A Mobile App with Real-Time Vehicle Dispatching in a Self-Driving Delivery System

Jyun-Ting Po
Institute of Computer and Communication Engineering
National Cheng Kung University

With the development of self-driving technology in the recent years, a lot of related industries and the applications gradually popped up. Among the applications, the freight transport and the sale of driverless trucks became a crucial research topic. However, due to the complexity of the road network and the constraint of traffic regulations, self-driving technology development companies have the difficulties with the complete system integration. Therefore, this thesis proposes a mobile application in a self-driving delivery emulation system. By using the dispatching mechanism, the application simulates the real user scenario of the parcel delivery service. As a result, the application can be viewed as a test platform before the autonomous vehicle is officially launched. In the system, self-driving technology development companies can implement their own path planning algorithms and evaluate the effectiveness of the delivery process. Moreover, the user can interact with the system by the parcel delivery order from the mobile device, which makes the delivery situation more realistic. The thesis implements a real-time parcel delivery service in urban-scale areas, builds a database, and establishes the communication interface between the simulator and the mobile application. Besides, the service contains the whole delivery process, which includes the pick-up and unloading tasks of parcels. Finally, the simulation result shows the compatibility for the ordinary delivery applications and offered the flexibility to adjust different logistics planning algorithms.

Keywords—self-driving, real-time, mobile application, parcel delivery service, traffic simulators

Acknowledgments

First of all, I would like to express my gratitude to my advisor, Professor Kuo-Feng Ssu, for his valuable advices in my research and the encouragement during my time at National Cheng Kung University. I am exceptionally thankful for his endless effort to provide the guidance of the paper reading and the thesis writing.

Besides, I am really grateful to Professors Hewijin Christine Jiau, Ye-Qi Wu and Wei-Zhong Hu for not only their contributions to the research but also serving on my dissertation committee. I am obliged to the members of Dependable Computing Laboratory, including Yu-Jung Change, Che-Ming Lee, Wei-Lun Hung, Wen-Chin Lai, and Kang Yen. They helped me a lot and gave me many opinions in my research. Finally, I would like to thank my family and friends for their support.

Table of Contents

Chapter

List of Figures

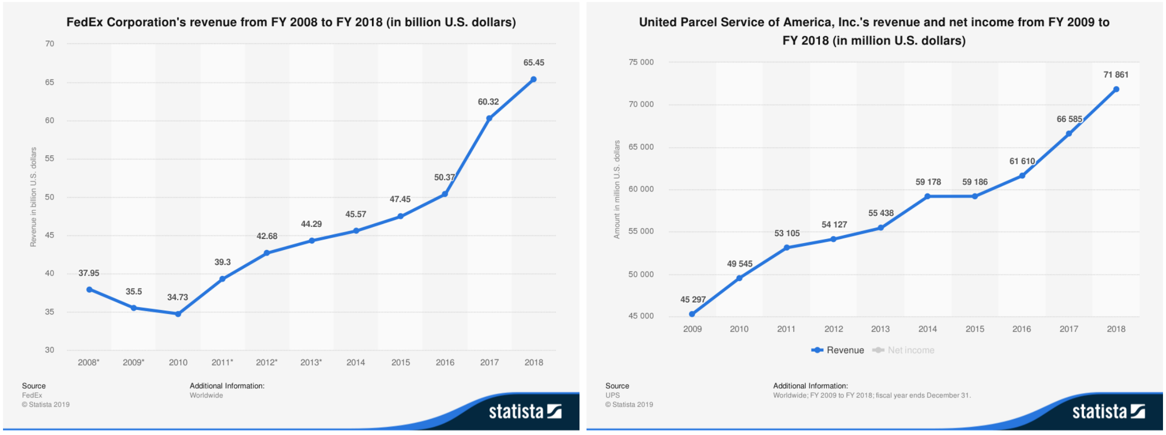
List of Tables

Chapter 1

Introduction

Nowadays the world's online marketplaces are growing rapidly because the online shopping has the advantages of the lower price and the rapid delivery service. The total e-commerce sales increased last year by 20% to $1.66 trillion according to Internet Retailer's released 2019 Online Marketplaces Report [1]. However, as the rapid development of the e-commerce such as Amazon, Alibaba and eBay, the novel problems and concerns occur. Compared to the tradition retailing, the online shopping encounters lots of challenges. For consumers' part, they must wait for the goods to arrive home instead of getting them immediately. Hence, the delivery time is a crucial factor for the consumer experience, and the parcel distribution becomes more and more important.

Figure 1.1 shows the revenues of two freight delivery companies, which have the highest market shares of couriers and local delivery service in the United States [2]. The trend chart represent that the demands of the express delivery service are growing year by year. In the domain of the express delivery, the same-day delivery is a significant service which aims to shorten delivery time as much as possible to improve the customer satisfaction.



(a) (b)

Figure 1.1: The revenue of top two couriers and local delivery service Inc. of America.

In the future, performing parcel delivery by self-driving vehicles is a promising way to improve the efficient of the route arrangement and to reduce the labor cost, thereby it can make it easier to achieve the same-day delivery.

The self-driving vehicle, also known as the robotic vehicle, the autonomous vehicle, or the driverless vehicle, is a vehicle which is able to sense the surrounding environment by sensors and drive on the road without a human driver. In Intelligent Transport System (ITS), the self-driving vehicle usually has one of many different communication technologies to communicate with other vehicles and the control center in the same system. With the self-driving technology becoming mature gradually, more and more companies invested in the development of self-driving vehicles, such as Google, Tesla, Waymo, Baidu, and so on.

However, in many countries, it is difficult and challenging to test the vehicles on public roads due to the complex road environment or policy restrictions. Therefore, this thesis proposes a self-driving delivery system which combines the mobile applications with the traffic simulation software. The system aims to simulate the process of the same-day delivery in urban areas. By this system, the self-driving technology development companies can test and adjust their path planning algorithms, vehicle deployments and the different system functions before their self-driving vehicles are official on the market. The mobile application is exploited to simulate the real usage situation and improve the user experience. In contrast with traditional home delivery service [3] which does not allow the parcel receiver to select the expected delivery time, the proposed application provides the real-time vehicle dispatching. The users including the sender and the receiver can select the arrival time of trucks at any time so that the failure rate of the delivery can be declined.

The simulation system contains the real-world road information, such as the longitude and the latitude of the vehicle, the cycle time of traffic lights, the different road types, and so on. The self-driving development companies can accomplish their expected functions by these data. Besides, this thesis designs an interface and the data format of the data transmission. Thus, the mobile application can be easily integrated into the actual self-driving transport system. By these advantages of this system, the self-driving vehicles can move seamlessly from the simulation to the real world.

The remainder of this thesis is organized as follows. Related work is discussed in Chapter 2. The system design, system functions and the usage scenarios are presented in Chapter 3. Chapter 4 describes the implementation of this system and the simulation environment. Finally, Chapter 5 concludes the thesis and discusses the future work.

Chapter 2

Related Work

According to Data Booklet of United Nations [4], the number of cities with at least one million citizens is 548 in 2018, and it will grow to 706 in 2030. As a result, the city logistic is getting more attention. The goal of city logistic issues is to find efficient and effective ways to transport goods in urban areas. There are some studies discussing the current opportunities and challenges of city logistics [5] [6].

Last mile is the delivery process that transports goods to the final destination within the city. Habault et al. presented an architecture of the delivery management system [7]. The system uses the data generated from the machine-learning mechanism to reduce the delivery travel time. Ewedairo et al. used a scenario thinking approach to identify the potential planning and transport system attributes that would obstruct the last mile delivery [8]. It mentioned that the modes of transportation have great impact on delivery costs. Hochstenbach et al. introduced the design of an unmanned aerial vehicle (UAV) for the autonomous parcel delivery [9]. Niels et al. presented a project in Munich, Germany, where the last mile package delivery is carried out by cargo bikes and eBikes [10]. GUO et al. presented a framework which exploits the underutilized capacity of crowdsourced public transportation systems (CPTS) such as the bus or the subway to perform the same-day parcel delivery [11].

Utilizing the self-driving vehicles to perform the parcel delivery is a novel and attractive research direction. Buchegger et al. proposed an autonomous transport vehicle which is capable of navigating in large-scale urban environments [12]. The vehicle in this study is based on a commercially available electric vehicle, and it had been adapted for autonomous operations. A new delivery framework for same-day delivery was proposed [13]. The goods are delivered to a nearby pickup station so that it can alleviate many of the shortcomings that autonomous vehicles have. Kocsis et al. presented a service in smart cities for delivering groceries using autonomous vehicles in urban areas [14]. The autonomous vehicles in this study was developed using an electric golf cart by the same authors.

Seth et al. presented a system, named Similitude, comprised of the SimMobility traffic simulator with Android emulators and an optional network simulator (ns-3) [15]. Compared to our system, Similitude focuses on the general transportation and it does not support the user interaction interface.

Chapter 3

System Design

3.1 System Overview

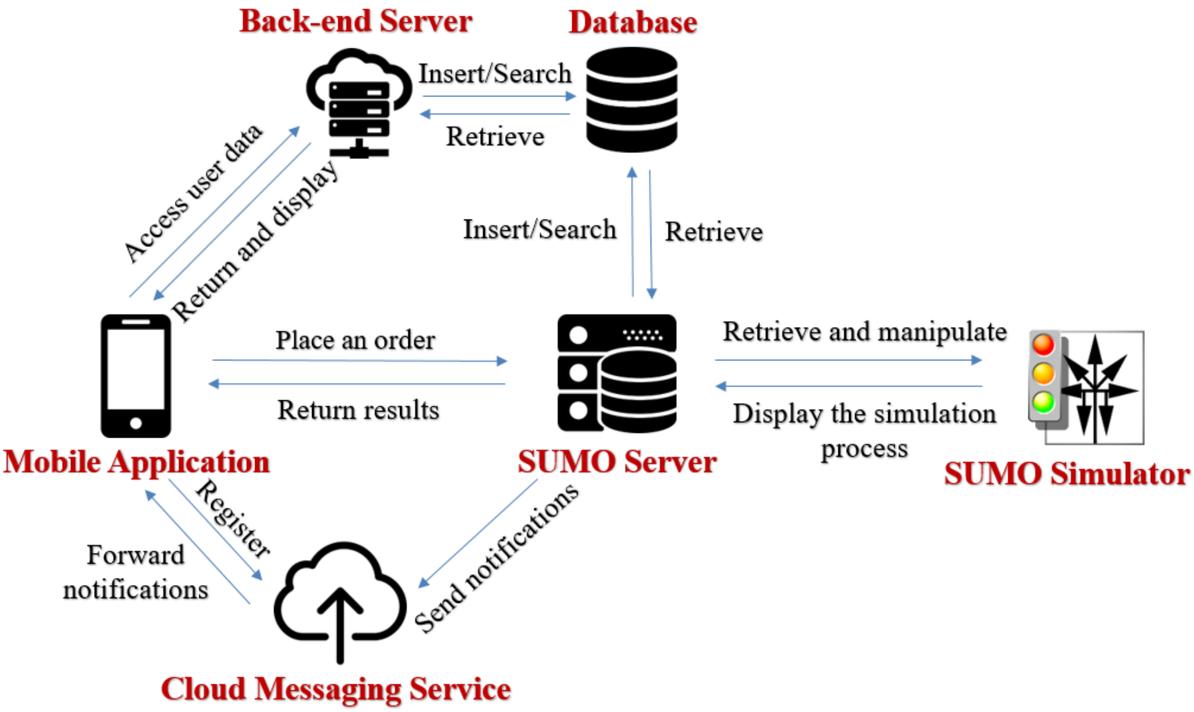


Figure 3.1: System overview.

This thesis designs a self-driving delivery simulation system, as illustrated in Figure 3.1. The system consists of five components, including the mobile application, the SUMO server, the SUMO simulator, the back-end server, and the database.

In this system, the mobile application is a platform in which the user can place a delivery order for shipping cargos and track the user's orders. After the user issues a delivery request on the mobile application, the request will be sent to the SUMO server. The server will execute the dispatching mechanism to assign a truck which meets the conditions to receive the cargo, and the movement of the truck will be displayed on the SUMO simulator.

Moreover, the information of orders will be uploaded to the database if the assignment is successful. The user then can check the order's status on the mobile application by connecting to the back-end server. When the truck arrives at the address of shippers or receivers on the simulator, the SUMO server will send a message to the mobile device for notifying the user of this arriving event through a cloud messaging service.

All the simulated process can be observed on the SUMO simulator. The features of each component will be described in detail in the following sections.

3.2 Mobile Application

In this system, the main object to develop the mobile application is to simulate the real usage scenario. The features of the mobile application include the registration and the login of user accounts, placing the delivery order, and tracking established orders. The functions of each part are depicted as follows.

3.2.1 Registration and Login

When the mobile application is launched for the first time, a token will be generated. If a user signs up and signs in his/her account, the token will be uploaded to the database. The SUMO server can send a notification to the mobile device by this token while the truck arrives at the target. On the other hand, the user-related data like username will be stored in the device's memory when the user signs in, and it can be used to generate the delivery order and track the user's history orders. The process will be described in the next two sections.

3.2.2 Placing a Delivery Order

In order to simulate the real usage scenario, the thesis implements a function that the user can place an delivery order. First, the user needs to fill out the order's information which includes container size, receiver's username, sender's address, receiver's address, and so on.

Then, the address will be displayed by Google Map service so that the user can confirm the correction of localization. After that, the user can choose the arrival time of the truck, and send the delivery request to the SUMO server. The request contains user-related data and the details of the order, and the server will perform the dispatching mechanism by these information and the road condition of the SUMO simulator. The order will be uploaded to the database if the request is accepted. Otherwise, the SUMO server will return a error message to the application to notify the user that there is no truck which can arrive the destination at the appointed time.

3.2.3 Tracking Established Orders

After the order is established successfully, the user can track the orders' status by using the function of tracking orders. In this function, users are divided into two categories such as a receiver and a sender.

Besides, if the user is a receiver, he can choose the pick-up time of the cargo after the truck arrives at the sender's address. Moreover, the receiver can only select the pick-up time after the cargo is loaded into the truck because the system must ensure the cargo is ready to be delivered. In addition, the system provides a function to examine the orders by inputting the order number instead of logging in. the function let the administrator search all orders of users conveniently on the mobile device.

3.3 SUMO Server and SUMO Simulator

In this system, the SUMO server is responsible for receiving the user's requests of delivery, executing the dispatching mechanisms, and updating the orders of users, and communicating with the SUMO simulator to get the data of road conditions and dispatch the truck to the destination on the simulator. This thesis deals with processing of received data and uploading the orders to the database.

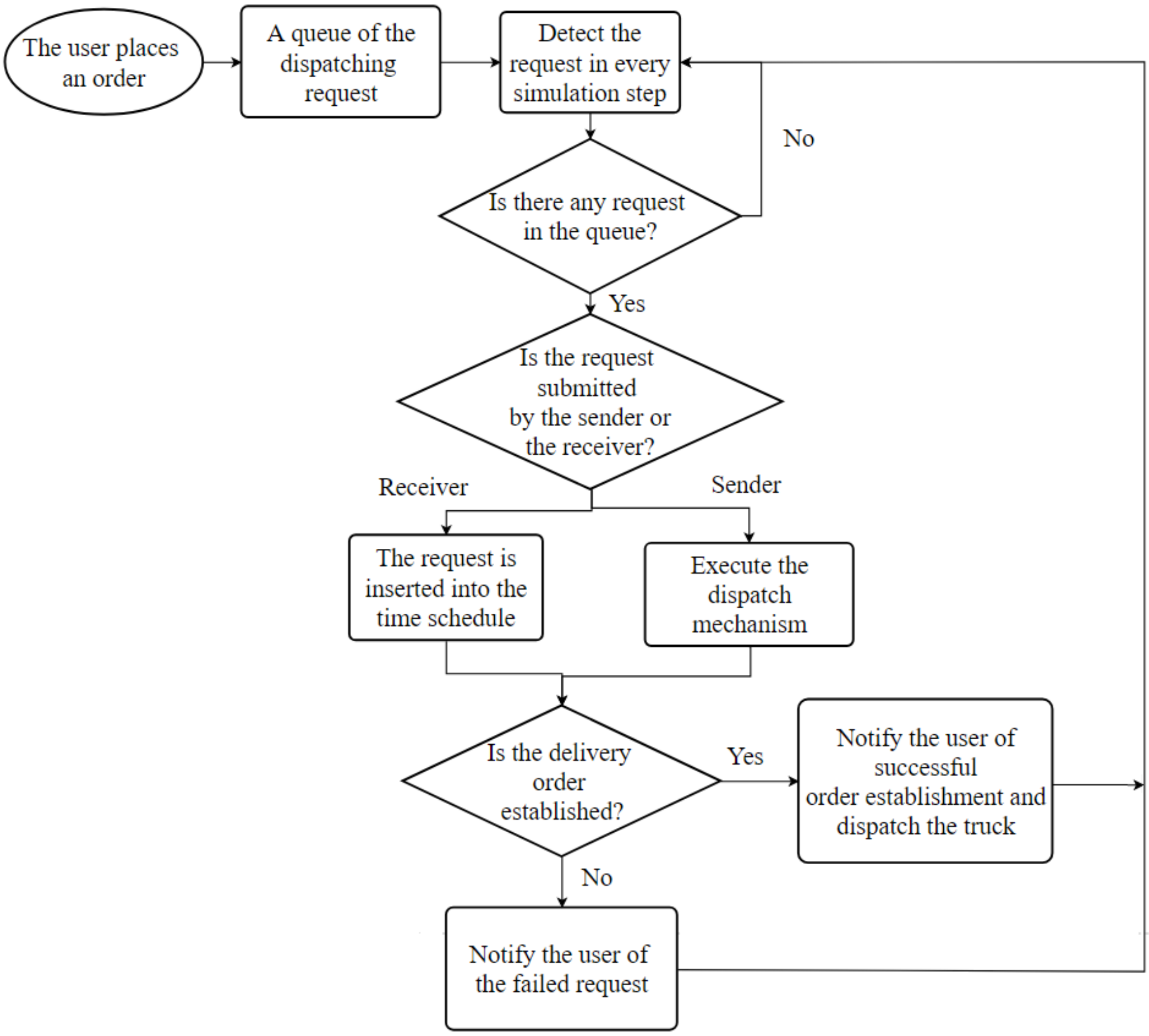


Figure 3.2: Flow chart of the delivery process.

As illustrated in Figure 3.2, after the user places an order with the SUMO server, the order will be stored in a queue of the dispatching requests. The SUMO server will distinguish between the sender's request and the receiver's request. If the request is sent by the receiver, the truck number and the container number would be stored because the server needs to know which trucks to be dispatched. The server will check the queue continuously, and do the dispatching mechanism if there are any requests in the queue.

If the dispatching is accepted, the server would insert or update the order information to the database with the database API and send a message to the mobile device through the socket. Otherwise, the server will notify the user that this arrival-time is unavailable with error message.

Moreover, the SUMO server updates the table of the dispatching schedule based on the above result. According to the table, the SUMO server will issue commands to change the routes of specific trucks in the simulator. After the delivery tasks are assigned to the trucks, they will drive along the designated routes. The system administrator can observe the trip and the parking place of each truck in the simulator. When a truck arrives at the destination, the SUMO server will send a notification to the mobile device through the cloud messaging service. The SUMO server will simulate the task of uploading and unloading. The status of the order will be updated to the database, and the server will notify the receiver to choose the preferred arrival time of the truck.

All of the above events will change the order, and the server will update them to the database each time. These changes can be observed on the mobile application by using the feature of tracking orders. Finally, the status of the order will be set to "received" by the server instead of being deleted when the delivery is over. Hence, the records of users' order can be viewed on the database or the mobile application. The implementation of the cloud messaging service will be described in Chapter 4.

3.4 Back-End Server

The SUMO server would spend much time executing the dispatching mechanism and simulating the package delivery process, and it will cause the computing delay. To keep the simulation precise, the gap between simulation time and actual time has to be shorten. Hence, the thesis builds a back-end server in order to reduce the workload of the SUMO server. In system, the back-end server is in charge of communicating between the mobile application and the database. It provides a interface by which the user can query or insert data to the database. The works of the back-end server include handling login and registration of user accounts, querying the order's data when the users track their orders, and showing the data of cities, counties and townships when the users are filling in an order form. The mobile application accesses to the database through the SUMO server only if users place a delivery order, so this process can reduce the computing burden of communication on the SUMO server as much as possible. The next section will introduce the design of data formats which are used when the back-end server executes the commands of the database.

3.5 Database

As discussed above, the simulation-related data in the system are stored in the database. The database is composed of four tables, they will be described and discussed in this section.

The first table named user-table keeps the account information of registered users. Table 3.1 shows the data format of user-table. As mentioned in Chapter 3.2.1, the attribute **device_key** is the token generated by the application. The token is assigned to the mobile device, and it will be updated when the user logs in on different devices. Therefore, the SUMO server can know which device is bound to the user by the unique token.

Table 3.1: User Table

Attribute	Description
user_id	The order of data stored in the table.
username	The name of the user.
password	A secret used to confirm the identity of a user.
gender	The state of being male or female.
phone_number	The number of the mobile phone.
register_time	The time when a user registered.
user address	The address where a user lives.
device_key	The registration token of the device produced by Firebase Cloud Messaging.

Table 3.2: County Table

Attribute	Description
id	The order of data stored in the table.
county_name	The name of the county in Taiwan.

Table 3.3: Township Table

Attribute	Description
id	The order of data stored in the table.
township_name	The name of the township.
county_name	The name of the county in Taiwan.

The second table and the third table are county-table and township-table. The data formats of the two tables are presented in Table 3.2, Table 3.3. The administrative district data of Taiwan is stored in the both tables. The data will be shown on the mobile application when the user selects the address of the shipper or receiver. Therefore, the typo can be reduced so that the process of geocoding can be performed better. The geocoding technology will be mentioned in Chapter 4. Furthermore, it provides the scalability of service regions, if the range of the simulation becomes larger, the administrator can easily expand regional options by adding new data to the database.

The last table is order table. As shown in Table 3.4, it retains the information of the user's order, and parts of the order details will be explained in this section. The attribute **status** represents the situation of the logistic activity, such as "on the way to the receiver's address". In addition, it is the criteria whether if the receiver can designate the arrival time. The attribute **sender_name** and **receiver_name** are used by the SUMO server to determine users which the server will notify. The latitude and the longitude transformed from senders' address is stored as **sender_lat** and **sender_lng**, and the same is true for the receiver' address. The attribute **sender_time** and **receiver_time** are the expected arrival time that users select on the mobile application. The SUMO server will exploit the locational and time information to generate a dispatching schedule and arrange the route of each truck.

Table 3.4: Order Table

Attribute	Description
id	The order of data stored in the table.
order_number	The unique number of package delivery order.
sender_name	The name of the sender.
receiver_name	The name of the receiver.
container_id	The unique container number.
in_time	The time when the cargo is load on to the truck.
out_time	The time when the cargo is unload the truck.
truck_id	The ID of the truck.
status	The shipping situation of the cargo.
weight	The weight of the cargo.
cargo_content	The content of the cargo.
size	The size of the selected container.
price	The amount of money for which the cargo is sold.
sender_lng	The longitude of the sender's appointed address.
sender_lat	The latitude of the sender's appointed address.
receiver_lng	The longitude of the receiver's appointed address.
receiver_lat	The latitude of the receiver's appointed address.
order_time	The time when the order is established.
sender_time	The time when the sender selects the expected the arrival time of the truck.
receiver_time	The time when the receiver selects the expected the arrival time of the truck.

Chapter 4

System Implementation

4.1 Mobile Application

Among all mobile operating systems, Android OS holds 86.7% market share according to the survey [16]. It is the most widely-used operating system in the world. Given that the self-driving delivery will be popularized in all walks of life soon after, Android OS is the most suitable system for developing the mobile application. The application is built by the Android Software Development Kit (Android SDK), and it is developed on Android Studio which is an Integrated Development Environment for Android. Android Studio provides a preview of the user interface and it has the built-in support of Google Cloud Platform. Consequently, it is an appropriate way to construct the mobile application of the system. Furthermore, Google Map SDK for Android and Google Geocoding API were added to the mobile application, the former enables the application to display the map of the simulation and mark the address, the latter is able to transform the street address into geolocations.

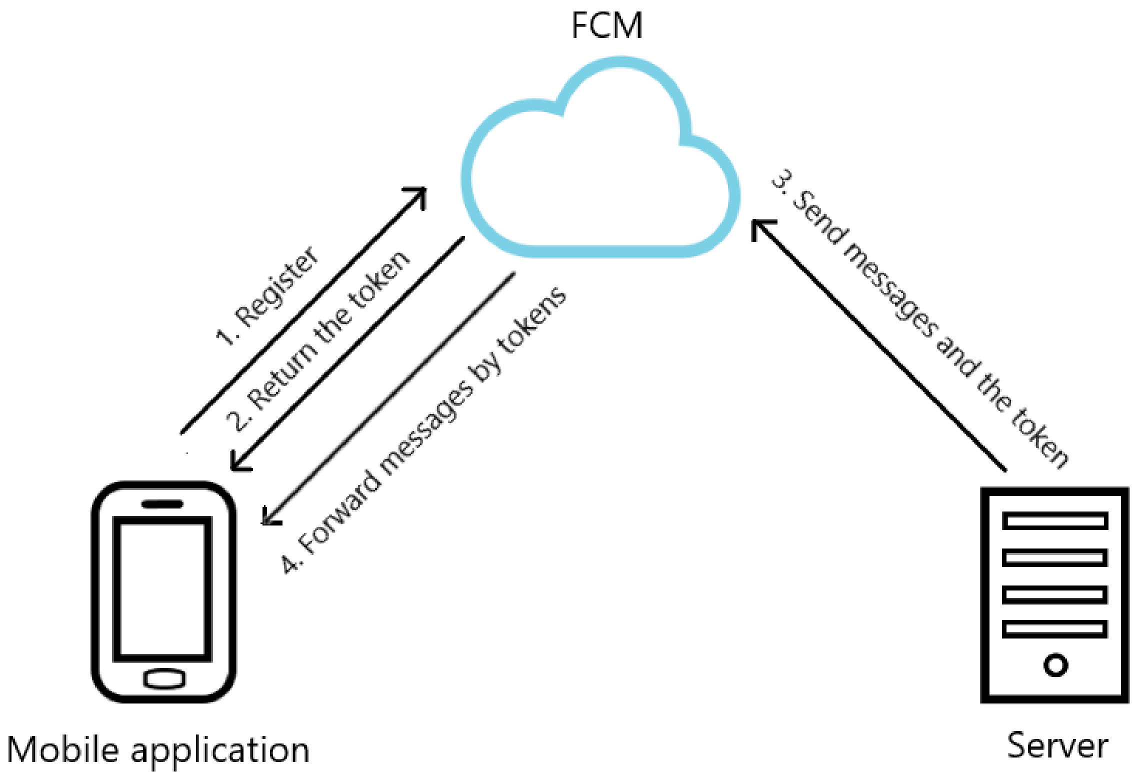
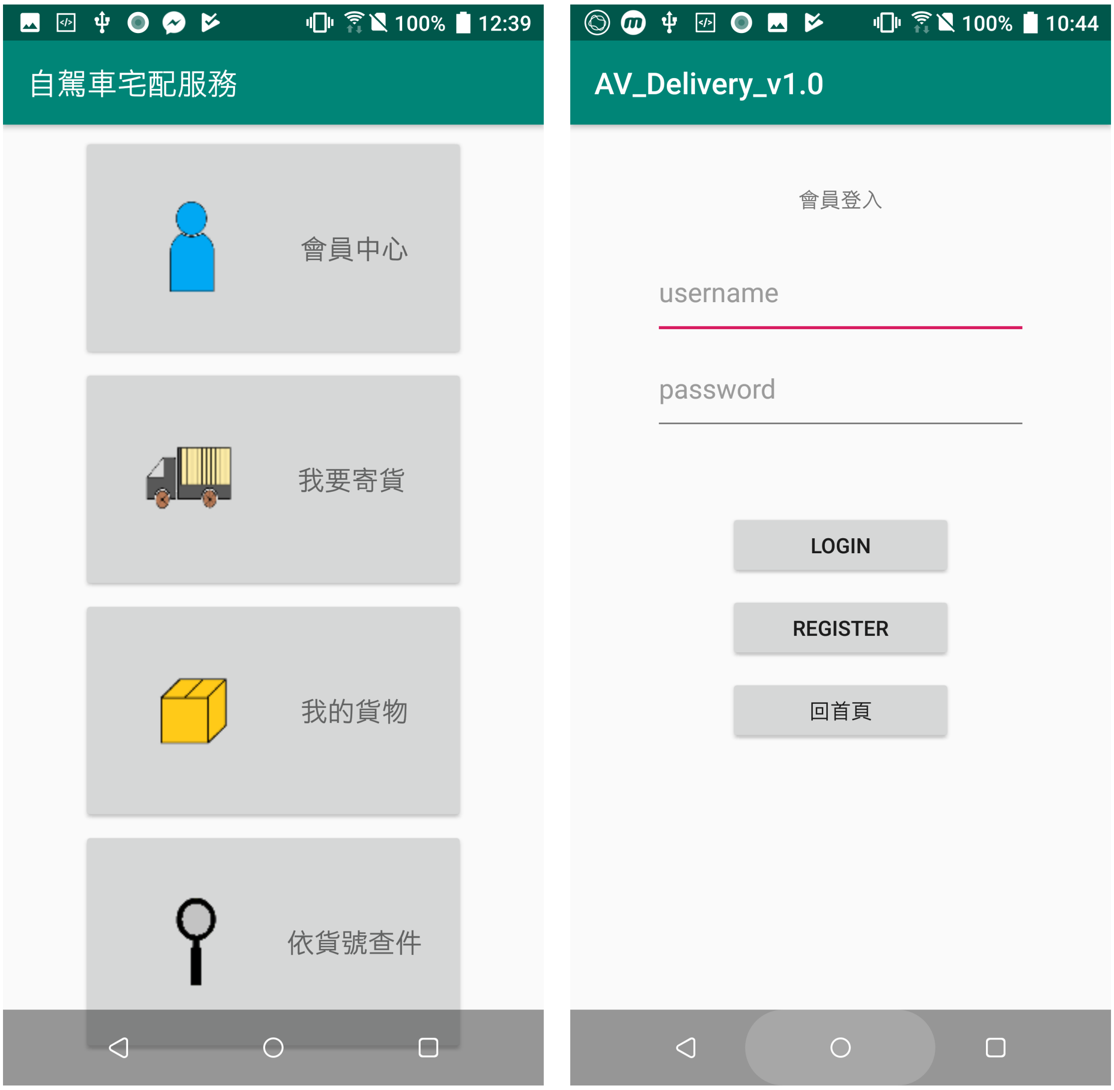


Figure 4.1: Firebase Cloud Messaging.

In order to notify the users that the truck has arrived to the destination, a cloud solution for messages and notifications is included in the self-driving delivery system. Firebase Cloud Messaging (FCM) is a service provided by Firebase, which is a subsidiary of Google, and it facilitates messaging between the mobile application and the server built by developers.

As illustrated in Figure 4.1, there are three components involving in the process of message passing, including the mobile application, FCM and the server that intends to send notifications (i.e., the SUMO server in this thesis). When the mobile application is installed on the mobile device and launched for the first time, the app will send a registration request to FCM through FCM SDK. Then, the FCM server will generate a corresponding token for the device, and return the token back to the mobile application. The token will be uploaded to the database if users log in. Conversely, when users log out, the token in the database will be removed to prevent the server from notifying the wrong users. According to the tokens stored in the FCM server, the messages sent from the SUMO server will be forwarded to the appointed device by FCM.

Figure 4.2 shows the initialization of the user interface in the mobile application. As displayed in Figure 4.2a, the homepage of the application is constituted by the buttons of the main functions. This study develops a simple registration function of user accounts, which is shown in Figure 4.2b. Before a user logs in, the FCM token is stored in the memory of the device. The user can be distinguished in different mobile devices with the unique token after logging in.



(a) Home Page. (b) Login.

Figure 4.2: Initialization of the user interface.

Figure 4.3 depicts the page of placing a package delivery order. The option of container size can be expanded based on the container type of the logistics companies.

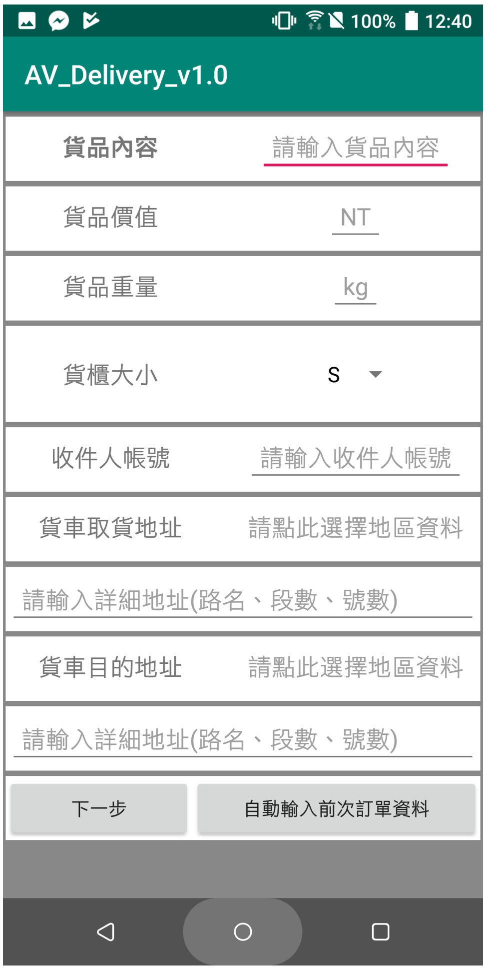
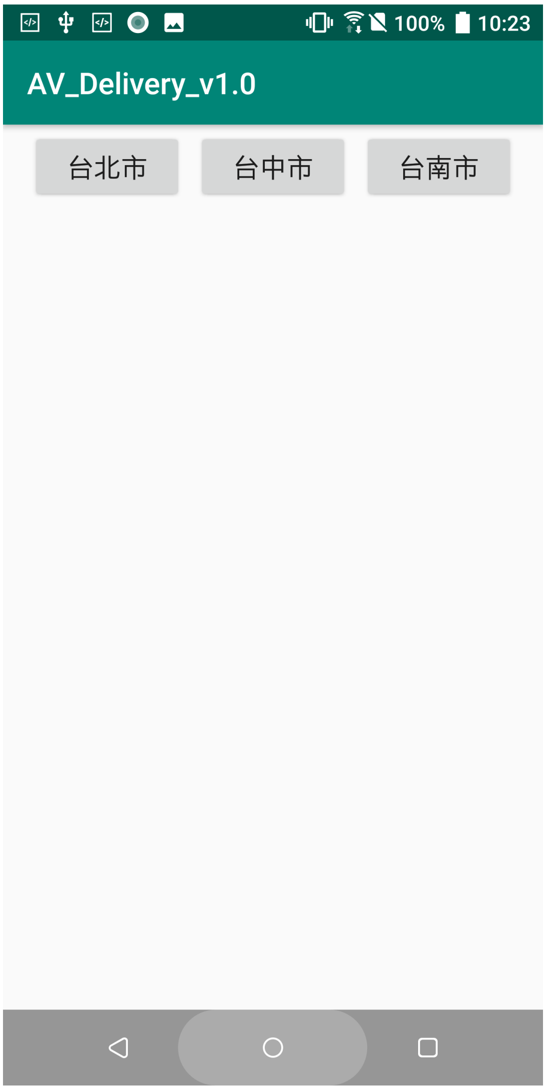
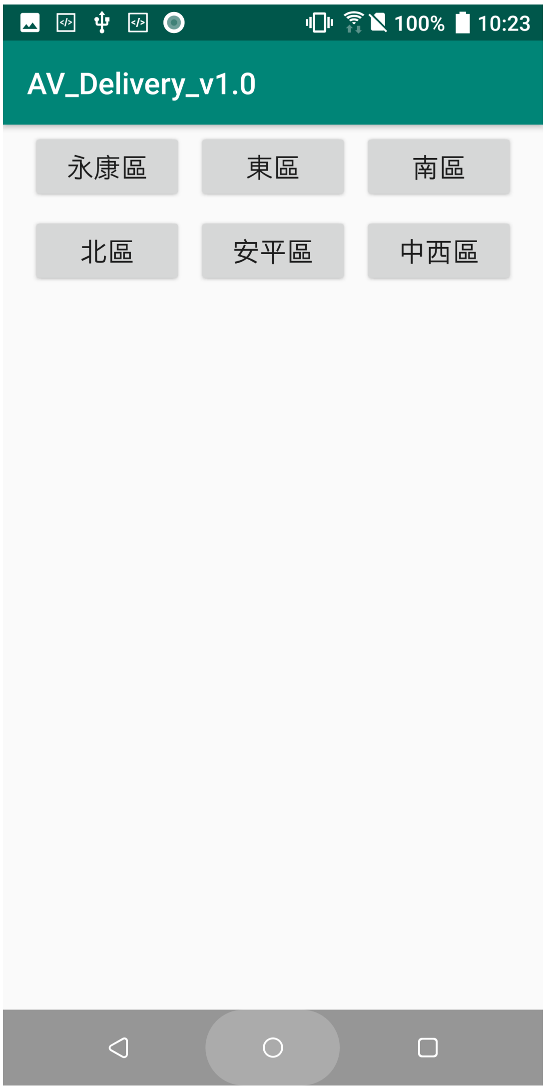


Figure 4.3: Place an order.

The user can press the **region selecting button** to get the administrative district of the simulation range, which is shown in Figure 4.4. This function make the user input data faster and reduce the typing errors. The simulation region contains parts of the downtown areas in Tainan City.



(a) City selection.



(b) District selection.

Figure 4.4: Region selection.

For the sake of convenience, the application will record the data entered by the user last time, and the data will be filled in the order automatically when the auto button is clicked. After filling out the order, the user can press the next button, and the application would check whether the user name of the receiver exists or not.

Figure 4.5: Localization of the destination.

As illustrated in Figure 4.5, the address that the user inputs will be transformed into geographic locations, and the shipper's address will be displayed on the Google Map. Then, the user can use the dropdown list to select the preferred arrival time of the truck. Finally, the order request will be sent to the SUMO server by the socket program. If there are eligible trucks in the simulated environment, the server would return a success message. Otherwise, the page will not be closed, and the user can choose other expected arrival time.

(a) The packages I sent.

(b) The packages I received.

Figure 4.6: Tracking history orders.

Figure 4.6 shows that the established order can be viewed on the page of tracking orders. This page has two cases. First case is to check the parcels the user sent. After sliding the tab above the page, the user can switches into the second case. The second case is to examine the history packages which the user received.

Figure 4.7: The details of the package.

Then, the details of the appointed package would be displayed while the user clicks any established order, which is shown in Figure 4.7.

If the user is a receiver in an order, he or she can choose the pick-up time of the cargo according to the status of the logistic activity, which means the shipping situation of the cargo. Then, the pick-up time page would pop up, which is the same as the page in Figure 4.5.

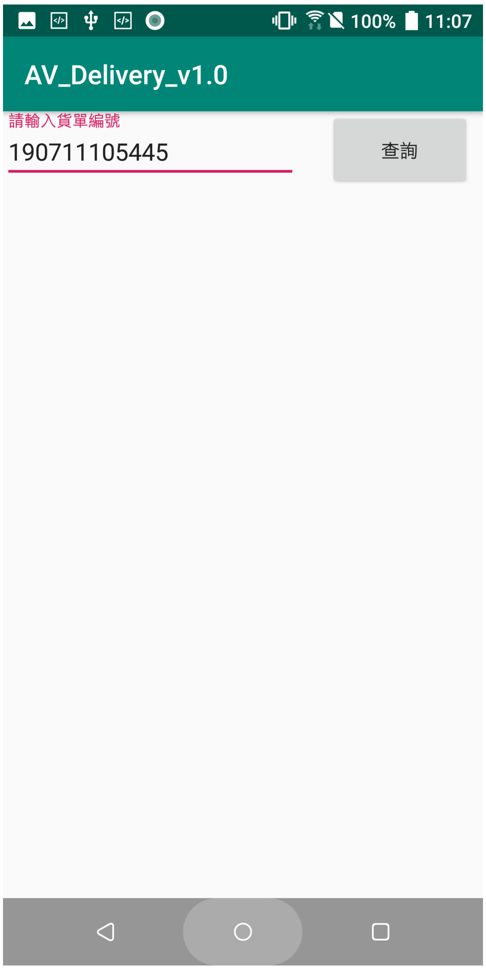


Figure 4.8: Entering the order number to track the package.

A system administrator can examine the history orders without logging in by the searching function. Figure 4.8 shows the searching page of the order. The order can be looked up by entering the unique order number.

4.2 SUMO Server and Simulator

4.2.1 Simulation Environment Setup

The simulation of the delivery process is implemented on SUMO [17], Simulation of Urban Mobility, which is an open source, highly portable, microscopic, and continuous traffic simulation package. The example of roads, vehicles, junctions, and other road information are included in the package. TraCI is an interface that provides the access to SUMO simulator by using a TCP based client/server architecture [18]. TraCI is implemented in various programming language, such as Python, C++, Java, and so on. In this thesis, TraaS is employed to be the control interface of the simulator. TraaS is a java library for working with TraCI, and it provides plenty of commands to manipulate the simulated objects. For instance, the developer can change the route of a vehicle by certain commands.

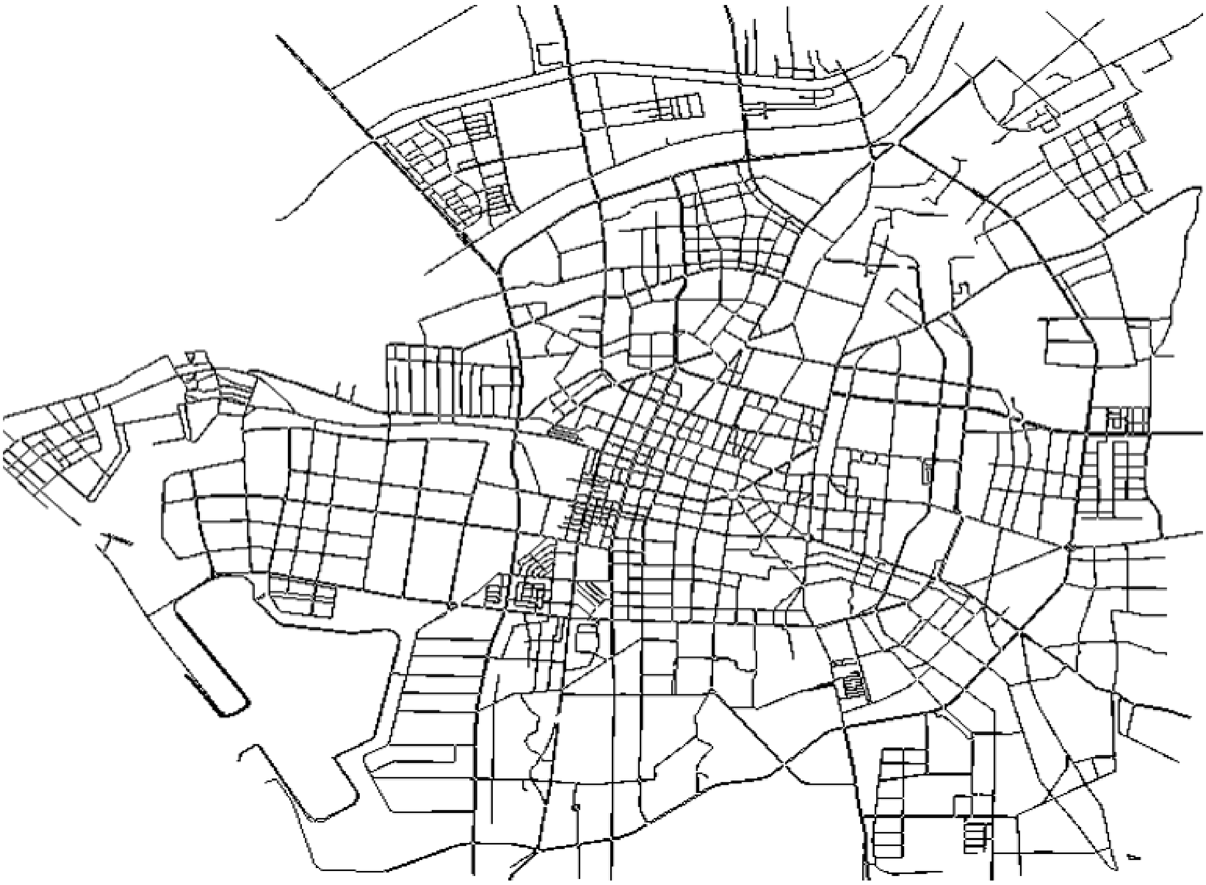


Figure 4.9: The service region on SUMO Map .

The self-driving delivery system is designed to supply a small-scale package delivery service in cities. As illustrated in Figure 4.9, the road network environment of the simulation is an approximate 50 km^2 area, which is a partial region of Tainan, a south city of Taiwan. The resource of the map data is gained from OpenStreetMap [19], which is a collaborative project to create a free editable map of the world. There are 25069 road segments and 6785 junctions in the simulation. The traffic lights are deployed on the junctions using the real world data. In the beginning of the simulation, three trucks are generated on the roads randomly, and they drive along random routes until the end of the simulation. The maximum speed of trucks is set to 5 m/s according to the average speed estimated by Google Map. The delivery task can be assigned in any time.

4.2.2 Build the Socket Server

For purpose of communicating between the mobile application and the simulator, the thesis builds a SUMO server to receive the data transferred from the mobile device which makes the simulator perform delivery task. The SUMO server is composed of a socket server and a main simulated program, which are both developed in Java language. This thesis copes with the socket server part and operations of database API.

When the simulation starts, the socket server will listen to the incoming connections from mobile applications. The socket server is programmed with a multi-client architecture, and the demand can be coped in a short period of time. Hence, the system can serve considerable users simultaneously. The user's request will be stored in the request queue when the user connects to the socket server, and the data format of the request is represented by JSON (JavaScript Object Notation). The simulated program would check the queue in each simulation second. If there is any request, it will be forwarded to the dispatching mechanism.

The order information would be uploaded to the database using JDBC if the dispatching execution is successful. JDBC (Java Database Connectivity) is an API for Java to access the database. The database and the SUMO server are built on the same computer because of the security concern. Then, the data can be transferred directly to the local side.

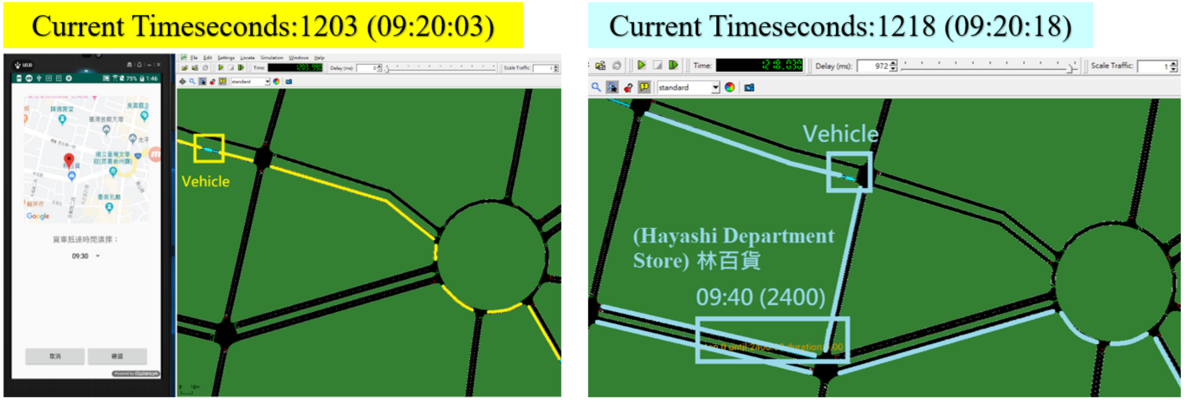


Figure 4.10: The changes of the route.

Figure 4.10 shows the change of the routes after dealing with the order request. In the left side of this figure, the vehicle follows the yellow route. However, after the dispatching command is appointed to the truck, the truck will follow the new route, whose color is blue in the right side of the figure. Thus, the simulation process will display on the GUI of the simulator so that the system administrator can confirm the information of trucks and containers.

Figure 4.11: The notification from FCM service.

Figure 4.11 shows that the mobile phone receives the notification from the FCM service. Moreover, the FCM service is also implemented in the SUMO server. When the truck is close to the destination, the system would notify the user of the coming message. Then, as the truck arrives to the destination, the system would inform the user of the arrival event via FCM server.

4.3 Back-end Sever

To avoid the leakage of the sensitive information of the database, users can not connect to the database directly. Thus, the system takes the back-end server as an interface to access to the database. The back-end server is a web server, which is built using Apache HTTP Server. The service on the back-end server receives query requests from users, and the service is implemented with PHP program. The mobile applications connect to the back-end server by HttpUrlConnection. It is Android function which makes the application communicate with the web server by HTTP protocol. According to the HTTP URL, the web server would run the PHP program to execute the SQL commands of the database. The result returned from the database will be forwarded to the mobile application by the web server.

4.4 Database

The database of the system is developed with MariaDB 10.1.32. MariaDB is a community-developed, commercially supported fork of the MySQL relational database management system. It inherits the advantages of MySQL. Hence, MariaDB supplies the JDBC driver, and the database API for PHP. This is highly useful for the development of our system. To deal with the administration of the database, phpmyadmin is employed in this thesis, it is a graphical, free and open source tool written in PHP. It allows the developer to manage the database with the use of a web browser. The records of orders can be exported for the subsequent data analysis.

The current goal of the system is to achieve the urban-wide delivery process. Therefore, the information of the administrative districts is stored in county table and township table of the database. There are three cities in county table currently, including Tainan City, Taichung City, and Taipei City. In the simulation of this thesis, only the map of Tainan City is constructed in the simulator, and the simulated districts involve East District, West Central District, North District, South District, Anping District and Yongkang District. The data are referenced from the Ministry of the Interior's Open Data (MOI Open Data).

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In order to make the self-driving technology development companies have a platform to test and adjust the product before it is officially on the market, a system was designed and implemented to simulate the freight process of the self-driving delivery. This thesis handled the developing of a mobile application and the data transmission between apps and the simulation system. In terms of the mobile apps, an account system was accomplished to store user information, and the order functions were devised to send and display orders. For the purpose of performing the dispatching mechanism, the SUMO server must be able to interact with mobile apps. Therefore, a socket server was built to receive and handle the data sent by apps. For similar reasons, FCM service was also be included in the system, and it can assist the SUMO server in notifying users of mobile apps. As regards the database, a back-end server was built to supply the access interface of the database to the mobile apps. In contrast, the SUMO server transfers data to the database directly because it is on the inner server side.

5.2 Future Work

There are some features that have not been implemented in this thesis. The features are not necessary for the delivery process, but they can be used to improve user experience or to manage the user data.

The functions of the real-time location inquiry and the arrival time prediction of the trucks are expected to be implemented. The information will be shown on the page of tracking orders. In addition, a management system will be established in the future, and it may be implemented on a web site. The management system allows the system administrator of the self-driving delivery system to view the data of the entire simulation environment. Furthermore, the administrator can change the truck's location, speed and even current routes. This function can allow the user to change the simulation deployment at any time during the runtime so that the simulation can be more flexible.

References

[1] "Key Findings from the 2019 Online Marketplaces Report," 2019, URL: https://www.digitalcommerce360.com/industry-resource/key-findings-from-the-2019-online-marketplaces-report/ [accessed: July 10, 2019].

[2] "U.S. Courier, Express and Parcels Market," 2019, URL: https://www.statista.com/topics/4063/courier-express-and-parcels-cep-market-in-the-united-states/ [accessed: July 10, 2019].

[3] L. Jiang, C. Liang, J. Dong, W. Lu, and M. Mladenovic, "A Disruption Recovery Problem with Time Windows Change in the Last Mile Delivery of Online Shopping," *Mathematical Problems in Engineering*, vol. 2018, pp. 1–10, 2018.

[4] United Nations-Department of Economic and Social Affairs-Population Division, "The World 's Cities in 2018 - Data Booklet," *The World 's Cities in 2018 - Data Booklet*, 2018.

[5] M. Savelsbergh and T. Van Woensel, "City Logistics: Challenges and Opportunities," *Transportation Science*, vol. 50, no. 2, pp. 579–590, Feb. 2016.

[6] T. G. Crainic, T. Bekt, T. G. Crainic, and T. V. Woensel, "From Managing Urban Freight to Smart City Logistics Networks," May 2015.

[7] G. Habault, Y. Taniguchi, and N. Yamanaka, "Delivery Management System Based on Vehicles Monitoring and a Machine-Learning Mechanism," in *Proceedings of 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. IEEE, Aug. 2018, pp. 1–5.

[8] K. Ewedairo, P. Chhetri, J. Dodson, and S. Rahman, "Building Last Mile Delivery Scenarios: A Case Study of Melbourne," in *Proceedings of 2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. IEEE, Dec. 2018, pp. 32–35.

[9] M. Hochstenbach, C. Notteboom, B. Theys, and J. De Schutter, "Design and Control of an Unmanned Aerial Vehicle for Autonomous Parcel Delivery with Transition from Vertical Take-off to Forward Flight – VertiKUL, a Quadcopter Tailsitter," *International Journal of Micro Air Vehicles*, vol. 7, no. 4, pp. 395–405, Dec. 2015.

[10] T. Niels, M. T. Hof, and K. Bogenberger, "Design and Operation of an Urban Electric Courier Cargo Bike System," in *Proceedings of 2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, Nov. 2018, pp. 2531–2537.

[11] G. Cheng, D. Guo, J. Shi, and Y. Qin, "Planning City-Wide Package Distribution Schemes Using Crowdsourced Public Transportation Systems," *IEEE Access*, vol. 7, pp. 1234–1246, Dec. 2019.

[12] A. Buchegger, K. Lassnig, S. Loigge, C. Mühlbacher, and G. Steinbauer, "An Autonomous Vehicle for Parcel Delivery in Urban Areas," in *Proceedings of 2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, Nov. 2018, pp. 2961–2967.

[13] M. W. Ulmer and S. Streng, "Same-Day Delivery with Pickup Stations and Autonomous Vehicles," *Computers and Operations Research*, vol. 108, pp. 1–19, Aug. 2019.

[14] M. Kocsis, J. Buyer, N. Sußmann, R. Zöllner, and G. Mogan, "Autonomous Grocery Delivery Service in Urban Areas," in *Proceedings of 2017 IEEE 19th International Conference on High Performance Computing and Communications*. IEEE, Dec. 2017, pp. 186–191.

[15] S. N. Hetu, V. S. Hamishagi, and L.-S. Peh, "Similitude: Interfacing a Traffic Simulator and Network Simulator with Emulated Android Clients," in *Proceedings of 2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*. IEEE, Sept. 2014, pp. 1–7.

[16] "Worldwide Smartphone Shipment OS Market Share Forecase," URL: https://www.statista.com/topics/4063/courier-express-and-parcels-cep-market-in-the-united-states/ [accessed: July 10, 2019].

[17] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo–Simulation of Urban Mobility: an Overview," in *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, Oct. 2011.

[18] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, "Traci: an Interface for Coupling Road Traffic and Network Simulators," in *Proceedings of the 11th communications and networking simulation symposium*. ACM, Apr. 2008, pp. 155–163.

[19] M. Haklay and P. Weber, "OpenStreet Map: User-generated Street Maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, Oct. 2008.

Vita

Jyun-Ting Po was born on November 15, 1992, in Taoyuan, Taiwan. He received the B.S. degree in Information Engineering from Chang Geng University, Taiwan, in 2017. In September 2017, he enrolled at the National Cheng Kung University as a master student in the Institute of Computer and Communication Engineering.