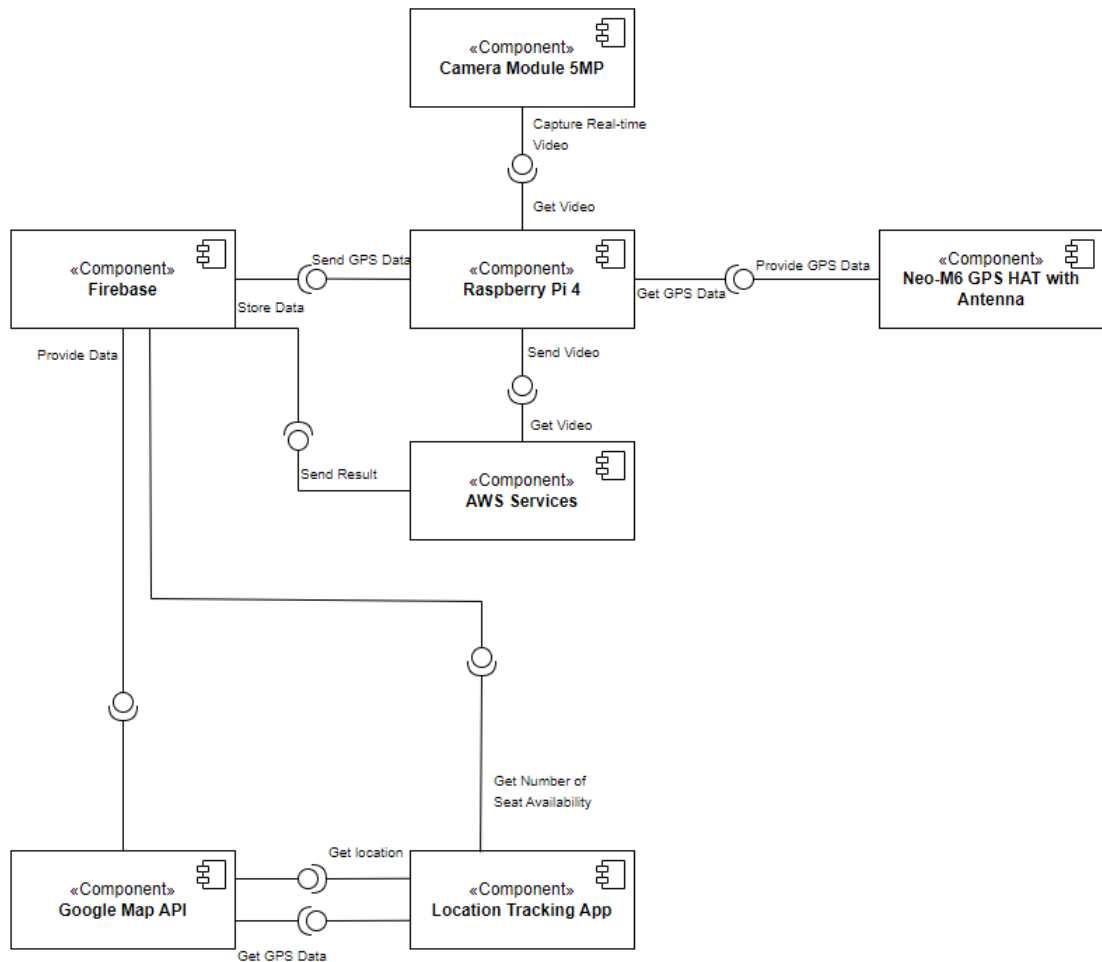


Figure 3-4 System Component Diagram

Firstly, Neo-M6 GPS HAT with Antenna module is connected to RPi 4. The GPS module can start to retrieve data from satellites. After that, the data received are send to RPi 4 to further process. A python script is coded in RPi4 to send the longitude and latitude data to the Firebase Realtime Database through API. Then, register for a Google Cloud account to generate a Google Maps API key and integrate it into Flutter app which is called UTAR Tracker App. The app will retrieve data from Firebase and display the marker locations on the map.

At the same time, the Raspberry Pi Camera Module 5MP which attached on the RPI 4 will capture the video and send it to the RPI 4. The video is then streamed to an AWS EC2 instance via WebSocket for calculating the number of passengers entering and exiting the bus. The final result is then uploaded to the Firebase Realtime Database and displayed on the app.

3.2.2 AWS Architecture Diagram

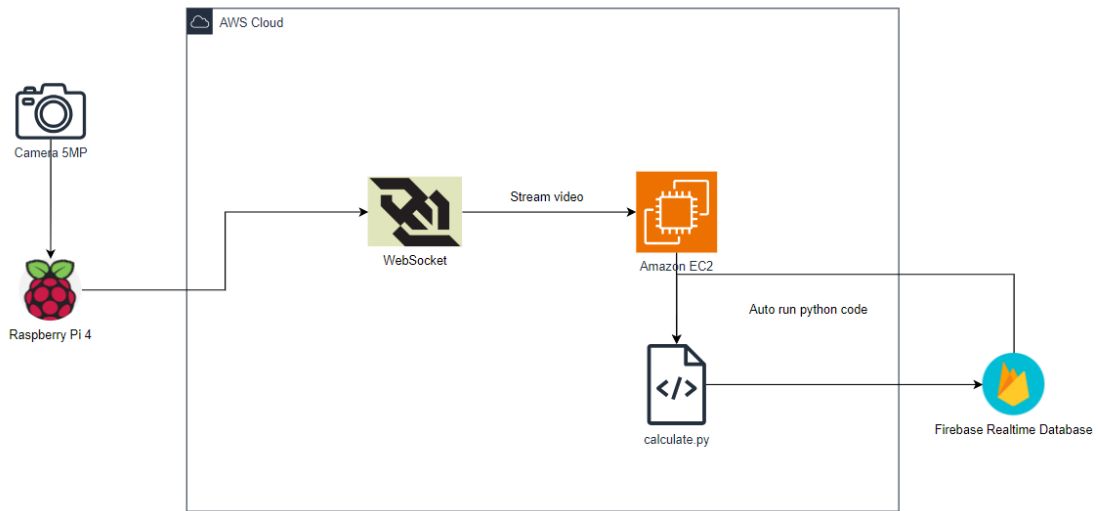


Figure 3-5 AWS Architecture Diagram

The calculation of the number of remaining seats is done by AWS EC2 instance due to its high processing power and fast speed of analysis. Firstly, the Raspberry Pi 4 continuously streams video captured by the 5MP camera module to the EC2 instance via WebSocket. WebSocket is a protocol that enables full-duplex communication channels over a single TCP connection which allow for real-time data transfer between the Raspberry Pi and the EC2 instance.

Once the video frames are received by the EC2 instance, the calculate.py script processes each frame in real-time. The results of these calculations are updated in the Firebase Realtime Database. Subsequent frames use the updated data for ongoing calculations, ensuring that the seat availability information is current and accurate.

3.2.3 Connection Diagram

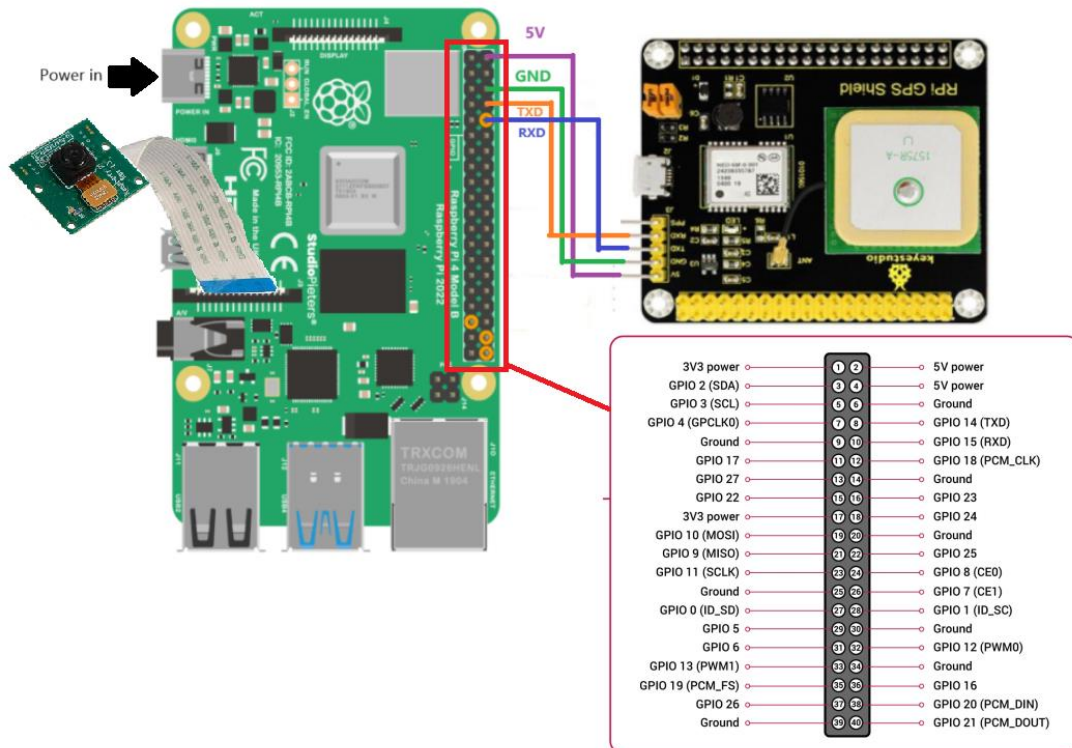


Figure 3-6 Connection diagram of RPi 4, GPS Module, and Camera Module

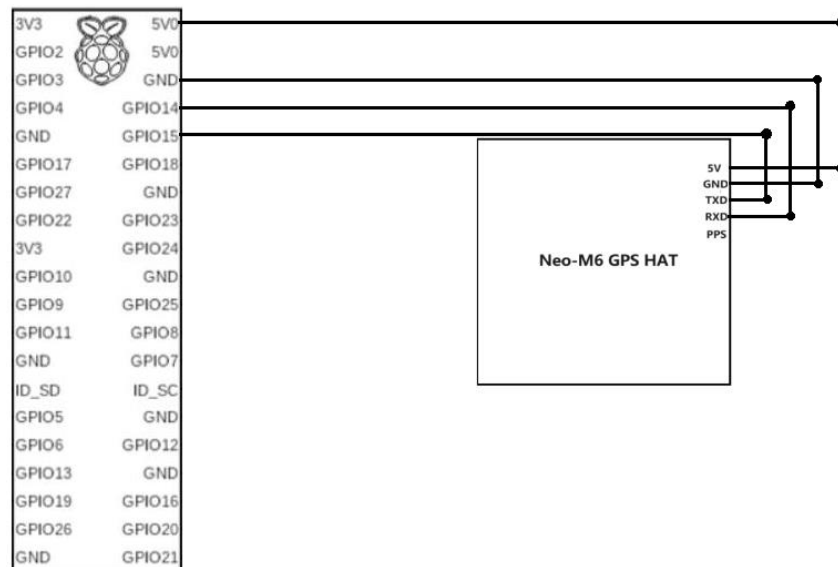


Figure 3-7 Circuit Diagram

In this system, a Raspberry Pi 4 and a Neo-M6 GPS HAT with Antenna module are needed. RPi 4 is used to provide power supply to the GPS module and process the data.

Additionally, the GPS module is used to receive the data from satellites and send to RPi 4 via TX pin from GPS module. In order to connect these two components, four female-to-female jumper wire are needed. Based on the Figure 3.3 above, purple jumper wire is connected from 5V pin of RPi 4 to 5V pin of GPS module; green jumper wire is connected from GND pin of RPi 4 to GND pin of GPS module; orange jumper wire is connected from TXD pin of RPi 4 to RXD pin of GPS module; blue jumper wire is connected from RXD pin of RPi 4 to TXD pin of GPS module. After connection successfully, the red LED will start blinking after a few minutes.

3.3 Deep Learning Model

3.3.1 YOLOv8 Model

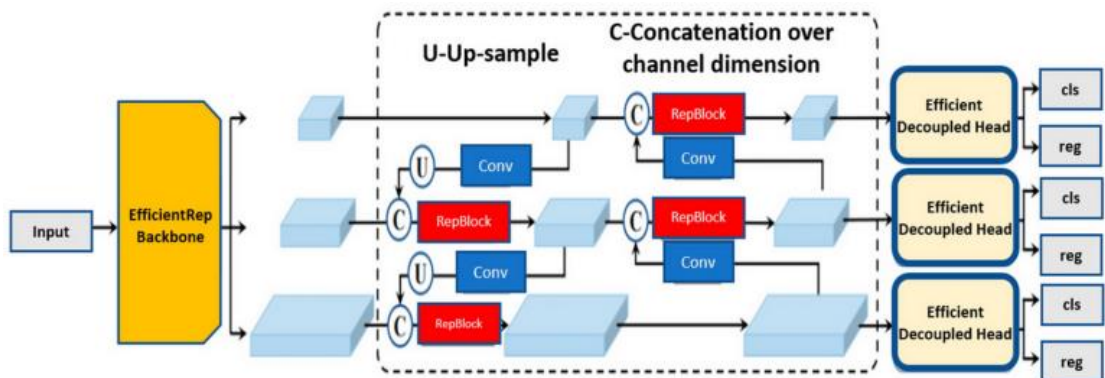


Figure 3-8 YOLOv8 architecture

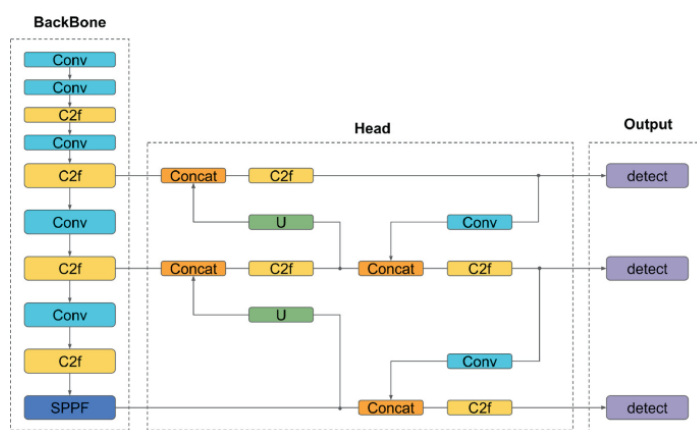


Figure 3-9 Detail YOLOv8 architecture

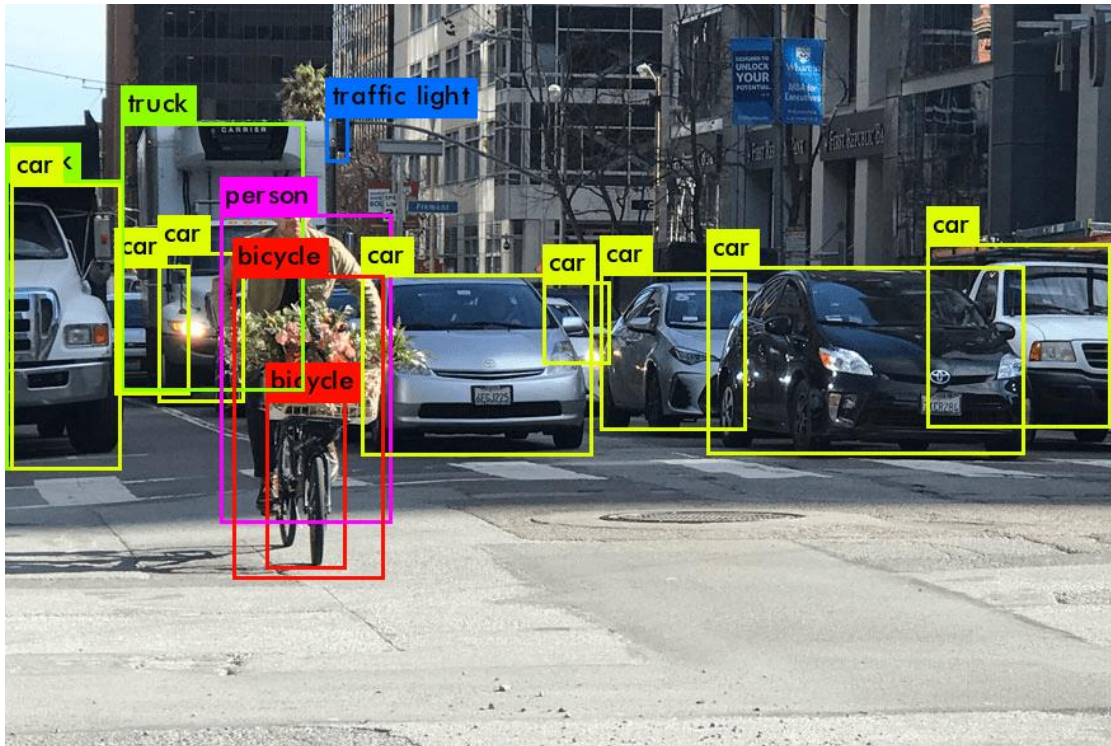


Figure 3-10 Object detection using YOLOv8

YOLO which also named as You Only Look Once is a deep learning model to detect objects and classify from an input image. YOLO consists of several versions which are from version one (v1) to version ten (v10) until now. In this project, YOLOv8 which is introduced in year 2023 is used due to its accuracy and real-time detection as well as high performance. By comparing with previous versions, YOLOv8 has improved its architecture by incorporating modern innovations in neural network design to improve both speed and accuracy. According to the architecture shown above (Figure 3-8), the input can be either a video or an image. If a video is used as input, the model will detect objects frame by frame, it is because a video is essentially a sequence of frames or images that combine together. At first, the image will pass to EfficientRep Backbone (yellow block) to extract basic features of the image such as shapes, lines, edges, textures and so on, then the network will detect more complex features when the image passes through deeper layers of the backbone. Next, the image passes through the Upsampling stage (blue blocks) to detect and focus on smaller details. For instance, if a person is standing far from the camera, the person will show smaller in size in the image. Hence, Upsampling helps the model pay attention to this smaller object and prevent it from being overlooked. After that, “C” represents concatenation which helps to combine the Convolutional Layer or large and small objects to get a full picture.

CHAPTER 3

Convolutional Layer (dark blue blocks) is a layer that filters images to perform tasks such as edge detection, sharpening, and blurring to improve the quality of image. The image is then pass through the RepBlock (red block) to detect deeper features which are difficult to distinguish. For instance, it will help to differentiate between similar-looking objects to enhance accuracy of detection. After feature extraction, the model uses Efficient Decoupled Heads for classification (cls) and regression (reg). For classification, it predicts the type of each detected object such as a person, a dog or a doll. While regression on the other hand will draw a bounding box around each detected object (Figure 3-10).

3.4 Mounting Locations of Hardware



Figure 3-11 Raspberry Pi is mounted at the upper right of the driver's seat

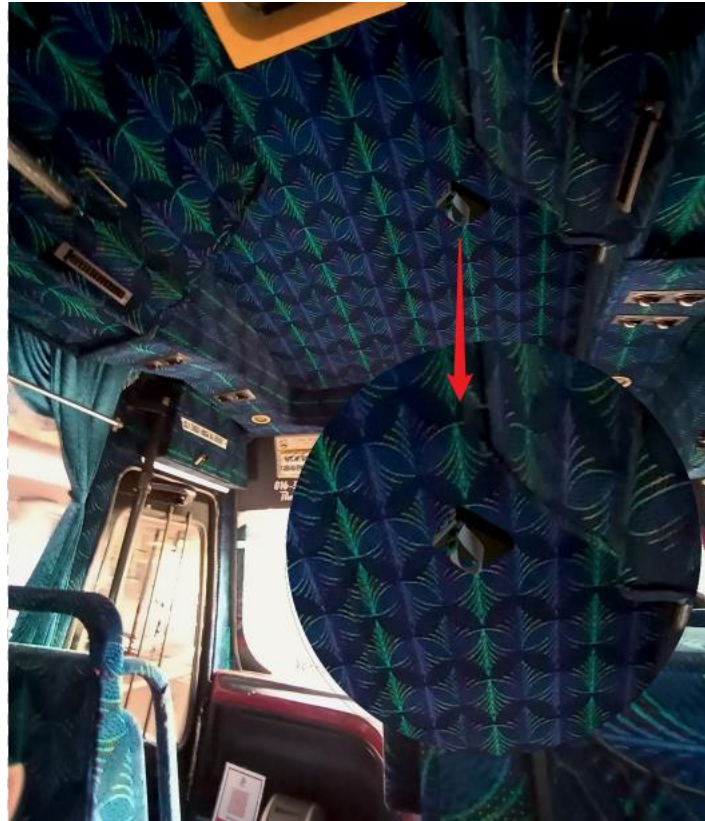


Figure 3-12 Raspberry Pi is mounted at the top of the bus

The Raspberry Pi can be placed in either of these two positions (Figure 3-11 and Figure 3-12), it is because Pi attached with a camera to capture the video of passenger getting on and off the bus. This video can use to calculate the number of available seats in the bus. Therefore, both positions provide a clear view of the passengers.

3.5 Implementation Issues and Challenges

The challenge occurred when connect Neo-M6 GPS HAT with Antenna to RPi 4 near the window. The red color LED did not blink regularly. It caused the GPS module failed to receive data from satellites. After doing some research, it stated that the new GPS module need put under open sky for around 15 minutes to get a fixed.

Besides, the data received from the GPS module and sent to the RPi 4 must be uploaded via Wi-Fi into Firebase Realtime Database. If the bus routes do not have

CHAPTER 3

network coverage, then the data in database cannot update and cause lack of accuracy in tracking.

Not only that, but the time also required to calculate the number of available seats using YOLOv8 on the RPI 4 is very long. It is because the GPU in RPI 4 do not support heavy processing. Therefore, the video captured by the camera module on the RPI 4 is stream to AWS EC2 instance for further calculations and then sent to Firebase Realtime Database.

CHAPTER 3

3.6 Timeline

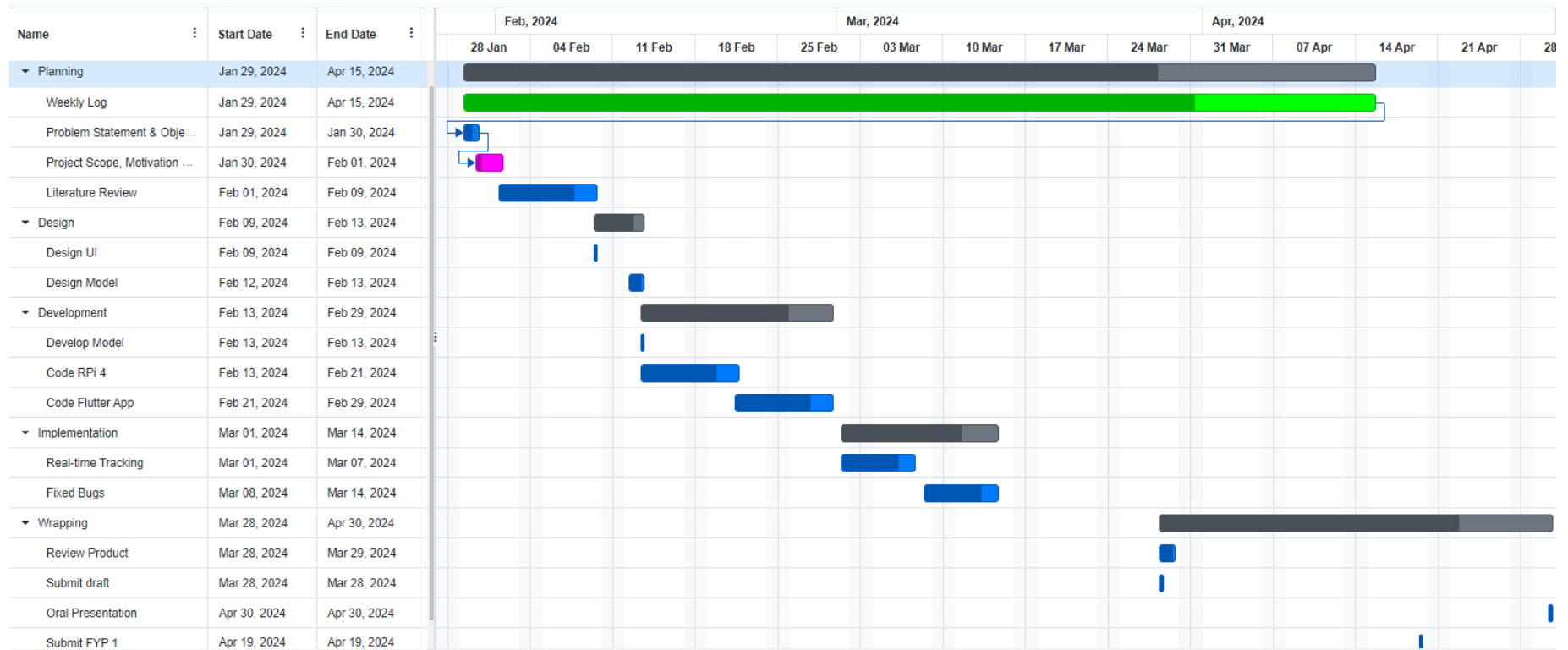


Figure 3-13 Gantt Chart for FYP 1

CHAPTER 3

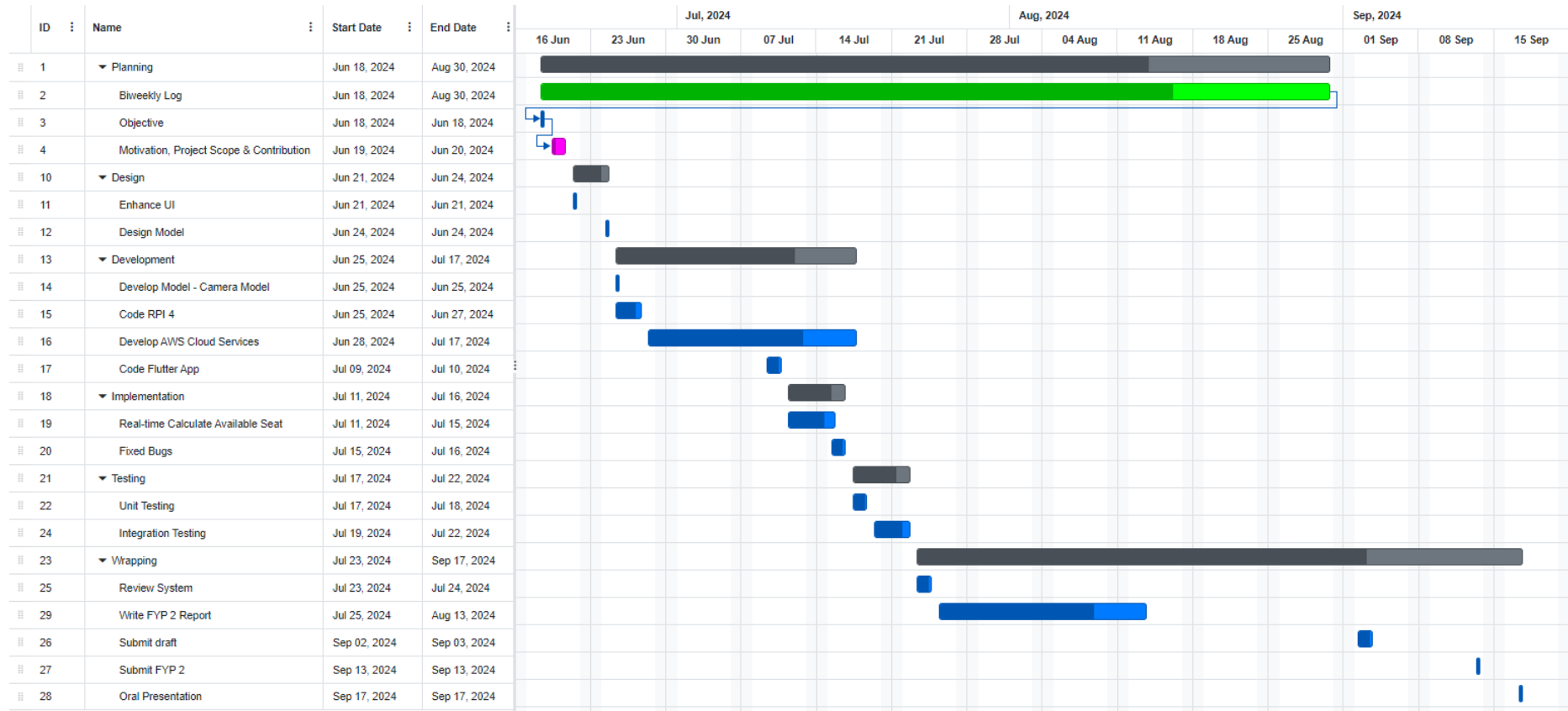


Figure 3-14 Gantt Chart for FYP 2

CHAPTER 4

Preliminary Work

The preliminary work of the UTAR Bus Management System includes setting up the entire model (both hardware and software) and initiating the system within the real situation.

4.1 Setting up

4.1.1 Hardware

For this bus management system, there are five hardware needed in the circuit setup:

1. RPi 4
2. female-to-female jumper wires
3. Neo-M6 GPS HAT with Antenna
4. Camera Module 5MP
5. Raspberry Pi power supply

Firstly, 4 jumper wires are utilized to make a connection between RPi4 and Neo-M6 GPS HAT with Antenna. One end of the first jumper wire is connected to the pin 2 with output of 5V on RPi4, while the other end of the jumper wire is connected to the pin header with the input of 5V on GPS module. Then, both GND pins on the RPi4 and GPS module are connected with another jumper wire. By these connections, the circuit is completed with 5V pins indicate the positive terminal of the power supply, while the GND pins describe the negative terminal of the power supply. For data connection, the TXD pin (pin 8 or GPIO 14) of RPi 4 is connected to the RXD pin of the GPS module, and vice versa, the RXD pin (pin 10 or GPIO 15) of RPi 4 is connected to the TXD pin of the GPS module.

Next, install the camera module in the camera slot near the HDMI port on RPi 4 by carefully lifting the plastic latch on the slot. Take the ribbon cable and insert it into the slot with the metal contacts on the cable facing towards the HDMI ports on the RPi

CHAPTER 4

4. After making sure the cable is fully inserted in the RPi 4, gently press the plastic latch back down to fix the position of cable. The other end of the ribbon cable is connected to the camera module with the metal contacts facing upwards. Finally, turn on the power after making sure that all the hardware has been installed correctly to prevent any damage to the components.

4.1.2 Software

For this bus management system, there are four software needed in the application setup:

1. Android Studio Hedgehog version 2023.1.1
2. Flutter
3. Firebase
4. AWS Cloud

Firstly, Android Studio, an IDE was installed in laptop with its function of code editing and application development. By using this IDE, a Flutter project was created as all users for platform such as iOS, Android, Web, MacOS, Linux and Window can be selected to use this application [10]. However, only iOS and Android users are considered because this project is focus on mobile application.

Secondly, create an AWS Cloud account to utilize the cloud services such as Amazon EC2. After all the functions in AWS Cloud services and Flutter application were coded, the Firebase Realtime Database was linked to the Flutter application so that the data collection can be retrieved.

4.2 Whole Prototype

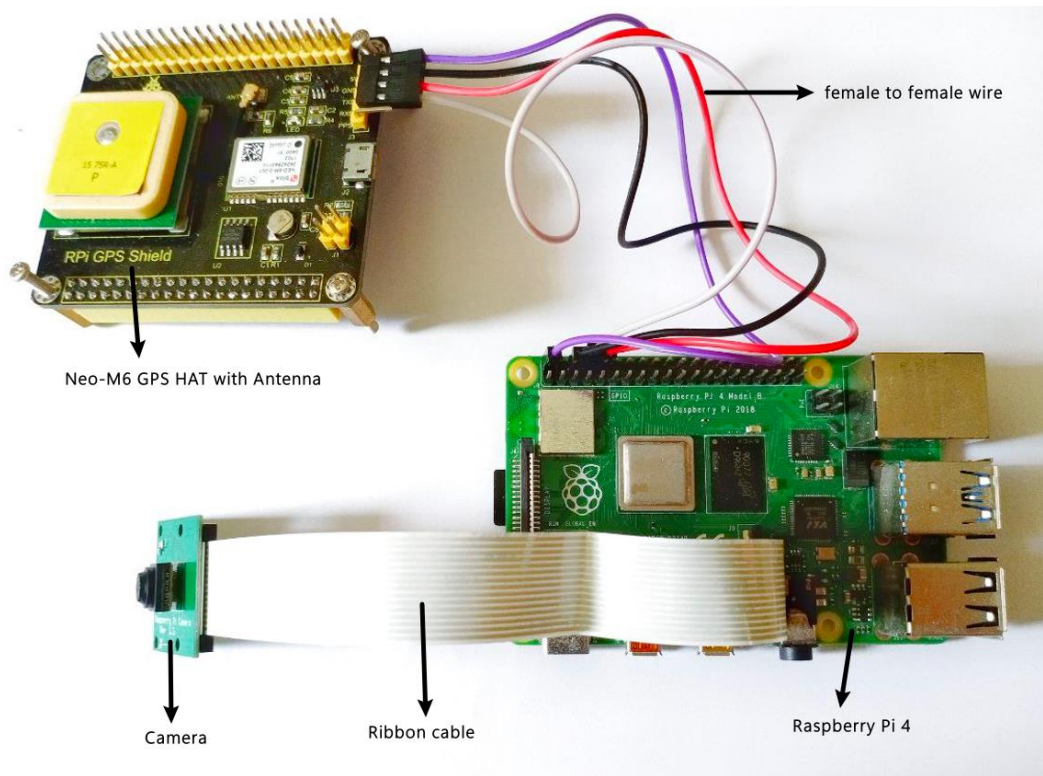


Figure 4-1 Whole prototype of Raspberry Pi

Table 4-1 Function of all components

Component	Function in Project
Raspberry Pi 4	It processes the data received from the camera and GPS.
Camera	It captures video of passengers getting on and off the bus for further analysis.
Neo-M6 GPS HAT with antenna	It provides real-time location data for the bus which can be used in tracking.
Female to female wires	It makes sure power and data are transmitted and received between the RPi and GPS.
Ribbon cable	It ensures power and data are transmitted between the RPi and camera module.

4.3 Accuracy of YOLO

Table 4-2 Accuracy of YOLOv8

Bus Number	Precision	Recall	F1-score	Accuracy
Bus 1	98%	99%	98%	97%
Bus 2	97%	99%	98%	98%
Bus 3	97%	98%	97%	96%
Average	97%	99%	97%	97%

The table above shows that the YOLOv8 model consistently delivers high precision, recall, F1-score and overall accuracy. Hence, YOLOv8 is a reliable choice for real-time object detection in bus environments.

4.4 Timeliness of YOLO Detection and Counter Update

Table 4-3 Timeliness of YOLOv8 detection and counter update

Time Interval	Object Detected (Passenger)	Counter Updates Per Minute	Detection Speed (FPS)	Processing Time per Frame (seconds)
0 – 1 minutes	7	7	30	0.032
1 – 2 minutes	5	5	30	0.029
2 – 3 minutes	10	9	28	0.035
3 – 4 minutes	9	9	29	0.033
4 – 5 minutes	4	4	30	0.028

According to the table above, the performance of object detection is evaluated in one-minute intervals over a five-minute period. It shows that YOLOv8 can maintain a high detection speed, typically around 29.4 FPS while keeping the processing time for each frame at 0.0314 seconds. Hence, this makes sure smooth and timely updates to the counter and enhance the efficiency of real-time calculations.

4.5 Bus Tracked in Google Map



Figure 4-2 Bus tracked at bus stop 1

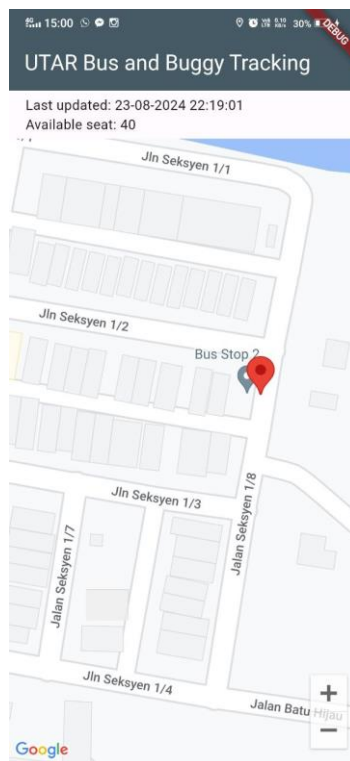


Figure 4-3 Bus tracked at bus stop 2

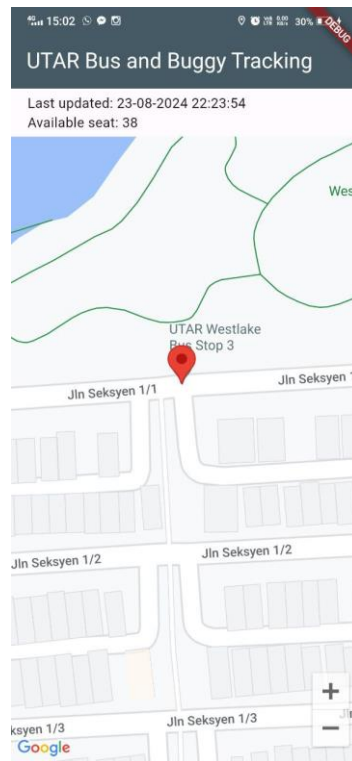


Figure 4-4 Bus tracked at bus stop 3

The above snapshots (figure, figure and figure) show real-time tracking of the UTAR bus using Google Maps which was integrated into a Flutter application named as UTAR Tracker App. It utilizes GPS data such as longitude and latitude captured by the GPS module to determine location of the UTAR bus.

4.6 Passenger Detection in Bus



Figure 4-5 Passenger detection when getting on bus part 1



Figure 4-6 Passenger detection when getting on bus part 2

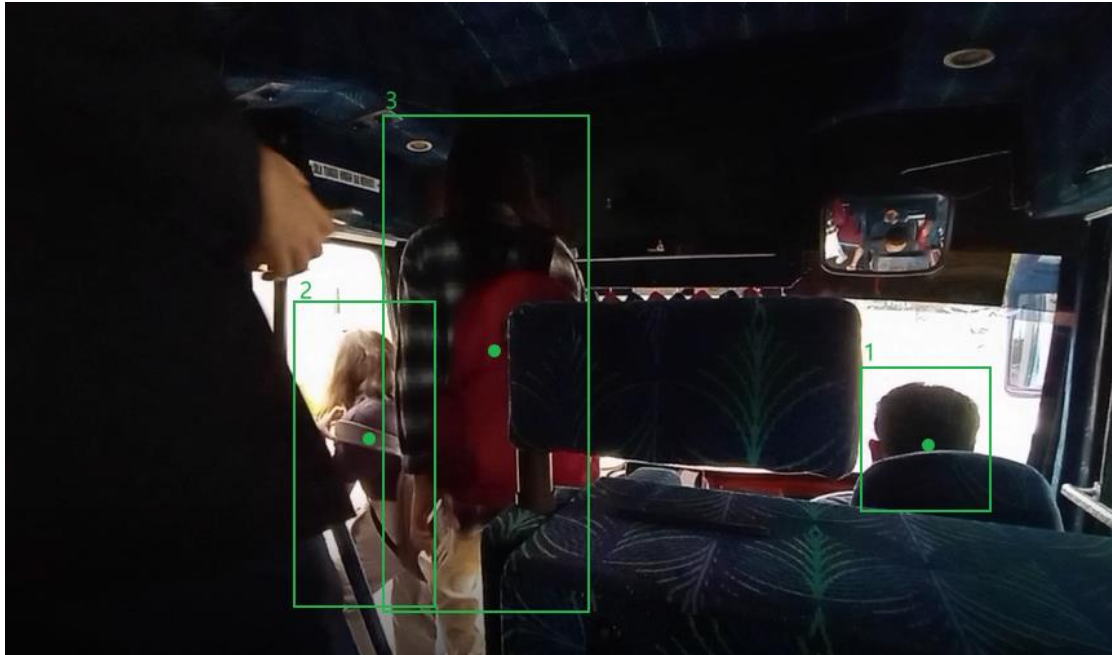


Figure 4-7 Passenger detection when getting off bus part 1

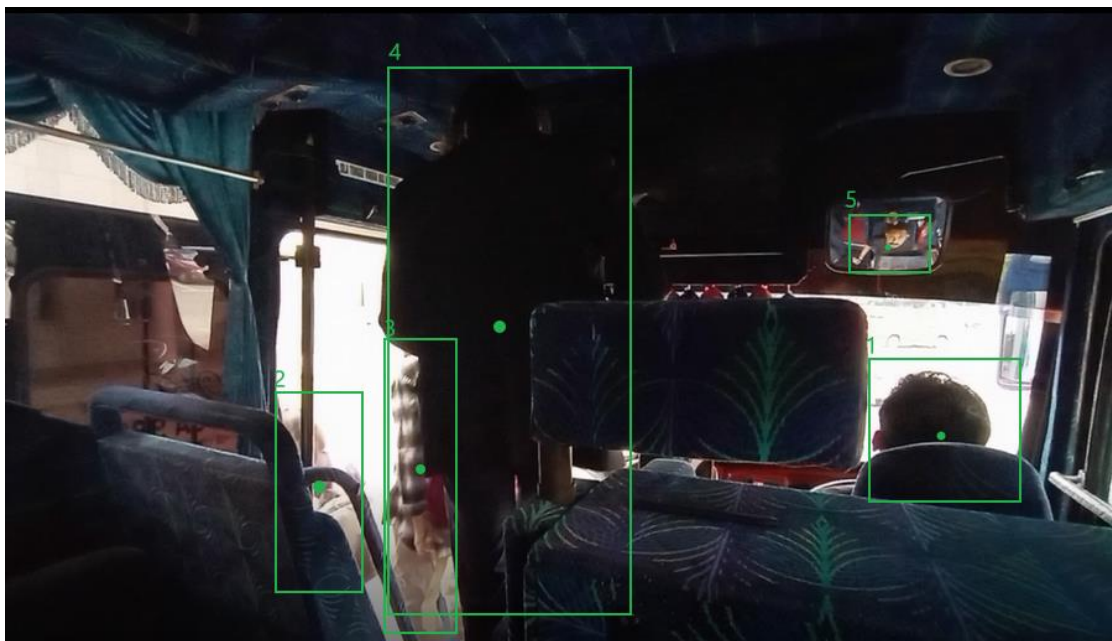


Figure 4-8 Passenger detection when getting off bus part 2

The figures above shows that there are a few students or passengers are detected when they are getting on and off the bus. The detected students or passengers are bounded by a green rectangle with unique ID on upper left of rectangle. Additionally, a central point, also known as the centroid, is located at the centre of the rectangle. This centroid is useful for further analysis in counting the number of available seats.

CHAPTER 5

System Implementation

5.1 System Initialization

After finishing the setup of hardware and software, the system was initialized by connecting the power supply to the RPi 4. The Neo-M6 GPS HAT with Antenna was placed near the window to receive the signal from the satellites. By looking at the blinking LED light, it indicates the GPS module was able to receive the signal successfully in the range of 5 to 10 minutes. Then, the signal data will be sent to the RPi 4 for data processing (longitude and latitude).

The following are the steps of data processing from the RPi 4:

- i. SSH into RPi 4 server by using server name, IP address and password.
- ii. The text editor was opened with the command “nano /home/janet/Desktop/GPS_Tracker/gps_send.py” for python code editing (Figure A-4).
- iii. Run the `gps_send.py` with the command “python /home/janet/Desktop/GPS_Tracker/gps_send.py”, the longitude, latitude data and current time of the location can be obtained (Figure A-5).
- iv. The obtained data was sent to the Firebase Realtime Database by using API (Figure A-6).
- v. The Flutter app retrieved the data from Firebase and the information was finally displayed on the app (Figure A-7).

Furthermore, the function of calculating the number of available seats was implemented by using a 5MP camera module to capture video footage of passengers entering and exiting the bus. The captured video is then analysed by YOLOv8 which is a deep learning model to accurately detect and count the number of people in each frame. However, due to the relatively limited processing power of the RPi 4, it may lead to delays or inefficiencies. Hence, the video data is stream to AWS Cloud services which is Amazon EC2 for high-performance computing. It provides the necessary

computational power to process the video data quickly and efficiently to ensure real-time seat availability updates.

The following are the steps for streaming video data to the AWS Cloud and processing it:

- i. Create an AWS EC2 instance.
- ii. Edit the inbound rules of the EC2 instance to ensure that port 8765 is allowed.
- iii. In RPi4, a python script was created with the command “nano /home/janet/Downloads/connect_device_package/client_web_socket.py”.
- iv. Write source code in client_web_socket.py to stream video to AWS EC2 instance (Table 4-1). Ensure that the Public IP is the current Public IP of the EC2 instance.
- v. Download WinSCP from <https://winscp.net/eng/download.php>. It is use for SSH access into the EC2 instance by using public and private keys of EC2 instance.
- vi. Write a Python script named tracker.py (Figure A-12) on the EC2 instance to track and assign a unique ID to detected objects.
- vii. Download YOLO pre-trained model file which is “yolov8n.pt” from <https://github.com/ultralytics/assets/releases/download/v8.1.0/yolov8n.pt>.
- viii. In EC2 instance, write a python script name as “calculate.py” (Table 4-2) to listen the port 8765 and receive all the frames.
- ix. Run calculate.py first to make sure it is ready to receive frames.
- x. Then, run client_web_socket.py to start sending the video frames to EC2 instance through WebSocket.
- xi. EC2 instance start to calculate the number of available seats using YOLO Version 8.
- xii. The total number of available seats will update on Firebase Realtime Database.
- xiii. After that, retrieve the updated data and display on Flutter app.

5.2 Feasibility of Bus Management System

In this project, the bus management system for a tracking application system is successful due to several reasons. One of its feasibilities is this system was lower in cost. There are just a few hardware and components involved in this RPi 4, with their advantages of affordable prices and easily to be found. However, although their prices are not expensive, RPi 4 can still provide a high performance and quality outcome. Secondly, the system was easily implemented. The time taken for whole progress for the tracking application system is short as the circuit setup of the model is simple. Therefore, it is very effective and suitable if the same models are utilized in different buses respectively. Additionally, the calculation is performed by an AWS Cloud service, specifically an AWS EC2 instance. This instance is chosen for its ability to provide high speed and enhanced performance to ensure the object detection and tracking processes are executed efficiently and in a timely manner. By leveraging the computational power of AWS EC2, the system can handle large video files and complex processing tasks with minimal latency, making it ideal for real-time applications like passenger counting on buses.

CHAPTER 6

System Evaluation and Discussion

6.1 Performance Evaluation

The processing speed of the calculation of the number of available seats will be determined by streamlining the video captured on the Raspberry Pi 4 to the EC2 instance.

1. Make sure the EC2 instance named as UTAR_Shuttle_Bus is running.

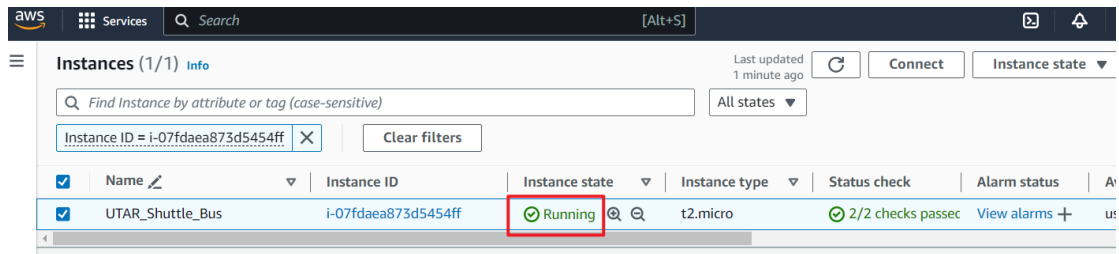


Figure 6-1 AWS EC2 instance

2. Copy the Public IP of EC2 instance and paste into client_web_socket.py.

```

/home/janet/Downloads/connect_device_package/client_web_socket.py - janet@192.168.
import asyncio
import websockets
from picamera2 import Picamera2
import numpy as np
import io
import cv2

async def send_video():
    uri = "ws://54.237.117.73:8765"

    picam2 = Picamera2()
    picam2.start()

    try:
        async with websockets.connect(uri) as websocket:

```

Figure 6-2 Public IP in client_web_socket.py

3. Run `calculate.py` using command “`python calculate.py`” to listen the port 8765.

```
admin@ip-172-31-60-213:~/FYP/Testing$ python3 calculate.py
WebSocket server started on port 8765
```

Figure 6-3 Run `calculate.py`

4. Run `client_web_socket.py` using command “`python client_web_socket.py`” to send frame to the EC2 instance.

```
janet@janet:~/Downloads/connect_device_package $ python client_web_socket.py
[0:22:41.578061152] [2100] INFO Camera camera_manager.cpp:313 libcamera v0.3
[0:22:41.625882153] [2103] WARN RPiSdn sdn.cpp:40 Using legacy SDN tuning -
[0:22:41.629174220] [2103] INFO RPI vc4.cpp:446 Registered camera /base/soc/
device /dev/media0
[0:22:41.629263059] [2103] INFO RPI pipeline_base.cpp:1104 Using configurati
[0:22:41.633618573] [2100] INFO Camera camera_manager.cpp:313 libcamera v0.3
[0:22:41.677375520] [2106] WARN RPiSdn sdn.cpp:40 Using legacy SDN tuning -
[0:22:41.680182135] [2106] INFO RPI vc4.cpp:446 Registered camera /base/soc/
device /dev/media0
[0:22:41.680310753] [2106] INFO RPI pipeline_base.cpp:1104 Using configurati
[0:22:41.687560646] [2100] INFO Camera camera.cpp:1183 configuring streams:
[0:22:41.688172161] [2106] INFO RPI vc4.cpp:621 Sensor: /base/soc/i2c0mux/i2c
- Selected unicom format: 640x480-pGAA
Connected to WebSocket server.
Frame sent.
Frame sent.
Frame sent.
```

Figure 6-4 Frames sent to EC2

5. At the same time, `calculate.py` on EC2 instance will continuously receive the frames and do processing to calculate number of available seats.

```

Seat available: 44
0: 480x640 (no detections), 160.7ms
Speed: 3.1ms preprocess, 160.7ms inference, 0.6ms postprocess per image at shape (1, 3, 480, 640)
Total in: 0
Total out: 0
Seat available: 44
0: 480x640 1 mouse, 163.4ms
Speed: 3.1ms preprocess, 163.4ms inference, 0.9ms postprocess per image at shape (1, 3, 480, 640)
Total in: 0
Total out: 0
Seat available: 44
0: 480x640 1 mouse, 165.9ms
Speed: 3.1ms preprocess, 165.9ms inference, 0.9ms postprocess per image at shape (1, 3, 480, 640)
Total in: 0
Total out: 0
Seat available: 44
0: 480x640 1 mouse, 160.4ms
Speed: 3.1ms preprocess, 160.4ms inference, 0.9ms postprocess per image at shape (1, 3, 480, 640)
Total in: 0
Total out: 0
Seat available: 44
0: 480x640 1 mouse, 160.3ms
Speed: 2.0ms preprocess, 160.3ms inference, 0.9ms postprocess per image at shape (1, 3, 480, 640)
Total in: 0
Total out: 0
Seat available: 44
0: 480x640 1 mouse, 162.2ms
Speed: 3.1ms preprocess, 162.2ms inference, 0.9ms postprocess per image at shape (1, 3, 480, 640)
Total in: 0
Total out: 0
Seat available: 44
0: 480x640 (no detections), 160.9ms
Speed: 3.1ms preprocess, 160.9ms inference, 0.6ms postprocess per image at shape (1, 3, 480, 640)
Total in: 0
Total out: 0
Seat available: 44
0: 480x640 1 mouse, 162.4ms
Speed: 3.1ms preprocess, 162.4ms inference, 0.9ms postprocess per image at shape (1, 3, 480, 640)
Total in: 0
Total out: 0
Seat available: 44

```

Figure 6-5 Output of processing

6. The speed of process is based on:
 - a. Preprocess time: time spent to prepare image for model input.
 - b. Inference time: time spent to perform object detection.
 - c. Postprocess time: time spent after detection, such as drawing bounding boxes on detected objects.
 - d. Processing time = preprocess time + inference time + postprocess time.

Hence, the average speed of processing time of each frame is around 163.0ms to 170.0ms.

- Then, the number of available seats will update on Firebase Realtime Database.



Figure 6-6 Updated seats number

In conclusion, the performance of the function that calculates the number of available seats is good. This is because the calculation is based on each frame, allowing for real-time updates of the number of available seats.

6.2 Accuracy Assessment

The accuracy of the GPS module on RPi 4 will be determined by compared the location of the longitude and latitude received by RPi4 with Google Map current location.

- Run the `gps_send.py` to get the longitude and latitude of current location from satellite through Neo-M6 GPS HAT with Antenna.

```

janet@janet: ~/Desktop/GPS_Tracker
janet@janet:~/Desktop/GPS_Tracker $ python gps_send.py
Sending Data to Firebase
Latitude=4.3282345 and Longitude=101.13583866666667
Data sent23-08-2024 01:35:43
Latitude=4.328238 and Longitude=101.13583633333333
Data sent23-08-2024 01:35:44
Latitude=4.328241 and Longitude=101.13583416666667
Data sent23-08-2024 01:35:45
Latitude=4.3282445 and Longitude=101.13583166666666
Data sent23-08-2024 01:35:46

```

Figure 6-7 Output of `gps_send.py`

CHAPTER 6

2. Then, open Google Map website to key in the longitude and latitude received just now into the search box.

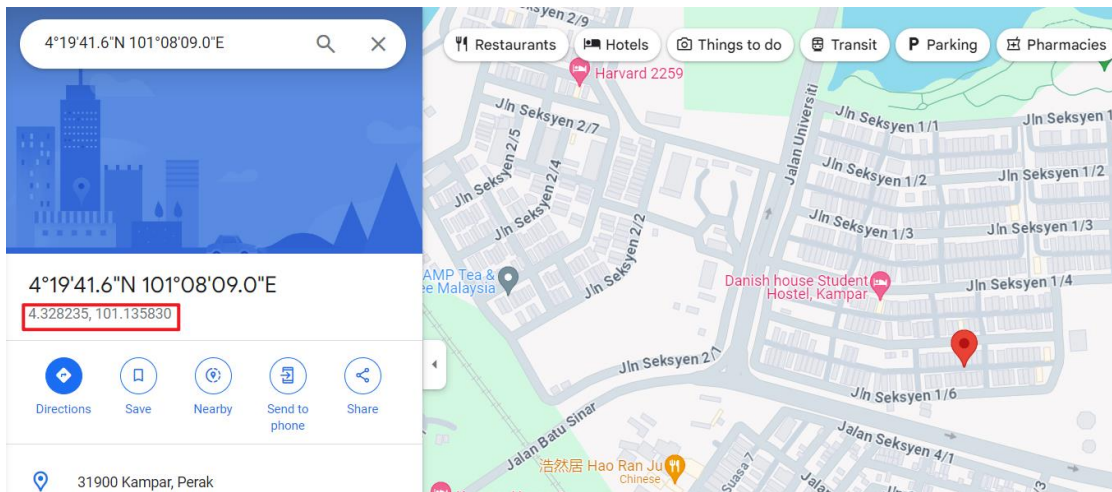


Figure 6-8 Key in longitude and latitude received to search for location

3. Compare the location with current location, it shows at the same place. Hence, the GPS module is running accurately.

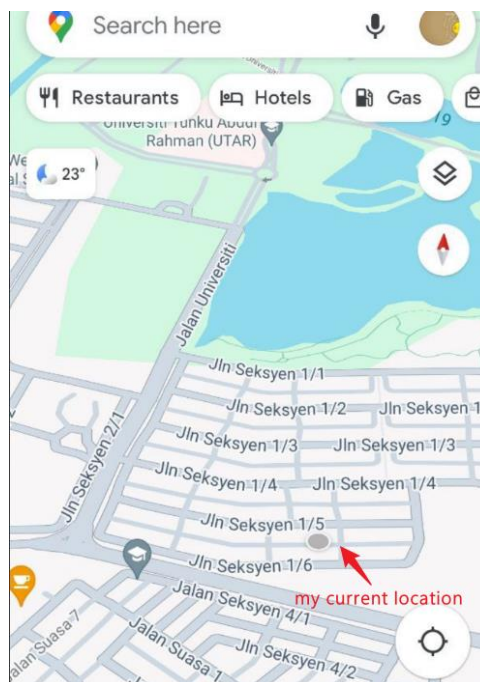


Figure 6-9 Current location display on Android phone

CHAPTER 6

The accuracy of the number of available seats will be evaluated by simulating the actions of bus passengers in and out from the bus, as well as the bus steps, rather than by capturing real bus passengers.

1. The images below are screenshots from the original demo video which is used in this evaluation that show three people moving up, simulating passengers boarding the bus.



Figure 6-10 Original video part 1



Figure 6-11 Original video part 2

2. The backend for calculating the number of passengers entering and exiting is based on a horizontal line. The person is represented by a bounding rectangle and a centroid. When the centroid crosses the line, it indicates the person is entering; otherwise, it indicates the person is exiting.



Figure 6-12 Backend of video processing

- The output file shows that the total number of people entering is 4, and the total number of people exiting is 1. It is because person's body may be unstable which cause the centroid to temporarily cross the horizontal line and then move back. However, the final result is still accurate, as the total available seats will decrease by 3 which reflect the entry of 3 people.

```

0: 640x384 1 chair, 126.3ms
Speed: 1.6ms preprocess, 126.3ms inference, 0.9ms postprocess per image at shape (1, 3, 640, 384)

0: 640x384 1 chair, 127.2ms
Speed: 1.6ms preprocess, 127.2ms inference, 0.9ms postprocess per image at shape (1, 3, 640, 384)

0: 640x384 1 chair, 126.2ms
Speed: 1.6ms preprocess, 126.2ms inference, 0.9ms postprocess per image at shape (1, 3, 640, 384)

0: 640x384 (no detections), 126.9ms
Speed: 1.7ms preprocess, 126.9ms inference, 0.6ms postprocess per image at shape (1, 3, 640, 384)
Total in: 4
Total out: 1
Seat available: 41

```

Figure 6-13 Result of available seats

6.3 System Integration Testing

This part will test the overall flow of data from Raspberry Pi 4 to AWS Cloud then display on Flutter app.

1. Run `gps_send.py` and `client_web_socket.py` at the same time to send the longitude and latitude to Firebase, as well as upload the video frames to AWS EC2 instances through port 8765 for further processing.

```

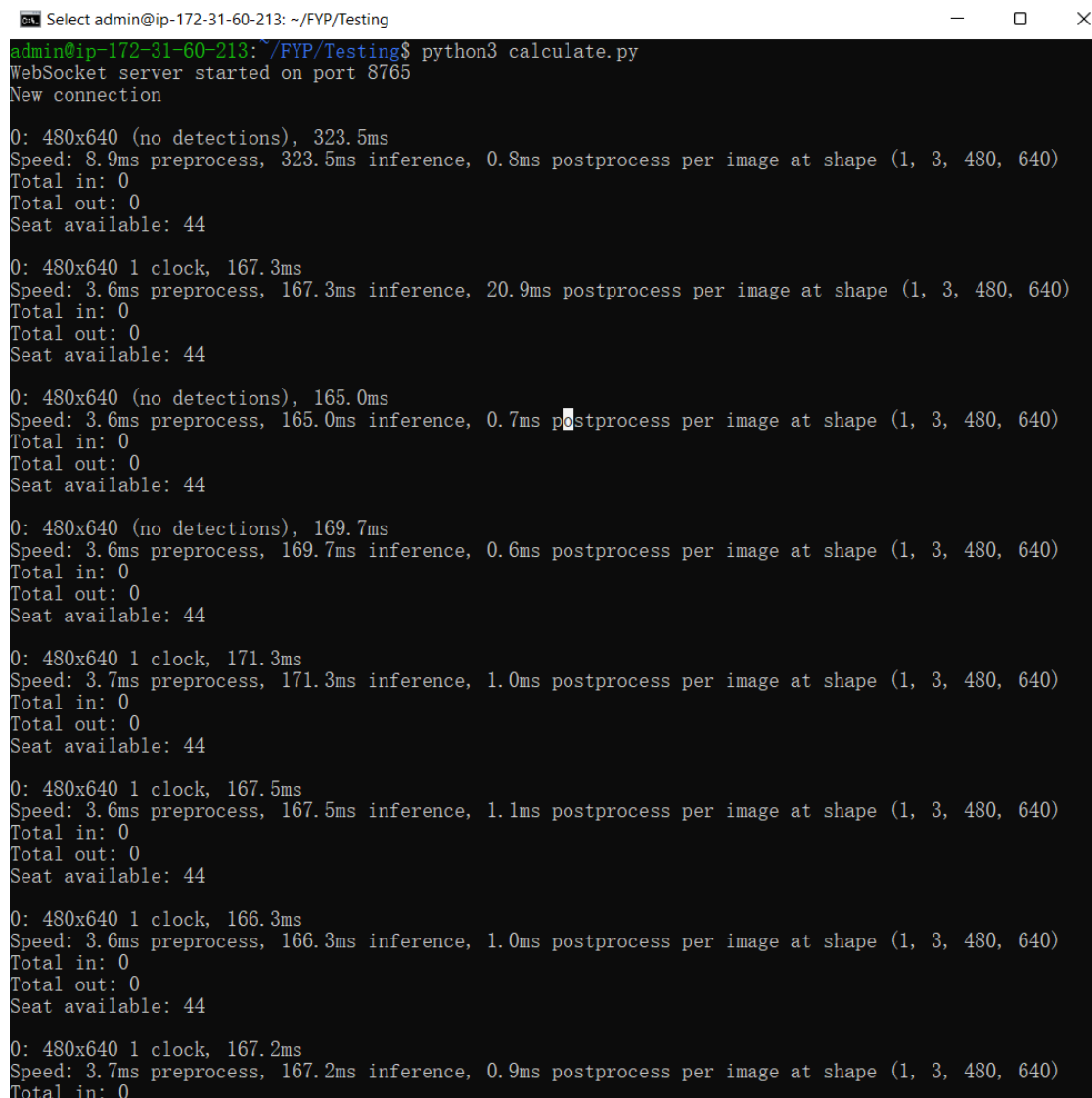
janet@janet: ~/Desktop/GPS_Tracker
janet@janet:~/Desktop/GPS_Tracker $ python gps_send.py
Sending Data to Firebase
Latitude=4.467744166666667 and Longitude=101.04380733333333
Data sent23-08-2024 22:15:10
Latitude=4.467753666666667 and Longitude=101.0438045
Data sent23-08-2024 22:15:17
Latitude=4.467754666666667 and Longitude=101.04380283333333
Data sent23-08-2024 22:15:18
Latitude=4.467755166666667 and Longitude=101.04380266666666
Data sent23-08-2024 22:15:19
Latitude=4.467755166666667 and Longitude=101.0438035
Data sent23-08-2024 22:15:20
Latitude=4.4677555 and Longitude=101.0438035
Data sent23-08-2024 22:15:21
Latitude=4.467756166666667 and Longitude=101.043803
Data sent23-08-2024 22:15:22
Latitude=4.467756166666667 and Longitude=101.04380316666666
Data sent23-08-2024 22:15:23
Latitude=4.4677535 and Longitude=101.04380483333334
Data sent23-08-2024 22:15:24
Latitude=4.4677533333333335 and Longitude=101.04380333333333
Data sent23-08-2024 22:15:25
Latitude=4.467756666666665 and Longitude=101.04379883333333
Data sent23-08-2024 22:15:26
Latitude=4.467761666666667 and Longitude=101.04379633333333
Data sent23-08-2024 22:15:27
Latitude=4.4677625 and Longitude=101.04379666666667
Data sent23-08-2024 22:15:28
Latitude=4.4677655 and Longitude=101.04379283333333
Data sent23-08-2024 22:15:29
Latitude=4.467763333333333 and Longitude=101.04379633333333
Data sent23-08-2024 22:15:30
Latitude=4.4677625 and Longitude=101.043798
Data sent23-08-2024 22:15:31

janet@janet:~/Downloads/connect_device_package
janet@janet:~/Downloads/connect_device_package $ ls
aws-iot-device-sdk-python-v2  Raspberry_Pi_private_key  VideoSend_AWS2.py
client_web_socket.py         Raspberry_Pi_public_key   VideoSend_AWS.py
Raspberry_Pi_cert.pem       root-CA.crt
Raspberry_Pi-Policy         start.sh
janet@janet:~/Downloads/connect_device_package $ python client_web_socket.py
[0:06:49.534196389] [2041] INFO Camera camera_manager.cpp:313 libcamera v0.3.0
+65-6ddd79b5
[0:06:49.583940395] [2046] WARN RPiSdn sdn.cpp:40 Using legacy SDN tuning - pl
ease consider moving SDN inside rpi.denoise
[0:06:49.588204170] [2046] INFO RPI vc4.cpp:446 Registered camera /base/soc/12
c0mux/12c01/ov5647036 to Unicam device /dev/media4 and ISP device /dev/media1
[0:06:49.588299809] [2046] INFO RPI pipeline_base.cpp:1104 Using configuration
file '/usr/share/libcamera/pipeline/rpi/vc4/rpi_apps.yaml'
[0:06:49.598128254] [2041] INFO Camera camera_manager.cpp:313 libcamera v0.3.0
+65-6ddd79b5
[0:06:49.649218416] [2049] WARN RPiSdn sdn.cpp:40 Using legacy SDN tuning - pl
ease consider moving SDN inside rpi.denoise
[0:06:49.652692769] [2049] INFO RPI vc4.cpp:446 Registered camera /base/soc/12
c0mux/12c01/ov5647036 to Unicam device /dev/media4 and ISP device /dev/media1
[0:06:49.652817926] [2049] INFO RPI pipeline_base.cpp:1104 Using configuration
file '/usr/share/libcamera/pipeline/rpi/vc4/rpi_apps.yaml'
[0:06:49.661685913] [2041] INFO Camera camera.cpp:1183 configuring streams: (0
) 640x480-YBGR8888 (1) 640x480-SGBRG10_CSI2P
[0:06:49.662401250] [2049] INFO RPI vc4.cpp:621 Sensor: /base/soc/12c0mux/12c
01/ov5647036 - Selected sensor format: 640x480-SGBRG10_1X10 - Selected unicam fo
rmat: 640x480-pGAA
Connected to WebSocket server.
Frame sent.
Frame sent.
Frame sent.
Frame sent.
Frame sent.
Frame sent.

```

Figure 6-14 Run `gps_send.py` and `client_web_socket.py`

2. EC2 instance start to process each of the frames.



```
Select admin@ip-172-31-60-213: ~/FYP/Testing
admin@ip-172-31-60-213:~/FYP/Testing$ python3 calculate.py
WebSocket server started on port 8765
New connection
0: 480x640 (no detections), 323.5ms
Speed: 8.9ms preprocess, 323.5ms inference, 0.8ms postprocess per image at shape (1, 3, 480, 640)
Total in: 0
Total out: 0
Seat available: 44
0: 480x640 1 clock, 167.3ms
Speed: 3.6ms preprocess, 167.3ms inference, 20.9ms postprocess per image at shape (1, 3, 480, 640)
Total in: 0
Total out: 0
Seat available: 44
0: 480x640 (no detections), 165.0ms
Speed: 3.6ms preprocess, 165.0ms inference, 0.7ms postprocess per image at shape (1, 3, 480, 640)
Total in: 0
Total out: 0
Seat available: 44
0: 480x640 (no detections), 169.7ms
Speed: 3.6ms preprocess, 169.7ms inference, 0.6ms postprocess per image at shape (1, 3, 480, 640)
Total in: 0
Total out: 0
Seat available: 44
0: 480x640 1 clock, 171.3ms
Speed: 3.7ms preprocess, 171.3ms inference, 1.0ms postprocess per image at shape (1, 3, 480, 640)
Total in: 0
Total out: 0
Seat available: 44
0: 480x640 1 clock, 167.5ms
Speed: 3.6ms preprocess, 167.5ms inference, 1.1ms postprocess per image at shape (1, 3, 480, 640)
Total in: 0
Total out: 0
Seat available: 44
0: 480x640 1 clock, 166.3ms
Speed: 3.6ms preprocess, 166.3ms inference, 1.0ms postprocess per image at shape (1, 3, 480, 640)
Total in: 0
Total out: 0
Seat available: 44
0: 480x640 1 clock, 167.2ms
Speed: 3.7ms preprocess, 167.2ms inference, 0.9ms postprocess per image at shape (1, 3, 480, 640)
Total in: 0
```

Figure 6-15 Output of calculate.py

3. The data also will update in the Firebase Realtime Database.



Figure 6-16 Information on Firebase

4. Open Flutter app to view the current location and number of available seats.



Figure 6-17 Location on flutter app

CHAPTER 6



As a result, the system has been successfully integrated to enable seamless data streaming and real-time display on the Flutter app, thereby providing users with an efficient and reliable experience.

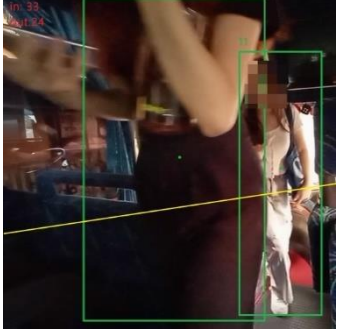
6.4 Case Study

Table 6-1 Testing parameters

Parameter	Description
Numbers of bus	3
Location	UTAR Kampar campus and Westlake
Duration	Half an hour
Number of passengers	Not fixed, based on how many getting on within the duration

Table 6-2 Accuracy in Bus 1, 2, 3

Bus	Image	Actual Passenger	Detected Passengers	Detection Accuracy (%)
Bus 1		30	30	100
Bus 2		25	25	100

Bus 3		33	33	100
-------	---	----	----	-----

6.5 Results on Questionnaire

Count of Before using the UTAR Tracker app, how long did you typically wait for the bus or shuttle (in minutes)?

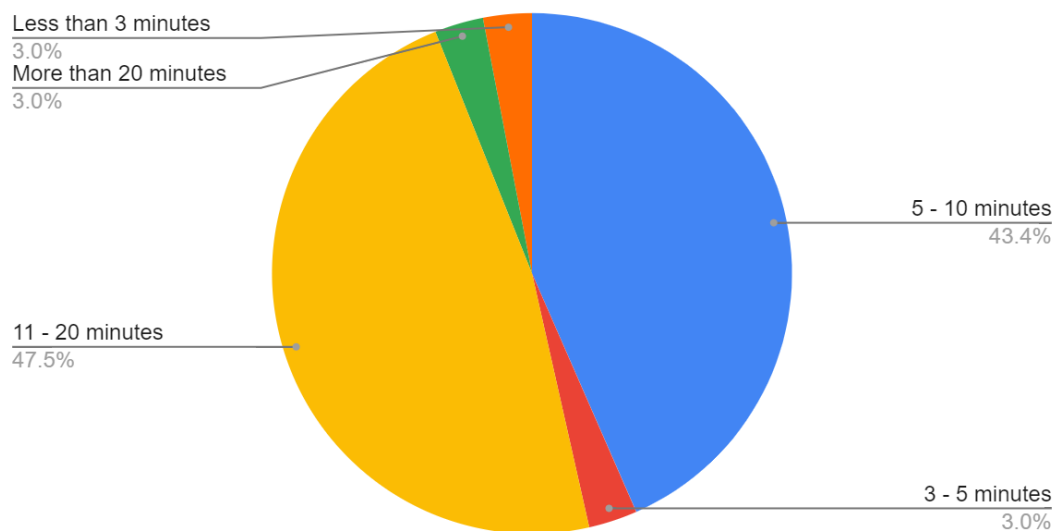


Figure 6-18 Pie chart before students used the app

According to the pie chart above, it shows that before UTAR students used the UTAR Tracker app, 3% of students waited for the bus for more than 20 minutes, 47.5% waited for 11 to 20 minutes, and 43.4% waited for 5 to 10 minutes. Only total of 6% of students waited for less than 5 minutes.

Count of After using the UTAR Tracker app, how long did you typically wait for the bus or shuttle (in minutes)?

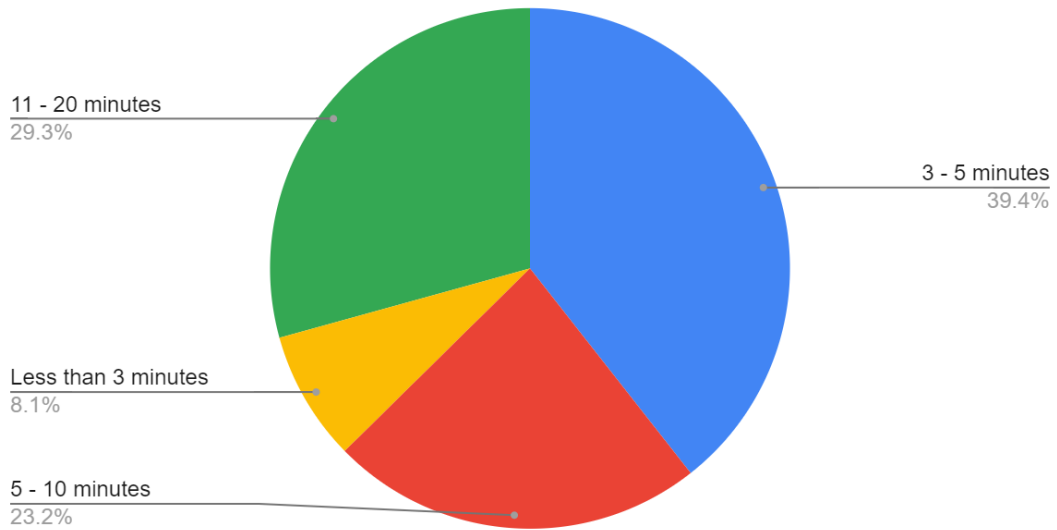


Figure 6-19 Pie chart after students used the app

After using the app to track bus locations and the number of available seats, 39.4% of students waited for 3 to 5 minutes, while only 29.3% waited for 11 to 20 minutes. Comparing the situation before and after using the app, the percentage of students waiting for 11 to 20 minutes decreased by approximately 18.2%.

On average, how much time do you think you have saved using the app?

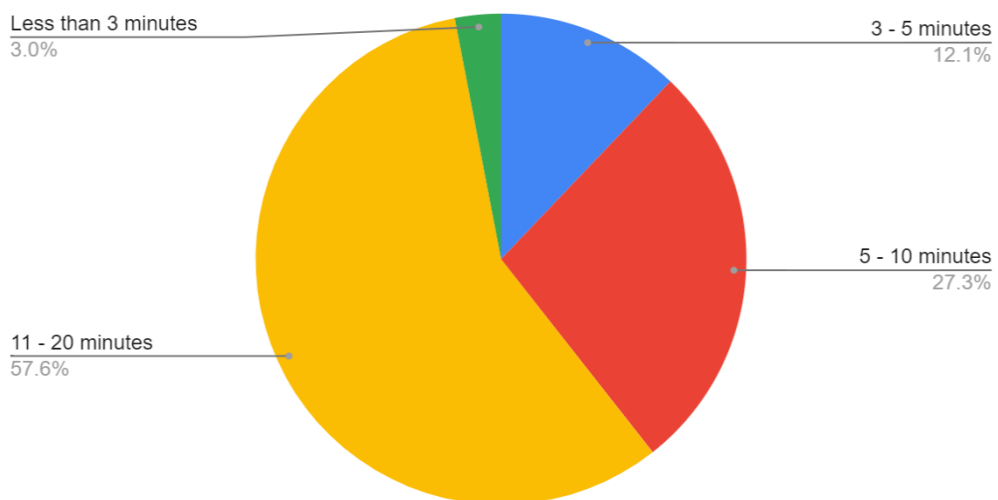


Figure 6-20 Pie chart of students saving time

CHAPTER 6

Next, the UTAR Tracker app has saved time for students. Specifically, 57.6% of students saved up to 20 minutes, 27.3% saved 5 to 10 minutes, 12.1% saved around 3 to 5 minutes, and only 3% saved less than 3 minutes.

CHAPTER 7

Conclusion

In conclusion, this system has solved the problem statements of inconsistent punctuality of transport arrivals and lack of real-time tracking applications as well as lack of information on seat availability. The motivation was driven by the students lack of real-time tracking system to track the location and number of passengers in and out of the bus. The students can only refer the predefined bus schedules which will be easily affected by unforeseen circumstances.

Therefore, a Raspberry Pi 4, Neo-M6 GPS HAT with Antenna, and a 5MP camera module were integrated to create a bus tracking management system. The GPS HAT was connected to the Raspberry Pi using female-to-female jumper wires via the 5V, GND, TXD, and RXD pins, while the camera module was attached via a ribbon cable.

Additionally, a Flutter application was created to display the location of buses on Google Map through Google Map API. The students can install the app on mobile phone to track the real-time location of buses and number of remaining seats before the students wait at the bus stops. Therefore, this system has solved inconsistent punctuality, unclear schedules and uncertain number of remaining seats which led to inconvenience and time waste.

Overall, this new system has integrated IoT into buses to enhance transportation experience for students. The research objectives which are “implements GPS sensor in bus to track real-time bus location” and “create an application to let students monitor the bus location” as well as “install a camera module to capture the passengers in and out of the bus” are achieved.

REFERENCES

- [1] S. McManus and M. Cook, *Raspberry Pi for dummies*, 3rd ed. Canada: John Wiley & Sons, Inc., p. 23.
- [2] “About Moovit: The leading Maas Solutions Provider,” Moovit, <https://moovit.com/about-us/> (accessed Apr. 11, 2023).
- [3] S. FRANCISCO, Transit App Moovit Expands Crowdsourcing to Masses. 2016.
- [4] Arrival updates: Always stay in the loop – help center, <https://support.moovitapp.com/hc/en-us/articles/13517706542866-Arrival-Updates-Always-Stay-in-the-Loop> (accessed Apr. 11, 2023).
- [5] M. Hargrave, “Crowdsourcing: Definition, how it works, types, and examples,” Investopedia, <https://www.investopedia.com/terms/c/crowdsourcing.asp> (accessed Apr. 11, 2023).
- [6] “Prasarana Malaysia Berhad official corporate website,” Prasarana Malaysia Berhad, <https://www.prasarana.com.my/> (accessed Dec. 11, 2023).
- [7] D. Tan, “Prasarana launches pulse journey planner app for Rapid KL Rail, bus services - PG, Kuantan mid-2021,” Paul Tan’s Automotive News, <https://paultan.org/2020/12/18/prasarana-launches-pulse-journey-planner-app-for-rapid-kl-rail-bus-services-pg-kuantan-mid-2021/> (accessed Dec. 11, 2023).
- [8] A. Zola, “What is google maps and how do you use it?,” WhatIs, <https://www.techtarget.com/whatis/definition/Google-Maps> (accessed Dec. 11, 2023).
- [9] E. Reid, “A look back at 15 years of mapping the world,” Google, <https://blog.google/products/maps/look-back-15-years-mapping-world/> (accessed Dec. 11, 2023).
- [10] “Building a web application with Flutter,” *docs.flutter.dev*, <https://docs.flutter.dev/platform-integration/web/building>

APPENDIX A

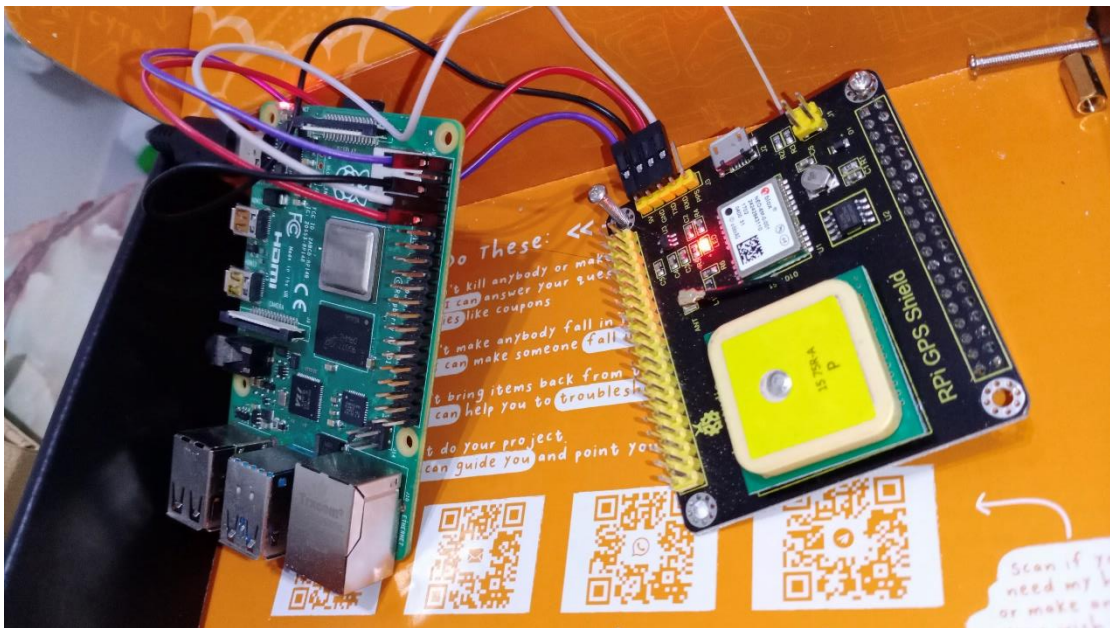


Figure A-1 Connection of Pi 4 with Neo-M6 GPS HAT with Antenna

```

janet@raspberrypi: ~/Desktop/GPS_Tracker
GNU nano 7.2 gps_print.py
import serial
import pynmea2

while True:
    port="/dev/ttyAMA0"
    ser=serial.Serial(port, baudrate=9600, timeout=0.5)
    dataout = pynmea2.NMEAStreamReader()
    newdata=ser.readline()
    n_data = newdata.decode('latin-1')
    if n_data[0:6] == '$GPRMC':
        newmsg=pynmea2.parse(n_data)
        lat=newmsg.latitude
        lng=newmsg.longitude
        gps = "Latitude=" + str(lat) + " and Longitude=" + str(lng)
        print(gps)

```

Figure A-2 Python code to read only longitude and latitude

```

janet@raspberrypi: ~/Desktop/GPS_Tracker
janet@raspberrypi: ~/Desktop/GPS_Tracker $ python gps_print.py
Latitude=4.467729666666667 and Longitude=101.04390983333333
Latitude=4.467729833333333 and Longitude=101.04391066666666
Latitude=4.467730166666667 and Longitude=101.04391116666666
Latitude=4.4677305 and Longitude=101.04391133333333
Latitude=4.467730666666666 and Longitude=101.04391133333333
Latitude=4.467730833333333 and Longitude=101.0439115
Latitude=4.467730666666666 and Longitude=101.04391166666667
Latitude=4.4677305 and Longitude=101.043912
Latitude=4.4677305 and Longitude=101.04391233333334
Latitude=4.467730166666667 and Longitude=101.04391283333334
Latitude=4.467729833333333 and Longitude=101.0439135
Latitude=4.46773 and Longitude=101.0439135
Latitude=4.467730333333333 and Longitude=101.0439135
Latitude=4.467730666666666 and Longitude=101.04391333333334
Latitude=4.467731 and Longitude=101.04391316666667
Latitude=4.4677315 and Longitude=101.043913
Latitude=4.467731666666666 and Longitude=101.04391283333334
Latitude=4.467732166666667 and Longitude=101.04391283333334
Latitude=4.467732666666667 and Longitude=101.0439125

```

Figure A-3 Longitude and latitude data

```

janet@raspberrypi: ~/Desktop/GPS_Tracker
GNU nano 7.2 gps_send.py
import pyrebase
from collections.abc import MutableMapping
import serial
import pynmea2
import datetime

firebaseConfig={
    "apiKey": "AIzaSyCOz--3ZRtEZn-eHGk-xxJ5lclDhpLd0TY",
    "authDomain": "tracker-50c3e.firebaseio.com",
    "storageBucket": "tracker-50c3e.appspot.com",
    "databaseURL": "https://tracker-50c3e-default-rtdb.firebaseio.com"
}

firebase=pyrebase.initialize_app(firebaseConfig)
db = firebase.database()

print("Sending Data to Firebase")

while True:
    port="/dev/ttyAMA0"
    ser=serial.Serial(port, baudrate=9600, timeout=0.5)
    dataout = pynmea2.NMEAStreamReader()
    newdata=ser.readline()
    n_data = newdata.decode('latin-1')
    if n_data[0:6] == '$GPRMC':
        newmsg=pynmea2.parse(n_data)
        lat=newmsg.latitude
        lng=newmsg.longitude
        x=datetime.datetime.now()
        y = x.strftime("%d-%m-%Y %H:%M:%S")
        gps = "Latitude=" + str(lat) + " and Longitude=" + str(lng)
        print(gps)
        data = {"LAT": lat, "LNG": lng, "DateTime": y}
        db.update(data)
        print("Data sent" + y)

```

Figure A-4 Python code to send data to firebase

```

janet@raspberrypi: ~/Desktop/GPS_Tracker
janet@raspberrypi:~/Desktop/GPS_Tracker $ janet@raspberrypi:~/Desktop/GPS_Tracker $
janet@raspberrypi:~/Desktop/GPS_Tracker $ python gps_send.py
Sending Data to Firebase
Latitude=4.4677135 and Longitude=101.04386616666666
Data sent19-04-2024 00:32:53
Latitude=4.4677141666666667 and Longitude=101.04386616666666
Data sent19-04-2024 00:32:54
Latitude=4.4677145 and Longitude=101.04386616666666
Data sent19-04-2024 00:32:55
Latitude=4.4677155 and Longitude=101.04386633333333
Data sent19-04-2024 00:32:56
Latitude=4.4677161666666665 and Longitude=101.04386666666667
Data sent19-04-2024 00:32:57
Latitude=4.4677166666666667 and Longitude=101.04386666666667
Data sent19-04-2024 00:32:58
Latitude=4.4677171666666667 and Longitude=101.0438665
Data sent19-04-2024 00:32:59
Latitude=4.4677173333333334 and Longitude=101.043866
Data sent19-04-2024 00:33:00

```

Figure A-5 Data sent into Firebase Realtime Database

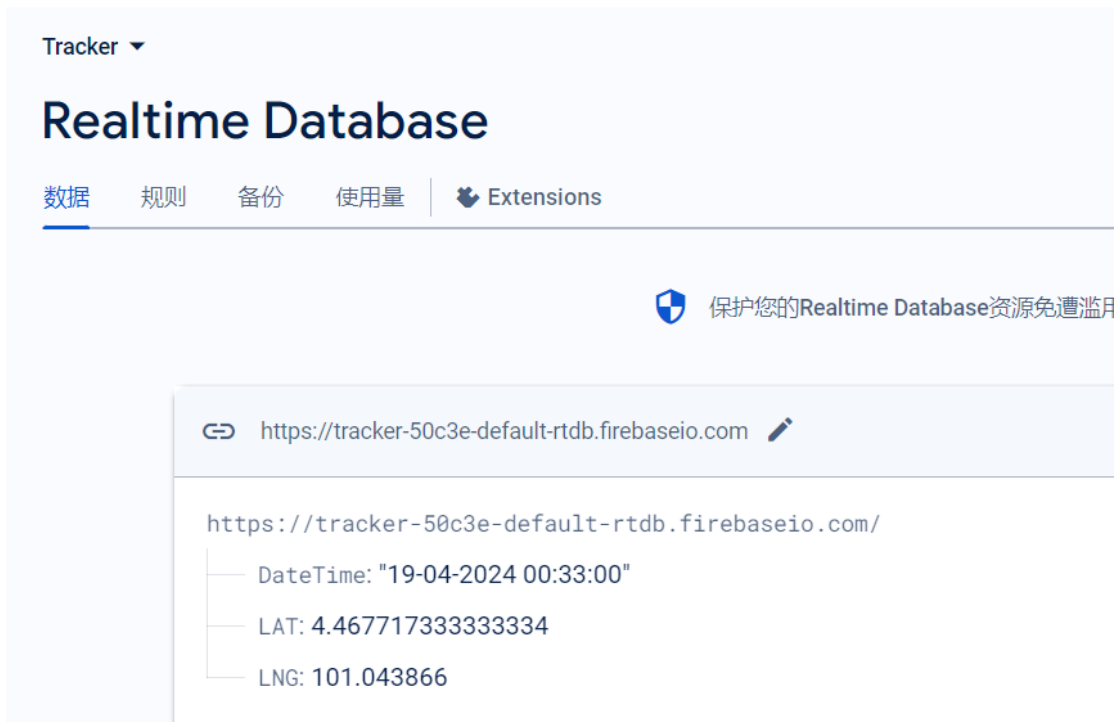


Figure A-6 Firebase Realtime Database



Figure A-7 Flutter app show updated location

APPENDIX

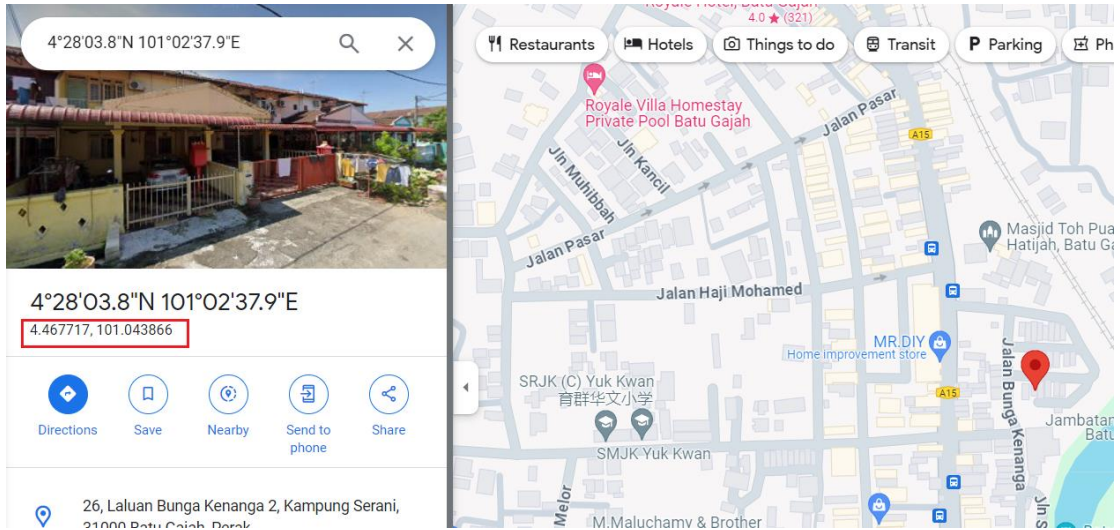


Figure A-8 Compare longitude and latitude on Google Map

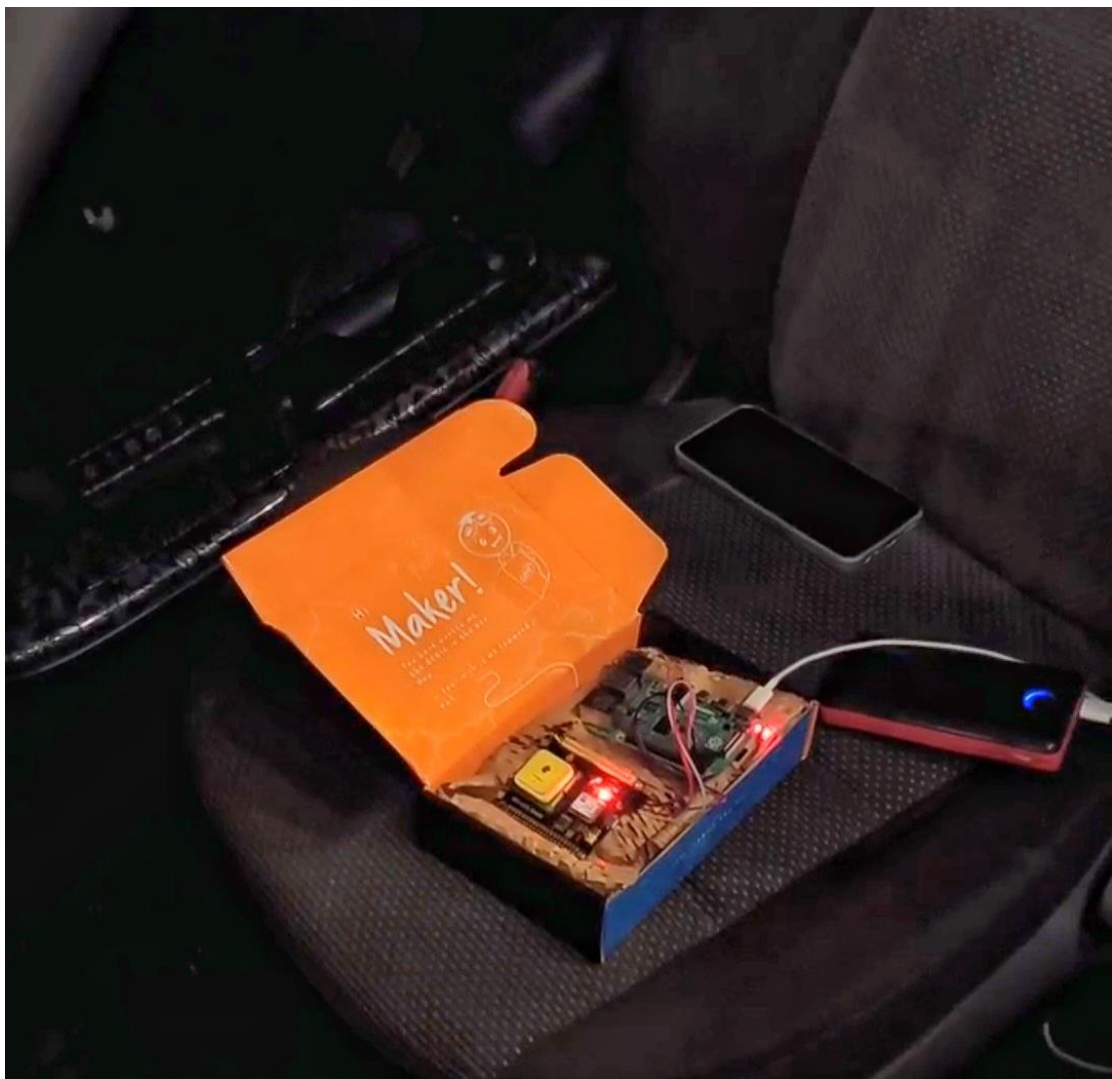


Figure A-9 Install GPS module on car using power bank as the power supply

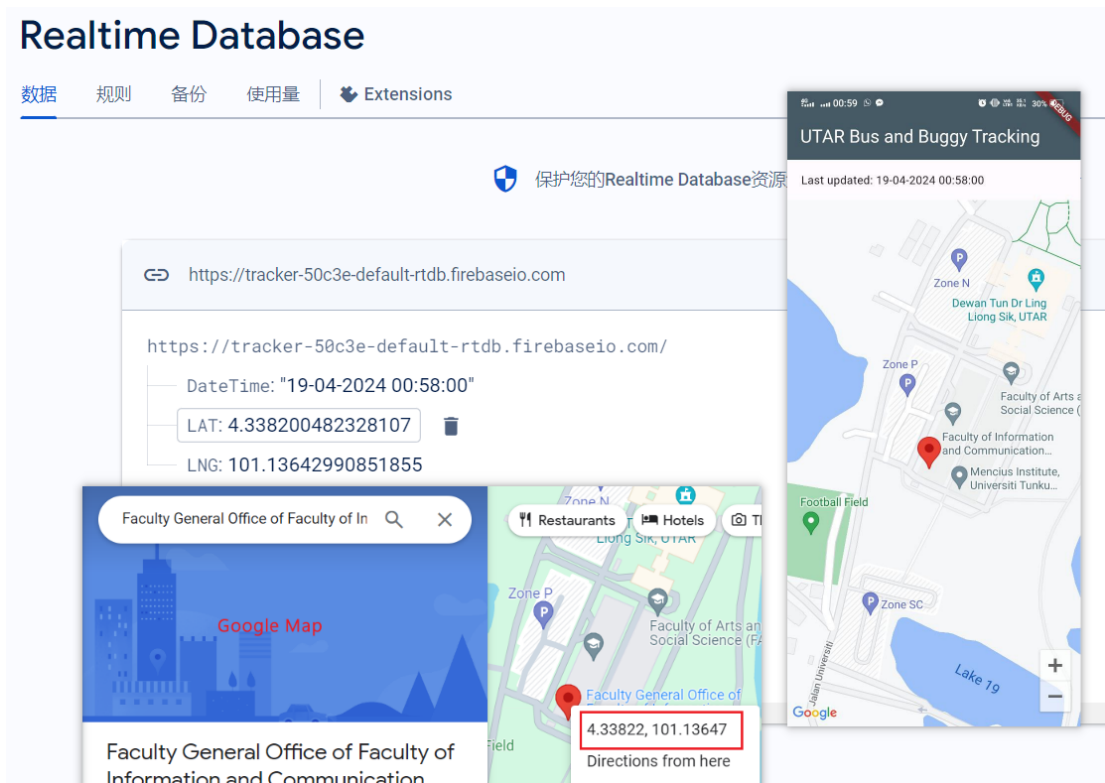


Figure A-10 Location updated when new data sent into database

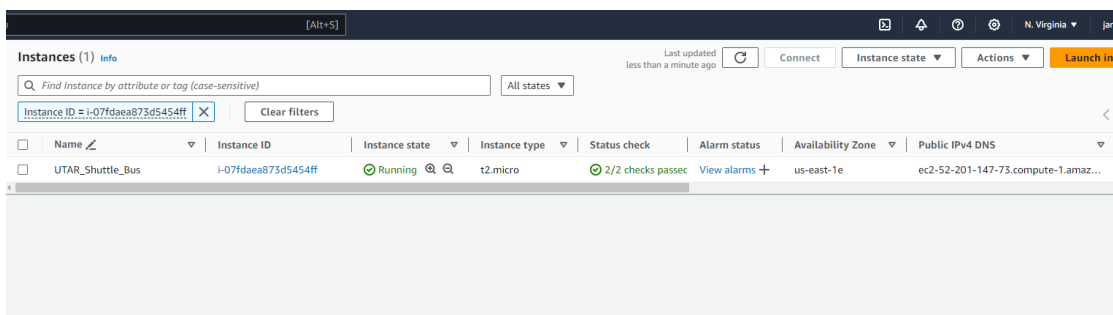


Figure A-11 AWS EC2 instance

Table A-1 Source Code of client_web_socket.py

	Source Code
	<pre>import asyncio import websockets from picamera2 import Picamera2 import numpy as np import io import cv2</pre>

	<pre> async def send_video(): uri = "ws://54.237.117.73:8765" picam2 = Picamera2() picam2.start() try: async with websockets.connect(uri) as websocket: print("Connected to WebSocket server.") while True: buffer = picam2.capture_array() image_data = np.array(buffer, dtype=np.uint8) _, img_encoded = cv2.imencode('.jpg', image_data) image_bytes = img_encoded.tobytes() await websocket.send(image_bytes) print("Frame sent.") except Exception as e: print(f"An error occurred: {e}") finally: picam2.stop() print("Camera stopped.") asyncio.get_event_loop().run_until_complete(send_video()) </pre>
--	--

Table A-2 Source code of calculate.py

	Source Code
	<pre> import cv2 import boto3 from tracker import Tracker import pyrebase from ultralytics import YOLO </pre>

```
import numpy as np
import json
import os
import asyncio
import websockets

firebaseConfig = {
    "apiKey": "AIzaSyCOz--3ZRtEZn-eHGk-xxJ51cldhpLdOTY",
    "authDomain": "tracker-50c3e.firebaseio.com",
    "storageBucket": "tracker-50c3e.appspot.com",
    "databaseURL": "https://tracker-50c3e-default-rtdb.firebaseio.com"
}

firebase = pyrebase.initialize_app(firebaseConfig)
db = firebase.database()

model = YOLO('/home/admin/FYP/yolov8n.pt')
class_list = model.names

def get_available_seats():
    try:
        seat_data = db.child("AvailableSeat").get()
        if seat_data.val() is not None:
            return seat_data.val()
        else:
            print("No seat data found in Firebase.")
            return 44 # Default seat count if no data is found
    except Exception as e:
        print(f"Error fetching seat data from Firebase: {e}")
        return 44 # Default seat count in case of error

async def handle_connection(websocket, path):
    print("New connection")
    try:
```

```
count = 0
total_in = 0
total_out = 0
seat = get_available_seats()
tracker = Tracker()
last_positions = {}

async for message in websocket:
    nparr = np.frombuffer(message, np.uint8)
    frame = cv2.imdecode(nparr, cv2.IMREAD_COLOR)

    count += 1
    if count % 2 != 0:
        continue

    pt1 = (0, 200)
    pt2 = (frame.shape[1], 520)

    results = model.predict(frame)
    boxes = results[0].boxes.xyxy.numpy()
    classes = results[0].boxes.cls.numpy()

    detections = []
    for i in range(len(boxes)):
        x1, y1, x2, y2 = boxes[i]
        d = int(classes[i])
        c = class_list[d]
        if 'person' in c:
            detections.append([int(x1), int(y1), int(x2), int(y2)])

    bbox_id = tracker.update(detections)
    for bbox in bbox_id:
        x3, y3, x4, y4, id = bbox
```

```

cx = (x3 + x4) // 2
cy = (y3 + y4) // 2

if id in last_positions:
    last_cx, last_cy = last_positions[id]

    if (last_cy - pt1[1]) * (pt2[0] - pt1[0]) > (last_cx - pt1[0]) *
(pt2[1] - pt1[1]) and \
        (cy - pt1[1]) * (pt2[0] - pt1[0]) <= (cx - pt1[0]) * (pt2[1] -
pt1[1]):
        total_out += 1
    elif (last_cy - pt1[1]) * (pt2[0] - pt1[0]) < (last_cx - pt1[0]) *
(pt2[1] - pt1[1]) and \
        (cy - pt1[1]) * (pt2[0] - pt1[0]) >= (cx - pt1[0]) * (pt2[1] -
pt1[1]):
        total_in += 1

    last_positions[id] = (cx, cy)

    seat_available = seat - (total_in - total_out)
    print(f"Total in: {total_in}\nTotal out: {total_out}\nSeat available:
{seat_available}")
    data = {"AvailableSeat": seat_available}
    db.update(data)

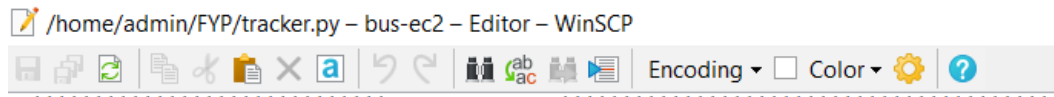
except Exception as e:
    print(f"An error occurred: {e}")

async def main():
    try:
        server = await websockets.serve(handle_connection, "0.0.0.0", 8765)
        print("WebSocket server started on port 8765")
        await server.wait_closed()

```

APPENDIX

```
except Exception as e:  
    print(f"Server error: {e}")  
  
asyncio.run(main())
```



```
import math  
  
class Tracker:  
    def __init__(self):  
        # Store the center positions of the objects  
        self.center_points = {}  
        # Keep the count of the IDs  
        # each time a new object id detected, the count will increase by one  
        self.id_count = 0  
  
    def update(self, objects_rect):  
        # Objects boxes and ids  
        objects_bbs_ids = []  
  
        # Get center point of new object  
        for rect in objects_rect:  
            x, y, w, h = rect  
            cx = (x + x + w) // 2  
            cy = (y + y + h) // 2  
  
            # Find out if that object was detected already  
            same_object_detected = False  
            for id, pt in self.center_points.items():  
                dist = math.hypot(cx - pt[0], cy - pt[1])  
  
                if dist < 35:  
                    self.center_points[id] = (cx, cy)  
                    print(self.center_points)  
                    objects_bbs_ids.append([x, y, w, h, id])  
                    same_object_detected = True  
                    break  
  
            # New object is detected we assign the ID to that object  
            if same_object_detected is False:  
                self.center_points[self.id_count] = (cx, cy)  
                objects_bbs_ids.append([x, y, w, h, self.id_count])  
                self.id_count += 1  
  
        # Clean the dictionary by center points to remove IDs not used anymore  
        new_center_points = {}  
        for obj_bb_id in objects_bbs_ids:  
            _, _, _, _, object_id = obj_bb_id  
            center = self.center_points[object_id]  
            new_center_points[object_id] = center  
  
        # Update dictionary with IDs not used removed  
        self.center_points = new_center_points.copy()  
        return objects_bbs_ids
```

Figure A-12 Source code of tracker.py

Survey of UTAR Bus and Shuttle Tracking System

Hi, my name is Chin Pei Shan, a final-year student from FICT-IA. I hope you can take around 5 minutes to fill out this form for my FYP project.

shansp010807@tutar.my [Switch account](#) 📁 Draft saved

🔒 Not shared

Before using the UTAR Tracker app, how long did you typically wait for the bus or shuttle (in minutes)?

- Less than 3 minutes
- 3 - 5 minutes
- 5 - 10 minutes
- 11 - 20 minutes
- More than 20 minutes

After using the UTAR Tracker app, how long do you typically wait for the bus or shuttle (in minutes)?

- Less than 3 minutes
- 3 - 5 minutes
- 5 - 10 minutes
- 11 - 20 minutes
- More than 20 minutes

Figure A-13 Survey part 1

APPENDIX

On average, how much time do you think you have saved using the app?

- Less than 3 minutes
- 3 - 5 minutes
- 5 - 10 minutes
- 11 - 20 minutes
- More than 20 minutes

How satisfied are you with the app's performance in reducing waiting time?

- Very satisfied
- Satisfied
- Neutral
- Dissatisfied
- Very dissatisfied

How easy was it to use the UTAR Tracker app?

Very easy 1 2 3 4 5 Very difficult

Would you recommend this app to others?

- Yes
- No
- Maybe

Figure A-13 Survey part 2

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3	Study week no.: 2
Student Name & ID: Chin Pei Shan 21ACB06723	
Supervisor: Dr Aun Yi Chiet	
Project Title: Shuttle IoT: UTAR Bus Management System	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Search information about Raspberry Pi 4 camera module.

2. WORK TO BE DONE

Purchase a camera module.

3. PROBLEMS ENCOUNTERED

No.

4. SELF EVALUATION OF THE PROGRESS

So far so good.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3	Study week no.: 4
Student Name & ID: Chin Pei Shan 21ACB06723	
Supervisor: Dr Aun Yi Chiet	
Project Title: Shuttle IoT: UTAR Bus Management System	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Install camera module on Raspberry Pi 4 and write the code to track the person.

2. WORK TO BE DONE

Calculate the number of remaining seats.

3. PROBLEMS ENCOUNTERED

Multiple tracks on a person.

4. SELF EVALUATION OF THE PROGRESS

Still in plan.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3	Study week no.: 6
Student Name & ID: Chin Pei Shan 21ACB06723	
Supervisor: Dr Aun Yi Chiet	
Project Title: Shuttle IoT: UTAR Bus Management System	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Fixed the bugs of multiple tracks on one person and successfully calculate total remaining seats in the bus.

2. WORK TO BE DONE

Display the data on Flutter app.

3. PROBLEMS ENCOUNTERED

The process is very slow when calculate the available seats.

4. SELF EVALUATION OF THE PROGRESS

So far so good.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3	Study week no.: 8
Student Name & ID: Chin Pei Shan 21ACB06723	
Supervisor: Dr Aun Yi Chiet	
Project Title: Shuttle IoT: UTAR Bus Management System	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Fixed the slow process by captured real-time video in bus using RPi 4 and upload the video into AWS S3 bucket.

2. WORK TO BE DONE

Choose a suitable AWS service to analyze person object.

3. PROBLEMS ENCOUNTERED

No.

4. SELF EVALUATION OF THE PROGRESS

So far so good.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3	Study week no.: 10
Student Name & ID: Chin Pei Shan 21ACB06723	
Supervisor: Dr Aun Yi Chiet	
Project Title: Shuttle IoT: UTAR Bus Management System	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Created AWS SQS to receive message after video successfully uploaded into S3. EC2 instance also created to save all the necessary file to calculate the number of available seats.

2. WORK TO BE DONE

Do final testing and write FYP2 report.

3. PROBLEMS ENCOUNTERED

No.

4. SELF EVALUATION OF THE PROGRESS

Still in my plan.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3	Study week no.: 12
Student Name & ID: Chin Pei Shan 21ACB06723	
Supervisor: Dr Aun Yi Chiet	
Project Title: Shuttle IoT: UTAR Bus Management System	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Completed final evaluation, test, and report.

2. WORK TO BE DONE

Submit draft to let supervisor do checking.

3. PROBLEMS ENCOUNTERED

No.

4. SELF EVALUATION OF THE PROGRESS

Overall good.



Supervisor's signature

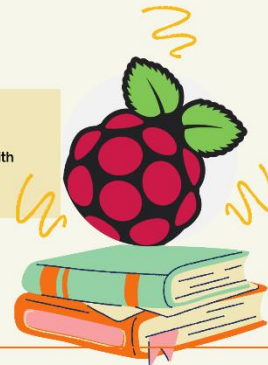


Student's signature

POSTER

SHUTTLE IOT: UTAR BUS MANAGEMENT SYSTEM

A real-time bus location tracking system, integrating IoT technologies within a Rapid Application Development (RAD) framework. The prototype includes a Raspberry Pi 4 paired with a Neo-M6 GPS HAT with Antenna that utilizes satellite communication channel to pinpoint geographical coordinates and a camera module to capture video for people counting.

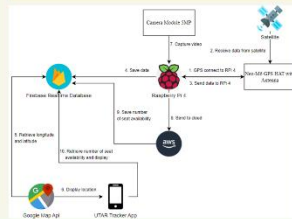


PROBLEM STATEMENT

- Inconsistent punctuality of transport arrivals.**
UTAR will predefine the schedules of each bus and send the schedules to students through UTAR email. The students can only look at the predetermined schedules to know the estimated arrival time of the targeted transport.
- Lack of real-time tracking applications**
UTAR do not have a real-time tracking application for tracking location of buses. The students do not know the current bus location and cannot estimate the arrival times of bus at bus stations, hence they can only wait until a bus arrives.

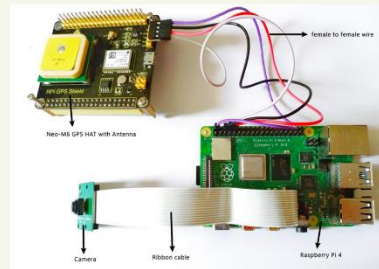
OBJECTIVE

- To develop a location tracking module for UTAR bus and shuttle using GPS-on-Pi.
- To integrate GPS coordinate on Google Map for intuitive tracking
- To develop a crowd counting method to count the number of passengers in real time using Yolov3



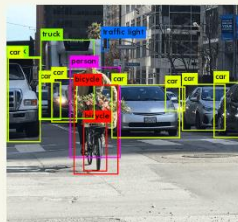
PRELIMINARY WORK

- Hardware:**
- RPI 4
 - female-to-female jumper wires
 - Neo-M6 GPS HAT with Antenna
 - Raspberry Pi power supply
 - Camera Module
- Softwares:**
- Android Studio Hedgehog version 2023.1.1
 - Flutter
 - Firebase
 - AWS Cloud



CONCLUSION

Overall, this new system has integrated IoT into buses to enhance transportation experience for students. The research objectives which are "to develop a location tracking module for UTAR bus and shuttle using GPS-on-Pi" and "to integrate GPS coordinate on Google Map for intuitive tracking" as well as "to develop a crowd counting method to count the number of passengers in real time using Yolov3" are achieved.



```

select admin@172.31.40.212 - ssh/ftesting
root@172.31.40.212: /# python3 calculate.py
WebSocket server started on port 8765
new connection
[ 480x640 (no detections), 323.5ms
speed: 0.9ms preprocess, 323.5ms inference, 0.9ms postprocess per image
total in: 0
total out: 0
not available: 44
[ 480x640 i clock, 167.3ms
speed: 2.6ms preprocess, 167.3ms inference, 20.9ms postprocess per image
total in: 0
total out: 0
not available: 44
[ 480x640 (no detections), 165.0ms
speed: 3.6ms preprocess, 165.0ms inference, 0.7ms postprocess per image
total in: 0
total out: 0
not available: 44
    
```

PLAGIARISM CHECK RESULT

PeiShan_Final_FYP.pdf			
ORIGINALITY REPORT			
4%	2%	2%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS
PRIMARY SOURCES			
1	Muhammad Fareez Mohd Ainul Hakeem, Norakmar Arbain Sulaiman, Murizah Kassim, Naimah Mat Isa. "IoT Bus Monitoring System via Mobile Application", 2022 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS), 2022 Publication	1%	
2	eprints.utar.edu.my Internet Source	1%	
3	fict.utar.edu.my Internet Source	1%	
4	Tejas Rathod, Vinay Patil, R. Harikrishnan, Priti Shahane. "Multipurpose deep learning-powered UAV for forest fire prevention and emergency response", HardwareX, 2023 Publication	<1%	
5	appmaster.io Internet Source	<1%	
6	Arduino Robotics, 2011. Publication	<1%	

PLAGIARISM CHECK RESULT

7	Irrnan N. Junejo, Hassan Foroosh. "Euclidean Path Modeling from Ground and Aerial Views", 2007 IEEE Conference on Computer Vision and Pattern Recognition, 2007 <small>Publication</small>	<1 %
8	umpir.ump.edu.my <small>Internet Source</small>	<1 %
9	link.springer.com <small>Internet Source</small>	<1 %
10	vdoc.pub <small>Internet Source</small>	<1 %
11	"Intelligent Embedded Systems", Springer Science and Business Media LLC, 2018 <small>Publication</small>	<1 %
12	Gulshan Shrivastava, Sheng-Lung Peng, Himani Bansal, Kavita Sharma, Meenakshi Sharma. "New Age Analytics - Transforming the Internet through Machine Learning, IoT, and Trust Modeling", Apple Academic Press, 2020 <small>Publication</small>	<1 %
13	Nana Yaw Asabere, Gare Lawson, Godwin Badu-Marfo, Lydia Kwofie, Daniel Opoku Mensah, Reginald Lartey. "Classification of Public Health Centres in Accra through a Web-Based Portal Integrated with	<1 %

PLAGIARISM CHECK RESULT

Geographical Information System (GIS)",
Journal of Healthcare Engineering, 2021
Publication

14	portal.bazeuniversity.edu.ng Internet Source	<1 %
15	www.mdpi.com Internet Source	<1 %
16	www.slideshare.net Internet Source	<1 %

Exclude quotes On
Exclude bibliography On

Exclude matches < 8 words

PLAGIARISM CHECK RESULT

Form Title: Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Chin Pei Shan
ID Number(s)	21ACB06723
Programme / Course	Bachelor of Information System (Honors) Information System Engineering
Title of Final Year Project	Shuttle IoT: UTAR bus management system

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceed the limits approved by UTAR)
Overall similarity index: <u>4</u> % Similarity by source Internet Sources: <u>2</u> % Publications: <u>2</u> % Student Papers: <u>0</u> %	
Number of individual sources listed of more than 3% similarity: <u>0</u>	
Parameters of originality required, and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note: Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Signature of Supervisor

Signature of Co-Supervisor

Name: Aun Yichiet

Name: _____

Date: 12/09/2024

Date: _____

CHECKLIST

FYP CHECKLIST



UNIVERSITI TUNKU ABDUL RAHMAN

**FACULTY OF INFORMATION & COMMUNICATION
TECHNOLOGY (KAMPAR CAMPUS)**

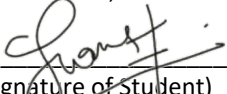
CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	21ACB06723
Student Name	Chin Pei Shan
Supervisor Name	Dr Aun Yi Chiet

TICK (√)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
/	Title Page
/	Signed Report Status Declaration Form
/	Signed FYP Thesis Submission Form
/	Signed form of the Declaration of Originality
/	Acknowledgement
/	Abstract
/	Table of Contents
/	List of Figures (if applicable)
/	List of Tables (if applicable)
/	List of Symbols (if applicable)
/	List of Abbreviations (if applicable)
/	Chapters / Content
/	Bibliography (or References)
/	All references in bibliography are cited in the thesis, especially in the chapter of literature review
/	Appendices (if applicable)
/	Weekly Log
/	Poster
/	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
/	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.


(Signature of Student)
Date: 11/9/2024