DEVELOPMENT OF A LIGHTWEIGHT AND EASY-TO-USE SMARTPHONE APPLICATION FOR TRAFFIC DATA COLLECTION: EZDATCOL

by

Berke Kaan Ülgen

B.S., Civil Engineering, Boğaziçi University, 2019

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Civil Engineering
Boğaziçi University
2021

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor Assoc. Prof. Ilgın Gokaşar for her continuous support and guidance throughout my undergraduate and graduate study. She provided me many opportunities to use my skills and work in a field that I enjoy. Her valuable ideas, vision and sincerity made all the classes I took and all the projects I participated in invaluable experiences. It was an honor to learn from her and work with her. I hope that we can work in many more projects together in the future.

I would like to thank Tekirdağ Municipality for their collaboration and support in the case study we conducted in Çorlu. Thanks to them, I was able to collect and process big traffic data and gain a precious experience. I would like to also thank Ari Bagamyan who kindly helped me conduct another case study by utilizing the application I've developed.

I would like to thank my friends in BOUN ITS Lab for their constant support, love, and respect. I thank Gazi Tülü for the excessive help and support he provided while conducting the case studies. I would like to also thank Alperen Timuroğulları, Emel Sadıkoğlu, Işık Okur, Semih Özkan, Burak Altın and Rahmi Şahin for their friendship and help throughout my graduate study.

Finally, I would like to thank my parents and my sister who encouraged me to pursue a graduate degree and supported me throughout this journey. They taught me the importance of being an honest and hardworking person. They have always believed in me, so I will always make them proud by achieving more.

ABSTRACT

DEVELOPMENT OF A LIGHTWEIGHT AND EASY-TO-USE SMARTPHONE APPLICATION FOR TRAFFIC DATA COLLECTION: EZDATCOL

Data analyses in transportation engineering require large amounts of recent traffic data of high quality, which are often unavailable. Conventional data collection methods such as in-vehicle GPS trackers are time-consuming and costly, which postpone the data analyses. These methods also often require additional training for the drivers, which delays the analysis parts even more. EZDatCol (Easy Data Collector) is a lightweight and easy-to-use Android application that can start collecting traffic data immediately after the APK file is distributed. As the application was designed to be compatible with Android devices of varying computing power and to be used all day long, the system demand of EZDatCol was set very low. EZDatCol collects GPS data continuously along with optional traffic data such as the number of passengers and stores collected data in an online database. In addition to the smartphone application, a Python-based toolkit was developed to automatize processing collected data. The functionality of the application and the data processing toolkit was tested in three case studies. The results of the case studies indicate that the application can successfully collect continuous traffic data for varying transport mode, duration, and distances. The Python-based toolkit also showed its usefulness in quickly processing big traffic data and exporting relevant spreadsheets, graphs, and maps to be examined by the traffic analysts. The Android application and the data processing toolkit can be easily implemented by future studies to be used during traffic data collection and processing.

ÖZET

TRAFİK VERİSİ TOPLAMAK İÇİN HAFİF VE KULLANIMI KOLAY BİR AKILLI TELEFON UYGULAMASI GELİŞTİRİLMESİ: EZDATCOL

Ulaştırma mühendisliği alanındaki veri analizleri, doğru sonuçlar verebilmek için yüksek miktarda kaliteli ve yakın zamana ait veriye ihtiyaç duyar. Araç içi GPS takipçileri gibi geleneksel veri toplama yöntemleri maliyetlidir ve uzun bir hazırlık süreci gerektirir, bu durum da veri analizin ötelenmesine yol açar. Aynı zamanda sürücülere, bu cihazlarla ilgili ilave eğitim verilmesi gerekebilir, bu durum da veri analizinin daha da ötelenmesine neden olur. EZDatCol (Kolay Veri Toplayıcı), APK dosyasının dağıtılmasının hemen sonrasında trafik verisi toplanabilmesini sağlayan, hafif ve kullanımı kolay bir Android uygulamasıdır. EZDatCol farklı işlemci gücüne sahip Android cihazlarla uyumlu olmak ve tüm gün kullanılmak üzere tasarlandığından dolayı bu uygulamanın sistem gereksinimi çok düşüktür. EZDatCol, aralıksız olarak GPS verilerinin yanı sıra yolcu sayısı gibi isteğe bağlı opsiyonel trafik verileri toplar ve topladığı verileri bir çevrimiçi veri tabanında saklar. Akıllı telefon uygulamasına ek olarak, toplanan verilerin işlenme sürecini otomatize eden Python tabanlı bir araç seti geliştirilmiştir. Mobil uygulamanın ve veri işleme araç setinin işlevselliği, üç farklı vaka çalışmasında test edilmiştir. Vaka çalışmalarının sonuçları, ulaşım biçimi, süresi ve mesafesi fark etmeksizin uygulamanın aralıksız trafik verisi toplayabildiğini göstermektedir. Python tabanlı araç seti ise, hızlıca büyük trafik verilerini işlemek ve trafik analistleri tarafından incelenmek üzere bu verilerle alakalı çizelgeleri, grafikleri ve haritaların çıktılarını alma konusunda yararlılığını göstermiştir. Android uygulaması ve Python tabanlı araç takımı, veri toplama ve işleme faaliyetlerinde faydalanmak üzere gelecek çalışmalar tarafından kolayca kullanılabilir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

API	Application Programming Interface
DIANA	Divisive Analysis Clustering
DP	Density-Independent Pixels
EZDATCOL	Easy Data Collector
GIF	Graphics Interchange Format
GIS	Geographic Information System
GPS	Global Positioning System
HTS	Household Travel Survey
IDE	Integrated Development Environment
ITS	Intelligent Transportation System
JSON	Javascript Object Notation
OS	Operating System
PPI	Pixels-Per-Inch
QR	Quick Response
SDP	Scalable Density-Independent Pixels
SP	Scale-Independent Pixels
SSP	Scalable Scale-Independent Pixels
UI	User Interface
XLSX	Extensible Markup Language Spreadsheet
XML	Extensible Markup Language

1. INTRODUCTION

Traffic data analyses in transportation engineering require processing big traffic data of high quality to yield accurate results. Such data, however, are not available unless collected for previous studies taking place in the same region. Collecting new data takes a great amount of time and organization, delaying the analysis for a large period. Furthermore, the recency of the collected data is lost after a certain period, requiring a new data collection study.

These problems can be easily solved by collecting traffic data continuously without intervening in traffic operations. In the case of requiring traffic data for analysis, up-to-date mass data would be available after such an implementation.

Mobile phones are capable of collecting GPS (Global Positioning System) data and they are always within reach of the driver. A lightweight, easy-to-use mobile application allows continuous GPS data collection along with optional data (e.g., number of passengers) provided by drivers. Such an application requires a one-time installation, and can be used every day after a brief tutorial. Using such an application in traffic data collection also saves the cost of installing external devices such as GPS trackers in vehicles.

In this thesis, a mobile application compatible with Android devices to be used during traffic data collection was developed. This application continuously collects GPS data and stores collected data in an online database. The practicality and the functionality of the application are shown with three case studies.

1.1. Goals and Objectives

The availability of traffic data of high quality is very important for traffic analyses. The goal of the thesis is to develop a method that allows collecting traffic data fast and conveniently. To achieve this goal, the following objectives are aimed:

(i) To develop a lightweight and easy-to-use Android application that can continuously collect traffic data
(ii) To conduct alpha and closed beta tests of the application
(iii) To develop a Python-based toolkit that automatizes processing collected traffic data
(iv) To test the functionality of the application and the toolkit in three case studies

1.2. Organization of the Thesis

The remainder of this thesis is organized as follows: Literature review on GPS data collection and processing is given in the next chapter. Then, the theory of the methods used in this thesis is stated in Chapter 3. Following thesis theory, the methodology of every step of this thesis is explained in detail in Chapter 4. Then, case studies conducted using the application and the data analyses of these case studies are given in Chapter 5. Finally, conclusions and recommendations are included in Chapter 6.

2. LITERATURE REVIEW

The location of users, vehicles, and equipment can be tracked using today's GPS technology. The GPS tracking units receive signals from satellites, calculate a position, and keep these calculations as coordinates. More than one method can be followed in these location calculations. Collected GPS data can be used for many different purposes. For instance, the trajectory data of GPS-equipped buses were used to analyze the influence distance of bus stops and the traffic behavior of buses around them [1]. As another example, GPS data collected through GPS devices carried by participants were used to carry out a route and mode choice analysis [2]. There exist many GPS data collection methods implemented by previous studies. Utilizing GPS trackers, custom GPS units, GPS sensors, third party smartphone applications and newly developed smartphone applications are some of the GPS data collection methods [3–6]

Collecting GPS data via GPS trackers, sensors or similar physical devices has been preferred by many studies. Cui *et al.* used GPS data collected through GPS devices equipped by taxis in Harbin, China to examine inconsistencies between urban travel demand and transport services. The GPS devices had been implemented to all licensed taxis as a security measure and they record the location data every 30 seconds [3]. Jun *et al.* collected GPS data through an in-vehicle trip data collector to observe the speed patterns of drivers [7]. Castro *et al.* collected GPS data through an in-vehicle GPS device to estimate the geometric model of a highway located in Madrid, Spain [8]. Alshibani and Moselhi collected location data via GPS units mounted on one truck in each fleet and used the collected data to estimate cycle times for each round trip of earthmoving operations to be used in time and cost estimation [9]. Rasmussen *et al.* developed a GIS-based algorithm that detects trip legs and travel modes and applied the algorithm to the GPS data collected through wearable GPS devices equipped by the participants [10]. Zhu collected GPS data through in-vehicle GPS devices in Minneapolis, United States to observe the behavior of commuters after the collapse and reopening of a bridge [11]. Huang and Levinson used the collected data to model home-based, non-work destination choice [12]. Shen *et al.* collected activity-travel diary of commuters through surveys and GPS trackers to observe day-to-day temporal, spatial, modal, and route flexibility in Beijing, China [13]. He *et al.* collected travel data of private passenger cars through GPS loggers to generate individual trip chain distributions to analyze customer acceptance for battery electric vehicles and evaluate the energy consumption of plug-in hybrid electric vehicles [14]. Patnaik *et al.* used Divisive Analysis Clustering (DIANA) to classify large amount of speed data collected through a GPS unit [15]. Liu *et al.* used long-term GPS tracking data and digital elevation map data to research the effect of road gradients on the electricity consumption of electric vehicles [16]. Necula implemented a statistical approach on 10.000 vehicle GPS traces from 3.600 drivers to analyze the traffic patterns on street segments [17]. Guo *et al.* applied an unsupervised deep learning model to the data gathered by GPS sensors in Shenzhen, China to study driving behavior and risk patterns [5]. Carli *et al.* proposed an algorithm to automate the analysis and evaluation of the congestion in urban areas and applied the algorithm to the GPS-generated data provided by a local transit bus tracking system [18]. Lu *et al.* used taxi GPS data to trace macroscopic traffic in large-scale and complex urban networks [19]. Ciscal-Terry *et al.* used a dataset of low frequency GPS coordinates to recognize vehicle trajectories and analyze the route choices of drivers [20]. Hast *et al.* collected population movement data through GPS loggers to identify the relationship between individual movement patterns and malaria risk in a high-transmission area in Zambia [21]. Ma *et al.* applied a series of data-mining algorithms to extract trip-chaining information from massive truck GPS data sets of several companies obtained through in-vehicle GPS devices [22]. Liu *et al.* proposed an algorithm based on Adaptive Kalman Filter to improve the precision of navigation information. Road tests were carried out and the algorithm was applied to the GPS data of autonomous vehicles [23]. Du and Aultman-Hall developed three methods to identify trip start points and these methods were applied to the GPS travel datasets collected in Kentucky, United States between 2002 and 2003 [24].

Collecting GPS data with smartphone applications is another method that gained popularity in recent years. Flake *et al.* compared trip rates obtained through Household

Travel Survey (HTS) and rMove, a travel data collection application developed by Resource Systems Group [25]. Gong *et al.* collected GPS data every 30 seconds through smartphones carried by participants. The application installed on the smartphones allowed users to enter destinations, trip purposes, and travel modes. Collected data were used to train a machine learning algorithm that identifies trip purposes and travel mode [6]. Korpilo *et al.* used GPS tracks collected through multiple smartphone applications (e.g. Sports Tracker, Strava) installed on the smartphones of volunteer participants to examine spatial patterns on paths runners and mountain bikers follow [26]. Shafique and Hato used GPS data collected in Kobe, Japan through an application installed on the smartphones of the participants to train an algorithm that detects travel modes [27]. Jackson *et al.* collected GPS data in Montreal, Canada through Mon RésoVélo, an application installed on the smartphones of cyclists [28]. Collected data were used by Strauss *et al.* to estimate bicycle volumes and injury risk of cyclists in the entire network [29]. Stipancic *et al.* collected GPS data to examine vehicle maneuvers via Mon Trajet, a smartphone application developed by Brisk Synergies [30]. Pluvinet *et al.* collected GPS data through a smartphone application to research the contribution of GPS survey techniques to urban freight route characterization and diagnosis [31]. Chen *et al.* used vehicle trajectories obtained through the smartphones of the drivers to analyze the particle matter emission of on-road vehicle braking events [32]. To build an Intelligent Transportation System (ITS), Pham *et al.* developed a smartphone application as a part of traffic data collection framework that incorporates traffic cameras, sensors, GPS data of vehicles and individuals [33].

In recent years, various technologies making use of Global Positioning System have been used in location data collection studies. While obtaining travel data containing location and time, device-related problems such as misreporting and unresponsiveness may be encountered. In addition, financial problems that stem from the cost of physical devices may arise in developing countries. Moreover, keeping track of whether the tracker is working properly is also problematic for the drivers. Smartphone applications, on the other hand, are easy-to-use and free. As the smartphone technologies became widespread, smartphone applications have become a great alternative for GPS trackers and other physical devices.

Nonetheless, existing third-party smartphone applications may not be appropriate for the project requirements as the data collection period and frequency cannot be adjusted. The application also may not be suitable to be used all day long. Additionally, problems may arise while retrieving the collected data. Mass data collection may be restricted by the application. Furthermore, collected GPS data may be shared with the owners of the application or public, which poses a threat to the privacy of users. Another privacy concern is that these third-party applications may request or force the users to share personal data. The application developed in this thesis can be used in any GPS data collection study regardless of travel duration or mode. It can also be adjusted by the project specifications, if desired. The application can be used all day long as it has low battery consumption. Finally, the application does not request any personal data from the user and the collected GPS data can be accessed by authorized personnel only.

3. THEORY

Availability of mass traffic data is very important for the accuracy of traffic analysis results. Unfortunately, there are many obstacles such as time and money cost and bureaucracy that impose difficulty on collecting traffic data or accessing collected traffic data. Having a smartphone application capable of collecting traffic data continuously can be very convenient for future traffic analyses.

EZDatCol (Easy Data Collector) developed in this thesis is an Android application that collects GPS data every second and stores collected data in an online database. The application was specifically developed to be highly backward compatible, easy-to-use, and battery-life-friendly. Hence, it is suitable for traffic data collection of any transportation mode, without any limitations such as time, distance, speed, vehicle type, or the number of passengers.

In addition to the Android application, a set of Python scripts were developed to quickly process the mass data collected by the application. Android application and Python scripts together create a great toolkit that easily collecting GPS data and processing the collected data easily. Development of the tools introduced in this thesis consists of the following steps:

(i) Developing Android application
   - Developing and linking user interface
   - Developing activities
   - Implementing miscellaneous features such as handling errors and permission checks

(ii) Conducting alpha and closed beta tests

(iii) Developing data processing Python scripts

The flowchart of the toolkit development is given Figure 3.1.

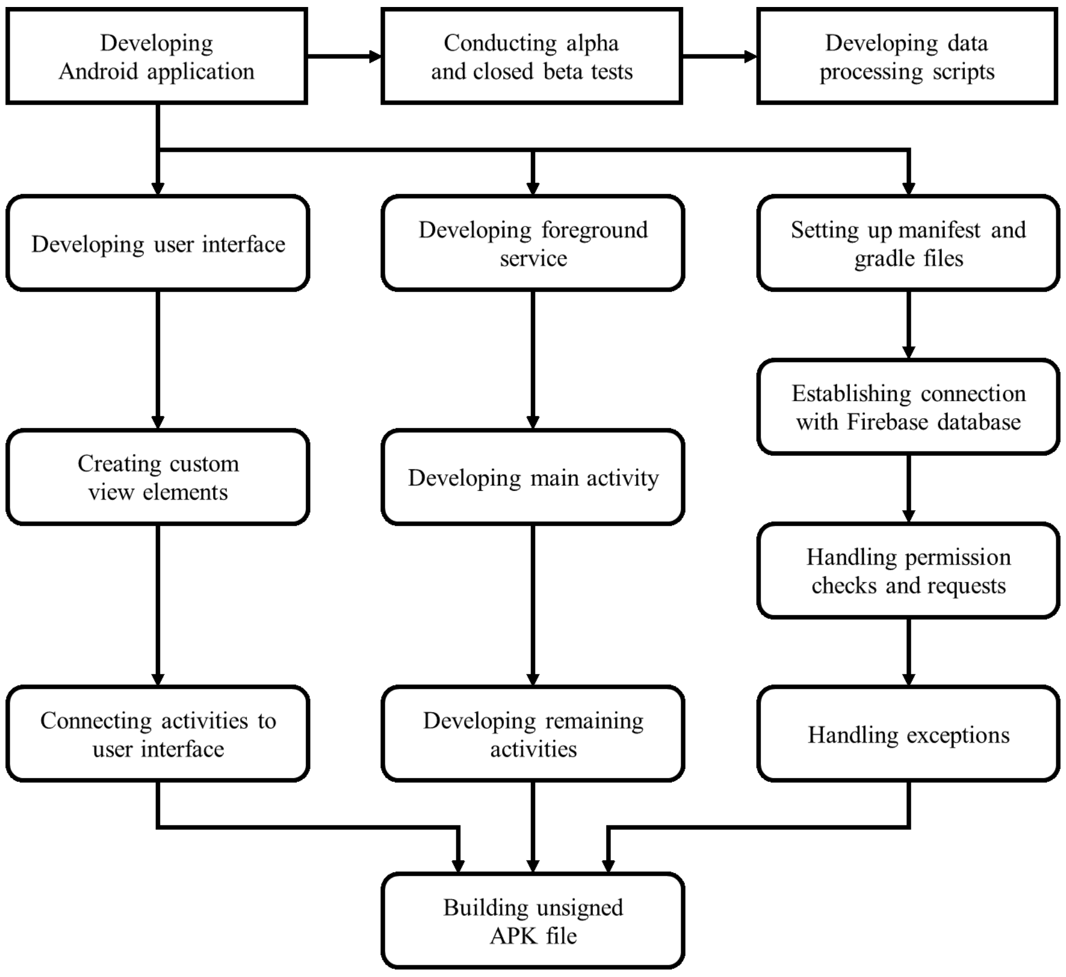

Developing Android application
Conducting alpha and closed beta tests
Developing data processing scripts
Developing user interface
Developing foreground service
Setting up manifest and gradle files
Establishing connection with Firebase database
Creating custom view elements
Developing main activity
Handling permission checks and requests
Connecting activities to user interface
Developing remaining activities
Handling exceptions
Building unsigned APK file

Figure 3.1. Toolkit development flowchart.

Selection of the programming language is important because each language has advantages and disadvantages in certain fields. An Integrated Development Environment (IDE) is software that allows developing programs in certain languages faster and more conveniently with the help of tools they provide such as code editor, compiler, developer console, and debugger. Hence, the selection of the IDE is also important as it can shorten the application development period significantly.

3.1. Developing Android Application

The smartphone application was developed on Android Studio version 4.1. Although other platforms can be used while developing an Android application, Android Studio is very convenient thanks to the various tools it provides. It also gets frequent updates published by Google.

Kotlin language was used while developing activities. Although the activities can still be developed with Java, Google increased the functionality and compatibility of Kotlin in the development of Android applications in recent years and it is possible that Google with continuing with only Kotlin in the future. XML language was used while developing UI.

Google Firebase was chosen as the online database platform to be used along with the application as it is easy to implement in Android applications. It also has low processing power demand thanks to high compatibility and does not require additional operations to communicate with the database such as running an additional Node.js script on the server-side. Free Google Firebase plan provides a realtime database size of 1 Gigabyte and enables up to 100 simultaneous connections.

3.2. Conducting Alpha and Closed Beta Tests

Alpha tests of the application were done on virtual devices via the emulator provided by Android Studio, and on physical devices connected to the computer in debugging mode. Debugging the application was done through the internal debugging tool in Android Studio. A checklist including certain tests concerning many different topics such as functionality, design, permissions, and limitations stemming from specific Android OS versions was followed. The application was tested in various devices and the tasks included in the test checklist were repeated a certain number of times order to check whether an error or a bug is encountered.

Closed beta tests were done on the physical devices of volunteer participants. The application was installed on the smartphones of testers through the unsigned APK file of the application. Signing the APK file is not necessary as the application will not be published in Google Play Store in the future. The participants were asked to use the application continuously for a specified duration of time and give feedback on certain features of the application such as functionality, accessibility, user interface, and ease of use.

3.3. Developing Data Processing Scripts

Python 3.6 was used to create scripts that process and modify the database exports. Python scripts handle files very fast and require minimal installation. Additionally, gmaps binary provides a very easy toolkit for creating custom maps with given datasets. Visual Studio Code is used as the IDE.

Multiple Python scripts were created to split data processing into smaller parts. This enables manually verifying the validity of the script output after each run, imposes smaller amounts of workload on the computer for each run, and reduces the time loss in case of encountering errors stemming from corrupted JSON or XLSX files.

4. METHODOLOGY

EZDatCol was designed to be lightweight and easy-to-use. In addition to the GPS data, the application also collects the number of passengers if the user chooses to provide information. Users are also able to select the genders of the passengers. Collecting such additional data can be very useful for analyses done for certain transportation modes such as a taxi. The reason that the number and gender of passengers were chosen to be collected is that both features are distinct and can be easily provided by the driver.

Another feature that was implemented in the application is a system that allows passengers to provide additional information about themselves if they choose to do so. Collecting such information is possible through anonymous surveys, which also ensures privacy between the driver and passengers. Survey questions can be adjusted by the project specifications.

4.1. Developing Android Application

To be used in a wide range of devices with varying Android versions, the minimum API level was chosen to be 17, which corresponds to Android 4.2 Jelly Bean released in November 2012. The targeted API level was selected as 29, which corresponds to Android 10 released in September 2019. The selected API level range was compatible with 99.2% of all Android devices as of September 2020.

The application consists of multiple interconnected activities. Most of the activities have a layout shown in the user interface (UI), which provides the users visual feedback and enables user input. Some activities run in the background, in other words not shown in the user interface, and some activities are not launched but accessed by other activities. The activity hierarchy of the application is given in Figure 4.1.

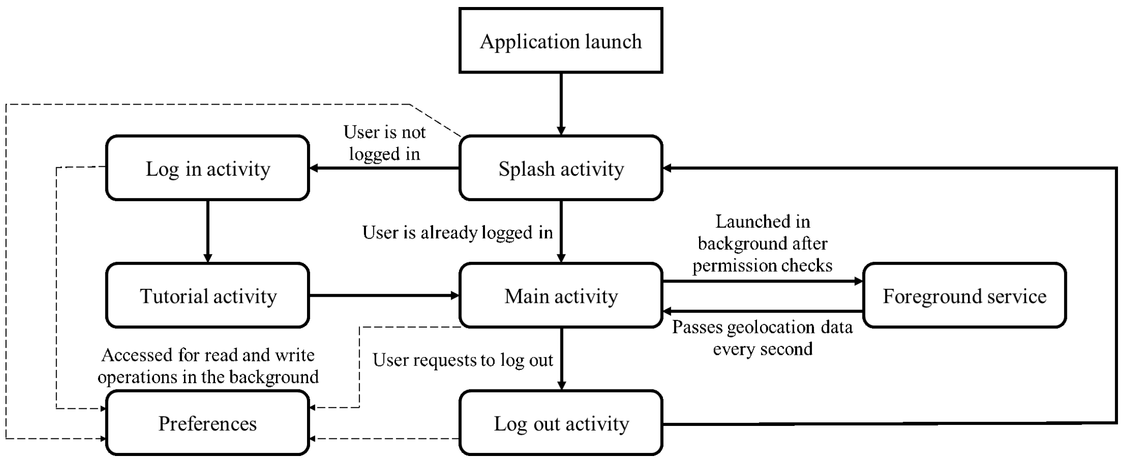


Figure 4.1. Activity hierarchy of the application.

As the survey questions may be modified in time, the survey link needed to be dynamic. Thus, it was stored in the database. It also needed to be easily accessible by the passenger. Even if the survey link was shortened beforehand, typing it into a mobile phone browser is very inconvenient. Hence, a system that generates QR (Quick Response) code of the link stored in the database was implemented in the application. The driver asks the passenger whether they want to participate in the survey, and can generate the QR code of the survey with a single tap. The passengers can access the survey within seconds by reading the QR code with their mobile phone's camera.

4.1.1. Setting Up Manifest and Gradle Files

AndroidManifest.xml file contains information about the application. All activities and services must be included in the manifest file to be launched. Launch activity was also stated here, which was splash activity in this case. Default settings except for screen orientation were used for most of the activities. Portrait mode was selected for all activities in order not to force the users to adjust their screen orientation constantly while driving.

Launch mode of the main activity was selected to be "singleTask" to run a single instance of it at a time and not restart it when the user returns to it from other activities or the home screen.

The “stopWithTask” attribute of the foreground service was selected as “True” to kill the service when the application is closed. This enables collecting data only when the application is running. Forcing the application to collect GPS data continuously even when the user closes the application is unacceptable as it is a big privacy breach.

Permissions of the application, which are given in detail in Section 4.1.7, were stated in the manifest file. Icon, name (i.e., label), and theme of the application were adjusted here.

Building configurations were stated in build.gradle files. One build.gradle file is for the top-level whereas the other one is for the module-level. External binaries used in the application were stated within module-level build.gradle file. In this case, external binaries were rarely used as the application was designed to be simple and battery-friendly.

Following external binaries were used in the application in addition to the ones defined by default:

(i) Binaries related to location services
- com.google.android.gms:play-services-location:*
- com.google.android.gms:play-services-maps:*

(ii) Binaries related to Firebase database
- com.google.firebase:firebase-database:*
- com.google.firebase:firebase-database-ktx:*

(iii) Binaries used in tutorial screen
- me.relex:circleindicator:*
- androidx.viewpager2:viewpager2:*
- pl.droidsonroids.gif:android-gif-drawable:*

(iv) Binaries used in QR code generation

- org.jetbrains.anko:anko-commons:*
- com.google.zxing:core:*

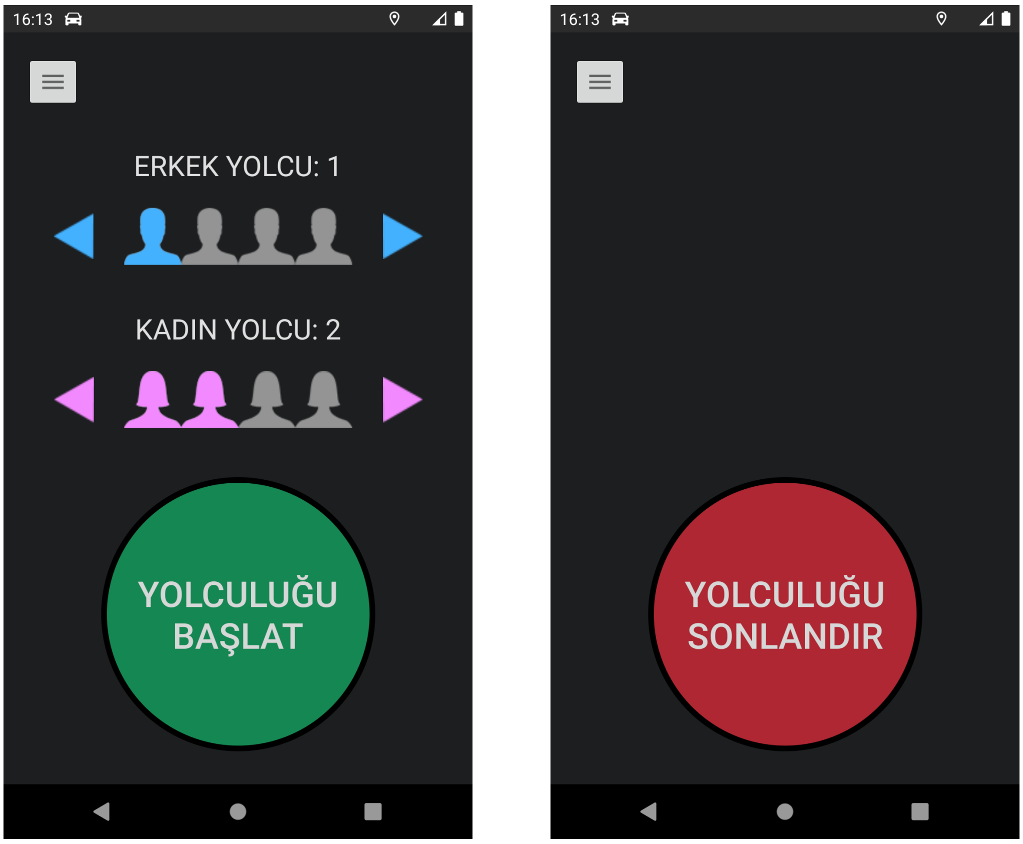
(v) Binaries used for scalable text and views

- com.intuit.sdp:sdp-android:*
- com.intuit.ssp:ssp-android:*

It should be noted that asterisks (*) correspond to the latest version for each binary.

4.1.2. Developing User Interface

The application was designed to be easily usable in traffic. Hence, it does not contain any audiovisual pollution. The main screen is composed of the number of male and female passenger selectors (i.e., rating bars), a large, round, green main button that is used to start a trip, and a menu button that reveals or hides buttons belonging to miscellaneous functions such as generating QR code and logging out. Rating bars get hidden when a trip is started and the main button is replaced with a red one in order for the driver to use the application easily in traffic. Starting and ending the trip requires the users to approve their actions through an alert box to prevent problems due to misclicks. The main screen in idle and trip mode is given in Figure 4.2.



(a) (b)

Figure 4.2. The main screen in idle mode (a) and trip mode (b).

Constraint layout was used in all activity layouts to prevent problems in screens with very high or very low pixel density. Scalable sp (scale-independent pixels) and scalable dp (density-independent pixels) were used for texts and UI elements (i.e., views), respectively. These units abbreviated as ssp and sdp, respectively, were implemented through appropriate binaries provided by Intiuit.

4.1.3. Creating Custom View Elements

Custom rating bar and button styles were created for the application. For the custom rating bar styles, male and female avatar icons were used as the rating bar stars. The background and tint colors of these avatars were adjusted to be easily distinguishable from each other and the remaining view elements. For each custom button style, different shapes and radii were used depending on the place of use. Stroke and background colors were also adjusted accordingly for neutral, focused, and pressed button states.

4.1.4. Developing Foreground Service

Google restricted the frequency of background location requests strictly with Android 8.0 Oreo (API level 26) to reduce power consumption. Hence, a service was created for the application as frequent location requests are needed. Service is an extension for an application with no user interface that runs simultaneously with the application itself. Foreground services can keep running in the foreground even if the application is brought to the background.

A custom service class that extends to the original service class was created in the application. Upon starting the service, it starts requesting precise location every second until it is stopped when the application is closed or the main activity is destroyed. Foreground services are launched along with a service notification; thus, a notification channel and a notification builder were set up within the service. Additionally, for the service to communicate with the main activity, a broadcast manager was set up within.

To achieve a better user experience and privacy, the application was designed to request location updates in the background so that the users can use other applications in the meantime, and stop requesting when it is closed so that the users can stop sharing location data and lower their power consumption. This was achieved by setting "stopWithTask" attribute of the service to "true" in AndroidManifest.xml. Additionally, the service is launched with "onStartCommand" function which returns "START_NOT_STICKY". This ensures that the system does not try to relaunch the service after it is killed. Content title and content text shown as the texts in notification title and description, respectively, were written to describe the purpose of the notification to the user. The importance of the notification was set to "IMPORTANCE_LOW" as higher notification importance levels make a sound when the notification is created, which was not desired in this case. Notification intent was set for the main activity, which means that the user is redirected to the main activity upon pressing the notification.

Upon creating the service “startLocationUpdate” function is run with “INTERVAL” and “FASTEST_INTERVAL” attributes of 1000. This results in requesting location updates every second. Besides, “PRIORITY” was set to “PRIORITY_HIGH_ACCURACY”, which means that the requested location data have the highest accuracy possible. Whenever “onLocationChanged” function is called, received location data are broadcasted via custom broadcast manager set up in the service to be received by a broadcast receiver within the main activity.

4.1.5. Establishing Connection with Firebase Database

The database reference and credentials are stored in the Google services JSON file provided by Firebase after setting up the database and stored in the application. An instance of the Firebase database is created within main activity after setting up permission checks and requests the details of which are given in Section 4.1.7. Database reference and credentials are retrieved from the mentioned JSON file.

4.1.6. Developing Main Activity

The main activity creates an instance of the preferences class upon launching to execute read and write operations in the device storage. A broadcast receiver compatible with the foreground service was set up within the main activity. Every time the main activity receives a broadcast from the service, a function that writes to the database is called. This write function creates a node with the value of the saved plate value. Under the plate node, the function creates a node with the value of the current date and another node under the date note with the value of the timestamp. Finally, trip parameters are written under the timestamp node. Depending on many parameters such as connection quality, Android version and computing power of the phone, the interval between consecutive write operations may rise for several seconds.

The trip parameters that the write function passes are current latitude, current longitude, number of male passengers, number of female passengers, and a Boolean indicating whether the trip is started or not. The trip parameters were optimized to reduce the size of the database and mobile data usage. Each second, a device uploads 80 bytes of data to the database, which results in using less than a third of a megabyte of mobile data every hour.

When the application is launched, writing to the database starts immediately if the user has already logged in. 0 is used as the default value for the number of male and female passengers until the user starts a trip. After the user starts a trip, the number of male and female passengers specified by the user is passed to the database. The number of passengers and the time trip starts are also stored in the preferences class.

A timer (i.e., fixed-rate time) runs in the main activity to remind the user that the trip is continuing every 10 minutes in case they forget to end the trip when the customer gets off the vehicle.

If the user closes the application while a trip is continuing and relaunch the application, they are asked whether they want to resume the previous trip or not. After the user ends the trip, default values for the number of passengers are used again. Trip parameters saved in the preferences class are also deleted.

A QR code of the survey link is generated when the main activity launches. The link is retrieved from the database in case that the survey link changes after the application is released. A function that creates the QR code of a given string as a bitmap was implemented in the main activity through appropriate functions provided by ZXing Core. The bitmap is scaled to the screen size and has a minimum dimension specified. For the devices with slow processors or high screen resolution, the function generating the bitmap takes a while to finish execution, delaying the launch of the main activity. Hence, the function generating QR code bitmap is run asynchronously in order not to delay activity launch. “doAsync” function provided by Anko was preferred for coroutines of Kotlin for simplicity and performance purposes.

When the user presses the survey link button in the menu, the QR code bitmap is shown with a layout inflator with a transparent background over the main activity, if it has been generated successfully. Using a layout inflator enables showing an image view without pausing/destroying the current activity or creating a separate layout for the image in the ”layouts” directory.

4.1.7. Handling Permission Checks and Requests

When the main activity launches, it checks whether all the permissions required for the application are granted and requests are made for permissions denied or not granted yet. If the user chooses not to grant any of the permissions, the application is closed as it cannot function.

Permission checks run at the beginning are repeated every time the application is brought to the foreground in case the user initially grants permissions and turns on location services and internet access but denies permissions or turns off location services or internet access after the application is launched.

Permissions the application requires are as follows:

(i) Permissions granted by default upon installation
- INTERNET: Required for the device to connect to the database and call read and write functions
- ACCESS_NETWORK_STATE: Required for the application to check for internet connectivity and alert the user if no connection is found
- FOREGROUND_SERVICE: Required for the application to run a foreground service that requests location data even if the application is in the background

(ii) Permissions to be granted by the user
   - ACCESS_COARSE_LOCATION: Required to access the approximate location of the device with an accuracy of a city block
   - ACCESS_FINE_LOCATION: Required to access the precise location of the device
   - ACCESS_BACKGROUND_LOCATION: Required for the application to request location data while it is running in the background

4.1.8. Developing the Remaining Activities

The application has a splash screen in which the logo of the application is shown for a short duration while certain tasks such as reading data saved in the phone storage are executed in the background. Afterward, the user is redirected to the appropriate activity depending on whether or not they have already logged in before.

If the user has not logged in before, log-in activity launches to input the license plate of their vehicle. The plate inputted is stored in the preferences class to be saved for future application launches. The user is redirected to the tutorial screen afterward. The tutorial screen is shown only once per log-in but also accessible through the tutorial button in the main activity.

The tutorial screen contains a slider (i.e., view pager) with a tip and a video of the tip being executed at each slide. A custom view pager adapter was created to be used for the view pager in the tutorial activity. The videos shown on the tutorial screen are in GIF format. Animated GIF files can be played by implementing the binary provided by Droids on Roids.

The log-out button in the main activity redirects the user to the log-out activity after validation through an alert dialog. log-out activity wipes saved data in preferences class and redirects the user to the splash screen.

4.1.9. Handling Exceptions

Exception handling procedures were implemented at necessary sections, especially at fragments that establish the connection to the database or require certain permissions to execute properly. When a function fails at execution, execution handling enables calling an alternative function or showing an error message to the user rather than crashing and relaunching the application.

As the application is compatible with all Android versions released within the last 8 years, version checks are executed while calling functions required for certain Android versions but deprecated in later versions or dealing with permission requirements added or removed in certain Android versions.

4.2. Conducting Alpha and Closed Beta Tests

Alpha tests were conducted on 6 physical and 3 virtual devices with OS versions ranging from Android 4.4 KitKat (API level 19) to Android 11 (API level 30) and pixel densities ranging from 216 to 538 pixels-per-inch (PPI). A checklist was created to assess the functionality and visual quality of certain aspects of the application including but not limited to permission checks and requests, activity launches and redirections, connection to the database, user interface, and notification tray. Each topic on the checklist has multiple subtopics and each subtopic has at least one test. The application is considered working properly if it passes every test.

No functionality error was observed during alpha tests. Minor cosmetic errors caused by some deprecated view attributes in certain API levels but not stated in documentations were fixed.

Closed beta tests were conducted by 3 testers for 2 days. The application was installed on the smartphones of volunteer testers through an unsigned APK file. The participants were asked to use the application continuously and give feedback on various aspects of the application such as functionality, accessibility, and user-friendliness. No problem was observed in the data collected. Testers did not encounter any problems while using the application.

4.3. Developing Data Processing Scripts

Multiple Python scripts were created to automatize data processing. Whole data processing was split into multiple parts to decrease execution time and workload of each step, and allow manually validating the script output after each run.

Following modules were imported in the Python scripts:

(i) Retrieving path of script file and creating folders
- pathlib
- os

(ii) Searching for and accessing multiple files in a certain path
- glob

(iii) Running command-line arguments to run other Python scripts
- subprocess

(iv) Reading JSON files
- json

(v) Creating, reading, and writing on XLSX files
- pandas
- xlsxwriter

(vi) Calculating the distance between two data points
- geopy.distance

(vii) Creating custom maps based on Google Maps
- gmaps

(viii) Exporting maps as HTML files
- ipywidgets

A total of 6 scripts were created. Data processing flowchart is given in Figure 4.3.

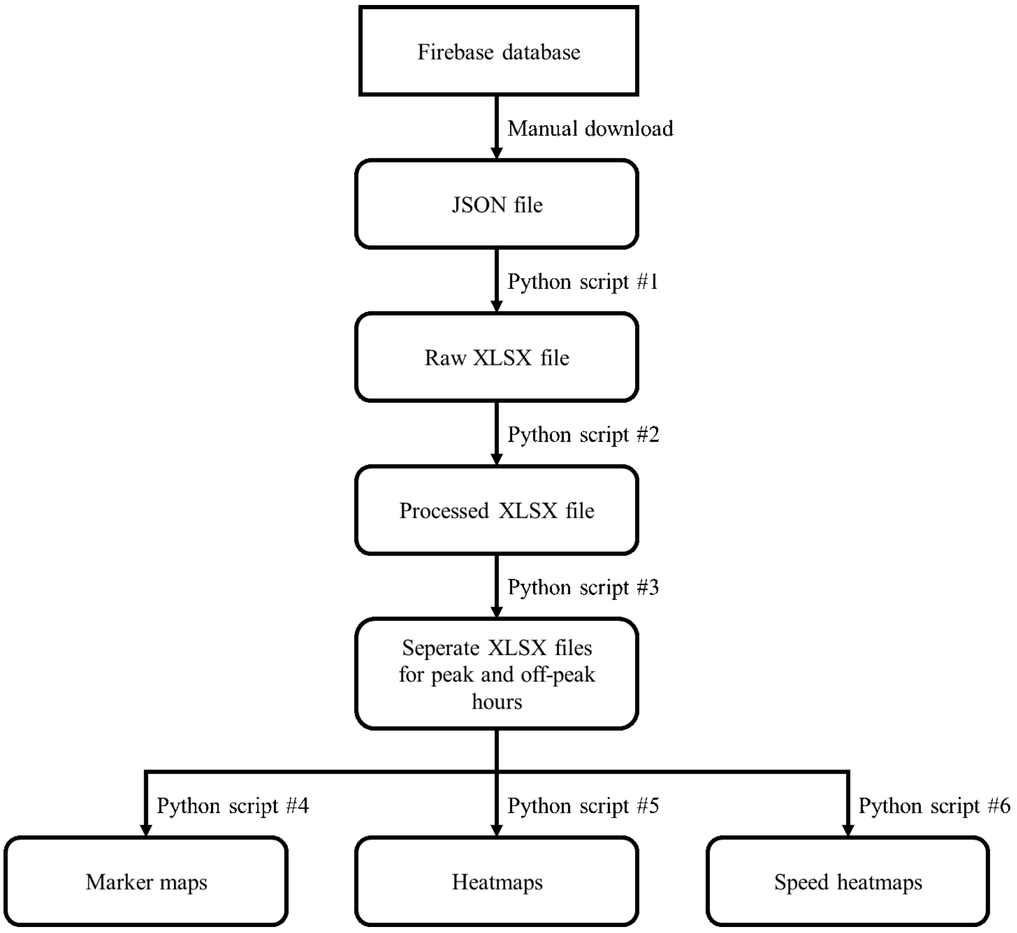


Figure 4.3. Data processing flowchart.

First Python script loads the JSON file, extracts trip data, and writes extracted data in a new XLSX file. Extracted trip data comprise location, date, time, number of passengers, vehicle occupancy and license plate. Each data type is retrieved as a dictionary object and converted to a transposed DataFrame object. All data collected from a vehicle throughout a day, which will be referred to as “travel log” in the future, is named based on the date and license plate, and stored in a separate XLSX sheet. Designing the hierarchy of the nodes in the database well while developing the Android application helps to scan and navigate through JSON files very easily.

Second Python script loads the initial XLSX file (i.e., raw XLSX file) and processes it. First, all cells and are formatted depending on the cell value to be easily readable. Second, time and distance differences between consecutive data points are calculated. The distance between two data points is calculated by using geopy.distance binary. Third, the instantaneous velocity is calculated for every point based on time and distance difference. Fourth, the total distance and time of the travel log are calculated and the average speed is found. Last, the speed vs. time graph is plotted for each travel log.

Third Python script loads the processed XLSX and splits it into four as morning peak, afternoon peak, evening peak, and off-peak by checking the timestamp of each data point and placing it in the appropriate XLSX file. Morning, afternoon, and evening peaks are defined as 07:00 – 10:00, 12:00 – 14:00, and 17:00 – 19:00, respectively.

Fourth, fifth and sixth Python scripts create the marker map, heatmap, and speed heatmap of the XLSX file, respectively. These scripts have a similar structure. They retrieve latitude and longitude values from the XLSX file, form a location list, create a gmaps figure, add a gmaps layer to the figure, and export the figure as HTML file. The type and attributes of the gmaps layer depend on the map type.

Marker maps contain the timestamp, latitude, and longitude of every data point. Hence, timestamps must be retrieved from the XLSX file in addition to latitude and longitude. Marker maps are useful to cross-reference with the XLSX file. They, however, have large file sizes and load slower.

Heatmaps show in which regions the data inputs are concentrated more. They do not, however, represent instantaneous speed well. Speed heatmaps are similar to regular heatmaps. Data points are, however, separated into groups based on instantaneous speed and each group is assigned a gradient. The color assigned for each speed group can be observed in the legends of maps. Then, each group is added to the figure separately.

An error that is likely to be encountered is that when multiple maps are exported at once with Visual Studio Code, file sizes get larger with each map. This is caused by the gmaps figure storing data of previous maps even though the figure is redeclared for each map. This error can be solved by exporting a single map at each time. However, this solution is very impractical and time-consuming.

In this case, a separate map-exporting script was created for each map type, instead. These scripts are called once for each map by the scripts that retrieve data from XLSX through Popen function of subprocess module. Information about the travel log to be exported as the map is passed to the second script through Popen function arguments. The remaining data such as latitude, longitude, and timestamp of data points are saved in a text file and read by the second script.

5. CASE STUDY ANALYSES

The functionality of the application was tested in three case studies. The application was used in collecting descriptive GPS data of different vehicle types. The data collection period and the number of vehicles vary for each case study. The same data processing scripts previously described in detail in Section 4.3 were used for all case studies.

5.1. Case Study 1: Bus Trips in Tekirdağ

In collaboration with Tekirdağ Municipality, the application was used in collecting GPS data of buses in Çorlu, Tekirdağ. For this case study, an alternative version of the application was developed. In this alternative version, the application does not ask for any user input other than granting permission requests and it starts collecting GPS data immediately after permission checks are completed. The user also does not input the license plate or number of passengers. Instead, a unique device ID is created at the first launch and the location data are stored in the database under a node with the value of the device ID.

The application was installed on 15 devices and data collection continued for four weeks. Collected GPS data were processed using Python scripts and stored as individual daily travel logs. A flowchart of the processing of collected bus data was previously given in Figure 4.2. Each travel log was named based on the date and the device ID. The total travel time and distance of the collected bus data are 430 hours, and 6860 kilometers, respectively. A summary of the collected data is given in Table 5.1.

Table 5.1. Summary of the collected bus data in Çorlu, Tekirdağ.

Time (DD.MM.YYYY)	**Number of Travel Logs**	**Total Duration (hh:mm:ss)**	**Total Distance (km)**	**Average Speed (km/h)**
30.11.2020 - 06.12.2020	42	128:50:46	1967.77	15.27
07.12.2020 - 13.12.2020	42	159:38:34	2674.67	16.75
14.12.2020 - 20.12.2020	29	84:17:37	1309.87	15.54
21.12.2020 - 27.12.2020	19	57:23:37	907.87	15.82
Total	132	430:10:34	6860.18	15.95

Using appropriate Python script, travel logs were split into parts corresponding to peak and off-peak hours, and the speed vs. time graph of each travel log part was plotted. As previously mentioned in Section 4.3, the morning, afternoon, and evening peaks are defined as 07:00 – 10:00, 12:00 – 14:00, and 17:00 – 19:00, respectively. Next, the marker map, heatmap, and speed heatmap of each travel log part were exported using appropriate Python scripts.

Two travel logs, B-12-01.6 and B-12-01.9 will be showcased in this section. Summary of travel log B-12-01.6 is given in Table 5.2.

Table 5.2. Summary of B-12-01.6.

Travel Log	**Peak/Off-peak**	**Duration (hh:mm:ss)**	**Distance (km)**	**Average Speed (km/h)**
B-12-01.6	Morning peak	02:59:57	49.48	16.50
	Noon peak	01:05:46	15.53	14.17
	Evening peak	-	-	-
	Off-peak	03:01:55	43.98	14.51
	Total	07:07:38	108.99	15.29

B-12-01.6 took place between 05:58 and 13:05 on 1 December 2020. Hence, it comprised the the morning and noon peaks, but not the evening peak. The average speed during the noon peak and off-peak hours were close to each other and the average speed during the morning peak was slightly higher. The speed vs. time graph of travel log B-12-01.6 during the morning peak, noon peak, and off-peak hours are given in Figure 5.1, Figure 5.2, and Figure 5.3, respectively.

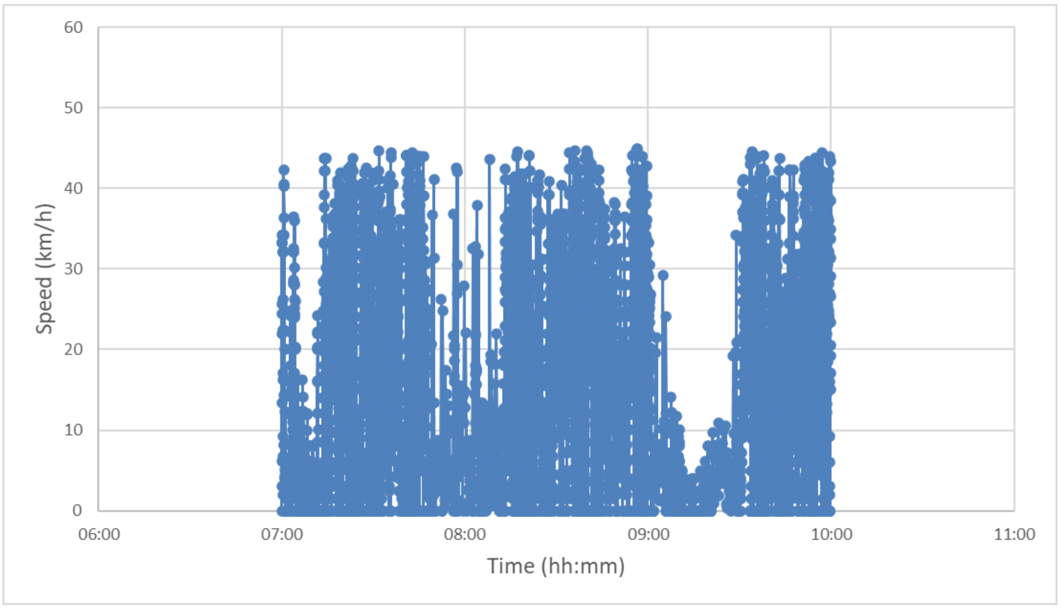


Figure 5.1. Speed vs. time graph of B-12-01.6 during the morning peak.

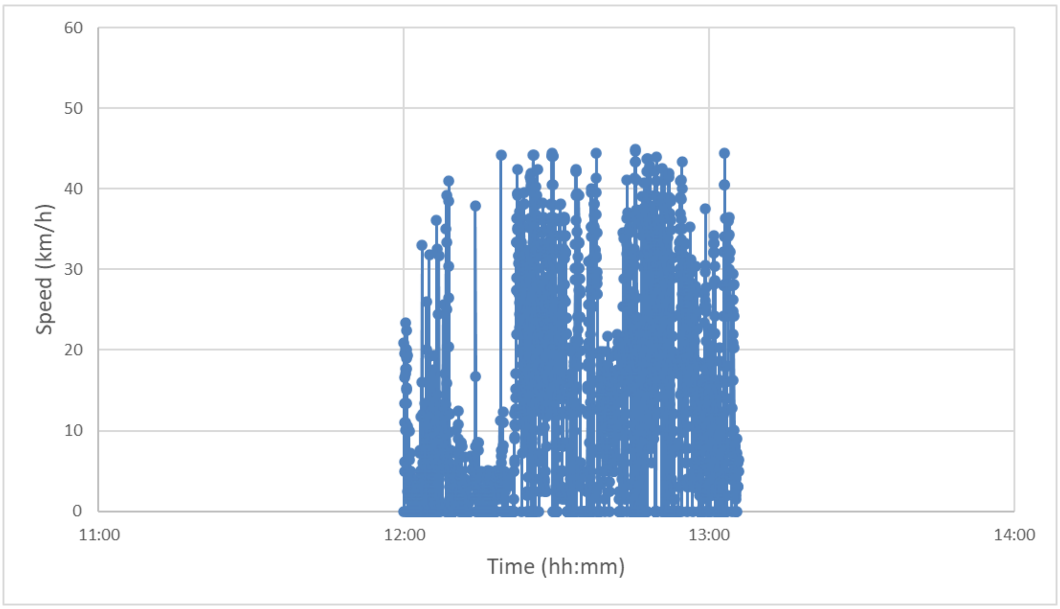


Figure 5.2. Speed vs. time graph of B-12-01.6 during the noon peak.

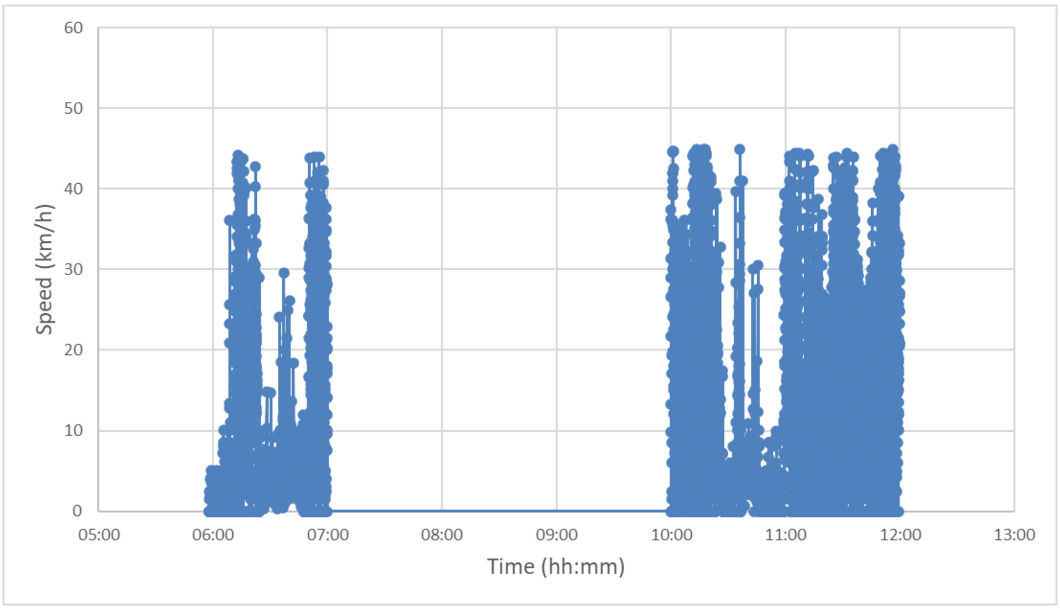


Figure 5.3. Speed vs. time graph of B-12-01.6 during off-peak hours.

Although speed profiles can be examined via the speed vs. time graphs, it is hard to keep track of speed on a minute-by-minute basis as the period is too long.

They also don't provide any information about the correlation between instantaneous speed and location. Maps, especially speed heatmaps, can be very useful in this regard. The speed heatmaps of travel log B-12-01.6 during the morning peak, noon peak, and off-peak hours exported through appropriate Python script are given in Figure 5.4, Figure 5.5, and Figure 5.6, respectively.

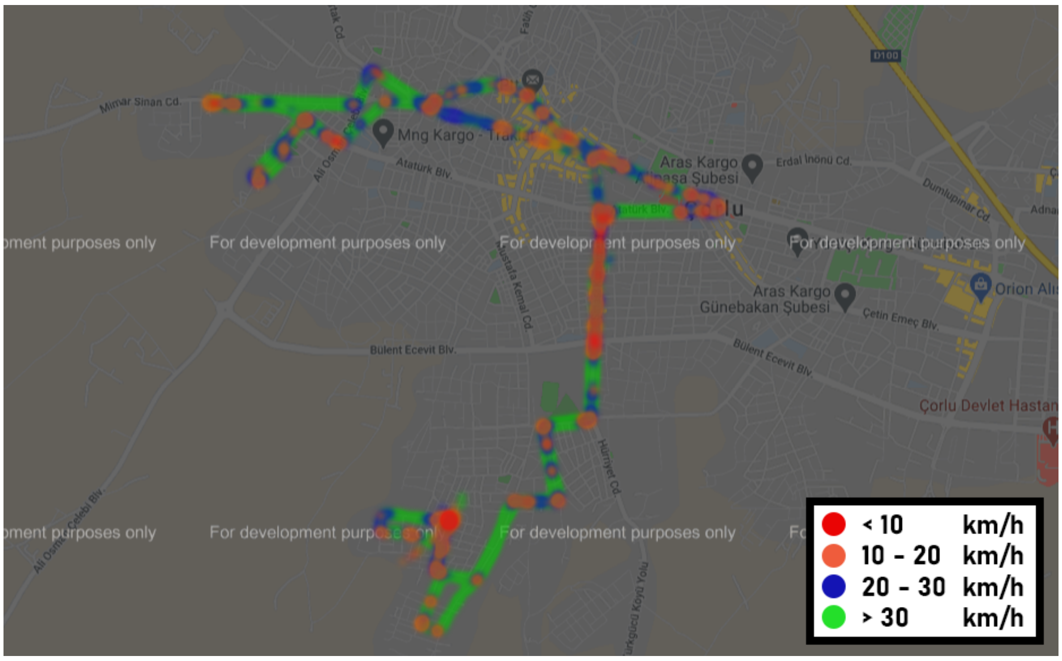


Figure 5.4. Speed heatmap of B-12-01.6 during the morning peak.

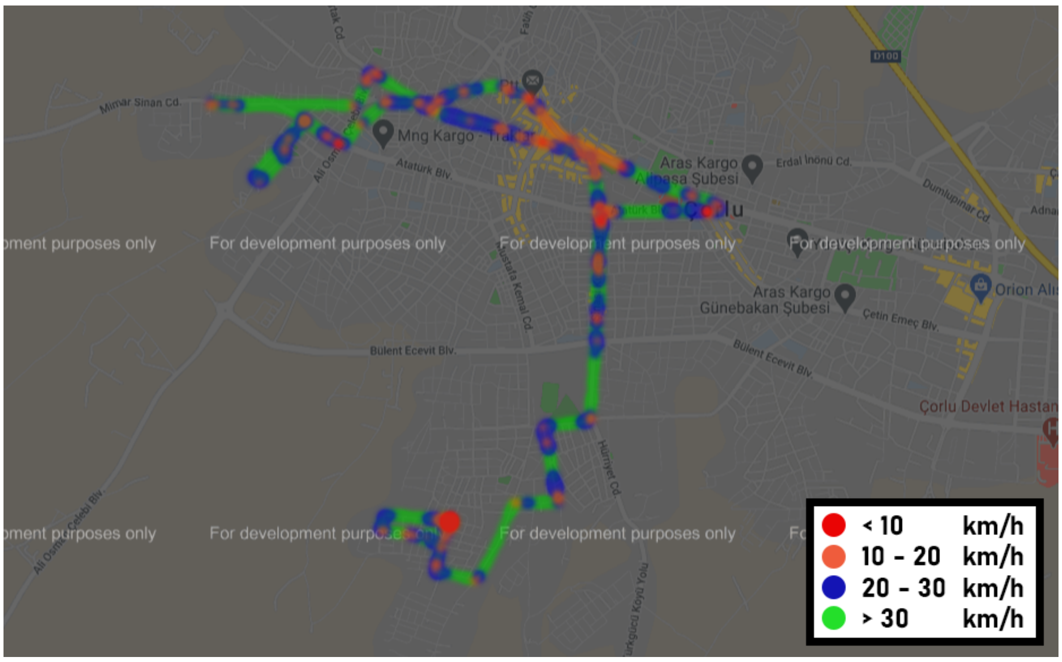


Figure 5.5. Speed heatmap of B-12-01.6 during the noon peak.

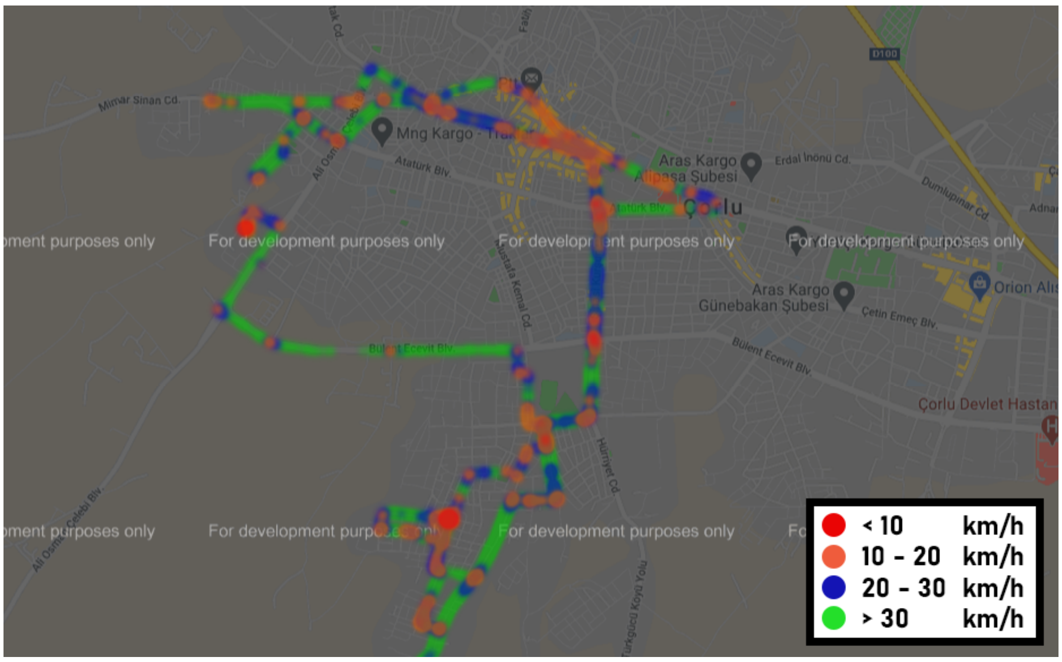


Figure 5.6. Speed heatmap of B-12-01.6 during off-peak hours.

During off-peak hours, the bus maintained relatively high speeds except in Salih Omurtak Avenue, Kumyol Avenue, and Şinasi Kurşun Avenue. A similar speed profile can be observed during the noon peak, although the speed drops are less severe in these three avenues. Harsh speed drops can be observed during morning peak in these avenues, especially in Şinasi Kurşun Avenue.

To showcase a travel log that includes evening peak, similar procedures will be repeated for travel log B-12-01.9. Summary of travel log B-12-01.9 is given in Table 5.3.

Table 5.3. Summary of B-12-01.9.

Travel Log	**Peak/Off-peak**	**Duration (hh:mm:ss)**	**Distance (km)**	**Average Speed (km/h)**
B-12-01.9	Morning peak	-	-	-
	Noon peak	01:20:20	19.00	14.19
	Evening peak	01:59:56	28.80	14.43
	Off-peak	04:27:08	64.25	14.43
	Total	07:47:24	112.09	14.39

B-12-01.9 took place between 12:39 and 20:27 on 1 December 2020. Hence, it comprised the noon and evening peaks, but not the morning peak. The average speed during the noon peak, evening peak and off-peak hours were close to each other. The speed vs. time graph of travel log B-12-01.9 during the noon peak, evening peak, and off-peak hours are given in Figure 5.7, Figure 5.8, and Figure 5.9, respectively.

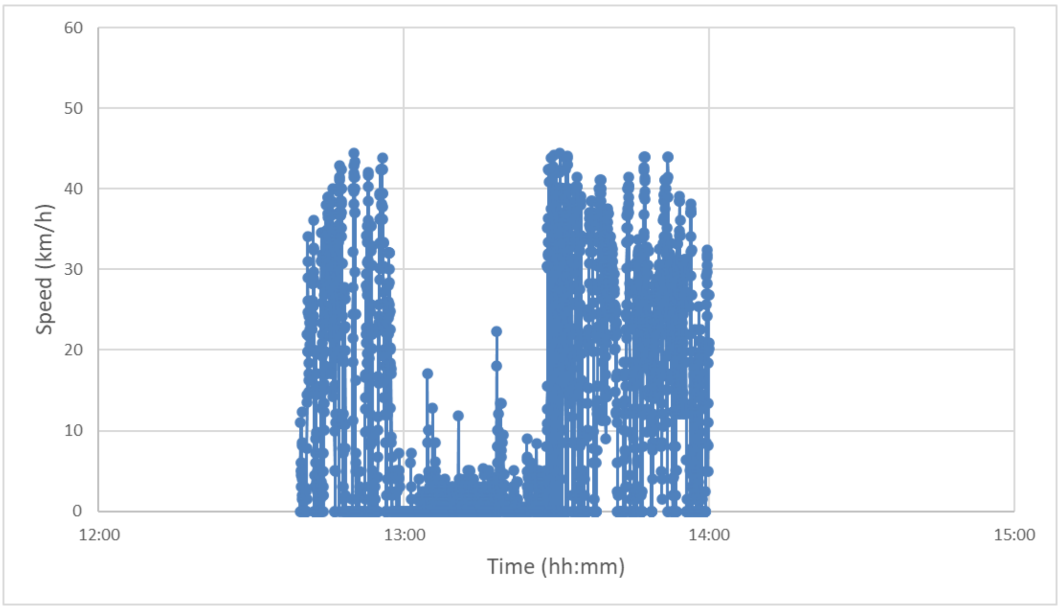


Figure 5.7. Speed vs. time graph of B-12-01.9 during the noon peak.

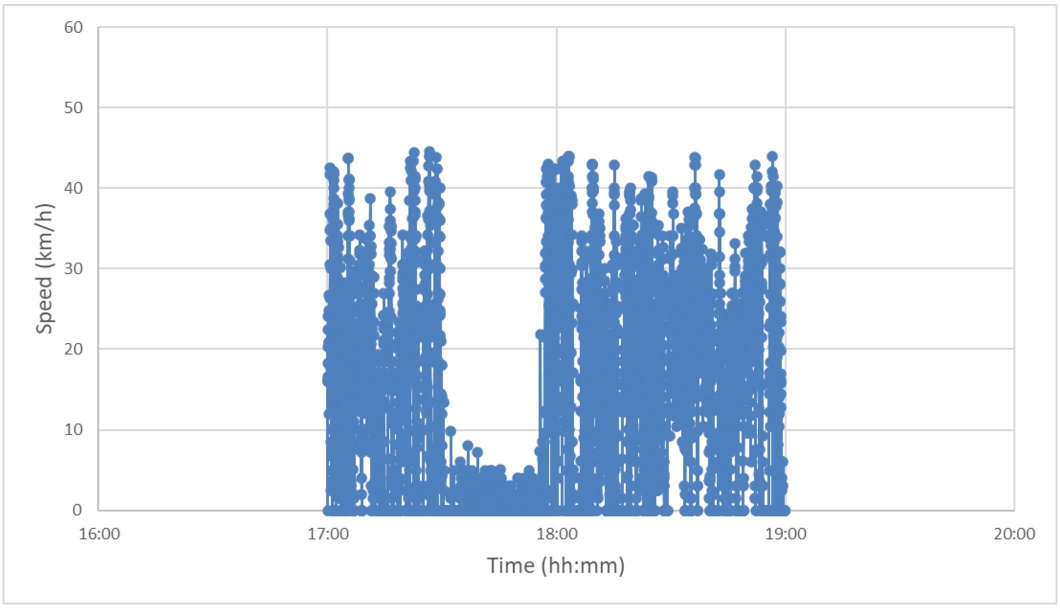


Figure 5.8. Speed vs. time graph of B-12-01.9 during the evening peak.

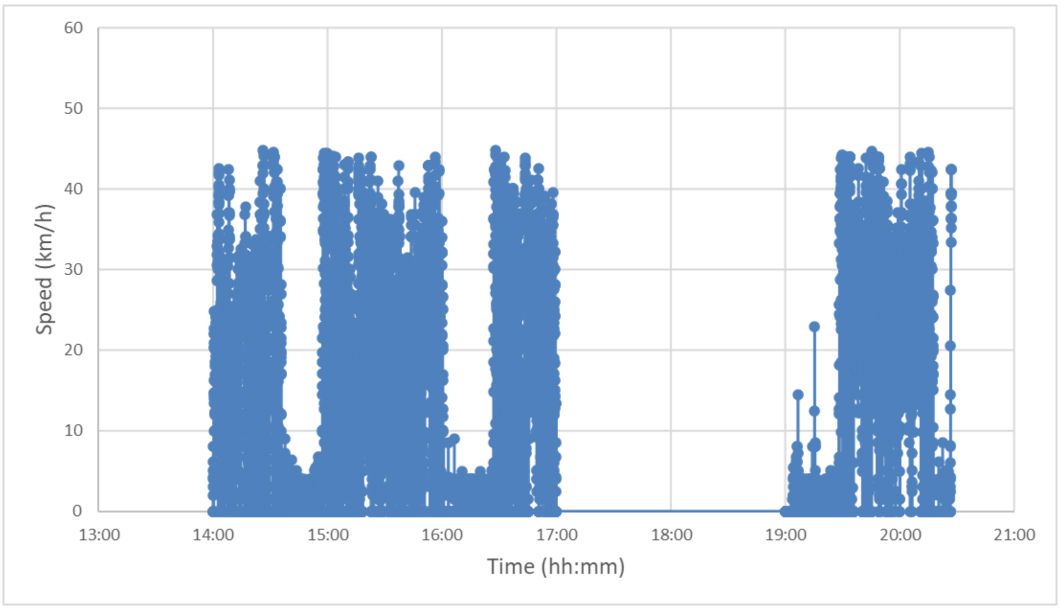


Figure 5.9. Speed vs. time graph of B-12-01.9 during off-peak hours.

Speed heatmaps of travel log B-12-01.9 during noon peak, evening peak, and off-peak hours are given in Figure 5.10, Figure 5.11, and Figure 5.12, respectively.

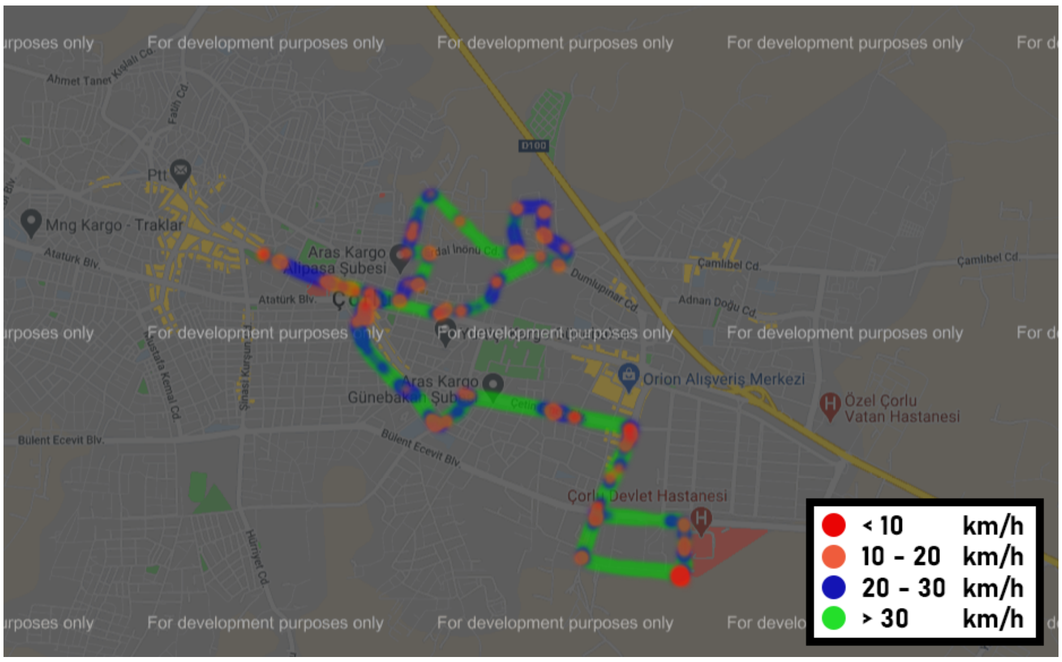


Figure 5.10. Speed heatmap of B-12-01.9 during the noon peak.

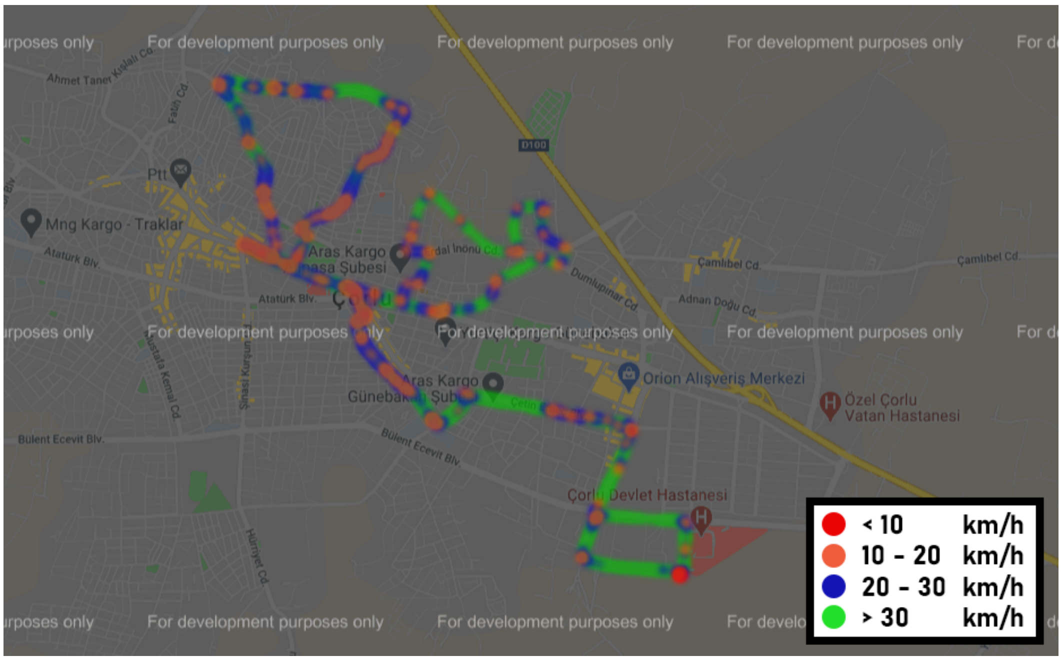


Figure 5.11. Speed heatmap of B-12-01.96 during the evening peak.

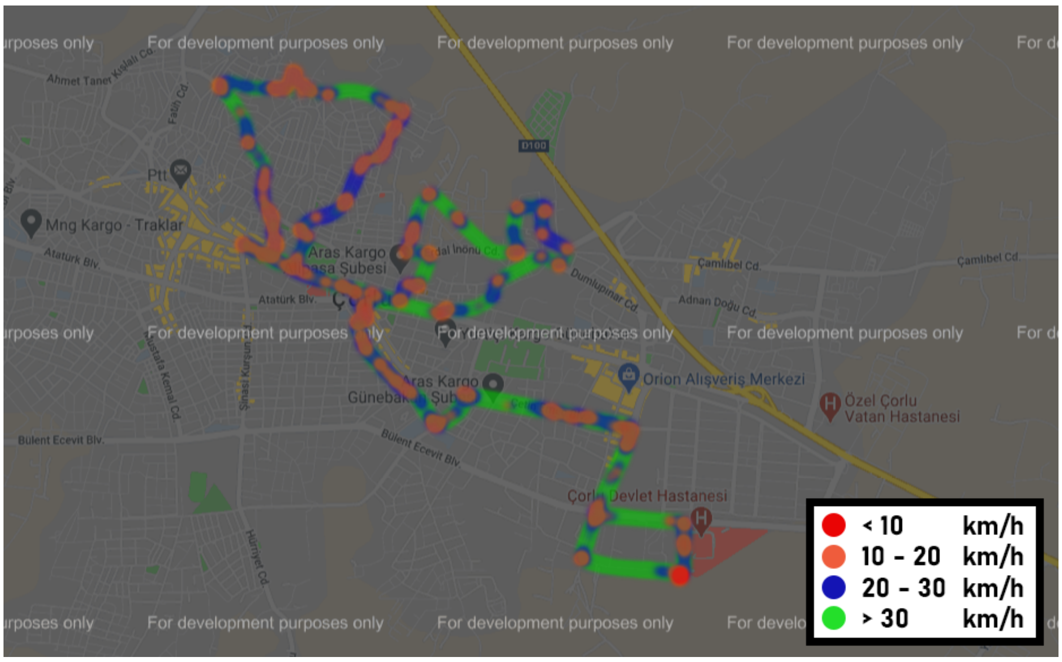


Figure 5.12. Speed heatmap of B-12-01.9 during off-peak hours.

Throughout the day, the bus maintained speeds between 10 and 30 km/h most of the time, especially in Salih Omurtak Avenue, Kırkova Avenue, Halit Ziya Uşaklıgil Avenue, Kömürcü Avenue, and Ahmet Priştina Avenue. Additionally, during off-peak hours, speed drops can be observed in certain parts of Çetin Emeç Avenue. Aside from these regions mentioned above, the bus maintained relatively high speeds.

5.2. Case Study 2: Taxi Trips in İstanbul

The application was used in collecting descriptive GPS data of taxis. Three taxi stands near Boğaziçi University campuses and one taxi stand in Fulya, Şişli was selected for the case study. Locations of the taxi stand near Boğaziçi University and in Şişli are given in Figure 5.13 and Figure 5.14, respectively.

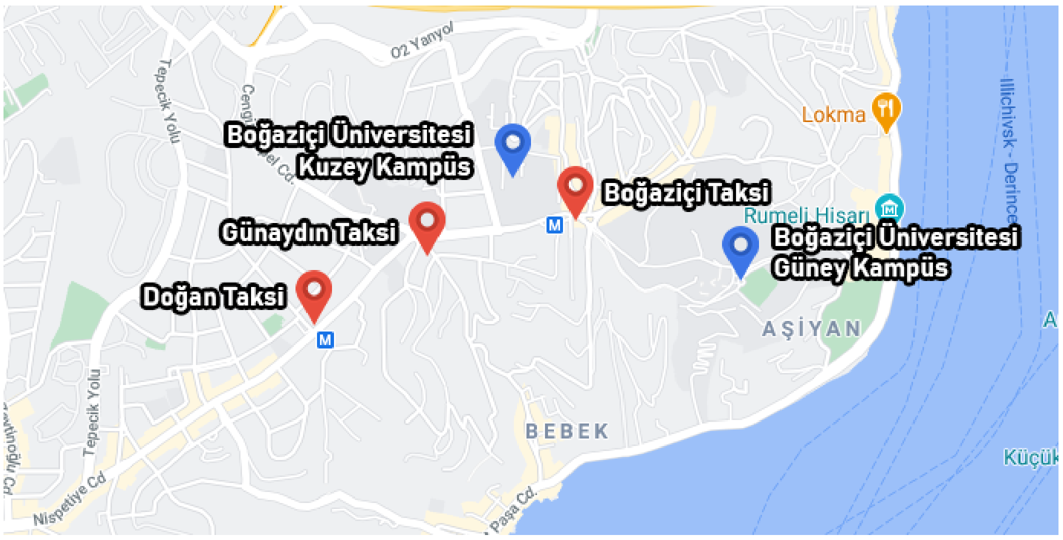

Boğaziçi Üniversitesi Kuzey Kampüs
Boğaziçi Taksi
Günaydın Taksi
Doğan Taksi
Boğaziçi Üniversitesi Güney Kampüs
Lokma
Rumeli Hisarı
AŞİYAN
BEBEK

Figure 5.13. Locations of taxi stands near Boğaziçi University.

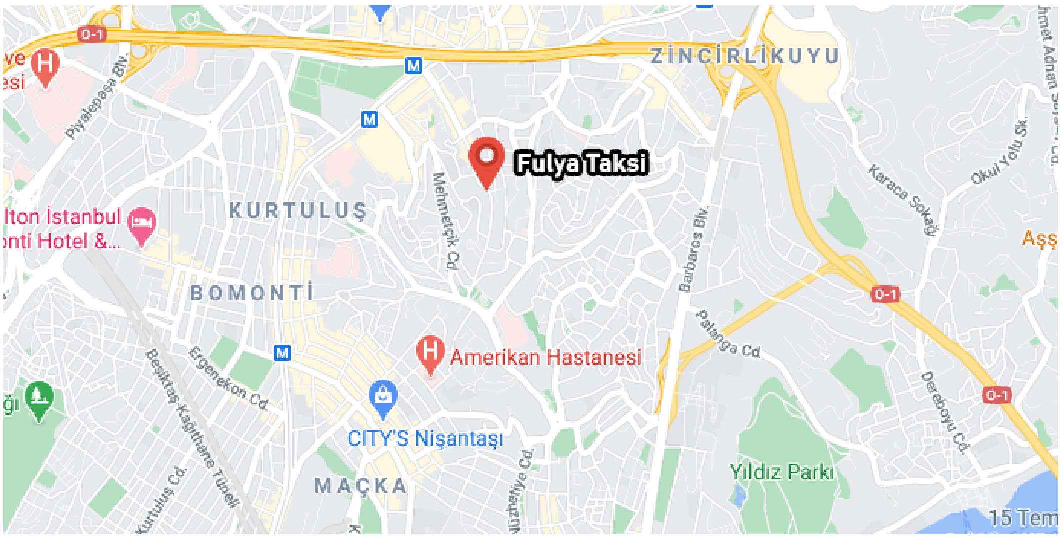


Figure 5.14. Location of the taxi stand in Fulya, Şişli.

The application was introduced to the taxi drivers and a brief tutorial on how to use it was given face-to-face. The application was installed on the mobile phones of four volunteer participants through the unsigned APK file built priorly. A PDF file containing instructions on how to install and use the application was sent to the drivers in case they would like to refer to the documentation. Contact information was also shared with the drivers if they would like to give feedback or ask questions. The data collection period continued for five days.

Using the appropriate Python script, the speed vs. time graph of each travel log was plotted. Travel log T-11-17.1, which took place between 12:10 and 12:56 on 17 November 2020, will be showcased in this section. Total duration, total distance, and average speed for travel log T-11-17.1 are 46 minutes, 15 kilometers, and 19.4 km/h, respectively. The speed vs. time graph of travel log T-11-17.1 is given in Figure 5.15.

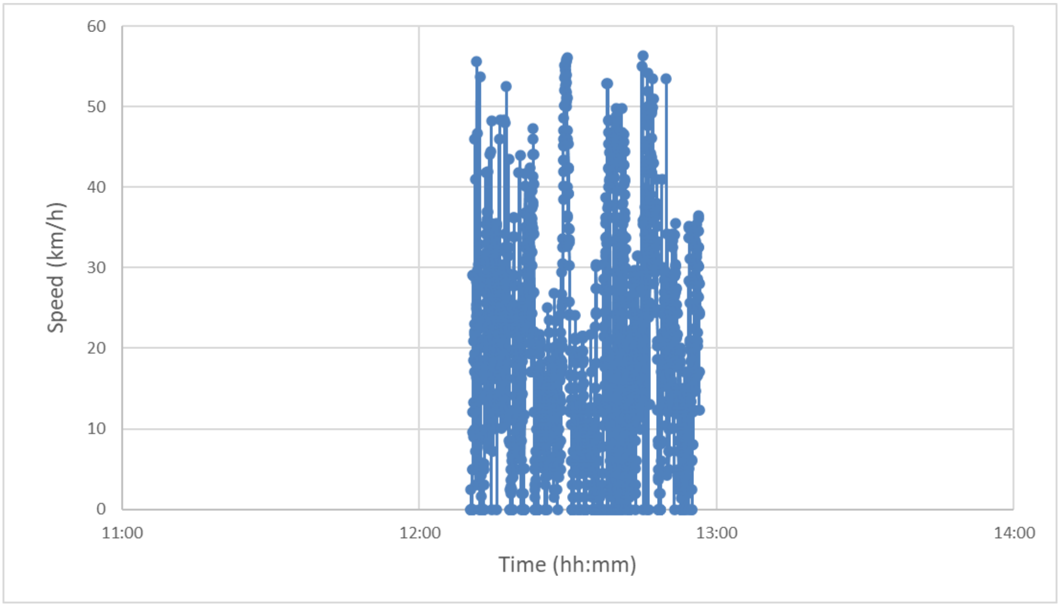


Figure 5.15. Speed vs. time graph of T-11-17.1.

Using appropriate Python scripts, three types of maps were exported for each travel log. Heatmap, speed heatmap, and marker map of travel log T-11-17.1 are given in Figure 5.16, Figure 5.17, and Figure 5.18, respectively.

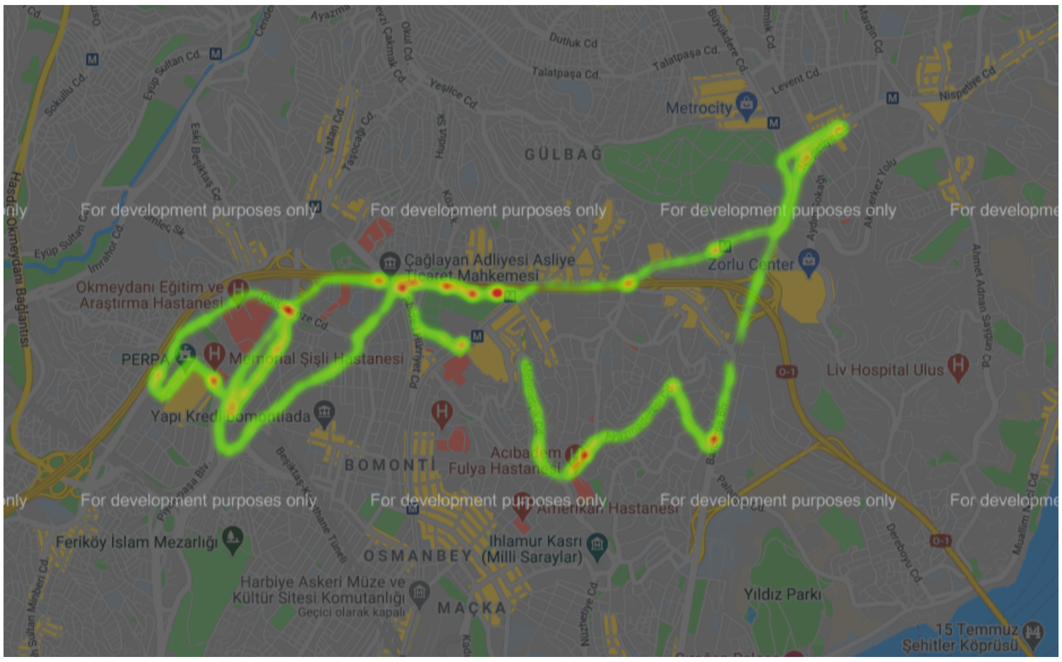


Figure 5.16. Heatmap of T-11-17.1.

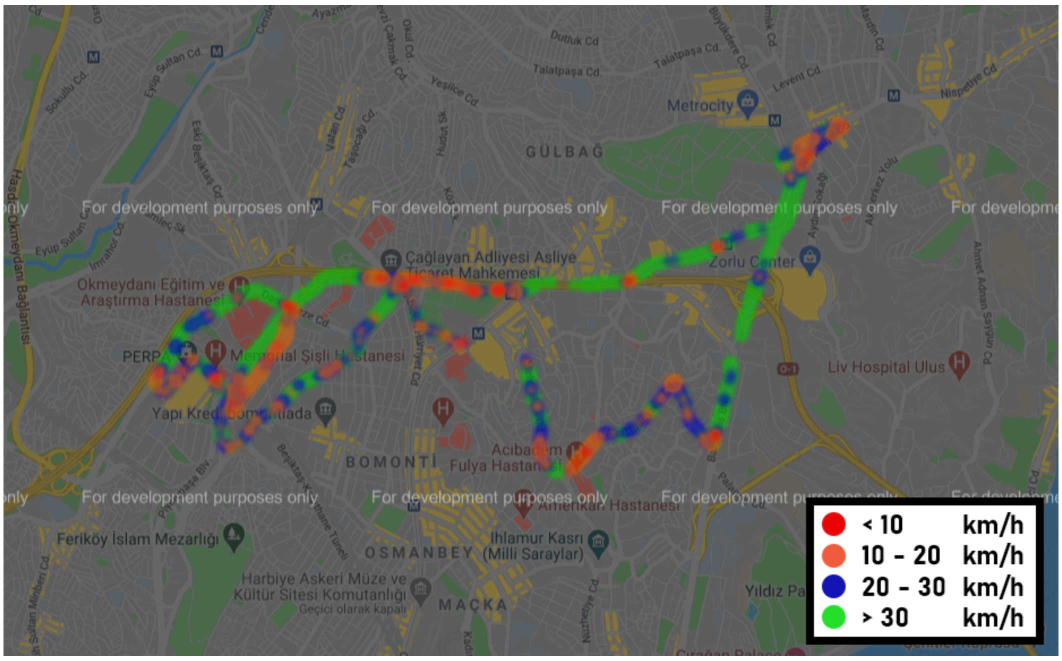


Figure 5.17. Speed heatmap of T-11-17.1.

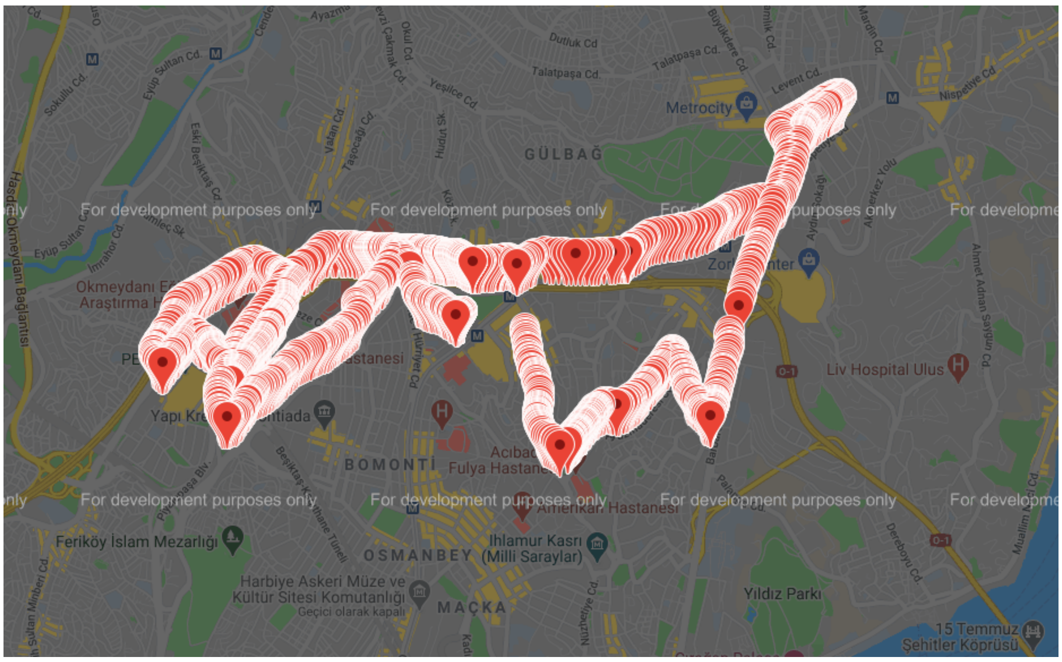


Figure 5.18. Marker map of T-11-17.1.

As seen in the maps, the travel log took place around Şişli and Beşiktaş, İstanbul. In Figure 5.17, it can be observed that the taxi maintained a speed over 30 km/h for nearly half of the trip. Most notable speed drops were observed around Halide Edip Adıvar, Şişli and Gayrettepe, Beşiktaş. Examining Figure 5.18, the exact time the vehicle visited each marker point can be found out. Likewise, using Figure 5.18, at which regions the density of data inputs is higher can be observed. These regions, naturally, match with the regions where the speed drops, as observed in Figure 5.17.

5.3. Case Study 3: Textile Firm Fleet Trips in İstanbul

The application was used in collecting GPS data of a voluntary vehicle belonging to a textile firm's fleet. Data collection continued for three weeks. Trip data recorded by the tester vehicle mostly consist of series of short trips between warehouses. The application was turned on by the driver every time the vehicle started moving and turned off when the vehicle reached its destination.

Using appropriate Python scripts collected data was processed. The speed vs. time graph of each travel log was plotted. Three types of maps of each travel log were exported. Travel log F-12-17, which took place between 11:20 and 16:27 on 17 December 2020, will be showcased in this section. Total duration, total distance, and average speed for travel log F-12-17 are an hour and 22 minutes, 23.5 kilometers, and 17 km/h, respectively. The speed vs. time graph of travel log F-12-17 is given in Figure 5.19.

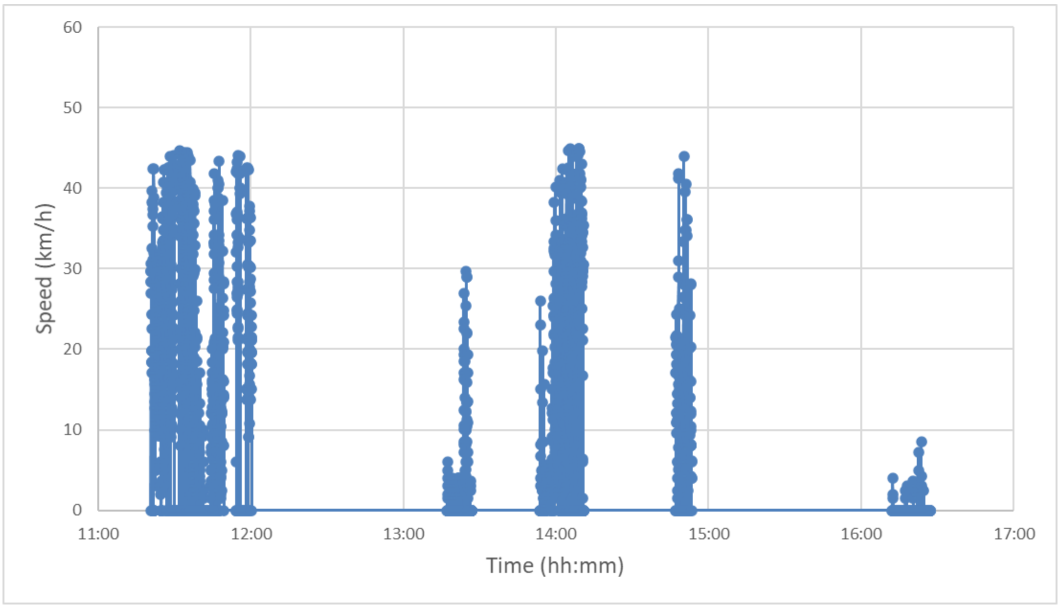


Figure 5.19. Speed vs. time graph of F-12-17.

As seen in Figure 5.19, the vehicle made multiple small trips throughout the day. These trips varied in duration and none of them exceeded an hour. The speed profile also varied for each trip. The route of each trip can be examined in detail with appropriate maps. Heatmap, speed heatmap, and marker map of travel log F-12-17 are given in Figure 5.20, Figure 5.21, and Figure 5.22 respectively.

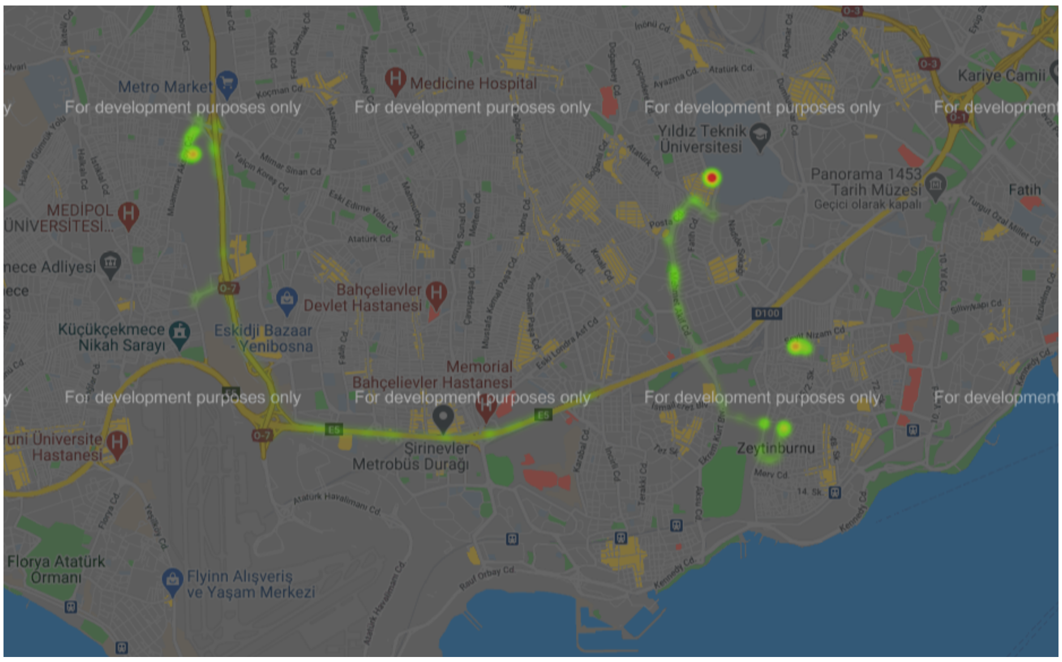

Metro Market
Medicine Hospital
Yıldız Teknik Üniversitesi
Kariye Camii
Panorama 1453 Tarih Müzesi
Fatih
MEDİPOL ÜNİVERSİTESİ
Bahçelievler Devlet Hastanesi
Küçükçekmece Nikah Sarayı
Eskidji Bazaar - Yenibosna
Memorial Bahçelievler Hastanesi
Şirinevler Metrobüs Durağı
Zeytinburnu
Florya Atatürk Ormanı
Flyinn Alışveriş ve Yaşam Merkezi
For development purposes only

Figure 5.20. Heatmap of F-12-17.

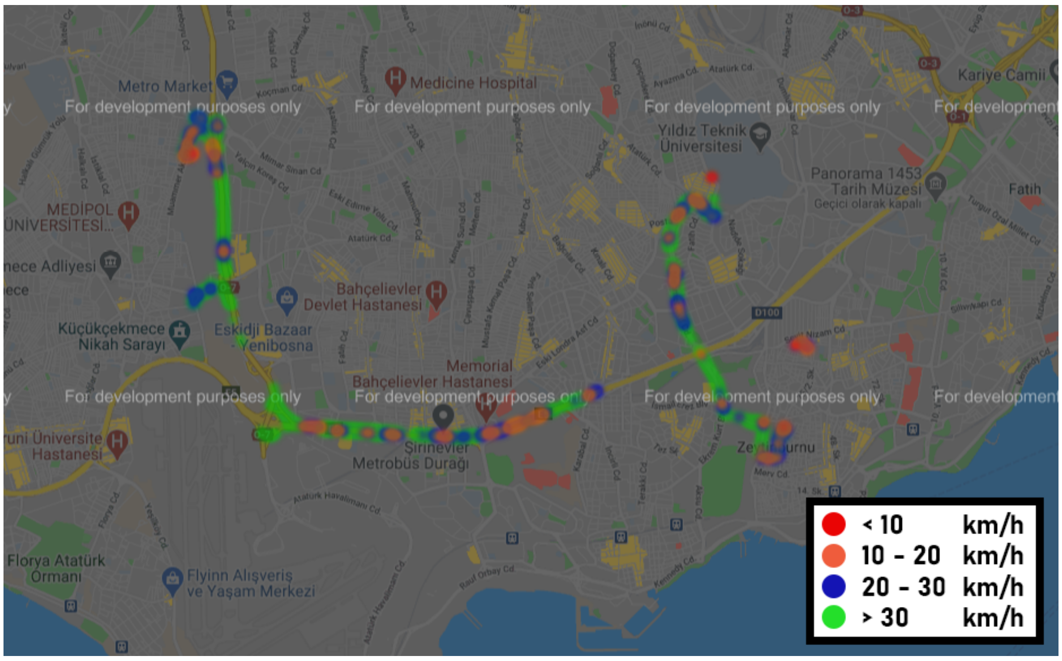

● < 10 km/h
● 10 - 20 km/h
● 20 - 30 km/h
● > 30 km/h
Metro Market
Medicine Hospital
Yıldız Teknik Üniversitesi
Kariye Camii
Panorama 1453 Tarih Müzesi
Fatih
MEDİPOL ÜNİVERSİTESİ
Bahçelievler Devlet Hastanesi
Küçükçekmece Nikah Sarayı
Eskidji Bazaar - Yenibosna
Memorial Bahçelievler Hastanesi
Şirinevler Metrobüs Durağı
Florya Atatürk Ormanı
Flyinn Alışveriş ve Yaşam Merkezi
For development purposes only

Figure 5.21. Speed heatmap of F-12-17.

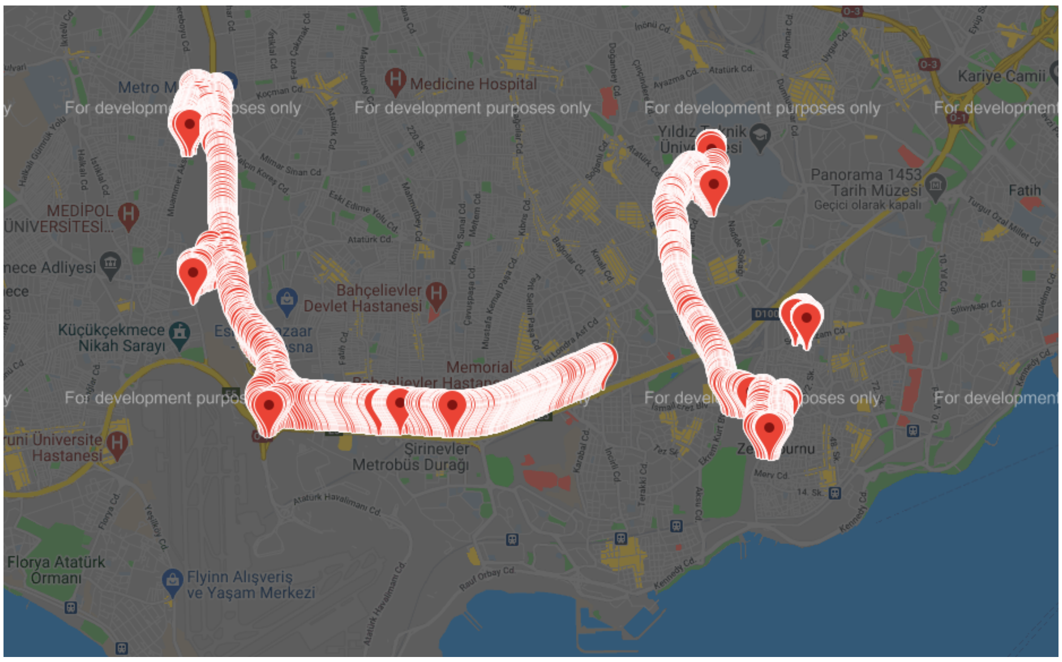


Figure 5.22. Marker map of F-12-17.

As seen in the maps, the vehicle made multiple short trips around Bahçelievler, İstanbul. In Figure 5.21, it can be observed that the vehicle maintained a speed over 30 km/h most of the time. Speed drops can be observed on certain parts of D100 Highway. Similar to the previous case studies, Figure 5.22 and Figure 5.20 can be used to determine data point timestamps and regions with higher data input density, respectively.

6. CONCLUSION

The application developed in this thesis was used in three different case studies for traffic data collection. As each case study had different specifications, it was shown that the application can be used in various types of projects regardless of data collection duration or transport mode. It was also shown that traffic data can be collected without the need to visit the project region; data collection can start immediately after distributing the application and giving a brief tutorial to the users. Additionally, Python-based data processing toolkit developed in this thesis proved its efficiency and functionality in processing collected data, plotting relevant graphs, and exporting relevant maps.

Although the main objectives of this thesis were to develop a smartphone application and a data processing toolkit, multiple case studies were conducted to test the functionality of them. As the drivers in the second and the third case studies were volunteer participants, data sets collected in these case studies are limited. In future studies that use the Android application and the Python-based data processing developed in this thesis, the number of participants and overall data size should be increased to be able to conduct detailed traffic analyses.

REFERENCES

1. Gokasar, I., Y. Cetinel and M. G. Baydogan, “Estimation of Influence Distance of Bus Stops Using Bus GPS Data and Bus Stop Properties”, *IEEE Transactions on Intelligent Transportation Systems*, Vol. 20, No. 12, pp. 4635–4642, Dec 2019.

2. Montini, L., C. Antoniou and K. W. Axhausen, “Route and mode choice models using GPS data”, p. 14, 2017.

3. Cui, J., F. Liu, D. Janssens, S. An, G. Wets and M. Cools, “Detecting urban road network accessibility problems using taxi GPS data”, *Journal of Transport Geography*, Vol. 51, pp. 147–157, Feb 2016.

4. Murakami, E. and D. Wagner, “Can using global positioning system (GPS) improve trip reporting?”, *Transportation Research Part C: Emerging Technologies*, Vol. 7, No. 2-3, pp. 149–165, Apr 1999.

5. Guo, J., Y. Liu, L. Zhang and Y. Wang, “Driving Behaviour Style Study with a Hybrid Deep Learning Framework Based on GPS Data”, *Sustainability*, Vol. 10, No. 7, p. 2351, Jul 2018.

6. Gong, L., R. Kanamori and T. Yamamoto, “Data selection in machine learning for identifying trip purposes and travel modes from longitudinal GPS data collection lasting for seasons”, *Travel Behaviour and Society*, Vol. 11, pp. 131–140, Apr 2018.

7. Jun, J., R. Guensler and J. Ogle, “Differences in observed speed patterns between crash-involved and crash-not-involved drivers: Application of in-vehicle monitoring technology”, *Transportation Research Part C: Emerging Technologies*, Vol. 19, No. 4, pp. 569–578, Aug 2011.

8. Castro, M., L. Iglesias, R. Rodríguez-Solano and J. A. Sánchez, “Geometric modelling of highways using global positioning system (GPS) data and spline approxi-

mation", *Transportation Research Part C: Emerging Technologies*, Vol. 14, No. 4, pp. 233–243, Aug 2006.

9. Alshibani, D. A. and D. O. Moselhi, "Productivity based method for forecasting cost time of earthmoving operations using sampling GPS data", *Journal of Information Technology in Construction (ITcon)*, Vol. 21, No. 3, pp. 39–56, Mar 2016.

10. Rasmussen, T. K., J. B. Ingvardson, K. Halldórsdóttir and O. A. Nielsen, "Improved methods to deduct trip legs and mode from travel surveys using wearable GPS devices: A case study from the Greater Copenhagen area", *Computers, Environment and Urban Systems*, Vol. 54, pp. 301–313, Nov 2015.

11. Zhu, S., "The roads taken: theory and evidence on route choice in the wake of the I-35W Mississippi River bridge collapse and reconstruction.", pp. 1–129, Sep 2010, `http://conservancy.umn.edu/handle/11299/99233`.

12. Huang, A. and D. Levinson, "A model of two-destination choice in trip chains with GPS data", *Journal of Choice Modelling*, Vol. 24, pp. 51–62, Sep 2017.

13. Shen, Y., M.-P. Kwan and Y. Chai, "Investigating commuting flexibility with GPS data and 3D geovisualization: a case study of Beijing, China", *Journal of Transport Geography*, Vol. 32, pp. 1–11, Oct 2013.

14. He, X., Y. Wu, S. Zhang, M. A. Tamor, T. J. Wallington, W. Shen, W. Han, L. Fu and J. Hao, "Individual trip chain distributions for passenger cars: Implications for market acceptance of battery electric vehicles and energy consumption by plug-in hybrid electric vehicles", *Applied Energy*, Vol. 180, pp. 650–660, Oct 2016.

15. Patnaik, A. K., P. K. Bhuyan and K. Krishna Rao, "Divisive Analysis (DIANA) of hierarchical clustering and GPS data for level of service criteria of urban streets", *Alexandria Engineering Journal*, Vol. 55, No. 1, pp. 407–418, Mar 2016.

16. Liu, K., T. Yamamoto and T. Morikawa, "Impact of road gradient on energy consumption of electric vehicles", *Transportation Research Part D: Transport and Environment*, Vol. 54, pp. 74–81, Jul 2017.

17. Necula, E., "Analyzing Traffic Patterns on Street Segments Based on GPS Data Using R", *Transportation Research Procedia*, Vol. 10, pp. 276–285, 2015.

18. Carli, R., M. Dotoli, N. Epicoco, B. Angelico and A. Vinciullo, "Automated evaluation of urban traffic congestion using bus as a probe", *IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 967–972, IEEE, Aug 2015, `http://ieeexplore.ieee.org/document/7294224/`.

19. Lu, S., V. L. Knoop and M. Keyvan-Ekbatani, "Using taxi GPS data for macroscopic traffic monitoring in large scale urban networks: calibration and MFD derivation", *Transportation Research Procedia*, Vol. 34, pp. 243–250, 2018.

20. Ciscal-Terry, W., M. Dell'Amico, N. S. Hadjidimitriou and M. Iori, "An analysis of drivers route choice behaviour using GPS data and optimal alternatives", *Journal of Transport Geography*, Vol. 51, pp. 119–129, Feb 2016.

21. Hast, M., K. M. Searle, M. Chaponda, J. Lupiya, J. Lubinda, J. Sikalima, T. Kobayashi, T. Shields, M. Mulenga and et al., "The use of GPS data loggers to describe the impact of spatio-temporal movement patterns on malaria control in a high-transmission area of northern Zambia", *International Journal of Health Geographics*, Vol. 18, No. 1, p. 19, Dec 2019.

22. Ma, X., Y. Wang, E. McCormack and Y. Wang, "Understanding Freight Trip-Chaining Behavior Using a Spatial Data-Mining Approach with GPS Data", *Transportation Research Record: Journal of the Transportation Research Board*, Vol. 2596, No. 1, pp. 44–54, Jan 2016.

23. Liu, Y., X. Fan, C. Lv, J. Wu, L. Li and D. Ding, "An innovative information

fusion method with adaptive Kalman filter for integrated INS/GPS navigation of autonomous vehicles", *Mechanical Systems and Signal Processing*, Vol. 100, pp. 605–616, Feb 2018.

24. Du, J. and L. Aultman-Hall, "Increasing the accuracy of trip rate information from passive multi-day GPS travel datasets: Automatic trip end identification issues", *Transportation Research Part A: Policy and Practice*, Vol. 41, No. 3, pp. 220–232, Mar 2007.

25. Flake, L., M. Lee, K. Hathaway and E. Greene, "Use of Smartphone Panels for Viable and Cost-Effective GPS Data Collection for Small and Medium Planning Agencies", *Transportation Research Record: Journal of the Transportation Research Board*, Vol. 2643, No. 1, pp. 160–165, Jan 2017.

26. Korpilo, S., T. Virtanen and S. Lehvävirta, "Smartphone GPS tracking—Inexpensive and efficient data collection on recreational movement", *Landscape and Urban Planning*, Vol. 157, pp. 608–617, Jan 2017.

27. Shafique, M. and E. Hato, "Travel Mode Detection with Varying Smartphone Data Collection Frequencies", *Sensors*, Vol. 16, No. 5, p. 716, May 2016.

28. Jackson, S., L. F. Miranda-Moreno, C. Rothfels and Y. Roy, "Adaptation and Implementation of a System for Collecting and Analyzing Cyclist Route Data Using Smartphones", *Transportation Research Board 93rd Annual Meeting*, 2014, `https://trid.trb.org/view/1289501`.

29. Strauss, J., L. F. Miranda-Moreno and P. Morency, "Mapping cyclist activity and injury risk in a network combining smartphone GPS data and bicycle counts", *Accident Analysis Prevention*, Vol. 83, pp. 132–142, Oct 2015.

30. Stipancic, J., L. Miranda-Moreno and N. Saunier, "Vehicle manoeuvers as surrogate safety measures: Extracting data from the gps-enabled smartphones of regular

drivers", *Accident Analysis Prevention*, Vol. 115, pp. 160–169, Jun 2018.

31. Pluvinet, P., J. Gonzalez-Feliu and C. Ambrosini, "GPS Data Analysis for Understanding Urban Goods Movement", *Procedia - Social and Behavioral Sciences*, Vol. 39, pp. 450–462, 2012.

32. Chen, J., W. Li, H. Zhang, W. Jiang, W. Li, Y. Sui, X. Song and R. Shibasaki, "Mining urban sustainable performance: GPS data-based spatio-temporal analysis on on-road braking emission", *Journal of Cleaner Production*, Vol. 270, Oct 2020.

33. Pham, D.-T., B. A. M. Hoang, S. N. Thanh, H. Nguyen and V. Duong, "A Constructive Intelligent Transportation System for Urban Traffic Network in Developing Countries via GPS Data from Multiple Transportation Modes", *IEEE 18th International Conference on Intelligent Transportation Systems*, IEEE, Sep. 2015.