ONLINE REAL-TIME SIMULATION OF VEHICLES

A Thesis

by

Mert Şahin

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Electrical and Electronics Engineering

Özyeğin University
September 2020

ONLINE REAL-TIME SIMULATION OF VEHICLES

Approved by:

Prof. H. Fatih Uğurdağ, Advisor
Department of Electrical and Electronics Engineering
Özyeğin University

Prof. Taylan Akdoğan
Department of Natural and Mathematical Sciences
Özyeğin University

Asst. Prof. Tacha Serif
Department of Computer Engineering
Yeditepe University

Date Approved: 20.08.2020

To My Family

ABSTRACT

Simulation technologies are a must-use in software development and testing processes in the automotive industry. In this thesis, an online real-time simulation environment is presented with the purpose of evaluating vehicle control algorithms considering their real-time response performance. Also, to perform an experiment driving a vehicle from the cloud with energy consumption prediction is intended. Additionally, reducing operational costs of Hardware-In-the-Loop (HIL) systems for small research groups and companies is aimed. Alternatives to HIL approach are also discussed. In the scope of this thesis, unique online real-time simulation use cases are defined. In this context, these use-cases are driving a vehicle from the cloud, evaluating performance of vehicle control algorithms, and design of an online HIL replacement with a novel approach to embedded software testing. From this approach, the presented tool is named Cloud-In-the-Loop (CIL). Moreover, to make the testing of controllers well-coordinated it is useful to have a model of a vehicle as a separate process on a separate computer in the network. It also makes the maintenance of the vehicle model and the whole simulator easier. The presented environment in this thesis is designed to have features such as terrain data extraction, graphical user interface, server/client communication with socket programming, multiple client handling with multi-threading, and ability to run vehicle models in the cloud. The developed environment was put under evaluation to test these features. As a result of our evaluations, our environment can drive vehicles from the cloud for coordinated cloud driving applications, can be used as a vehicle control algorithm evaluation tool in terms of real-time response performance, and can serve as CIL service to replace HIL systems.

ÖZETÇE

Simülasyon teknolojileri, otomotiv endüstrisindeki yazılım geliştirme ve test süreçlerinde kullanılması zorunludur. Bu tezde, araç kontrol algoritmalarını gerçek-zamanlı yanıt performanslarıyla değerlendirmek için çevrim içi gerçek-zamanlı bir simülasyon ortamı sunulmuştur. Ayrıca, enerji tüketimi tahmini ile buluttan araç sürüş deneyinin yapılması amaçlanmaktadır. Son olarak, küçük araştırma grupları ve şirketler için döngü içi donanım (HIL) sistemlerinin işletme maliyetlerinin azaltılması hedeflenmektedir. Bu tez kapsamında, özgün çevrim içi gerçek zamanlı simülasyon kullanım senaryoları tartışılmış ve alternatifleri tanımlanmıştır. Bu bağlamda, bu kullanım senaryoları bir aracı buluttan sürmekte, araç kontrol algoritmalarının performansını değerlendirmekte ve gömülü yazılım testine yeni bir yaklaşımla çevrim içi bir HIL değişiminin tasarımını içermektedir. Bu yaklaşım çerçevesinde, sunulan ortamın adı döngü içi bulut (CIL) olarak belirlenmiştir. Ayrıca, kontrolörlerin testini daha iyi koordine etmek amacıyla, ağdaki ayrı bir bilgisayarda ayrı bir işlem olarak bir araç modeline sahip olmak daha verimli bir yöntemdir. Bu durum, araç modelinin ve tüm simülatörün bakımını da kolaylaştırmaktadır. Tezde sunulan ortam, arazi verisi çıkarma, grafik kullanıcı arayüzü, soket programlama ile sunucu/istemci iletişimi, çok izleklilik ile çoklu istemci kullanımı ve araç modellerini bulutta koşturma gibi özelliklere sahip olacak şekilde tasarlanmıştır. Değerlendirmelerimizin bir sonucunda, ortamımız koordineli bulut sürüş uygulamaları için araçları buluttan sürebilir, gerçek zamanlı yanıt performansı açısından araç kontrol algoritması değerlendirme aracı olarak kullanılabilir ve HIL sistemlerinin yerini alacak CIL hizmeti olarak hizmet verebilir.

ACKNOWLEDGEMENTS

I am grateful to my thesis advisor Prof. H. Fatih Uğurdağ for his invaluable guidance and enlightenment he provided with his experience and wisdom. I would like to thank Prof. Taylan Akdoğan and Asst. Prof. Tacha Serif for serving on my thesis examination committee. I would like to further thank all my lab mates in nEMESys Lab at Özyeğin University as well as all my other friends for their help and support.

This thesis is partly based on work previously supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK). During the initial phases of this work, I was a research assistant for TÜBİTAK project No. 115E127 (PI: H. Fatih Uğurdağ). Last but not least, I must acknowledge the tuition waiver I received from Özyeğin University throughout my graduate studies as well as the great research and working environment the university provides.

I would also like to thank to my boss M. Ali Gözüküçük for his unlimited support and encouragement during my MS studies.

Finally, I would like to express my gratitude to my beloved wife Sezen Başak Özünur Şahin and my parents Zerrin and Ali Fuat Şahin for their love and support. Without them, it would have been impossible for me to complete this thesis.

Contents

List of Tables

List of Figures

Chapter I

INTRODUCTION

With changing trends in the automotive industry, a new approach needs to be implemented every day in the area of production and development. Especially when it comes to software development and implementation. In software development, simulating existing work plays a crucial role to catch bugs that may be overlooked. Simulations are also a must to verification and validation process of a software. Moreover, with the increase of computational power in processors, running complex algorithms in real-time for simulation of vehicle models becomes an easy task to do. Real-time Simulations also are used as an online tool that helps to develop software without the need for a target environment.

In line with these developments, an online real-time simulation environment is presented as the scope of this thesis. Real-time simulation of a vehicle accessible as a server process on a network is needed in multiple use cases. One such use case is coordinated driving. Coordinated driving will eventually require us to control vehicles from the cloud. From the viewpoint of a coordinated driving controller, every vehicle can be regarded as a process running somewhere on the network. One implication is that both the controller and vehicle must live with the delays introduced by the network and the controller itself. The controller and vehicles must survive the test when some commands are late or lost. Commands sent to a vehicle should be time-stamped, and the vehicle should have a default policy when no command is received for a particular time point at or before that time point. It is also useful to have the model of a vehicle as a separate process and also on a separate computer (as long as it is accessible through a network) because it makes the testing of controllers

much cleaner, especially when the controllers are developed by different groups. It also makes the maintenance of the vehicle model and the whole simulator cleaner. Moreover, using the presented simulator as Hardware-In-the-Loop (HIL) replacement is possible. The possibility of an online HIL tool makes operations of HIL research and development much cheaper and easier.

The presented environment is put to test on whether it can predict the energy consumption of a vehicle, can it compare vehicle control algorithms in terms of real-time response performance, and last but not least, if it can be used as HIL replacements. Evaluations show that the presented tool can be used in all use-cases mentioned above. Before diving into simulation details, there are several questions needed to be discussed.

1.1 Problem Statement

The problems will be stated while asking some questions about the field of software testing. The main problem that discussed about this field, as the scope of the thesis, is why the need for online real-time simulation of vehicles. The question needs to be divided into sub-questions so the answers can add up to understand and solve parent questions. It is important to argue and understand the questions below;

- Why the Use of Simulation?
- Why Real-Time?
- Why the Use of Online Environment?

These questions will help to understand the scope of this thesis.

1.1.1 Why the Use of Simulation?

All research products need to be put in testing to be shipped. These tests are either done in the final product or in an environment that behaves like a final product. When

it comes to the automotive industry it requires a considerable amount of money and effort to do software testing on an actual vehicle. Also, any mistake or bug will cause too much trouble in terms of health, money, and time-frame. For these reasons, every vehicle manufacturer tests its software in the simulation environment. To get all possible inputs and outputs combinations these simulations need to be as much as close to reality. If there is any error or bug in the software that put under tests with the help of simulations any damage will be prevented with zero cost.

Simulations can also be used in the algorithm developing phase. In the area of Energy Management Algorithm (EMA) development, any research and development process needs to be tested. The performance evaluation of an EMA needs to be tested in real vehicles to see efficiency increase results. But for projects that have limited funding, it is very difficult to test their product on actual vehicles. Simulations become very handy tools at this point. In a simulation environment testings of these algorithms can be done very efficiently. The presented tool in the thesis is well suitable for comparing, developing, and evaluating EMAs.

The answer to the questions of why simulation of vehicles became obvious at this point. Simulations are much cheaper test environments than testing on the actual final product. Also, It is much less risky to do mistakes or face bugs in a simulation environment than the real-world. A good simulation tool that has good features is always a necessity when it comes to embedded software testing of vehicles.

1.1.2 Why Real-Time?

The second question is why real-time? Being real-time is important in the aspect of real-time vehicle control application usage of control algorithms. EMAs will require the prediction of energy consumption to optimize inputs of vehicles. To predict energy consumption, a vehicle model needs to be simulated as realistic as possible. To drive

a vehicle with this energy prediction and optimization, simulations need to be real-time and respond to the vehicle in real-time. Simulations will return the energy consumption and after that EMA will optimize the vehicle's certain inputs. This thesis serves the purpose of comparing and monitoring control algorithms and EMAs in real-time. This use case is discussed in our work [3].

1.1.3 Why the Use of Online Environment?

Simulation tool in an online network and interfacing through web services solves a couple of problems at once. The first problem is related to HIL systems that able to run complex vehicle models. HIL systems are very expensive hardware which causes small research groups to struggle to research HIL systems. However, with being online there is no need of buying costly hardware. Only buying service and configuring the online server is enough which will be a lot cheaper than buying the HIL system itself. With this amount of cost reduction, small research groups even research themselves can research this online simulation environment.

Another problem with HIL systems has portability issues. This means that setting up a HIL system requires too much software installation, and some libraries or .dll files may be missing or corrupted in the target environment. Then buying technical support will be needed which will also increase the cost. However, being online and interfacing through web services makes the presented environment much more flexible and easier to adapt to any project with a little tweak of configuration.

To sum up and answer the big question, why the need for online real-time simulation? The answer is the need of a flexible, online, and no maintenance cost solution for small research groups and for companies that don't want to spend loads of money on expensive HIL systems for a single project. The main motivation for this thesis is to create an alternative solution for real-time simulation problems instead of HIL. Moreover, other use cases also worth study to show interesting uses that can help to

create curiosity in different research areas.

1.2 The Vision of Cloud-In-the-Loop

Our CIL vision mainly focuses on eliminating the disadvantages of HIL systems in automotive industry. HIL system requires a harness setup to work as a final product. Working on a complex harness setup may take away the focus on software development and cause wasting too much time. HIL systems are not so flexible also they require too much software installation and the prerequisite libraries, files, and powerful hardware. This issue brings up the problem of setting up a HIL system is too complex. With CIL it is possible to eliminate all harness management and also the complex set-up procedures.

To give more insight information, embedded software testing methods in the automotive industry are explained in upcoming sections.

1.2.1 Embedded Software Testing Methods

There are steps to test the embedded software of the vehicle control unit. Embedded software testing is the process of validation and verification of both hardware and software. These tests are done to ensure there are no defects in both hardware and software. Also, testings help to ensure that the designed system meets the requirements that come from the end-user.

1.2.1.1 Model-In-the-Loop

The first step of the verification part of Model-Based Design is Model-In-the-Loop (MIL). MIL basically feeding every possible input to the controller logic algorithm. The first process of MIL is to develop a Plant model (hardware model). These development phases usually take place in the MATLAB/Simulink environment. Then the development phase of the controller model starts. With this phase, it can be verified whether the controller can control the plant as requirements specified or not. In the

MIL phase controller logic is being tested on the simulated plant model. If the results of MIL are satisfactory then the next phase on testing is Software-In-the-Loop (SIL). Fig. 1 shows the general architecture of MIL testing.

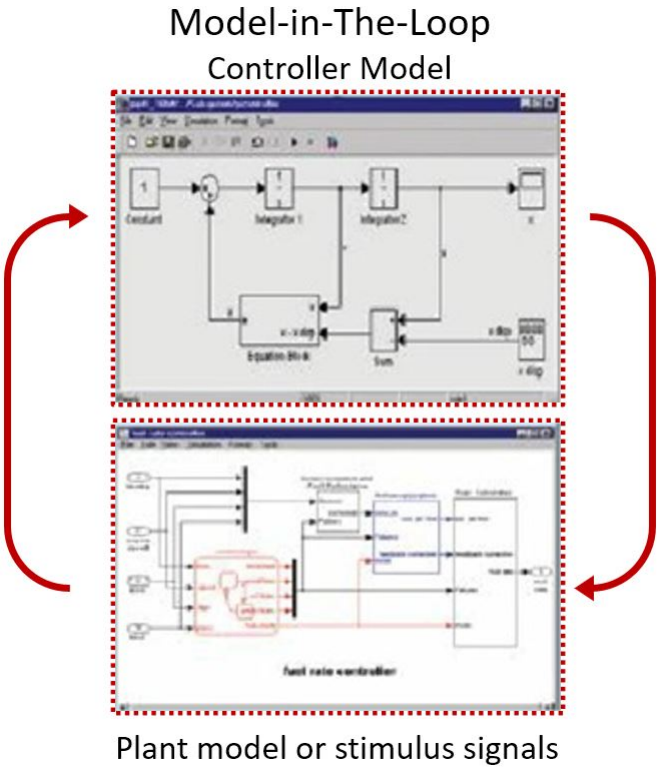

Figure 1: General architecture of MIL testing

1.2.1.2 Software-In-the-Loop

Since we are done with MIL testing now the code generation takes place. SIL is basically checking whether you generated C code works on the Plant model or not. In other words, this step is to check control logic that tested in MIL testing can be convertible to generated C code and to see if code is hardware implementable. The expected output on this step should be the same as MIL results. If there is a huge difference between SIL and MIL results, a step back to MIL testing is recommended. Fig. 2 shows the general architecture of SIL testing.

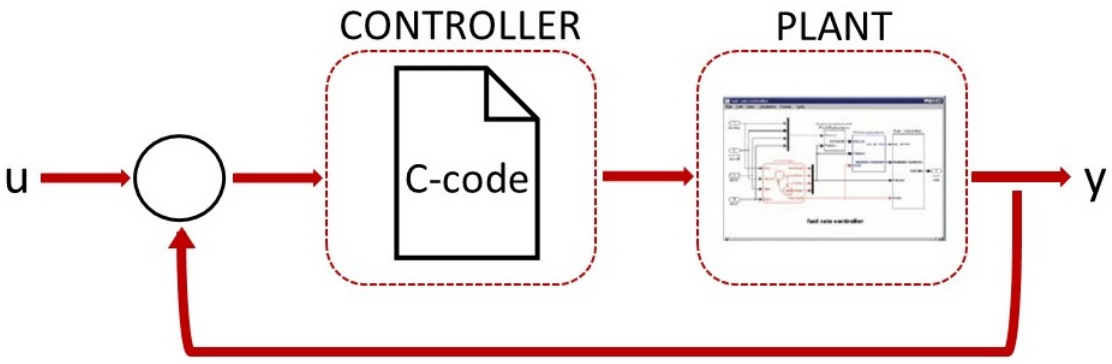

Figure 2: General architecture of SIL testing

1.2.1.3 Processor-In-the-Loop

The next phase of testing is Processor-In-the-Loop. PIL is where the generated C code is put to the actual hardware processor. In the PIL phase, the plant model is replaced by an actual processor. This test puts the developed model and the processor under investigation whether they can work together in harmony or not. Checking if the developed controller model can be run by the target embedded system. The result of the PIL testing should be the same as SIL and MIL results. If there are any differences between these testings, developed software should be revised and but under the testing again as the same order of MIL, SIL, and PIL. Fig. 3 shows the general architecture of PIL testing.

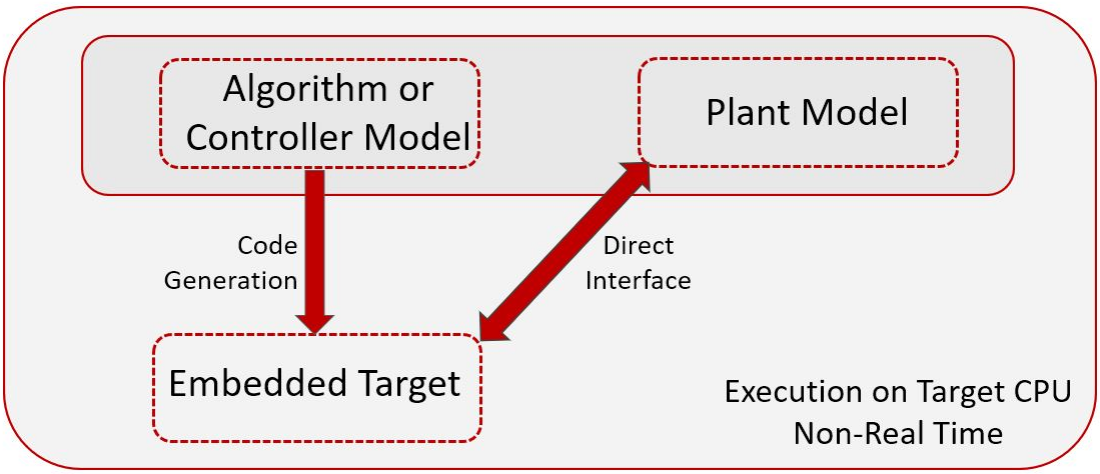

Figure 3: General architecture of PIL testing

1.2.1.4 Hardware-In-the-Loop

After verifying the designed model in PIL testing, now the model can be tested in real-time PC. This real-time PC is known as Hardware-In-the-Loop (HIL) system and testing is HIL testing. Before putting the controller model into final product hardware, it should be testing in the HIL system. HIL system is behaving exactly like final product hardware and feeds controller model's inputs with every possible combination. This testing show every possible behavior of the controller model. In Fig. 4 it can be seen that the HIL system hosts a plant model that feeds ECU with inputs.

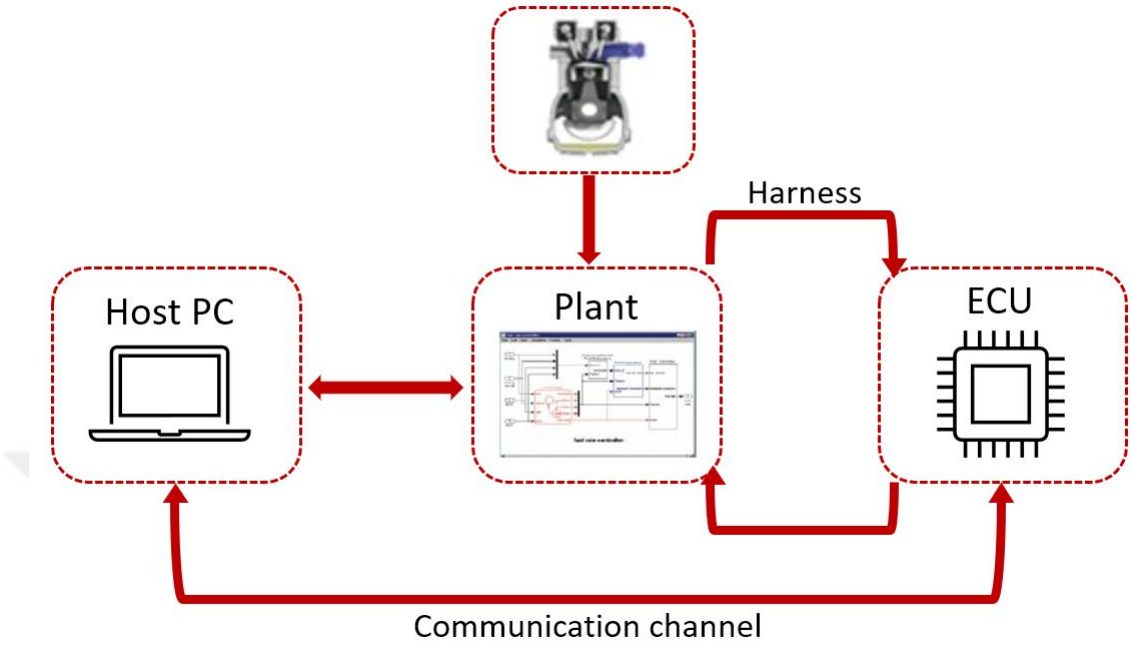

Figure 4: General architecture of HIL testing

1.2.1.5 Cloud-In-the-Loop

CIL is the presented use case for the developed simulation tool. It is possible to do all the testing in CIL instead of HIL. CIL is online so has very flexible features for installation and setup. HIL's maintenance requires a lot of workforces and can be very complex but CIL is running online, so it has no installation or maintenance just needs to be given correct inputs. With this flexibility, CIL becomes a very good replacement as HIL for small research groups. CIL also not requires expensive hardware as HIL. Just a service needs to be bought. With this feature, it is also great for small research groups. Instead of harness in HIL, web-service interfaces will provide communication. The process of model development and testing in CIL is the same as HIL. Fig. 5 shows the general architecture of CIL testing.

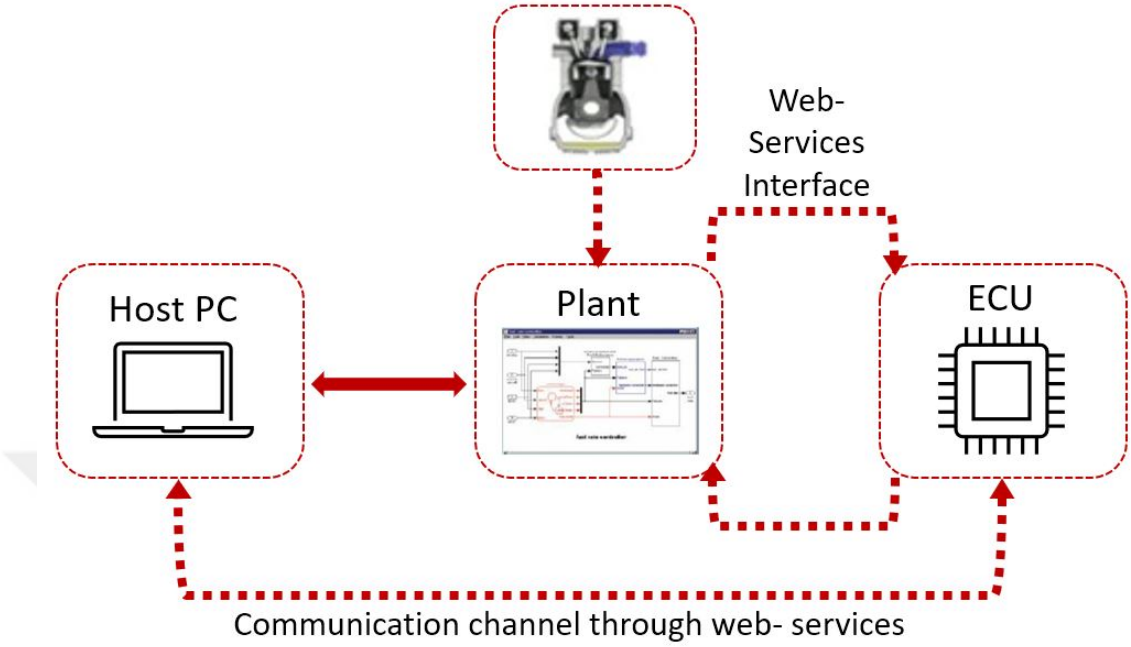

Figure 5: General architecture of CIL testing

1.2.2 Racing Control Algorithms

Another use case for the offered online real-time simulation environment is the evaluation of vehicle control algorithms in-terms of real-time response time performances. By performance evaluations, it is meant that the response time of EMA. To be more specific, EMA X can be very good at optimizing energy consumption but it may require too much time to do its' calculations which is not suitable for real-time vehicle control applications. On the other hand, EMA Y can be very good at real-time response time but it may not be that successful with the optimization of energy consumption. EMA Y can be a better option than EMA X for real-time vehicle control applications. At the end of the day, EMAs can be penalized or rewarded for their response time.

1.3 Previous Work

There are many works that contribute to the area of embedded software testing methods. In this scope first topic to investigate is MIL testing. Franco's [4] study demonstrates a valuable use case of MIL testing in the vehicle's dashboard system with CAN interface. Cakmakci [5] presents a MIL system that verifies the vehicle controller model and also has a plant model running in the MIL system.

The next step for testing is SIL and Taut's [6] study examines the SIL testing environment and also gives insight information about the software testing process. How MIL, SIL, PIL, and HIL operations progress. Moreover, Muresan [7] also investigates a SIL system in the aspect of embedded code testing. Muresan compares HIL and SIL in terms of data consistency and states that there is no difference between HIL and SIL testing [7]. Furthermore, he states that SIL testing can be started before HIL when the embedded system hardware is not present. For PIL testing Hu [8] presents a PIL system that tests and validates space vehicle embedded software.

There are many studies for HIL systems that validate embedded software of vehicles. Kendall [9] presents a work to test the vehicle body control unit with HIL systems. King's [10] study discusses how efficient HIL makes software development for vehicle control units. A utilized HIL system for the generation of railway vehicles' bending vibration is presented by Kabutomori [11]. Another HIL testing while developing a pure electric vehicle power train controller is presented by Hongyu [12]. Gozukucuk [13] also explains how testing of the proposed energy management algorithm in the HIL system is done.

Another area that is focused on during this project is web services. There are many studies on different areas that help to understand the usage of Google Maps API ([14], [15], [16], [17]). Mufid [14] presents an interesting formula to calculate estimated fuel consumption and is using Google Maps API to get distance data between two points. Another study is published by Gao [15]. Gao implements a trip planner with multiple

waypoints and optimizes the issue multiple waypoint trip path. Implementation is done in Google Maps API Javascript API. Pejic [16] implements an expert system for trips. He uses Google Maps API with AJAX interfaces and has great insight information on the development process. Konarski [17] implements a web portal on .NET framework. He explains the implementation of the web portal step by step and it is a great guide on understanding Google Maps APIs.

Another important topic was the communication interface during the development of the study. Matloff [18] has a great guide on network programming and explains the pillars of it clearly. Regnier [19] presents an alternative solution to increasing the efficiency of the TCP/IP socket data process. He also provides a great overview of Transmission Control Protocol/Internet Protocol (TCP/IP) Processing which helps to understand overall architecture, connection, receiver-side, and transmitter-side processes. Kalita [20] also provides a indepth details on socket programming over TCP/IP and compares it with User Datagram Protocol (UDP). Xue [21] also presents a Client/Server (C/S) application and gives a deep understanding of TCP and UDP.

There are also several use cases for the proposed tool one of them is Coordinated Driving([22], [23], [24], [25]). Gerla [22] presents an interconnected autonomous driving method and states how safety and traffic congestion can be reduced by vehicular cloud applications. There are also other researches on improving traffic flow by coordinated driving and Xiaoping's [23] work is one of them. Sasaki [24] also proposes a remote control system for coordinated driving and investigates cloud-based approaches. Another cloud-based application for the automotive sector presented by Zheng [25]. He also presents the network requirement for coordinated cloud driving.

1.4 Contributions of the Thesis

The contributions of this thesis to the literature are the following:

- An energy consumption prediction tool over the cloud in the context of the coordinated cloud driving.
- A vehicle control algorithm comparison tool in-terms of real-time response performances is proposed.
- Cloud-In-the-Loop environment that can be used as a HIL replacement.

1.5 Outline of the Thesis

Chapter II presents the tools that are used during the development of the CIL environment. Chapter III presents the simulation details and implementation. Chapter IV shows the evaluation of the environment as an energy calculation prediction of a light-duty commercial vehicle. Also presents the vehicle control algorithm comparison results in terms of real-time response performances. Last but not least, the result of CIL as HIL replacement. Chapter V is the last chapter that has the final discussion and future work proposals.

Chapter II

CLOUD-IN-THE-LOOP SOLUTION

Our CIL solution consists of a combination of different tools. These tools are detailed in this section to give a better understanding of the development of the presented project. This project relies heavily on terrain data collection and realistic vehicle models to make precise energy consumption. It also requires a good network architecture to provide reliable communication between a vehicle and a cloud-based server.

2.1 Terrain Data Collection

Terrain data collection is a crucial part of this dissertation because input data must be accurate as it is in the real world to make energy consumption realistic. There are plenty of web services that provide terrain data and Google Maps API is the famous one among them. Others are OpenStreetMap API, Yandex Maps API, and HERE Maps API. In this thesis, Google Maps API is used because of several reasons these reasons will be explained in next chapter named method.

Google Maps Web Services

There are a lot of Google Maps Web services from Wi-Fi locator to Route finder between points. In this project there are several Google Maps APIs are used. Background work about these web services is presented in upcoming sections.

Geo-Coding API.

Geo-coding API is a crucial sub-component of this project. Because to interact with the driver in-terms of coordinate points, origin and destination points need to

be requested. Svennerberg [26] has a good explanation about Geo-coding API and the rest of the web services as well. Geo-coding takes street addresses and converts them into coordinate points like latitude and longitude. Geo-coding also has reverse Geo-coding function which converts latitude and longitude points into actual street addresses. Another good study is Lemke's [27] on the accuracy of Google Maps Geo-coding API to OpenStreetMap Geocoding API.

Directions API

Directions API is the main API in which road information is gathered from. Directions API requests origin point and destination point in place ID format from geo-location API and then returns route between input points. Socharoentum [28] has a really good comparison and very deep study to understand how Google's and other mapping algorithms work. Also, Socharoentum is explaining the distance calculation methods of different Web Mapping APIs (WMAs).

Another good example is Li's [29] work to understand how routing services are working. An in-depth explanation of how traffic data can be added to adjust routes throughout the trip. Adding traffic information can be an accuracy increasing feature for this thesis.

Maps Elevation API

This project is mainly focused on the automotive/transportation area. In the automotive industry, to make accurate simulations, it is needed that the input data, which is the terrain, slope, elevation, etc., must be accurate. Han [30] conduct a very beneficial study about the accuracy of Google Maps Elevation API. Which explains that how it has better accuracy on different route profiles. Han [30] states that measured data with GPS and gathered data from Google Elevation API has in the range of 50 cm to 70 cm difference than adds that Elevation API can be useful on low-resolution elevation data required automotive applications.

2.2 Vehicle Modeling

Vehicle modeling is paramount for this project. In order to get accurate energy consumption, the vehicle model that runs in the server should be identical to the real world. To achieve this goal there are several methods. To decide on which method to use firstly vehicle power-train type should be decided. There are several power-trains to consider and details of these power-trains will be given in the upcoming section.

Vehicle Power-train Model

Details about vehicle power-trains will be given in this section.

- Internal Combustion Engine (ICE) Power-train
- Full-Electric Power-train
- Hybrid Power-trains
  - Series Hybrid Power-train
  - Parallel Hybrid Power-train
  - Power-split Hybrid Power-train

Internal Combustion Engine (ICE) Power-train

ICE is the most traditional power-train type that is used since the early 1880s in the automotive industry which we can call the modern car industry. Only one engine is turning wheels in this type of power-trains. The engine is connected to transmission and transmission is connected to wheels. Power is generated by burning fuel inside cylinders of ICE. ICEs cylinder timings are shown in Fig.6.

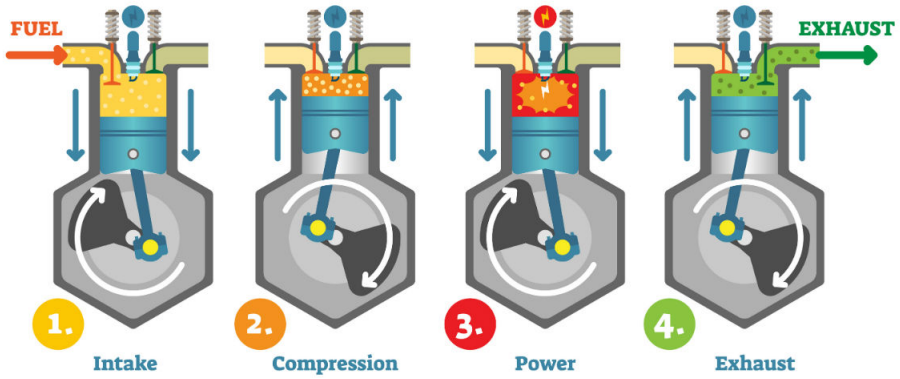

Figure 6: Four stroke cylinder timing [1]

General configuration of ICE power-train shown in Fig.7.

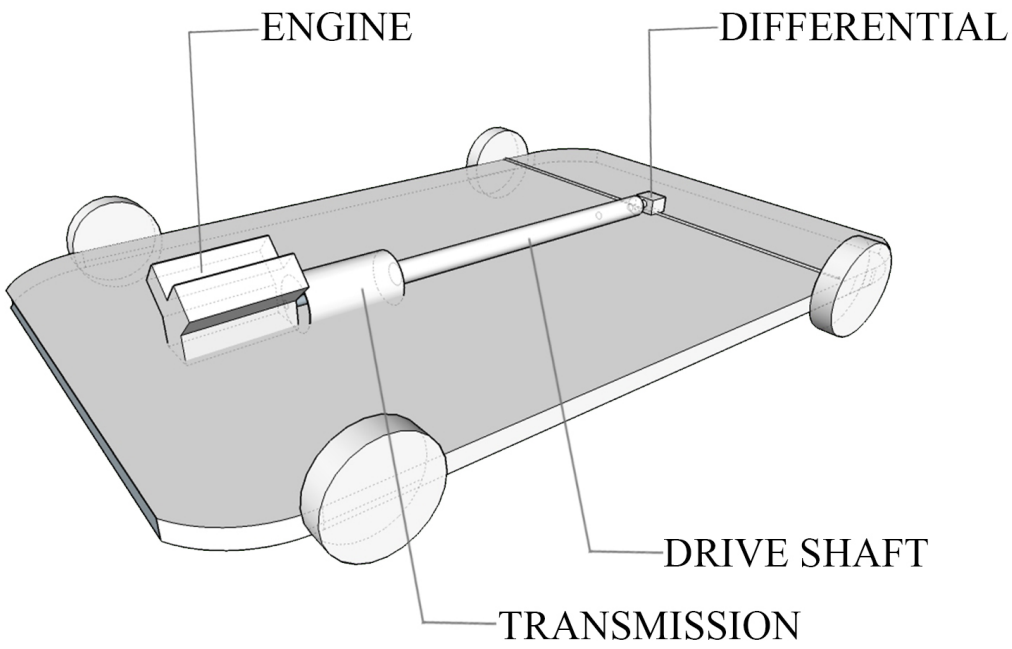

Figure 7: General ICE power-train configuration

The modeling of the ICE power train depends on the level of detail. In some simulations valve timing, cylinder timings might also be needed then the model of

engine will be very complex. If only power output needed than a simple mathematical model can do the job. Cook's study [31] is use full to understand the modeling approach for ICEs. Another study on automotive engine modeling has been done by Weeks [32] to show a wide range of use of engine modeling methods.

Full-Electric Power-train

Full electric power-train is the most straightforward power-train among all other configurations. It only consists of a battery, a traction inverter, and an electric motor in terms of torque generation. General configuration of full-electric power-train shown in Fig.8.

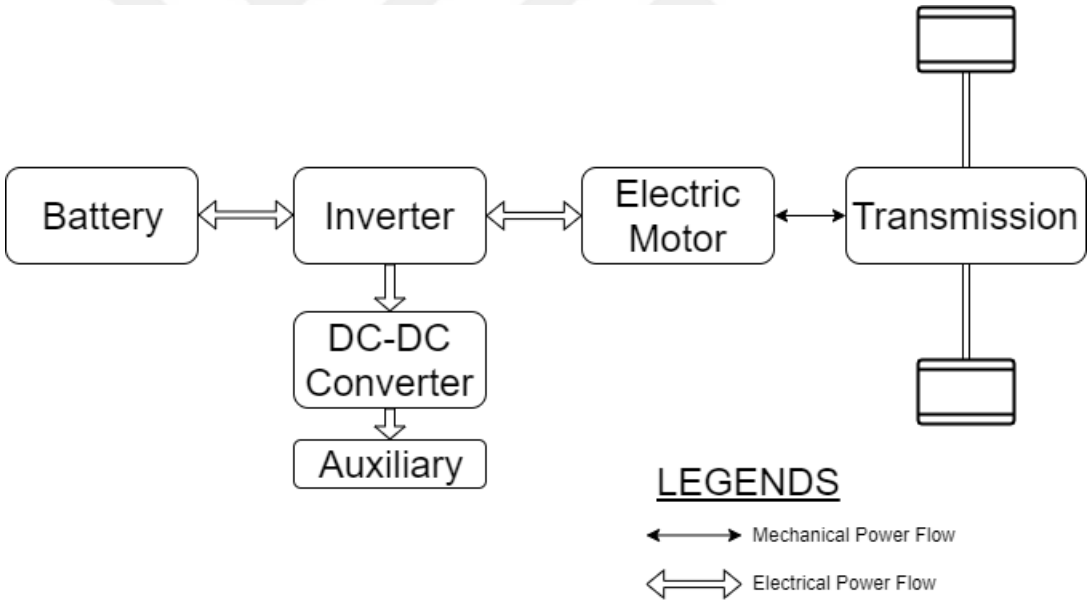

Figure 8: General electric power-train configuration

Walker [33] has an explanatory study about full-electric power-train modeling. Walker states that there are four main components of full-electric power-train modeling which are battery pack, electric motor, transmission, vehicle, and driver model. He explains in detail with mathematical models of each component.

Hybrid Power-trains

- **Series Hybrid**

  Series hybrid is also known as range-extended hybrid vehicles (REEV) where an ICE motor turns the power generator only to provide electrical energy to vehicle batteries. Evangelou's [34] study presents the modeling of various series hybrid electric vehicle components. Fig. 9 shows the general configuration of series hybrid power-trains.

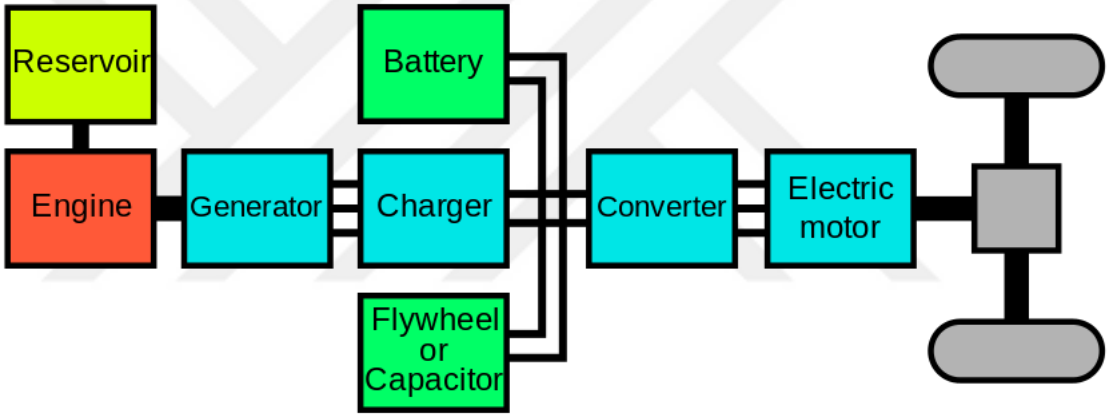

Figure 9: Series hybrid power-train configuration [2]

- **Parallel Hybrid**

  Parallel hybrid is where bot ICE and Electric Motor turns the wheels individually. These types of power trains are the most common one that is used in the hybrid-electric vehicle (HEV) market since 2016. Enang [35] has a comprehensive review that sums up different modeling approaches on parallel hybrid vehicles. Fig. 10 shows the general configuration of parallel hybrid power-trains.

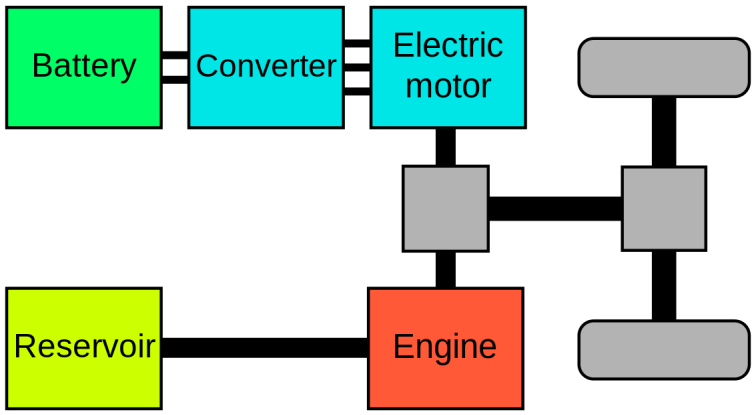

Figure 10: Parallel hybrid power-train configuration [2]

- **Power-Split Hybrid**

  Power-split hybrid power-train where gets most out of both series and parallel power-train types. It requires less torque on the electric motor with a combination of ICE, especially for large vehicles. Toyota Prius is a famous example of a power-split hybrid power-train. Liu's [36] work takes Prius as an example and presents a three-state dynamic vehicle model for power-split hybrid vehicles. Fig. 11 shows the general configuration of the power-split hybrid power-trains.

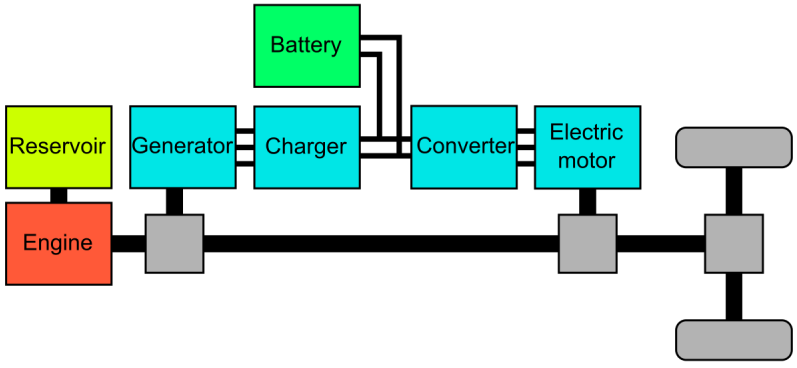

Figure 11: Power-split hybrid power-train configuration [2]

2.3 Network

To create a simulation environment, a communication protocol, and a communication system needed to be implemented. Python Low-level Networking Interfaces are used. Socket programming is one of them and it is used as the main interface to create communication between vehicle and server. There are several internet protocols (IP) to create this type of communication environment. The most commonly used one is the TCP. Another protocol is UDP which is not suitable for this project and reasons will be explained later on the dissertation. Sarker [37] explaining these protocols beautifully and presenting why TCP should be used in this type of project that requires in-order delivery, receiver acknowledgment, error detection, and flow and congestion control. These services provided by TCP service is crucial for this project. To communicate with the vehicle in-order of segmentation points, in-order delivery is needed. To detect if the vehicle receiving our data successfully, receiver acknowledgment is needed. To detect if any error occurred during the transfer of data to the vehicle, error detection is needed. While communicating with multiple vehicles at once, control of data packages is need to manage package transfer. Details of the Network Architecture will be given in the upcoming chapter.

Chapter III

SIMULATION DETAILS

This project presents an online real-time simulation environment. *Algorithm 1* is a pseudocode for the simulation process. Also, its process can be described as follows. Firstly, a route is created with the points taken from the driver by Google Maps APIs [38]. These points are the origin point and the destination point. After that, the route is divided to create segments, which helps to track the vehicle on the route. These segment beginning and endpoints are sample points and at each of these points, the simulated vehicle's EMA is requested to send velocity values in real-time while at simulation run-time. EMA used in this project is developed in previous work [13]. Velocity is represented as V in Fig. 7. Upon successfully receiving the velocity value from the EMA energy calculation algorithm takes place.

The energy calculation process takes a certain amount of time, this time is denoted as Δv. After Δv time passed, the energy that is calculated by simulation is sent back to EMA. Energy is denoted as E in Fig.7.

Another important point that needs to be considered and simulated is that communication loss. If a loss occurs, previous data is fed to simulation again to assume that vehicle is going at a constant speed. If the time to calculate energy consumption takes too much time and vehicle passes the target segment point without getting the energy consumption, then simulation aborts the calculation and starts a new iteration of energy consumption calculation with freshly received speed data from the vehicle control unit.

input : Origin Point and Destionation Point of the Route
output: Energy Consumption of Vehicle

```
Receive(originPoint, destinationPoint)
FindRoute(originPoint, destinationPoint)
RouteSegmentation(originPoint, destinationPoint)
StartSimulatingVehicle():
  CheckConnection:
    IF connectionOk:
      Receive(location, initialV, targetV)
      CalculateEnergy(location, initialV, targetV)
    ELSE:
      CheckConnection:
    ENDIF
  END CheckConnection
  CalculateEnergy:
    FOR eachSegment:
      energy = CalculateEnergyCurrentSegment(segment, initialV,
  targetV)
      SendEnergy(energy)
      Receive(initialV, targetV)
      IF receiveOk:
        SetNewV(initialV, targetV)
      ELSE:
        SetOldV(initialV, targetV)
      ENDIF
    ENDFOR
  END CalculateEnergy
END StartSimulating
```

Algorithm 1: Pseudocode of simulation process

Below Fig.12 represents the data exchange between the energy management algorithm which is the mentioned vehicle as a general representation.

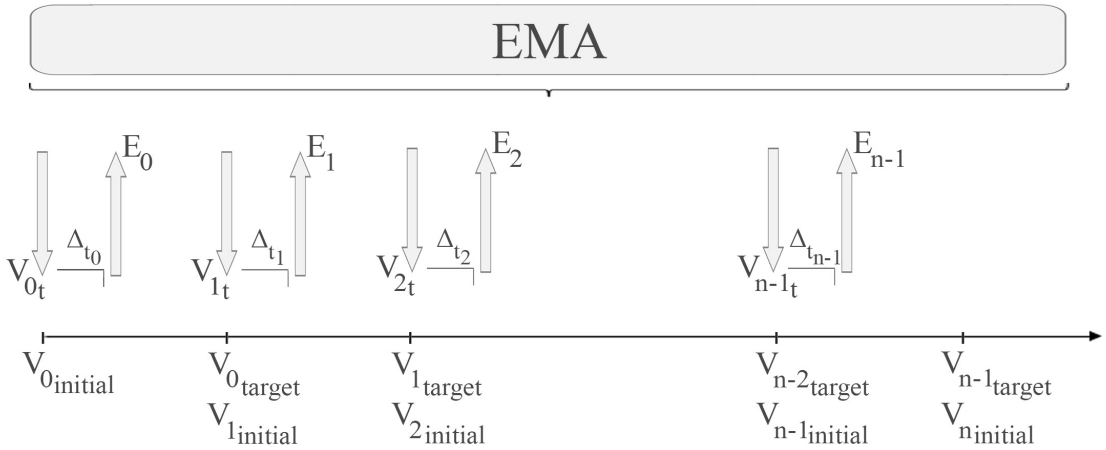

Figure 12: Simulation process

3.1 Terrain Data Gathering

Terrain data gathering is a crucial part of this project. Without proper data, energy consumption prediction will not be accurate as it needs to be. To get proper terrain data Google Maps APIs [38] are used. To be more specific Google Maps Elevation API, Geo-coding API, and Direction API.

The process of data gathering can be expressed like this: Driver gives input of origin and destination points. After that, geo-coding API takes these addresses and converts them into unique place IDs. Then, these two points will be feed into Directions API as inputs to generate a route between origin and destination point.

Directions API gives route legs divided at turning points which is non-uniform. To put it in better words, if there is a long straight part of a road that is for example five kilometers long, Directions API will give one leg that is five kilometers long.

Problem with long route legs, speed, and elevation changes will not be sampled good enough to make accurate energy consumption prediction. To understand the problem statement give above, Fig. 13 shows the non-uniform sampled route. Markers on the map represent the sampling points that are acquired from Google Maps Directions API.

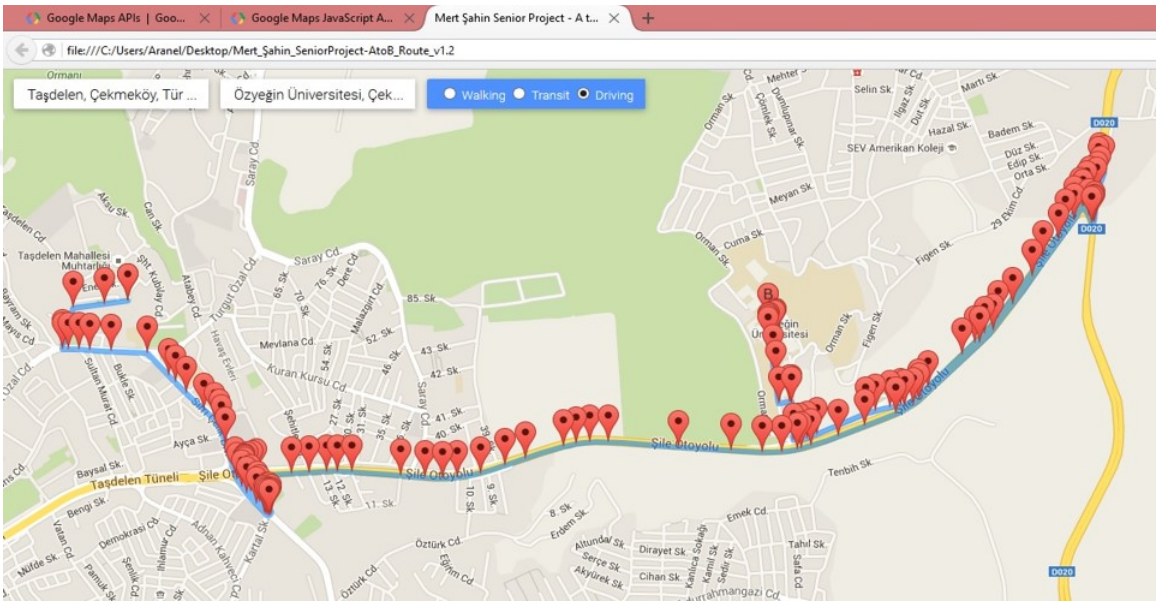

Figure 13: Non-uniformly sampled route example

Elevation API has good functionality to overcome this long leg segmentation problem which is called elevation along a path. Elevation along path function takes two inputs. The first input is the path which consists of start and end coordinates of the path. The second input is the number of samples to return. Then elevation API returns the equally sampled points, in distance-wise, in the path and the elevation of these points. So that, the route leg output of the Directions API can be given to Elevations API to get an elevation map of route legs. Below represented Fig. 14 which has a uniformly sampled route with the help of elevation along a path function of Elevations API.

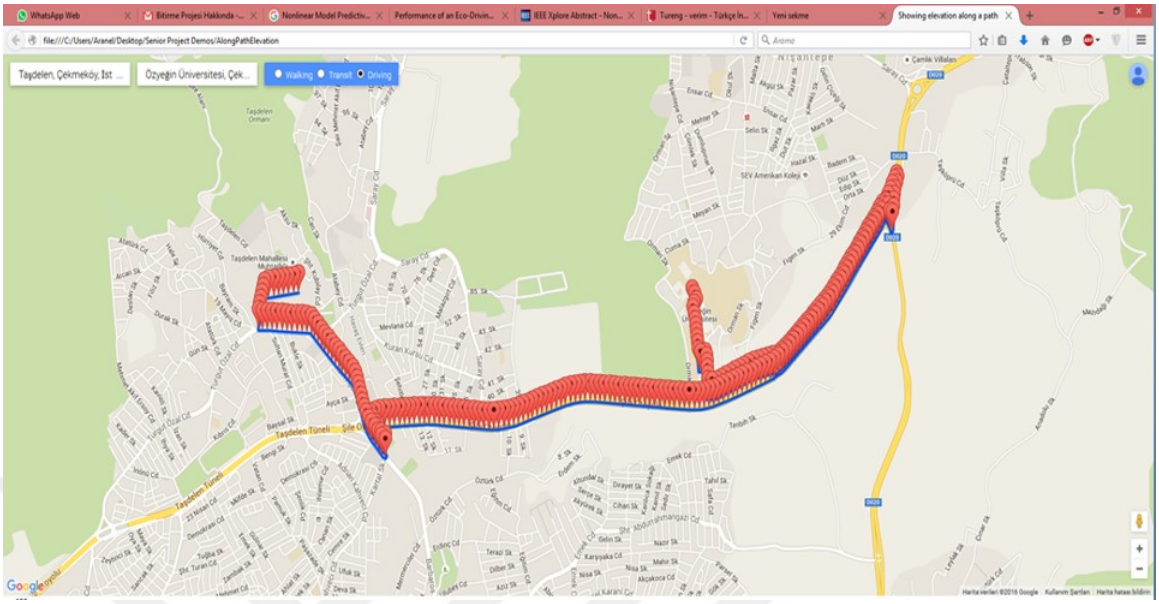

Figure 14: Uniformly sampled route example

Elevation along path serves two purposes, it divides the route into equal-length segments to increase the accuracy of route sampling. The second purpose is to get elevation values in the route that is acquired without using any other function. This means fever communication traffic between Google servers and applications. API requests are spending most time resources during the simulation process.

Another difficulty while gathering data is segmentation rate. Increasing the segmentation rate increasing the energy calculation accuracy by getting more data during drip which helps to catch small velocity fluctuations. However this causes another problem during the simulation, it will be very difficult to respond VCU in real-time as it passes the segmentation start and endpoints too fast. The requirement to accomplish health communication is either have a super-powerful server that can run vehicle models too fast or increase the segmentation intervals according to what you have in terms of computational power

To have a better understanding of segmentation points, segmentation intervals, and sampled route below shown Fig. 15. It is the view of the development mode

of GUI. In development, mode markers represent the segmentation points. Each of the markers is put in a sampling point which allows us to track the vehicle going through the route. In the figure you can see the origin point is "Sanayi, Sabiha Gökçen Uluslararası Havalimanı (SAW), Pendik/İstanbul, Turkey" and the Destination Point is "Nişantepe, Özyeğin Üniversitesi, Orman Sokak, Çekmeköy/İstanbul, Turkey" and sampling interval is defined as 5 meters.

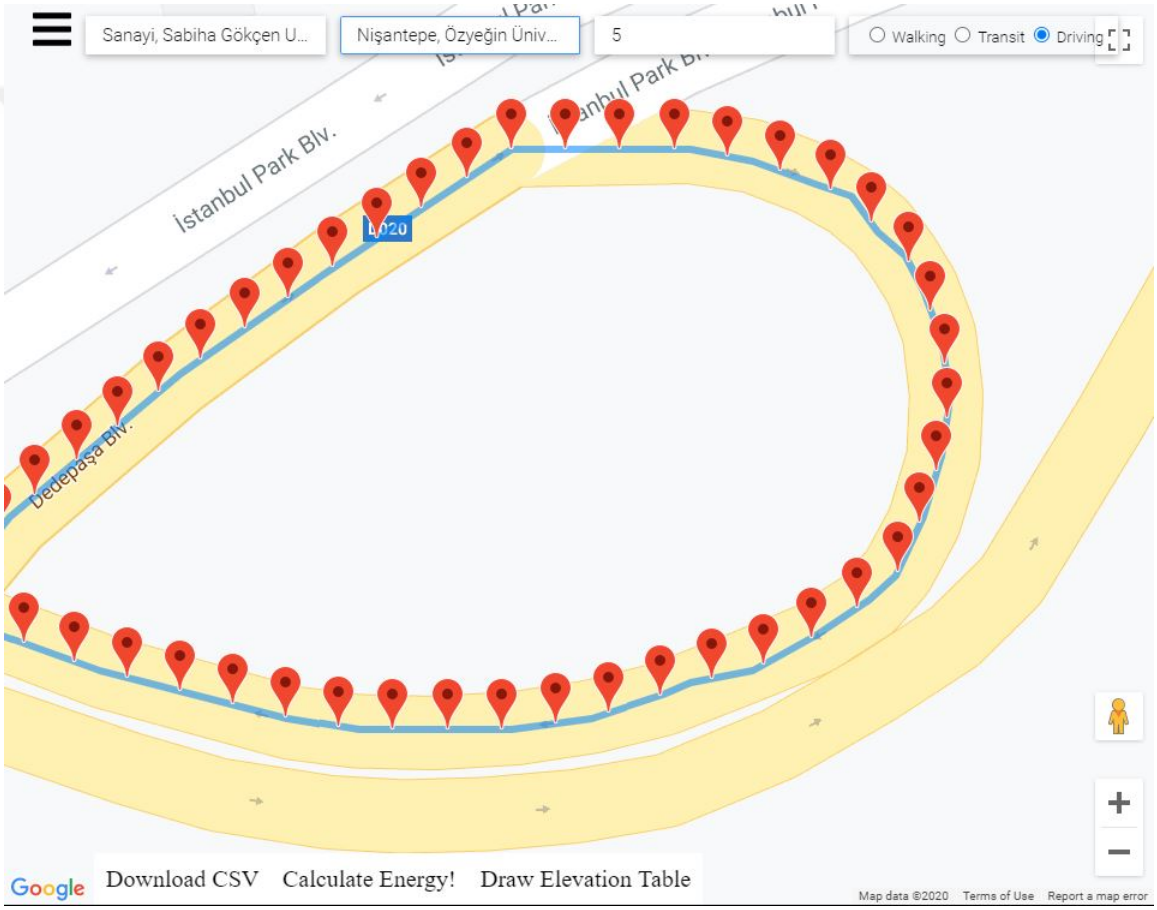

Figure 15: Sampled route example

3.2 Vehicle Dynamics Modeling

In this dissertation a plug-in electric vehicle taken as a simulation vehicle. The studied commercial plug-in electric vehicle has a battery of 66kWh lithium-ion battery capacity, a transmission with a ratio of 18.5, and an electric motor with peak power

of 75 kW. The architecture of the studied vehicle can be seen in Fig.16.

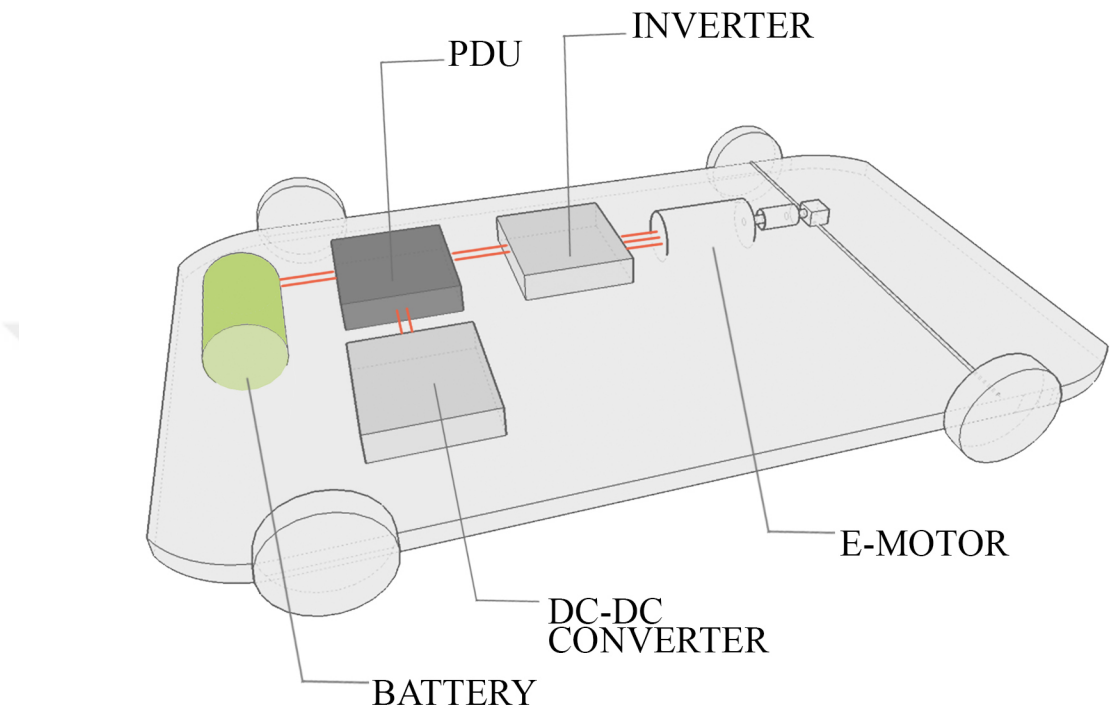

Figure 16: Electric vehicle architecture

In-vehicle modeling, the vehicle dynamics modeling approach has been used in this project. Vehicle dynamics are represented by the motion of a mass which is second of Newton's law of motion. Gillespie's [39] book has pretty good explanations on the vehicle dynamics. Below is the equation off Newton's second law of motion.

$$\sum f = m \cdot \frac{\partial v}{\partial t} \tag{1}$$

In a vehicle, while moving there are several forces acting on it. These forces can be listed as:

- Aero-dynamic resistance drag force (F_{drag})
- Force on wheels (F_{wheel})
- Rolling resistance force (F_{roll})
- Gravitational force (F_g)

Fig 17. shows the forces acting on a vehicle on a slope. As can be seen from the figure, the vehicle needs to overcome aerodynamic resistance, rolling resistance, and gravitational force with acceleration force.

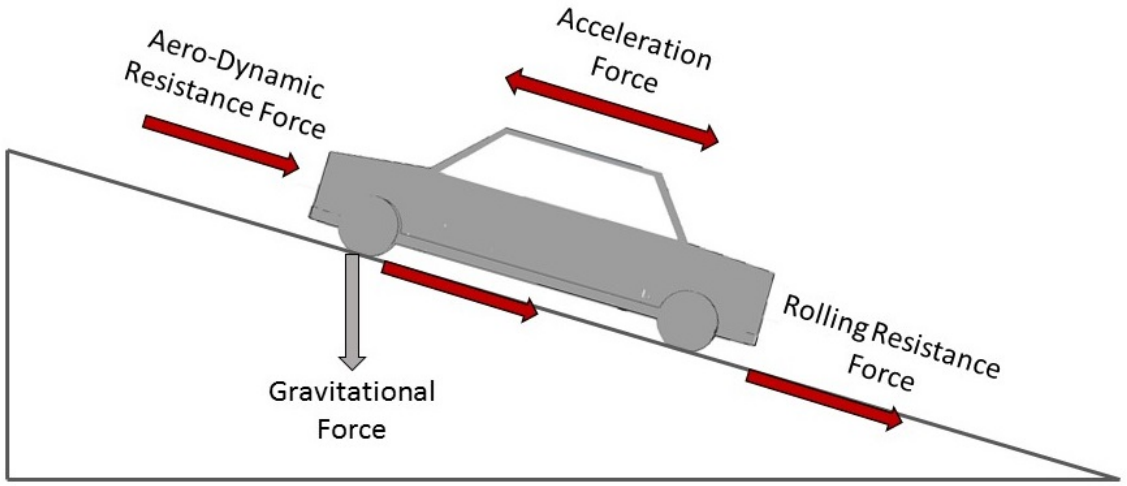

Figure 17: Forces acting on a vehicle

To explain these forces we shall start with the aerodynamic resistance force which is also known as a drag force. Drag force's equation has drag coefficient C_d which unique to every geometrical shape that faces to fluid. The equation also has the frontal area of the vehicle (A), air density (ρ) which vehicle moves in and the square of vehicle's speed V^2. Hucho's book [40] has all the basics of the aerodynamics of vehicles. It is recommended to scan this book to understand vehicle dynamics. Drag force equation is given below:

$$F_{drag} = \frac{1}{2} \cdot \rho \cdot C_d \cdot A \cdot V^2 \tag{2}$$

Another force that acts on the vehicle is the rolling resistance force. Rolling resistance force is highly effected by road friction and tire grip level. Higher the road friction and tire grip, the higher the rolling resistance force. DeRaad's study [41] is perfect to understand road surface effect rolling resistance force. Another great study on tire condition effect on rolling resistance force has been done by Grosch [42]. These two publications are very good readings to understand the concept of rolling resistance force. Rolling resistance force is calculated by rolling force coefficient(C_{rr}) multiplied by mass(m) times gravity(g) Rolling resistance force equation is given below:

$$F_{roll} = C_{rr} \cdot m \cdot g \tag{3}$$

Third force is the force on wheels $\mathrm{F_{wheel}}$. Force on the wheels is determined by generated torque by e-motor(T_{motor}), the gear ratio of transmission(G_b) and the radius of the tire(r_{tire}). Its equation can be expressed as below:

$$F_{wheel} = \frac{G_b \cdot T_{motor}}{r_{tire}} \tag{4}$$

Fig. 18 shows the torque path on a vehicle from engine to wheels. Also, it shows how torque is affected by different components of power-train. The torque generated by the engine is multiplied with gear ration and then drive shaft transfers the torque that is multiplied with gear ratio to the differential. Lastly, differential transfers torque to wheels, and in the wheels, torque is multiplied by tire radius.

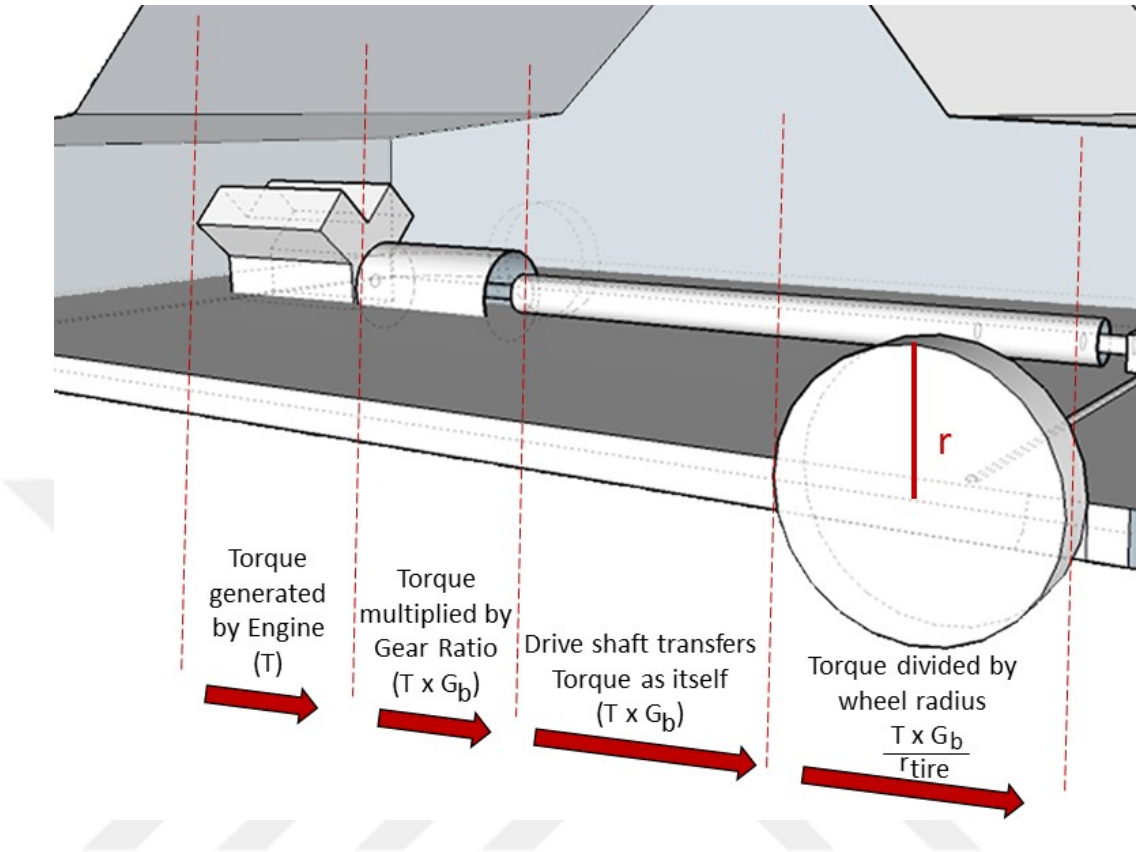

Figure 18: Torque path in a vehicle

Last but not least the gravitational force. Gravitational force(F_g) represents the force acting on the vehicle on uphill, it depends on the slope(θ) of the road segment, vehicle mass(m) and gravitational constant(g). Following equation represents the gravitational force on the vehicle:

$$F_g = m \cdot g \cdot \sin\theta \tag{5}$$

To be accurate on energy consumption calculations it is needed that the calculation must consider most if possible all real-life energy consumption of the vehicle. To achieve this, another energy consumption that needs to be taken into account is auxiliary power consumption. It is taken as constant through the simulation and it assumed that the high voltage comfort systems are not activated through the trip. Constant auxiliary power consumption, in this project, is $300Wh$.

To sum up, all the forces acting on vehicle and to get a Time-varying rate of change on vehicle speed an expression can be created as below:

$$\frac{\partial v}{\partial t} = \left(\frac{1}{m + \frac{I_{motor} \cdot G_b^2}{n_g \cdot R_{tire}^2}}\right) \cdot (F_{tire} - F_{roll} - F_{drag} - F_{grav}) \tag{6}$$

Used constant in equations 2 - 6 are described below with their values. These parameters are taken from a plug-in full electric light-duty commercial vehicle that is operated on the road. Parameters are presented in Table.1.

Table 1: Electric vehicle parameters

Symbol	*Value*	*Unit*	*Description*
C_d	0.55	–	Drag coefficient
m	3500	kg	Vehicle mass
A	5	m^2	Vehicle frontal area
G_b	18.5	–	Transmission gear ratio
r_{tire}	0.354	m	Tire radius
c	0.008	N	Rolling resistance coefficient
g	9.80665	$\frac{m}{s^2}$	Gravitational constant
n_g	0.9	–	Tire specific coefficient
I_{motor}	0.11	$Kg \cdot m^2$	Motor inertia

3.3 Network Architecture

For the experiments, which explained upcoming sections, a network communication architecture has been set up. The setup consists of 3 main components. First, the simulator server which has most of the computation load throughout this experiment. The second one is Google Maps Web Services which provides In this experiment vehicle control unit is sending origin and destination point to Google Maps Web Services

and Google Maps APIs returns elevation and route between origin and destination point. Details of this process have been explained in section name gathering terrain data. After getting the route and elevation details, route segment data sent to the simulator server one by one as the trip starts. The simulator server calculates energy consumption prediction and speed profile. There is another detail on the speed profile because it can be just energy consumption if the simulation is set not to calculate the optimum speed profile. Fig. 19. shows the general architecture of this simulation environment.

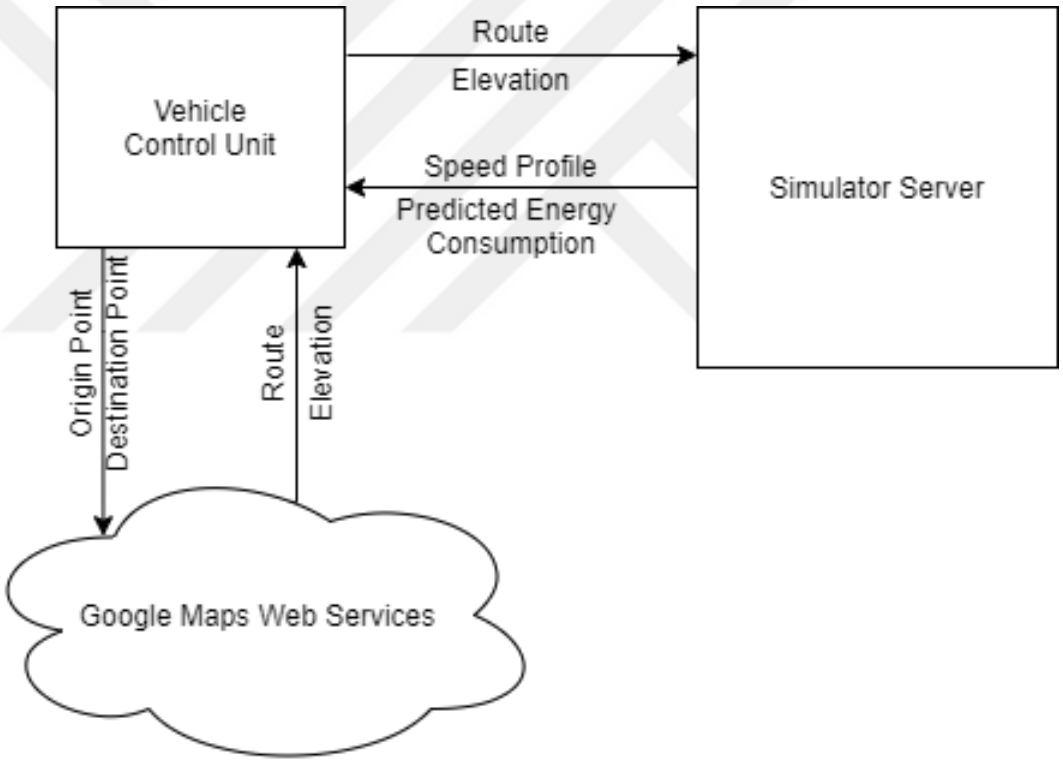

Figure 19: General network architecture

3.3.1 Client Side

For the experiments as Vehicle Control Unit an arbitrary personal laptop used. In this VCU, There is a GUI that offers an interface to the driver and Python program that manages the communication between the server. Below presented the general

workflow of the client in Fig. 20.

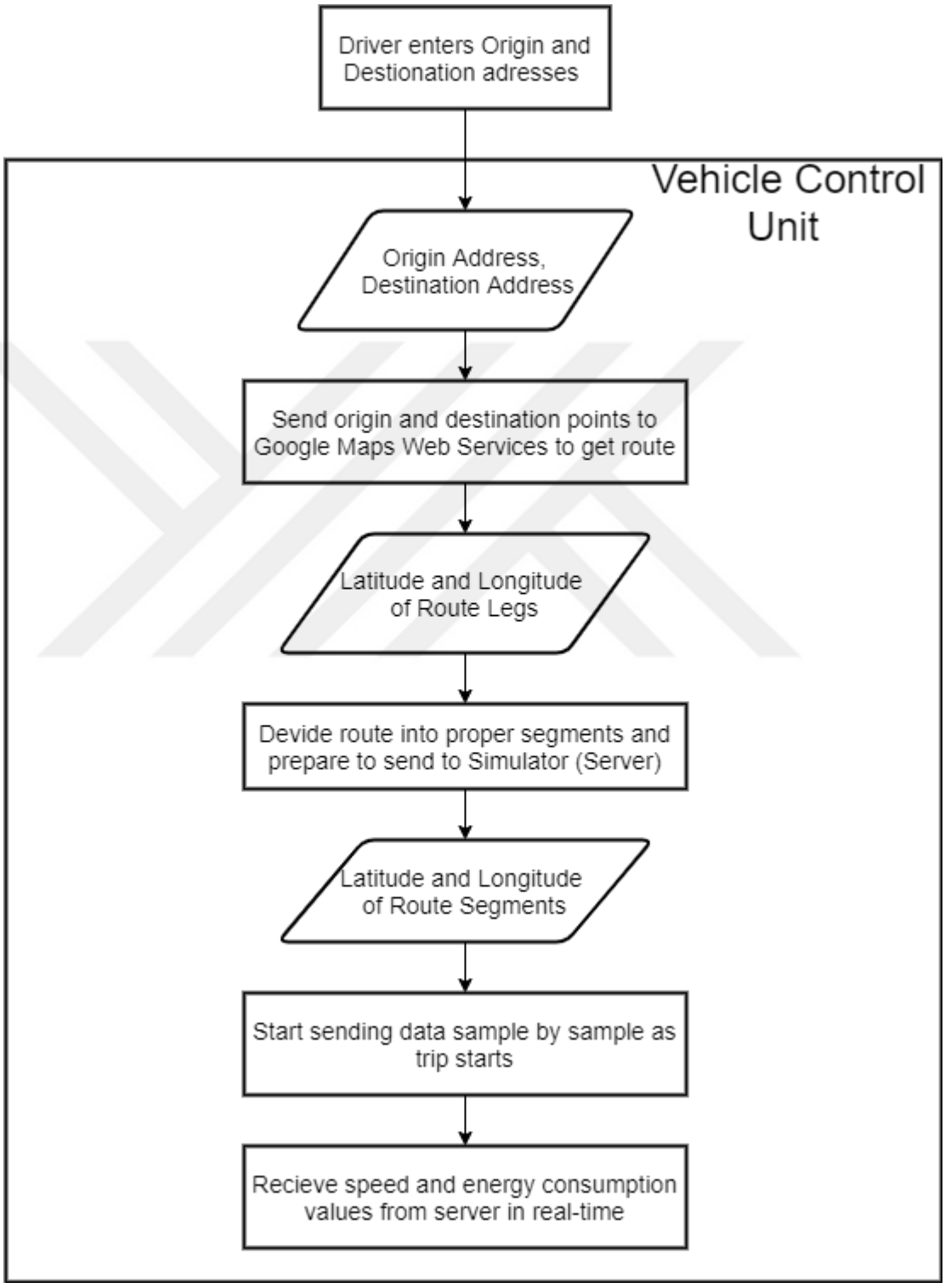

Figure 20: Client VCU workflow

A Graphical User Interface(GUI) is developed for these simulations and it is a part of the Vehicle Control Unit that offers an interface to the driver. On the front-end of this interface, there are HTML codes for the overall structure, CSS codes for visual layout. In the back-end of the web page JavaScript code for general Google functions and Ajax queries for data exchanges. Also, PHP codes to start a Python program. Firstly, the Integration of Google visuals and functions had to be done. For this purpose JavaScript API of Google Maps, Web Services is used. However, JavaScript API requests are using call-back functions which interrupts the program routine which is not suitable for our application because we wanted to make elevation requests in-order as each segment point gathering. For this purpose Ajax [43] offers an easy and great solution which is called synchronous queries. With Ajax, the developer has great control on data exchange with Google Maps API servers. synchronous queries require to be completed before the next declaration can be called. In other words, stops the program execution until the Ajax query is completed.

Another point while developing GUI is the server was operating in python language and the communication between server and client has been done in python. The challenge was to communicate JavaScript with Python Socket Program. To overcome this challenge, PHP [44] offered a great solution which is called shell execution. Shell Execution is running a shell command from the windows shell command window and returns the complete output as a string. Essentially, Python used to simulate the vehicle as it was going on the route. It was communicating with a simulator server which is also developed in Python language, and sending route samples one by one to the simulator as if a real vehicle was passing trough these sample points. Fig. 21 shows the front-end of GUI.

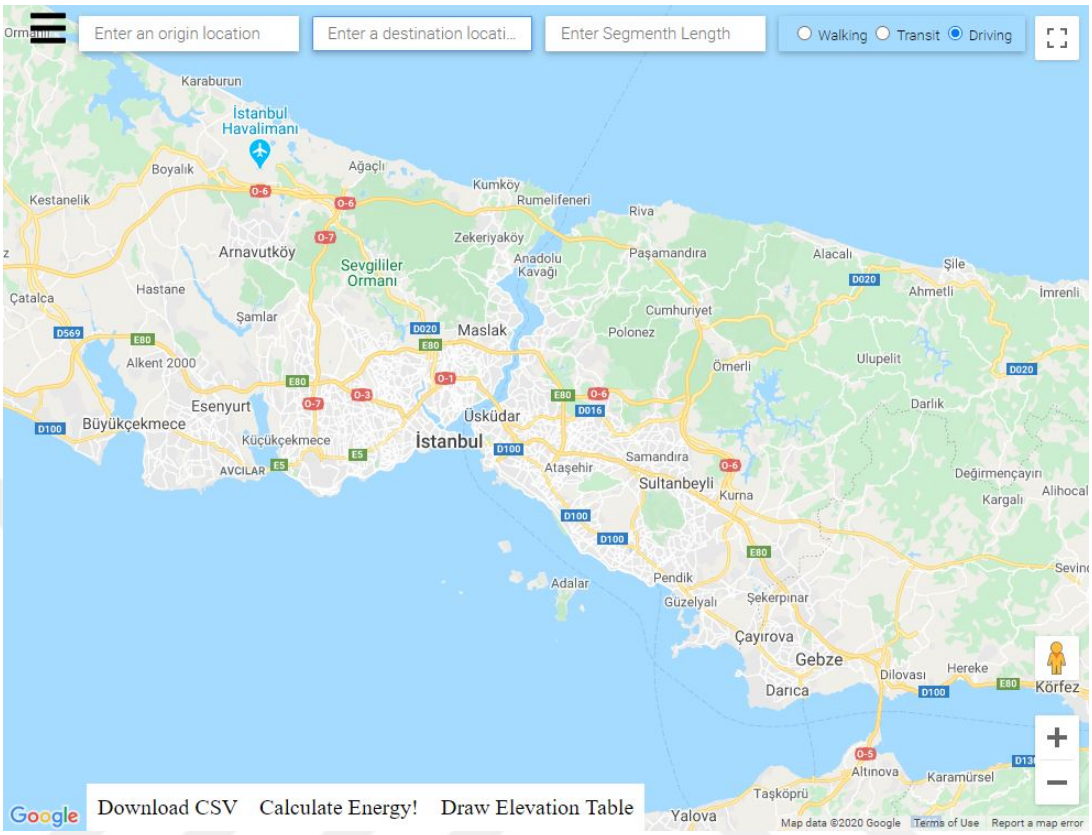

Figure 21: GUI of front-end

3.3.2 Server Side

For Simulator Server, a desktop relatively has more performance resources that a laptop has been used. The simulator server is started simulating by VCU requests if there is a connection from the VCU server starts a new thread for active connection and start simulation as soon as the segment data arrives. The workflow of simulator shown in Fig.22.

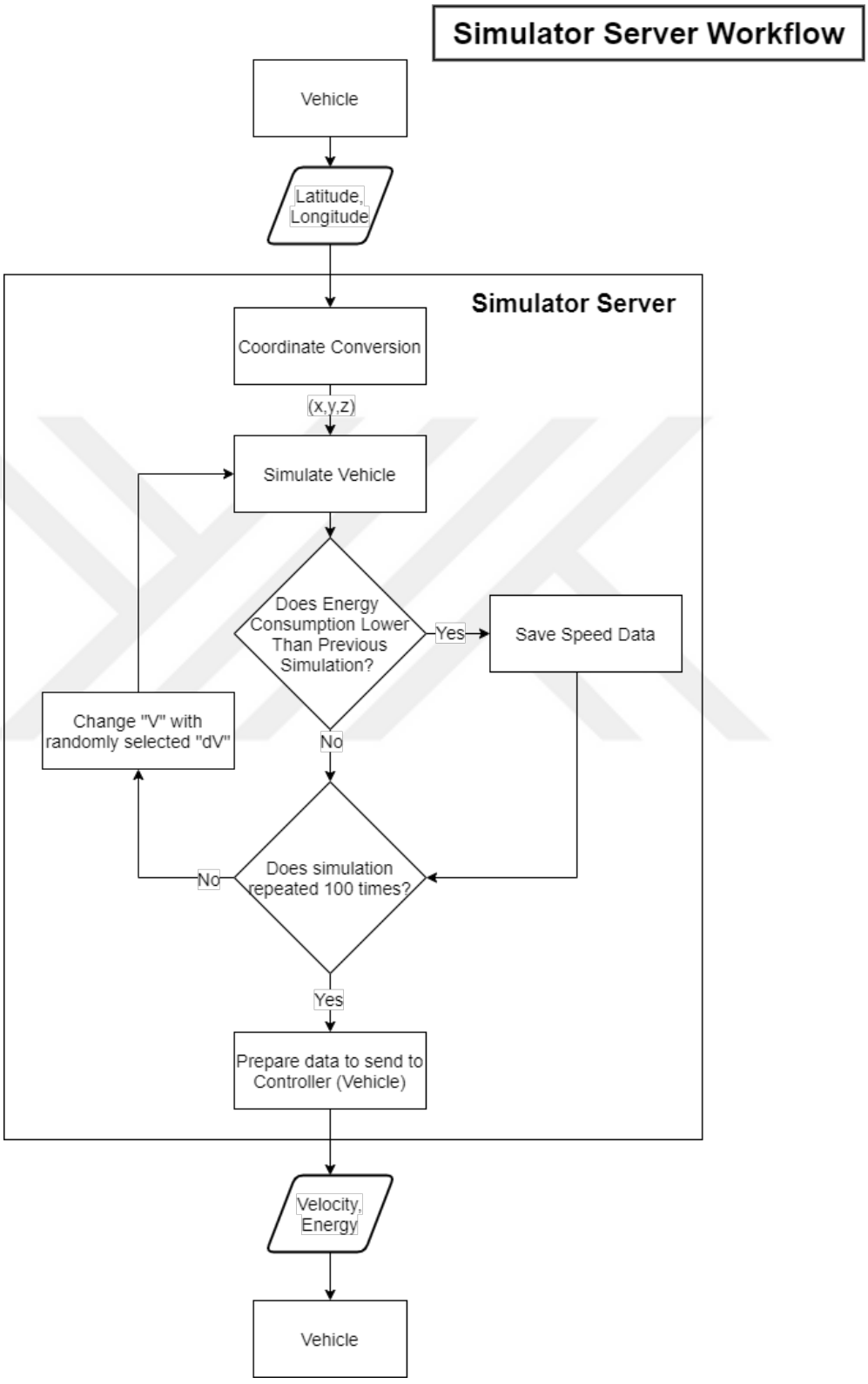

Figure 22: Server simulator workflow

During the development of Simulator Server, Python Socket Programming is used. With Python's socket interface the communication between client and server was provided. TCP is used because it provides in-order delivery and error report. Also, the vehicle model is running on the server. To develop vehicle model Numpy, Math, and Mat-Plot packages of Python are used.

Another important feature of the server is allowing multiple vehicle connections at once. To provide this feature Python thread package is used. Threading is basically executing your code in the separate flow which means multiple executions happening at once. Below Fig. 23 represents the server behavior on multiple client connections.

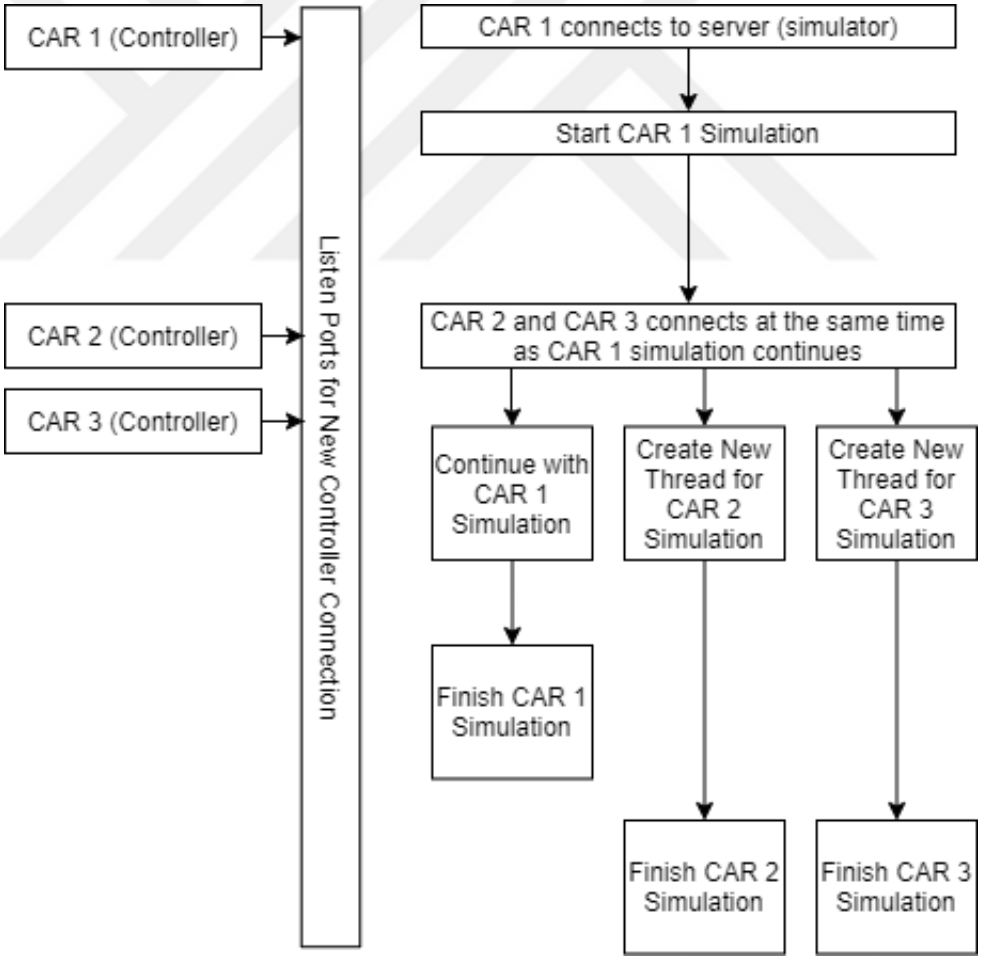

Figure 23: Server behavior on multiple TCP client connection

Lastly, a sequential flow diagram can be seen in fig. 24.

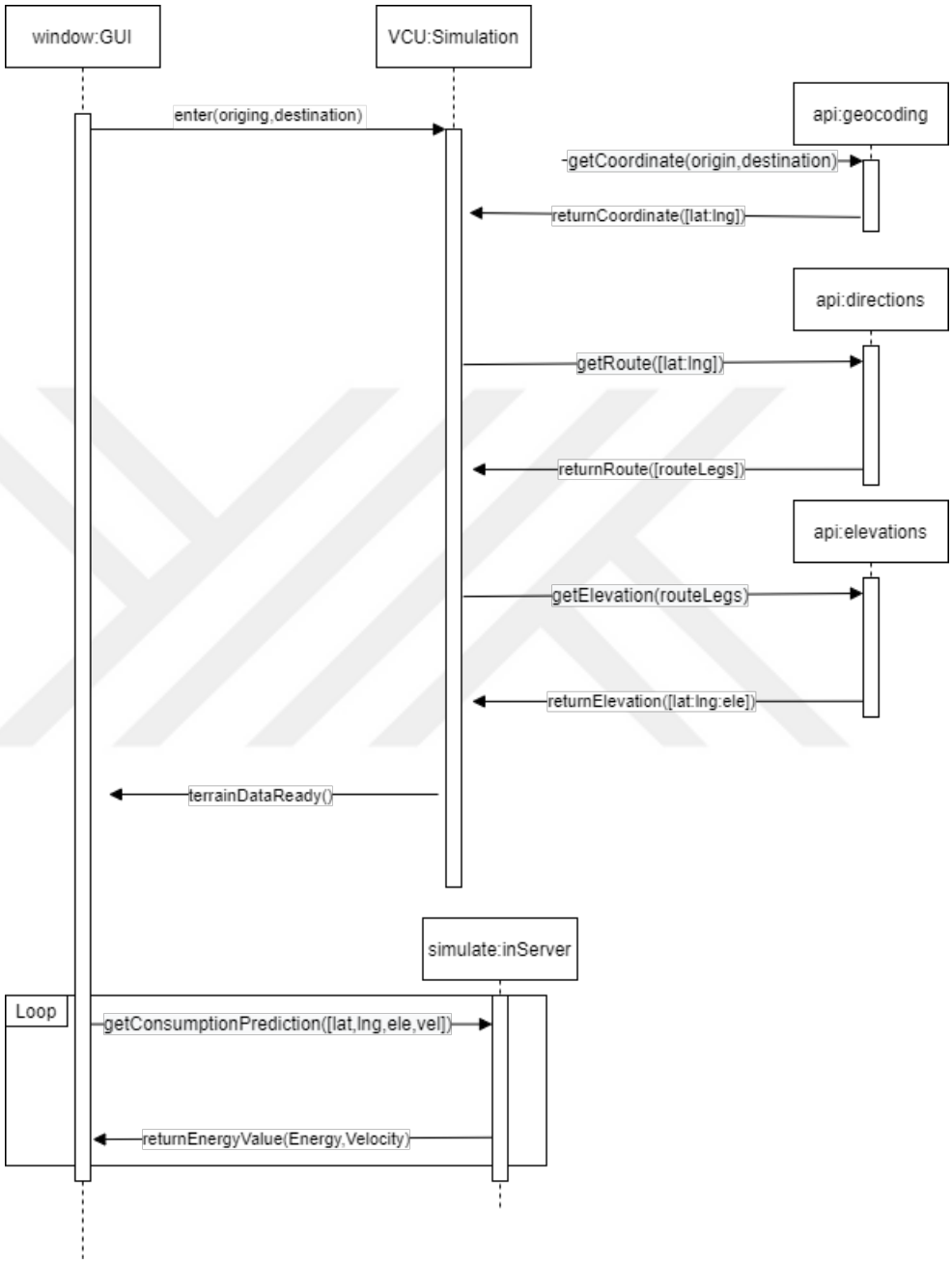

Figure 24: Sequential flow diagram of simulation process

Chapter IV

EVALUATION

4.1 Energy Consumption Prediction

Energy consumption simulation is done in the local network area which has standard protection protocols and open ports for connection. The simulator server and vehicles' virtual machine specification is given in Table 2.

Table 2: Specification of simulator server PC and vehicles' virtual machines

Specification	Simulator Server	Virtual Machines
RAM	8 GHz	2 GB
CPU	3.2 GHz	3.2 GHz
Cache	6 MB	6 MB

Simulation results prove that the server can respond to the vehicle control unit while the vehicle goes from road segment x_i to road segment x_{i+1}. Time comparison of real-time vehicle time spends during road segment and simulation time that takes the simulator server to calculate energy consumption shown in Fig.25. The vehicle starts its trip from 0 kph and eds trip with stopping to 0 kph. That is the reason why vehicle takes so much time in first and the last segments of the road.

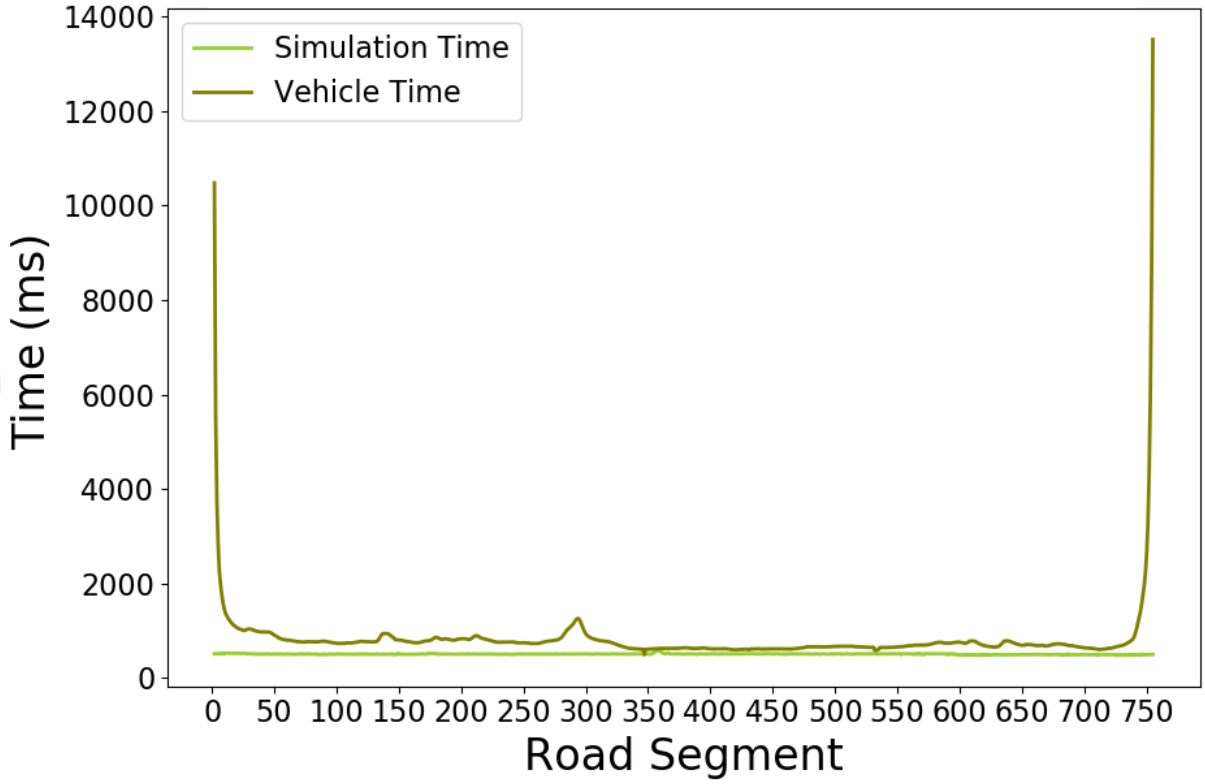

Figure 25: Simulation and vehicle time comparison

Communication loss is also compensated successfully as you can see from Fig.26. At the beginning of the trip, vehicle speed drops down to zero on a couple of data packets. It can be a network problem or vehicle that is unable to send data. But as you can see simulation continues with the last valid speed data as if the vehicle is moving at a constant speed.

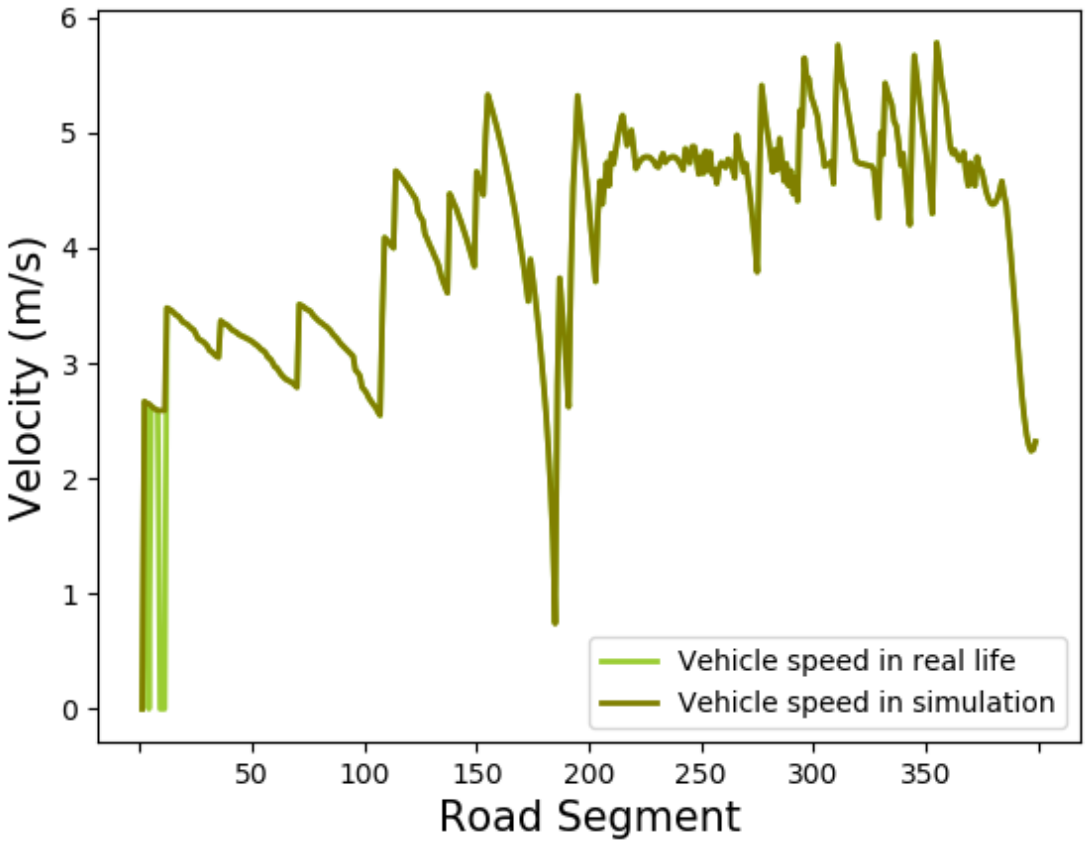

Figure 26: Simulation and vehicle speed comparison

Simulation sends energy consumption prediction at every segment. Energy consumption per segment is shown in Fig.27.

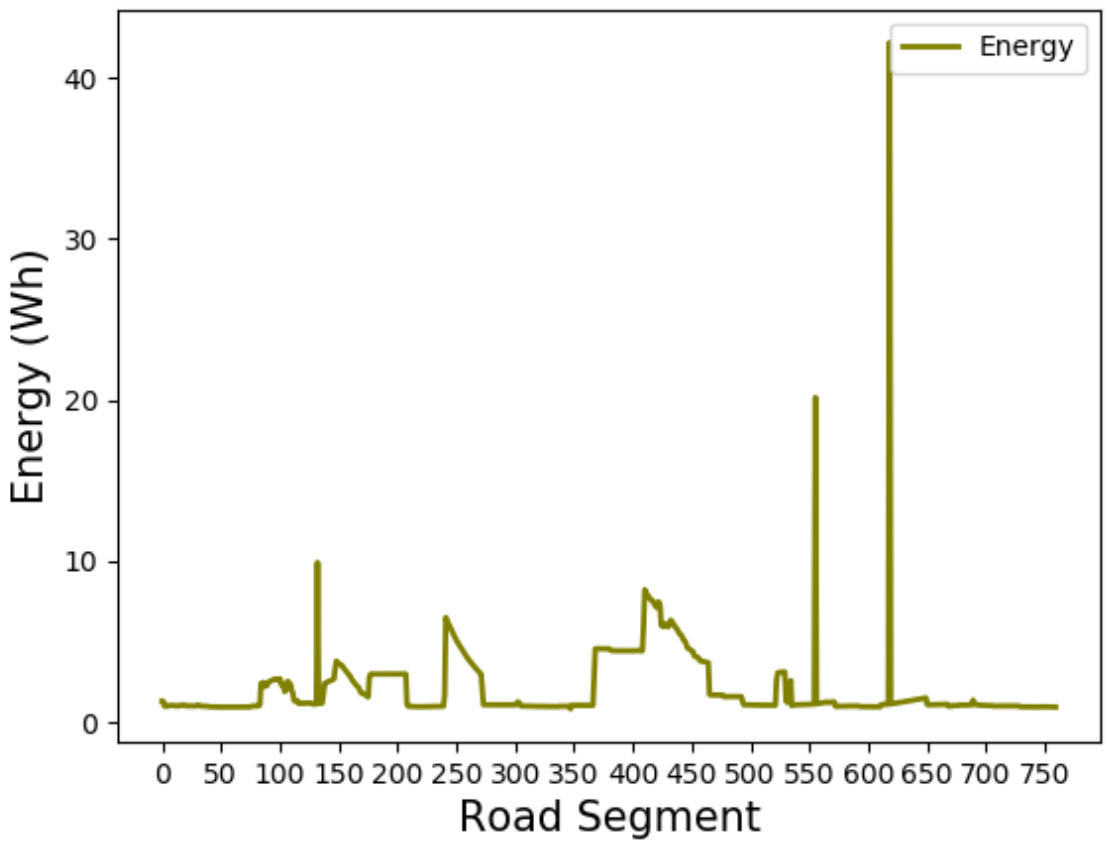

Figure 27: Calculated energy per segment

Cumulative Energy consumption also is shown in Fig. 28.

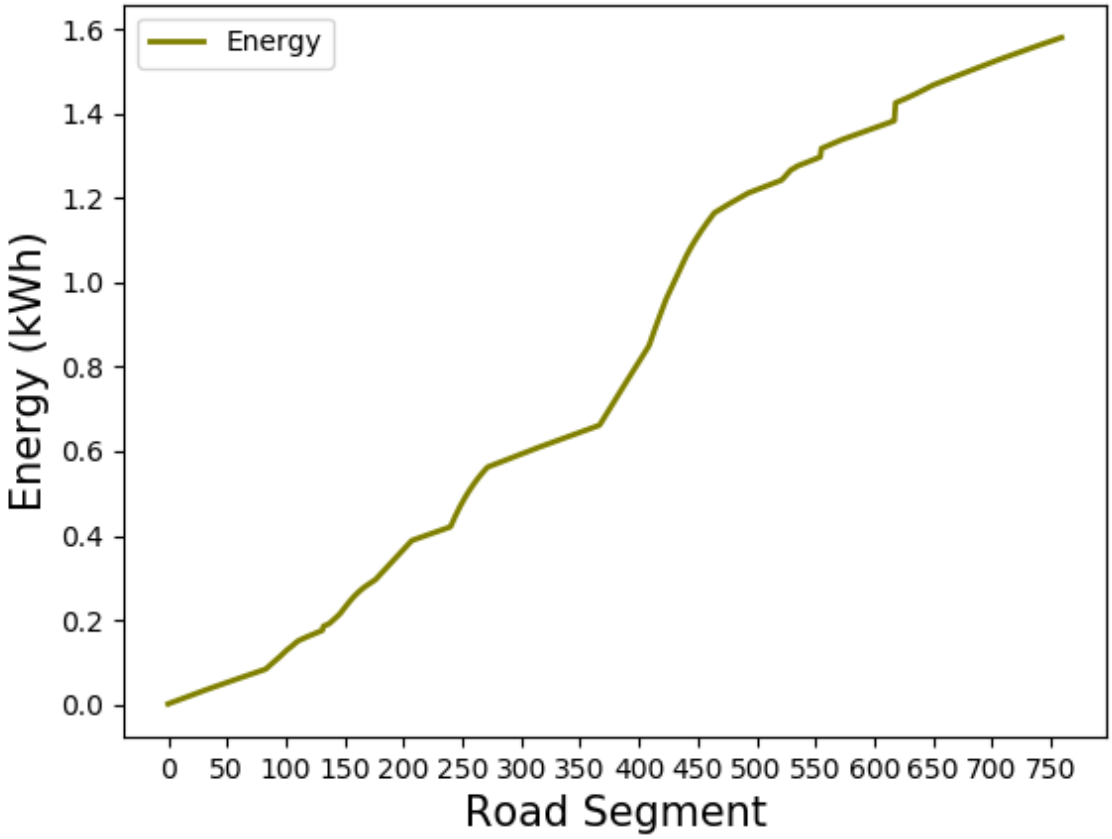

Figure 28: Cumulative energy consumption on trip

Results show that the created simulation environment can be used as a vehicle driving from the cloud tool.

4.2 Multiple Vehicle Simulation with Optimization

Multiple vehicle simulation services also an important experiment for this project because it shows the potential of the simulator to be used in real-life applications. During these simulations, Monte Carlo optimization method is used. Monte Carlo step number is 10. In the Monte Carlo optimization method, the server runs the vehicle model in the road segment 10 times and decides optimum speed.

These simulations take place on the same road but they are starting their trip at different times. It shows the effect of multiple thread creation to simulation time. In this case, it simple multiples simulation time to run vehicle by two. These specific simulations were calculating optimal speed with the Monte Carlo optimization method. Running vehicle model 10 times were decided by single-vehicle simulations. Vehicle speed was making simulation run 10 times, for example, if the vehicle model runs 50 times, simulations could not catch a vehicle that goes on the road with 30 kph. 30 kph seems slow but it should not be forgotten that the sampling intervals in these simulations were 5 meters which roughly equals the length of the vehicle. Choosing sampling intervals closer to the vehicle length will increase the accuracy of energy consumption prediction.

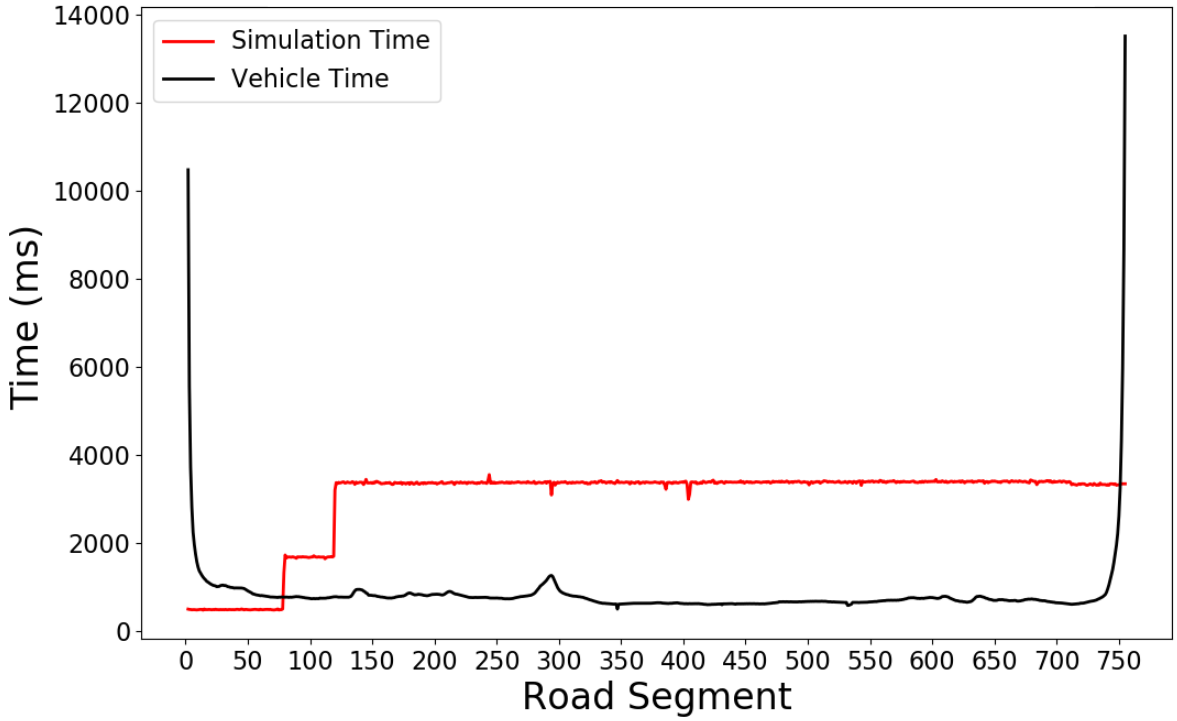

Figure 29: Simulation of vehicle 1 with optimization

Fig 29. shows the time comparison of vehicle and simulation of Vehicle 1. It is the first vehicle that connects to the server and the first 60 road segments were simulation around 500 milliseconds. It can be observed that if other vehicles did not connect to the simulator server vehicle 1 simulation time would be sufficient enough to respond VCU request in real-time while moving on the road.

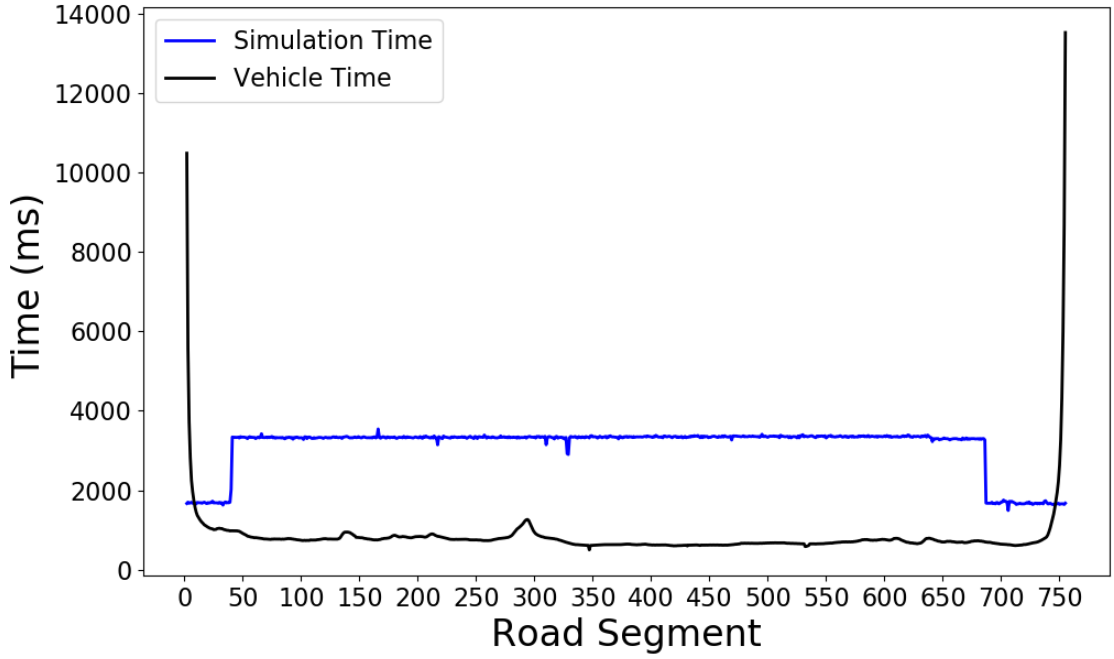

Figure 30: Simulation of vehicle 2 with optimization

Fig 30. shows the details of the simulation of vehicle 2 in-terms of time that it takes. It can be seen that vehicle 2 simulation time starts around 1600 milliseconds and then increases to 3300 after Vehicle 3 connects to the simulator server. Also, little jumps can be seen trough out the simulation. It can be because of other processes running in the simulator. There are also 100 milliseconds lower simulation time to the end of the simulation before the vehicle 1 simulation ends. It is because the screen of the simulator turned off after 10 minutes of inactivity. Long story short every other

process on server simulator effects the simulation time.

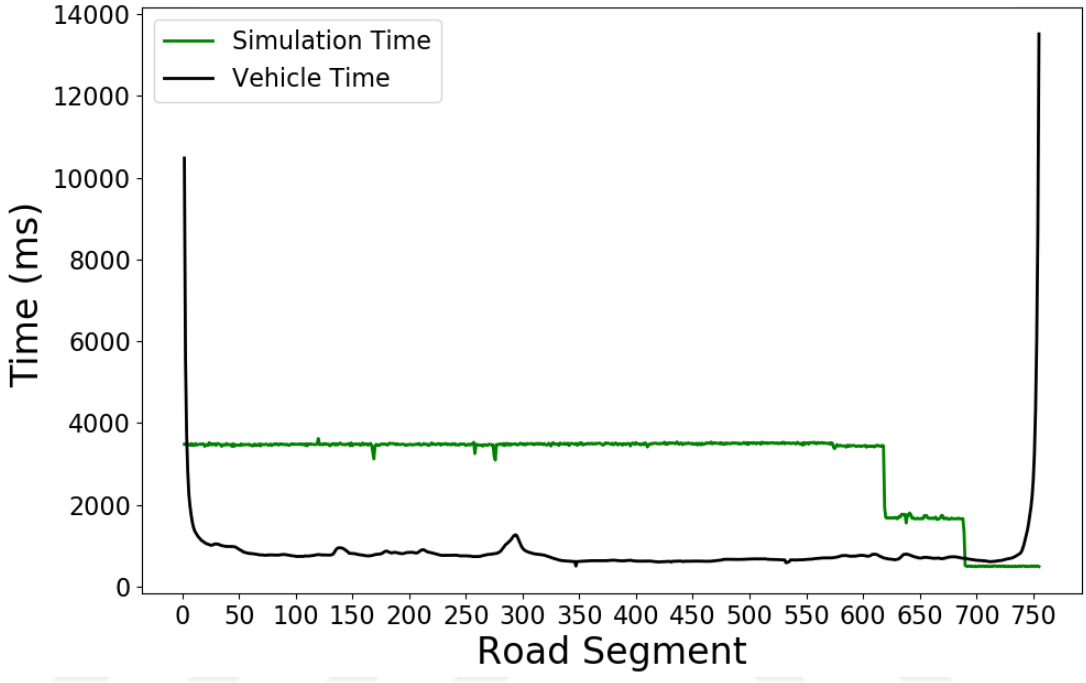

Figure 31: Simulation of vehicle 3 with optimization

Fig. 31 shows simulation of vehicle 3. Vehicle 3 is the last vehicle that connects to a simulator. Simulation time can be observed in Fig 26. As can be seen in figure simulation time starts around 3200 milliseconds and it is because there are already two-vehicle simulations that are running. After vehicle 1 simulation ends the simulation time drops to 1600 milliseconds and after vehicle 2 simulation ends the time that 1 iteration of simulation drops to 600 milliseconds.

All these three vehicle connections show that the resources of the simulator server are not enough to run simulations in real-time. The processor power is a key factor that affects the simulation time. With increased processor clock speed, simulation times can be dropped and possible to respond VCU in real-time with increased simulation repetitions. Another way to decrease simulation time is by decreasing the

number of Monte Carlo simulations. In the current simulation case, the server runs every road segment simulation 10 times. It can be decreased which will help to decrease the response time of the simulator. But increasing the number of Monte Carlo simulations will also decrease the chance to find the optimum speed that will cause the lowest energy consumption. The configuration of these must be made according to the computational power of the server and the complexity of the vehicle model.

4.3 Multiple Vehicle Energy Consumption Prediction

In this section results of multiple vehicle simulations without optimization are presented. And it can be seen from the results that the simulator is responding to all three VCUs in real-time. This proves the fact that the presented simulation environment is suitable for real-life applications.

Below Fig. 32 shows that even after the connection of 2 vehicles, the simulator still responds to Vehicle 1 in real-time. In the beginning simulation time is around 50 milliseconds. After Vehicle 2 connects to the server it increases to 100 milliseconds and connection from vehicle 3 causes it to increase 150 milliseconds. In the meantime vehicle goes from segment x_i to x_{i+1} in 500 milliseconds. As a result of this experiment, it can be said that the simulator is a good tool to calculate energy consumption prediction in real-time. Real-time energy consumption prediction tools can be also used in V2X applications. Fig. 33 shows the simulation of vehicle 2 and Fig. 34 shows the simulation of vehicle 3.

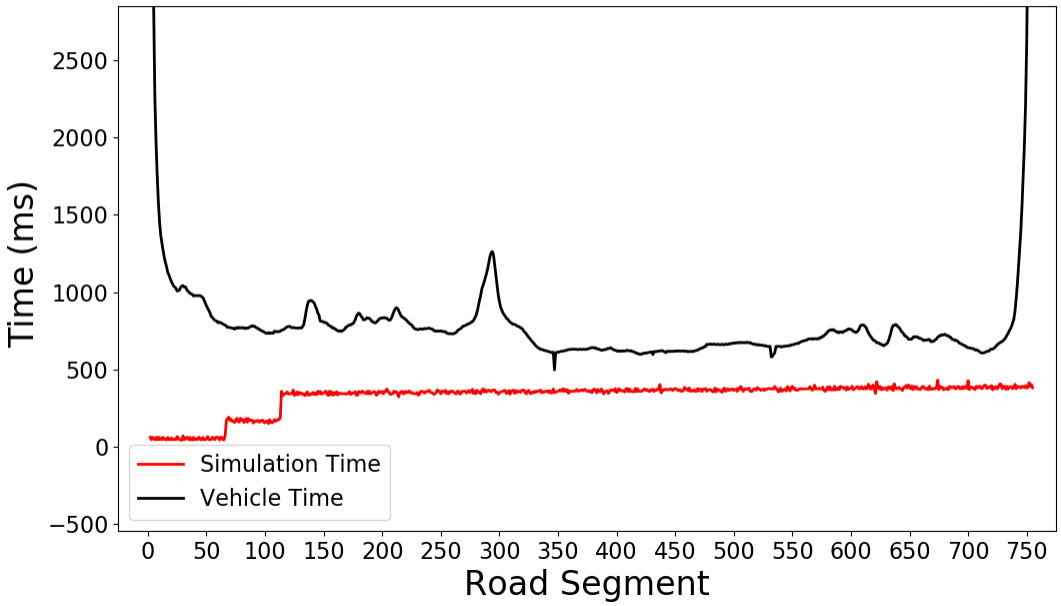

Figure 32: Simulation of vehicle 1 without optimization

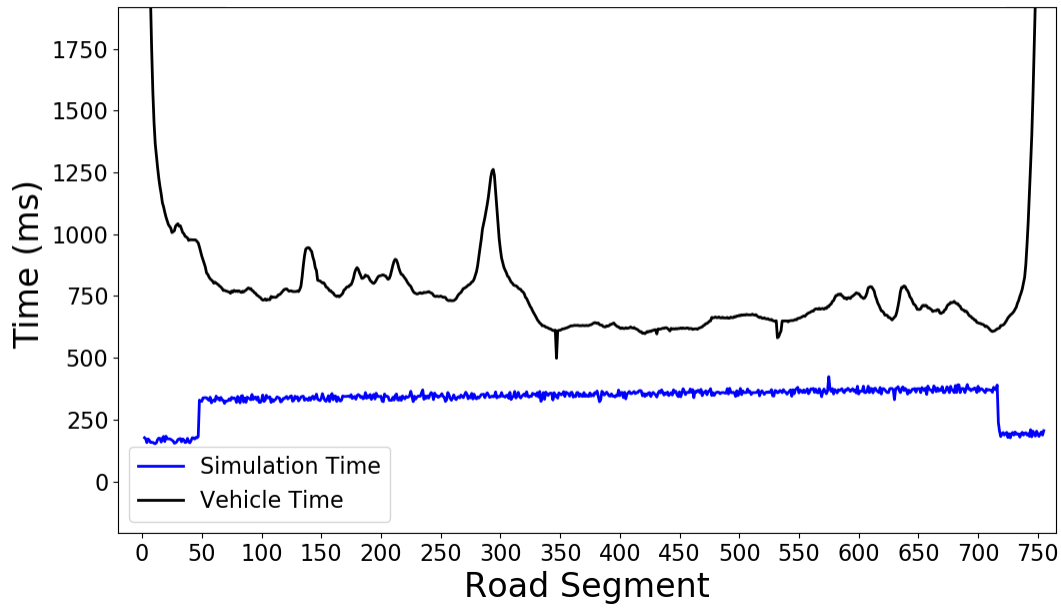

Figure 33: Simulation of vehicle 2 without optimization

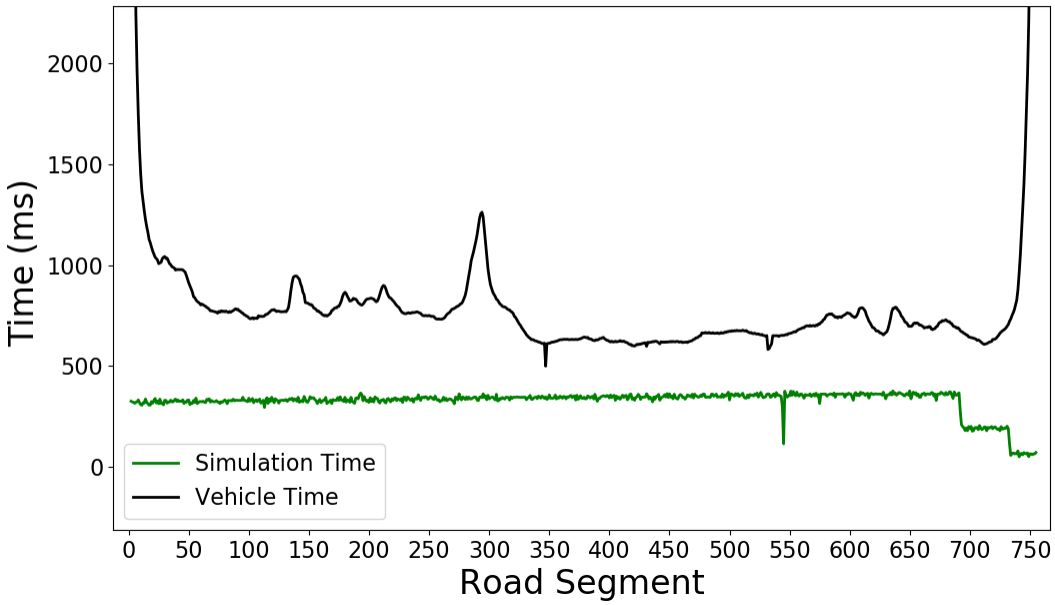

Figure 34: Simulation of vehicle 3 without optimization

Again these results prove that the simulation environment presented in this thesis is suitable for the applications that require prediction of energy consumption. Also, it can be said that servicing multiple clients are way easier without optimization. Because optimization algorithms usually use an iteration of the whole situation over and over again which will require too much computation power. If you don't have enough computation power it will not be possible to respond VCU request in real-time.

4.4 Control Algorithm Comparison

Another aspect of this dissertation is that it is possible to develop control algorithms and a comparison of them can be done in the presented simulation environment. Monte Carlo and Dynamic Programming control algorithms are compared here. The results presented below. Fig. 35 shows the response time of the Monte Carlo algorithm. The average response time of the Monte Carlo algorithm is around 125

milliseconds.

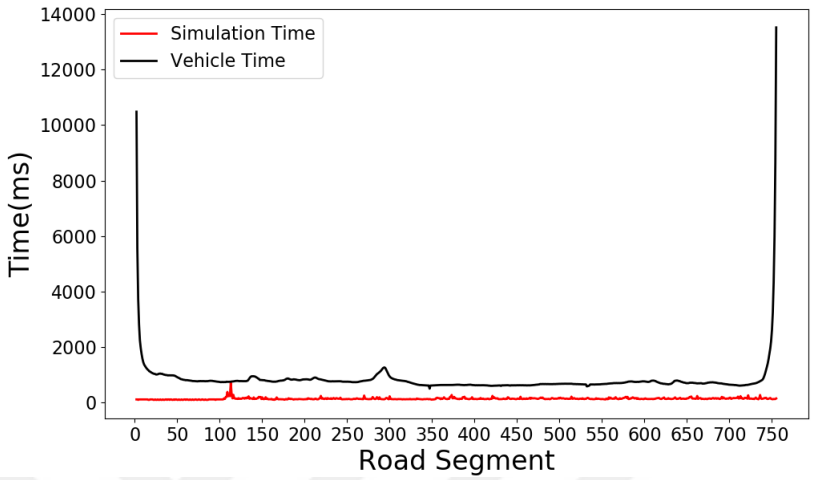

Figure 35: Monte Carlo response time

Fig. 36 shows the response time of the Dynamic Programming algorithm. The average response time of dynamic programming is 105 milliseconds.

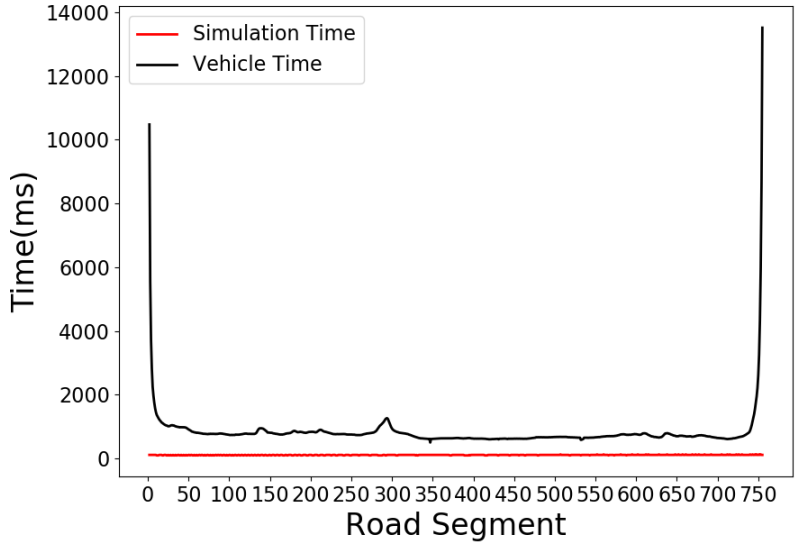

Figure 36: Dynamic programming response time

Figures show that the presented simulation environment is suitable for this use case. A comparison of control algorithms can be done with this setup.

4.5 Hardware-In-The-Loop Replacement

Another crucial contribution of this thesis is that it is possible to use this simulation environment as HIL replacement. Below is the Fig 37. that represents communication between HIL and client computers. This scenario is that sensor data is given by client and sinus and cosine values are requested from HIL. In this experiment, the maximum delay time is 50 milliseconds but the overall result is very satisfying. With proper hardware, this simulation environment can be used as a replacement of HIL.

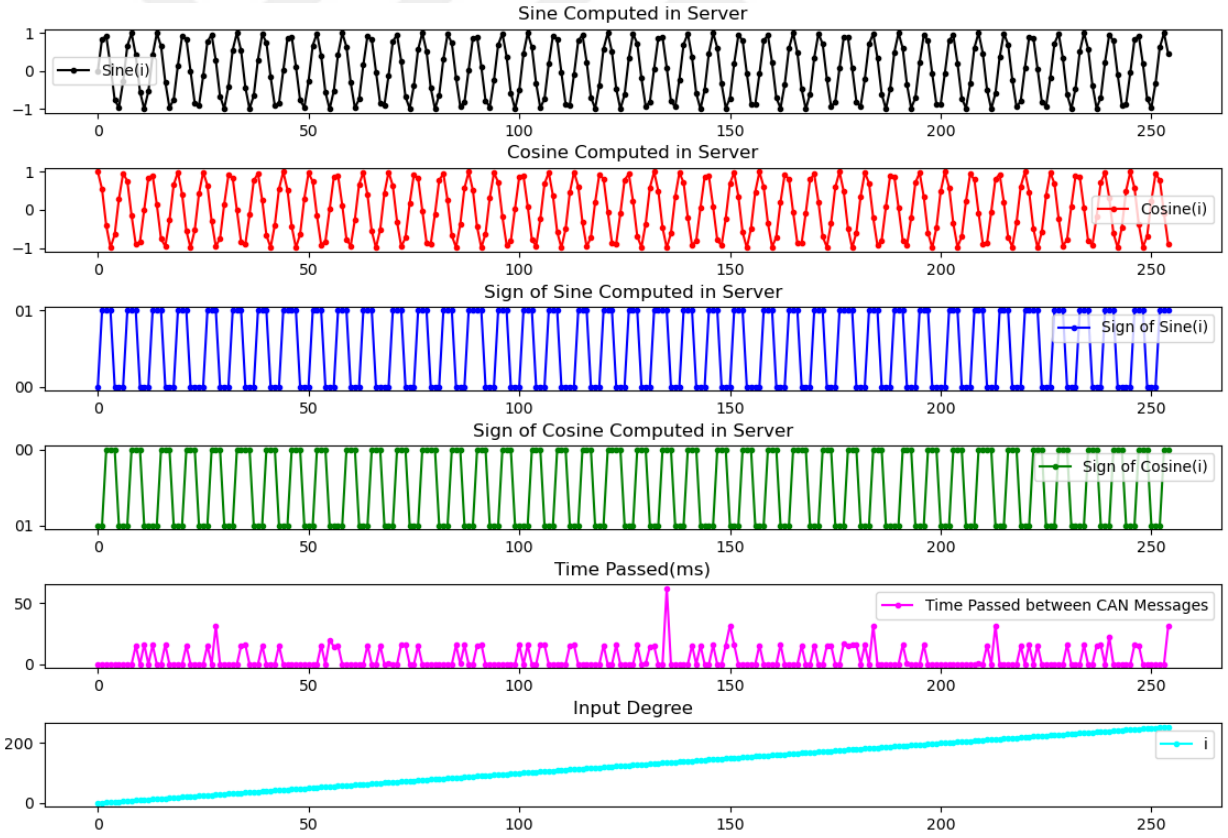

Figure 37: HIL replacement experiment

Chapter V

CONCLUSIONS AND FUTURE WORK

With the rising interest in V2X communication technologies, it has been proven that energy foreseeing simulation is a useful technic for energy management technologies. In this thesis, a simulation environment has been presented for energy calculation with communication for a virtual vehicle, and results are presented. Results show that the system can respond to VCU requests in real-time which makes this setup a good tool for V2X communication technologies such as coordinated driving.

A comparison of control algorithms is a project worth to be studied. With an on-line real-time simulation environment, any control algorithm can be punished/rewarded with its computation time. A control algorithm can be very good in terms of energy efficiency, but it can be also too slow for real-time applications. On the other hand, another control algorithm can be not so good in terms of energy efficiency, but it can be good at computation time, which can make it more suitable for real-time vehicular applications. Two control algorithms are compared in this dissertation and results are presented. This indicates that the presented simulation environment can be used in the comparison of control algorithms.

Being a HIL alternative was another motivation for our approach. HIL setups are costly and come with their challenges (wiring, big computer case, etc.). We carried out the tests done on a HIL in a cloud-based simulation environment which we called Cloud-In-the-Loop. CIL helps us replace HIL. And results are promising compared to response time and accuracy shows that the presented system can be used as HIL replacement. The trustworthiness of the presented system is promising that it can be used in automotive replacements.

For future work, the proposed system can be implemented on a powerful PC to have a better real-time response time. For the prediction of energy consumption, traffic data integration seems a feature that can be added to improve accuracy. Since acquiring live data was not possible trough Google Maps Web services, it should be acquired from other reliable service providers. Another possible update can be done on the GUI. Real-time moving vehicle representation would make the GUI more appealing.

Bibliography

[1] Kia Motors Corporation, "Auto School 1: How Does My Engine Work?." [Online]. Available: `https://www.friendlykia.com/blog/how-does-my-engine-work/`. Accessed: Jul. 14, 2020.

[2] Wikimedia Foundation, "Hybrid Electric Vehicle." [Online]. Available: `https://en.wikipedia.org/wiki/Hybrid_electric_vehicle`. Accessed: Jul. 12, 2020.

[3] M. Şahin, M. A. Gözüküçük, and H. F. Uğurdağ, "Online Real-Time Simulation of Vehicles," in *Proceedings of International Conference on Computer Science and Engineering (UBMK)*, Sep. 2020. to be published.

[4] F. Franco, G. D. Stella, J. H. Neme, M. M. D. Santos, S. L. Stevan, and J. N. H. da Rosa, "Teaching Model-In-the-Loop: A Case Study for Controller of Distributed Dashboard in a Road Vehicle," in *Proceedings of International Symposium on Industrial Electronics (ISIE)*, pp. 863–868, Jun. 2016.

[5] M. Çakmakci, Y. Li, and S. Liu, "Model-In-the-Loop Development for Fuel Cell Vehicle," in *Proceedings of American Control Conference (ACC)*, pp. 2462–2467, Jun. 2011.

[6] M. A. Taut, G. Chindris, and A. C. Taut, "Software-In-the-Loop System for Motor Control Algorithms," in *Proceedings of International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pp. 419–426, Oct. 2019.

[7] M. Muresan and D. Pitica, "Software-In-the-Loop Environment Reliability for Testing Embedded Code," in *Proceedings of International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pp. 325–328, Oct. 2012.

[8] M. Hu, G. Zeng, H. Yao, and Y. Tang, "Processor-in-The-Loop Demonstration of Coordination Control Algorithms for Distributed Spacecraft," in *Proceedings of International Conference on Information and Automation (ICIA)*, pp. 1008–1011, Jun. 2010.

[9] I. Kendall and R. Jones, "An Investigation Into the Use of Hardware-In-the-Loop Simulation Testing for Automotive Electronic Control Systems," *Control Engineering Practice*, vol. 7, pp. 1343–1356, 1999.

[10] P. J. King and D. G. Copp, "Hardware-In-the-Loop for Automotive Vehicle Control Systems Development," in *Proceedings of UKACC Control Mini Symposia*, pp. 75–78, Sep. 2004.

[11] M. Kabutomori, T. Murai, S. Ota, and Y. Terumichi, "Development of a Test Stand for Maglev Vehicles Using Hardware-In-the-Loop Simulation," in *Proceedings of International Symposium on Linear Drives for Industry Applications (LDIA)*, pp. 1–5, Sep. 2017.

[12] W. Hongyu, Y. Zhiqiang, F. Yong, Q. Yunqian, and L. Zhiming, "Development of Pure Electric Vehicle Powertrain Controller Based on Hardware-In-the-Loop Platform," in *Proceedings of IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 498–502, Sep. 2015.

[13] M. A. Gozukucuk, T. Akdogan, W. Hussain, T. K. Tasooji, M. Sahin, M. Celik, and H. F. Ugurdag, "Design and Simulation of an Optimal Energy Management Strategy for Plug-In Electric Vehicles," in *Proceedings of International Conference on Control Engineering Information Technology (CEIT)*, pp. 1–6, Sep. 2018.

[14] M. R. Mufid, A. Basofi, I. Syarif, F. Sanjaya, and Wajib, "Estimated Vehicle Fuel Calculation Based on Google Map Realtime Distance," in *Proceedings of International Electronics Symposium (IES)*, pp. 354–358, Sep. 2019.

[15] S. Gao, J. Niu, and D. Sun, "A Tourist Tour Node Model and Tour Support," in *Proceedings of International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 1168–1173, Dec. 2016.

[16] A. Pejic, S. Pletl, and B. Pejic, "An Expert System for Tourists Using Google Maps API," in *Proceedings of International Symposium on Intelligent Systems and Informatics (SISY)*, pp. 317–322, Sep. 2009.

[17] M. Konarski and W. Zabierowski, "Using Google Maps API Along With Technology .NET," in *Proceedings of International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET)*, pp. 180–182, Feb. 2010.

[18] N. Matloff, "Tutorial on Network Programming With Python." [Online]. Available: `http://index-of.co.uk/Python/PyNet.pdf` Accessed: May 14, 2020.

[19] G. Regnier, S. Makineni, I. Illikkal, R. Iyer, D. Minturn, R. Huggahalli, D. Newell, L. Cline, and A. Foong, "TCP Onloading for Data Center Servers," *Computer*, vol. 37, pp. 48–58, 2004.

[20] L. Kalita, "Socket Programming," *International Journal of Computer Science and Information Technologies (IJCSIT)*, vol. 5, pp. 4802–4807, 2014.

[21] M. Xue and C. Zhu, "The Socket Programming and Software Design for Communication Based on Client/Server," in *Proceedings of Pacific-Asia Conference on Circuits, Communications and Systems (PACCS)*, pp. 775–777, May 2009.

[22] M. Gerla, E. Lee, G. Pau, and U. Lee, "Internet of Vehicles: From Intelligent Grid to Autonomous Cars and Vehicular Clouds," in *Proceedings of IEEE World Forum on Internet of Things (WF-IoT)*, pp. 241–246, Mar. 2014.

[23] D. Xiaoping, L. Dongxin, L. Shen, W. Qiqige, and C. Wenbo, "Coordinated Control Algorithm at Non-Recurrent Freeway Bottlenecks for Intelligent and Connected Vehicles," *IEEE Access*, vol. 8, pp. 51621–51633, 2020.

[24] K. Sasaki, S. Makido, and A. Nakao, "Vehicle Control System for Cooperative Driving Coordinated Multi-Layered Edge Servers," in *Proceedings of IEEE International Conference on Cloud Networking (CloudNet)*, pp. 1–7, Oct. 2018.

[25] K. Zheng, Q. Zheng, H. Yang, L. Zhao, L. Hou, and P. Chatzimisios, "Reliable and Efficient Autonomous Driving: The Need for Heterogeneous Vehicular Networks," *IEEE Communications Magazine*, vol. 53, pp. 72–79, 2015.

[26] G. Svennerberg, *Beginning Google Maps API 3*. New York, NY, USA: Apress, 2 ed., 2010.

[27] D. Lemke, V. Mattauch, O. Heidinger, and H. Hense, "Who Hits the Mark? a Comparative Study of the Free Geocoding Services of Google and OpenStreetMap," *Das Gesundheitswesen*, vol. 77, pp. 160–165, 2015.

[28] M. Socharoentum and H. A. Karimi, "A Comparative Analysis of Routes Generated by Web Mapping APIs," *Cartography and Geographic Information Science*, vol. 42, pp. 33–43, 2015.

[29] Rui Li, Chun Cheng, Mingyao Qi, and Weiwei Lai, "Design of Dynamic Vehicle Routing System Based on Online Map Service," in *Proceedings of International Conference on Service Systems and Service Management (ICSSSM)*, pp. 1–5, Jun. 2016.

[30] S. H. Han, J. D. Lee, and H. B. Ahn, "Geospatial Data Acquisition Using the Google Map API," *International Journal of Contents*, vol. 8, pp. 55–60, 2012.

[31] J. A. Cook and B. K. Powell, "Modeling of an Internal Combustion Engine for Control Analysis," *IEEE Control Systems Magazine*, vol. 8, pp. 20–26, 1988.

[32] R. W. Weeks and J. J. Moskwa, "Automotive Engine Modeling for Real-Time Control Using Matlab/Simulink," *SAE Technical Paper 950417*, vol. 104, pp. 295–309, 1995.

[33] P. D. Walker, S. A. Rahman, N. Zhang, W. Zhan, Y. Lin, and B. Zhu, "Modelling and Simulation of a Two Speed Electric Vehicle," in *Proceedings of International Conference on Sustainable Automotive Technologies*, pp. 193–198, Jan. 2012.

[34] S. A. Evangelou and A. Shukla, "Advances in the Modelling and Control of Series Hybrid Electric Vehicles," in *Proceedings of American Control Conference (ACC)*, pp. 527–534, Jun. 2012.

[35] W. Enang and C. Bannister, "Modelling and Control of Hybrid Electric Vehicles (A Comprehensive Review)," *Renewable and Sustainable Energy Reviews*, vol. 74, pp. 1210–1239, 2017.

[36] J. Liu and H. Peng, "Modeling and Control of a Power-Split Hybrid Vehicle," *IEEE Transactions on Control Systems Technology*, vol. 16, pp. 1242–1251, 2008.

[37] M. O. Faruque Sarker and Sam Washington, *Learning Python Network Programming*. Packt, 1 ed., Jun. 2015.

[38] Google LLC, "Google Maps Platform Documentation." [Online]. Available: `https://developers.google.com/maps/documentation`. Accessed: Sep. 25, 2017.

[39] T. Gillespie, *Fundamentals of Vehicle Dynamics*. Warrendale, PA, USA: SAE International, Feb. 1992.

[40] W. Hucho and G. Sovran, "Aerodynamics of Road Vehicles," *Annual Review of Fluid Mechanics*, vol. 25, pp. 485–537, 1993.

[41] L. W. DeRaad, "The Influence of Road Surface Texture on Tire Rolling Resistance," in *SAE Technical Paper 780257*, Feb. 1978.

[42] K. A. Grosch, "The Rolling Resistance, Wear and Traction Properties of Tread Compounds," *Rubber Chemistry and Technology*, vol. 69, pp. 495–568, 1996.

[43] JS Foundation, "Jquery API Documentation." [Online]. Available: `https://api.jquery.com/jquery.ajax/`. Accessed: Jul. 13, 2020.

[44] PHP Documentation Group, "Jquery API Documentation." [Online]. Available: `https://www.php.net/manual/`. Accessed: Jul. 13, 2020.

VITA

Name: Mert Şahin

Date of Birth: 16 July 1993

Languages: Turkish, English

EDUCATION

– MS, Özyeğin University, Electrical and Electronics Engineering, 2016 - 2020

– BS, Özyeğin University, Electrical and Electronics Engineering, 2011 - 2016

– High School, Antalya Atatürk Anatolian High School, 2007 - 2011

WORK EXPERIENCE

FEV Turkey Research and Engineering | Istanbul/Turkey

Automotive Software Engineer, Dec. 2018 - Ongoing

As a member of the software development team my responsibilities are;

– ISO 26262 level 2 safety software implementation, documentation, unit test case definition, and documentation using MATLAB/Simulink.

– Basic software requirement management on safety-related signals.

– Link-Layer Software development, testing, and documentation

– Python tool development and testing.

– Development of the signal database in the CAN and J1939 standards.

– Failure mode and effects analysis (FMEA) of safety related components and software functions.

– Working together with other branches of FEV on a variety of projects.

Hexagon Studio Engineering | Istanbul/Turkey

Electronic Design and Software Engineering, Jun. 2017 - Dec. 2018

As a member of Electronic Integration and Software Team I was responsible for the below issues;

– Development of application software using MATLAB/Simulink for eVCU.

– Investigating and solving the problems that occurred during the testing of components and vehicles at Bosch Engineering in Germany and Hexagon Studio in Turkey.

– Giving engineering sign-off of prototype vehicles' software and electrical components. Collaborating with different suppliers for outsourcing works and components (e.g Bosch Engineering, TM4, Pi Innovo).

Özyeğin University | Istanbul/Turkey

Graduate Research Assistant, Sep. 2016 - Jun. 2017

– Worked as a Research Assistant for the project "Energy Management Based on Topographic Forecasting for Electric Vehicles". This project funded by TÜBİTAK (Scientific and Technological Research Council of Turkey) under projects 115E097 and 115E127.

Özyeğin University | Istanbul/Turkey

Graduate Teaching Assistant, Sep. 2016 - Jun. 2017

– Worked as a graduate TA for the courses in Mathematics for Engineering II and Mechatronics as a Head TA.

Özyeğin University | Istanbul/Turkey

Undergraduate Teaching Assistant, Mar. 2014 - Jun. 2016

– Worked as TA for the courses digital design, microprocessors, and computer architecture.

PUBLICATIONS

- M. A. Gözüküçük, T. Akdoğan, W. Hussain, T. K. Tasooji, M. Şahin, M. Çelik and H. F. Uğurdağ, "Design and Simulation of an Optimal Energy Management Strategy for Plug-In Electric Vehicles," in *Proceedings of International Conference on Control Engineering Information Technology (CEIT)*, pp. 1–6, Sep. 2018.

- M. Şahin, M. A. Gözüküçük and H. F. Uğurdağ, "Online Real-Time Simulation of Vehicles", in *Proceedings of International Conference on Computer Science and Engineering (UBMK)*, Sep. 2020. (Accepted, Not yet Published)