Cloud your bus: real-time energy consumption prediction for electric city buses

PDEng THESIS REPORT

Cloud Your Bus: Real-time Energy Consumption Prediction for Electric City Buses

Dhruv Jagga
October/2020
Department of Mathematics & Computer Science

PDEng AUTOMOTIVE SYSTEMS DESIGN
Track AUTOMOTIVE SYSTEMS DESIGN

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Cloud Your Bus: Real-time Energy Consumption Prediction for Electric City Buses

Dhruv Jagga

October 2020

Eindhoven University of Technology
Stan Ackermans Institute - Automotive/Mechatronic Systems Design

PDEng Report: 2020/045

Confidentiality Status:
Open Access

Partners

Sycada (Cloud Your Bus Consortium) Eindhoven University of Technology

Steering Group

Project Owner: Dhruv Jagga
Project Manager: Peter Heuberger
Project Mentors: Rogier Mulder, Kristian Winge
Project Supervisor: Igo Besselink

Date October 2020

Composition of the Thesis Evaluation Committee:

Chair:	Dr. ir. Igo Besselink
Members:	Prof. dr. Henk Nijmeijer
	Dr. Peter Heuberger
	Dr. Ion Barosan
	Ing. Rogier Mulder
	Kristian Winge, MSc/MBA
	Ir. Riske Meijer

The design that is described in this report has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct.

Date October, 2020

Contact address Eindhoven University of Technology
Department of Mathematics and Computer Science
Automotive Systems Design
MF 5.073
P.O. Box 513
NL-5600 MB
Eindhoven, The Netherlands
+31 402743908

Published by Eindhoven University of Technology

PDEng Report 2020/045

Abstract

In the present days, electric city buses have captured a major market share as a result of ambitions set by modern cities to reduce the local carbon emissions. This leads to more research and development in electric vehicles, especially in modeling to approximate the actual behavior of these vehicles. Current energy consumption prediction methodologies for electric vehicles suffer several practical limitations and challenges for dealing with uncertainties. This research explores an energy consumption prediction approach for electric city buses based on measurement data for a given bus route. An online correction algorithm is designed based on a recursive algorithm and Kalman filter to improve real-time energy consumption prediction capabilities. Results show that the offline model can give a rough prediction before the trip, and the online model can improve the estimate to a promising accuracy.

Keywords Real-Time Energy Consumption Prediction, Electric City Buses, Electric Vehicle, Energy Consumption Model, Estimation Algorithm, Real-World Data

Preferred reference Cloud Your Bus: Real-Time Energy Consumption Prediction for Electric City Buses, *A transition to zero-emission public transport in Europe*. Eindhoven University of Technology, PDEng Report 2020/045, October 2020.

Partnership This project was supported by Eindhoven University of Technology and Cloud Your Bus Consortium; an EM Europe Research and Innovation Project

Disclaimer Endorsement

Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the Eindhoven University of Technology and Company name. The views and opinions of authors expressed herein do not necessarily state or reflect those of the Eindhoven University of Technology and Cloud Your Bus Consortium, and shall not be used for advertising or product endorsement purposes.

Disclaimer Liability

While every effort will be made to ensure that the information contained within this report is accurate and up to date, Eindhoven University of Technology makes no warranty, representation or undertaking whether expressed or implied, nor does it assume any legal liability, whether direct or indirect, or responsibility for the accuracy, completeness, or usefulness of any information.

Trademarks

Product and company names mentioned herein may be trademarks and/or service marks of their respective owners. We use these names without any particular endorsement or with the intent to infringe the copyright of the respective owners.

Copyright

Foreword

Foreword (by Kristian K. Winge, CEO of Sycada)

The energy consumption of electric buses has proven to be more sensitive to driving style and external conditions, such as ambient temperature, than their fossil fuelled cousins. This sensitivity, coupled with a shorter range and the need to opportunity charge during the day, means that the operations in public transport is exposed to more volatility as well as planning uncertainty. This volatility requires bus operators to invest in additional battery capacity, or to acquire more assets, or to adopt new tools and technologies that bring more visibility and adaptability to the operation. From an economical and societal point of view, the latter would seem the better choice.

Accurately predicting the energy consumption of a bus in a given route is one of those critical tools, but it is not trivial task to accomplish. Current energy consumption prediction models suffer from several practical or computational limitations and more often than not fail to factor in environmental and contextual parameters. Hence have limited real value in a dynamic planning context. As a result, most, if not all, bus operators plan their zero-emissions operations based on limited historical data-sets for buses and routes. But these estimations are inherently inaccurate with an error margin up to 40%.

The unique online energy consumption prediction model developed in the context of the Cloud-Your-Bus (CYB) innovation programme has demonstrated to potentially bring this error margin down to an average close to 1%. When made available to bus operators, the more accurate information can facilitate better and faster decision making and help optimise route and charge planning throughout the day. This in turn has a massive positive impact on both capital and operational expenses and will help accelerate the transition to zero-emission public transport in Europe and beyond.

The energy prediction tool that has been developed as part of this project is a living example of how academia and business can work together to develop new tools and to create a positive impact on one of the biggest challenges we collectively face: creating a more sustainable world.

At Sycada, we are proud of the work that has been achieved and I want to take this opportunity to express our sincere thanks to Dhruv Jagga and his colleagues and mentors at TU/e for having brought this work from research to practical implementation.

Kind Regards,

Kristian K. Winge
CEO, Sycada

Foreword

Foreword (by Igo Besselink)

In the transition from an internal combustion engine to a battery electric powertrain, various new challenges arise. The prediction of the energy consumption becomes very important due to the limited energy density of a battery in comparison to diesel. This applies in particular to city busses, which are used intensively and have to operate as (cost) efficiently as possible. As part of the "Cloud Your Bus" project a novel energy consumption prediction method has been developed.

The fact that city busses drive fixed routes is exploited. This requires collecting, analyzing and processing historical trip data, which was done by Dhruv and Yuzhe together. The estimate for the total energy consumption for a specific trip is adapted while the bus is driving to provide near real time updates. Dhruv has designed and evaluated the parameter estimation process. Furthermore he worked closely with Dan to get a prototype of the software running on an embedded device, which will be evaluated on a real bus in the near future.

The results obtained so far show great promise and Dhruv has made important contributions to the development of this energy consumption prediction tool. I also think that it was a very useful experience for him to be involved in the process from idea and concept development to realization and implementation, with all (practical) issues that come along the way.

Kind Regards,

Igo Besselink
Associate professor, Mechanical Engineering
TU Eindhoven
October 2020

Preface

This report is one of the deliverables of the graduation project for the degree of Professional Doctorate in Engineering in Automotive Systems Design (PDEng, ASD). This report comprehends the design, development, testing, and integration of a real-time energy consumption prediction system. It aims to allow the technical reader of this report to understand the procedure in detail and give the opportunity to either replicate the results or extend the method of the electric vehicle's energy consumption prediction system.

This report describes new ideas to incorporate real-world data into physical models and make them more realistic and reliable.

This report gives a detailed explanation of the implementation of prediction algorithms for energy consumption prediction systems in real-time.

This report describes the process of system design. It explains the step-by-step transition of the system development, starting from the stakeholder expectations.

This report can serve as a reference for implementation of energy consumption prediction systems in the commercial ecosystem.

Ir. Dhruv Jagga

October 2020

Acknowledgements

The culmination of this PDEng project would not have been possible without some external support. First, I would like to thank all the program community people for maintaining a stimulating environment, which has helped shape the trainee's overall personality during this two-year tenure at the Eindhoven University of Technology.

I want to express my deepest gratitude to my supervisor, dr.ir. Igo Besselink for providing me the opportunity to work on such a promising research topic. I would also like to thank him for his continuous support and guidance throughout the project. I would like to extend my gratitude to my project mentors from Sycada B.V., Kristian Winge, and Rogier Mulder for having fruitful discussions and feedback sessions. I am also indebted to the discussions carried out in the update meetings with Prof. dr. Henk Nijmeijer that helped me to think more about the subject leading to improving my work.

I also would like to thank my colleagues Camiel Beckers and Yuzhe Ma. They helped me get started with a relatively new domain and assisted me in expanding my knowledge base in energy modeling of electric vehicles by sharing useful resources and conducive discussions on the topic. A big thanks also go to Dan Chirascu. Later in the project, he translated the prototype software developed by me to an executable version that needed to be run on the device, which helped me validate my system and conclude my project.

In addition, I am honored to have my program manager as dr. Peter Heuberger, whose honest guidance and support were always available during the past two years, and helped me grow in this fellowship program. I would also like to thank Ellen van Hoof-Rompen for her continued support during the program. Furthermore, I cherished each and every discussion made with my colleagues from the PDEng ASD/MSD program, which helped me to improve my work - especially Siddhesh Rane, Arash Arjmandi, Mohamed Kamel, and Navaneeth Bhat.

Finally, I would like to express my deepest gratitude and love towards my family and my mentor Dr. Daisaku Ikeda, to provide unconditional support, love, and faith in me.

Special thanks to my dear friends Ajar, Hemant, Sabyasachi, Swaraj, Kirti, Ketan, and Ishant for always being there.

Eindhoven University of Technology
September 22, 2020

Executive Summary

Zero-emission transportation is blooming swiftly with the earliest movers to be public transport in this field. Electric buses are replacing conventional ICE engine buses in the city areas in entire Europe to deal with the issues concerning greenhouse-gas emissions. The bus operators are amongst the top pioneers to adapt to the leading-edge technologies to make this transition possible.

The transformation of public transportation from classical ICE buses to electric buses brings about startling challenges in operational ambiguities, vulnerabilities, and costs. The ambition of the EU funded **Cloud Your Bus** project (an EM Europe Research and Innovation Project) is to stimulate the transition to zero-emission public transport in Europe. The business goal revolves around reducing the risks and costs of this transition to zero-emission bus operations. The project concentrates on establishing an online synergic electric bus (e-Bus) data platform to establish operational excellence in zero-emission public transportation. This data platform is intended to be kept independent from the original equipment manufacturer (OEM) and will reinforce the operational usage of a fleet of electric city buses.

The transition to electric counterparts in the public transport system turns up to bring the unforeseen challenges with the higher factor of operational uncertainties. The electric buses drive significantly less distance as compared to their fuel-based counterparts on a fully charged battery. It requires special occasions for charging during the operation. Furthermore, the energy usage pattern of electric buses is more uncertain than the diesel buses. This induces more dynamicity in operation and tactical planning and hence introduce more vulnerability.

The high operational vulnerability can be resolved by adding more assets to the fleet, a cost-inefficient approach. Another approach is to have real-time insights on the status of the electric vehicles and their batteries and the progress of their charging cycles. With this real-time information, the dynamic re-planning of operations of these buses can be facilitated. Within the project, solutions are being introduced to reduce the risks and costs involved.

A cutting edge energy modeling technique is used amalgamated with the advanced energy estimation algorithm to create a toolbox to give more accurate predictions on the energy estimates. This toolbox has capabilities to self-learn from the data perceived from the environment and handle any perturbations in the operational domain of the large fleet of vehicles while ensuring the robustness in the efficiency and accuracy of energy predictions. These predictions in the current energy estimation infrastructure can go about 40 % off from the actual consumption. The real-time energy estimation system is capable to bound the error in energy estimations to $< 2\%$ on an average. When available with the bus operators, the more accurate information can facilitate in making better and faster decisions in optimizing the re-scheduling of the electric city buses. This provides a sustainable solution that can lead to avoiding operational uncertainties and vulnerabilities and helps reduce the costs of operation.

Glossary

ASD	Automotive Systems Design
OEM	Original Equipment Manufacturer
PDEng	Professional Doctorate in Engineering
PSG	Project Steering Group
TU/e	Eindhoven University of Technology
RMSE	Route Mean Squared Error
GPS	Global Positioning System
RCI	Route Characteristics Indicator
HMI	Human Machine Interface
CO_2	Carbon-di-oxide
UNFCCC	United Nations Framework Convention on Climate Change
EV	Electric Vehicle
ICE	Internal Combustion Engine
BEV	Battery Electric Vehicle
HEV	Hybrid Electric Vehicle
PHEV	Plug-in Hybrid Electric Vehicle
FCEV	Fuel Cell Electric Vehicle
ECU	Electronic Control Unit
UC	Ultra Capacitors
AC	Air-Conditioning
SOC	State of Charge
MVLR	Multi-Variate Linear Regression
e-Bus	Electric Bus
OCPP	Open Charge Point Protocol
OCPI	Open Charge Point Interface
GR	General Requirements
CAN	Controller Area Network
SW	Software
ID	Identification
RI	Route Information
CSV	Comma Separated Value
LQE	Linear Quadratic Estimator
AKA	Also Known As
ARM	Advanced RISC Machine
API	Application Programming Interface

List of symbols

E	Energy
m	Mass
m_{eff}	Mass (Effective)
F_r	Rolling Resistance Force
F_x	Propulsion Force
a_x	Acceleration
f_r	Rolling resistance coefficient
g	Gravitational constant
α	Road Slope
F_{aero}	Aerodynamic Drag
ρ	Air Density
C_d	Aerodynamic drag coefficient
A_f	Front Area of Vehicle
v	Velocity
W	Wind Speed
F_g	Road Slope Force
η	Drive-train Efficiency
t, T	Time
s, S	Distance
m	Mass
P_{aux}	Auxiliary Power
P_{bat}	Battery Power
P_{drive}	Drive-train Power
V_{drive}	Drive-train Voltage
I_{drive}	Drive-train Current
V_{bat}	Battery Voltage
I_{bat}	Battery Current
ψ	Regressor Vector
θ	Parameter Vector
K	Gain Matrix
P	Error-Covariance Matrix
γ	Gamma

List of Tables

List of Figures

Contents

1 Introduction

This chapter describes the context of the project while diving deeper into the current infrastructure or the status of the technologies considered for the project. This chapter also describes the project background and scope for the advancement of the technology.

1.1 Zero Emission Transportation

Climate change has become an enormous concern around the world; this requires an immediate acknowledgment from the global leaders to address the CO_2 emission problems caused by transportation. To ensure this, in 2016, the Paris Agreement was signed within the United Nations Framework Convention on Climate Change (UNFCCC), which deals with the issues concerning the greenhouse-gas-emissions mitigation, adaptation, and finances by the members of UNFCCC [1]. This pact involves a portfolio of strategies that must be engaged to subdue the transportation-related pollution and its dependencies on fossil fuels. The industry, public agencies, and the research bodies, therefore, promote the electrification of transportation. Electric vehicles (EVs) are among the most effective transiting options towards a low-carbon emission transportation system. To promote the electric vehicles, some countries are proposing the strict laws on developing internal combustion engine (ICE) vehicles. On the one hand, Norway is aiming to have 100% of its new car sales as EVs by 2025. France, United Kingdom, and California in the United States have indicated diminishing the sales of its ICE vehicles by 2040. China is promoting its EVs sales and aims to reach sales of 7 million units annually by the year 2025. This huge shift towards the EVs is resulting in a paradigm shift in the automotive industry towards the electric drive-train [2]. This shift can be observed by analyzing the sales of these vehicles around the world, as shown in Figure 1.1 [3].

1.2 Electric Vehicles

In most general terms, the electric vehicles are road vehicles whose propulsion unit is electricity instead of burning fossil fuel. The fleet of the electric vehicle consists of battery electric vehicle (BEVs), hybrid electric vehicle (HEVs), plug-in hybrid electric vehicle (PHEVs), and fuel cell electric vehicle (FCEVs). A full electric vehicle configuration includes three major subsystems: electric propulsion, energy source, and auxiliary systems. The electric propulsion subsystem usually consists of a motor, transmission, power converters, and electronic control units (ECUs). The main constituents of the energy source subsystem are energy storage unit, energy management unit, and charging unit. In operation, the most extensively used energy storage device is the battery. The auxiliary subsystem includes power steering unit, HVAC, lighting, brakes, and air suspension.

The vehicle under consideration for this project is battery electric city buses. The operating range of these vehicles is directly dependent on the capacity of the battery and various other factors, including

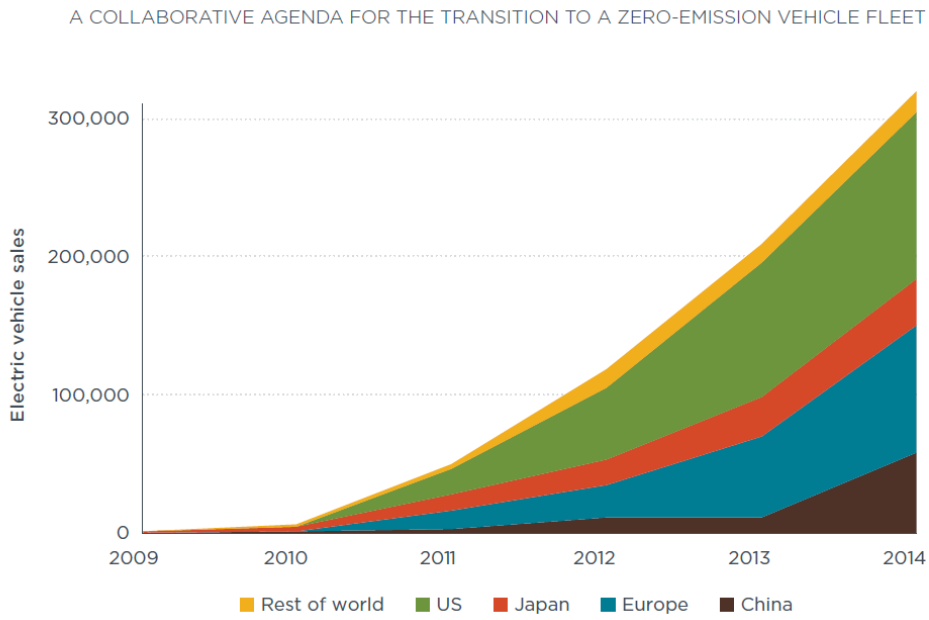


Figure 1.1: Annual Global Electric Vehicle Sales [3]

vehicle characteristics, e.g., weight, configuration, driving style, road, weather, traffic, and payload conditions [4].

1.3 State of the Art

In the present day, electric city buses have captured the market interest due to ambitions set by cities to reduce the carbon emissions. This leads to more research in the field of electric vehicles, especially in creating efficient models to approximate the actual behavior of these vehicles. These models can further facilitate to develop advanced technologies, One of such technologies is energy consumption estimation.

Taxonomy of Influential Variables on EV Energy Estimation

Numerous factors affect the energy consumption of an electric vehicle. These factors are broadly classified into four major categories: vehicle component, vehicle dynamics, traffic, and the environment.

- **Vehicle Component:** The operating states of the critical parts of the propulsion (e.g., electric motors, mechanical transmissions), and energy flow in energy storage and auxiliary systems are governed by vehicle component related parameters. Motor and transmission efficiencies play a crucial role in determining the withdrawal of the energy from the source used for the propulsion. The energy withdrawal can vary based upon specifically chosen configurations of EVs and the motor and transmission technology. The initial state of charge of the battery can also aggravate or mitigate the EV driver's range anxiety. This has subsequent effects on their driving behavior and hence the energy consumption by the vehicle. The auxiliary power is required for air conditioning, ventilation, radio, monitoring panel, lights, power steering, and pneumatics. This is also a significant contributor to the energy demand from the battery for

the electric city bus case. A basic energy consumption scheme for the EVs can be observed in Figure 1.2

- **Vehicle Dynamics:** Vehicle dynamics include aspects of the motion of the vehicles comprising speed, acceleration, and tractive/brake torque. This leads to an energy demand from the vehicle, which is governed by the laws of physics. The road load, physically related to rolling resistance, aerodynamic drag, and road gradient, can be specifically linked to the key parameter; speed [5].

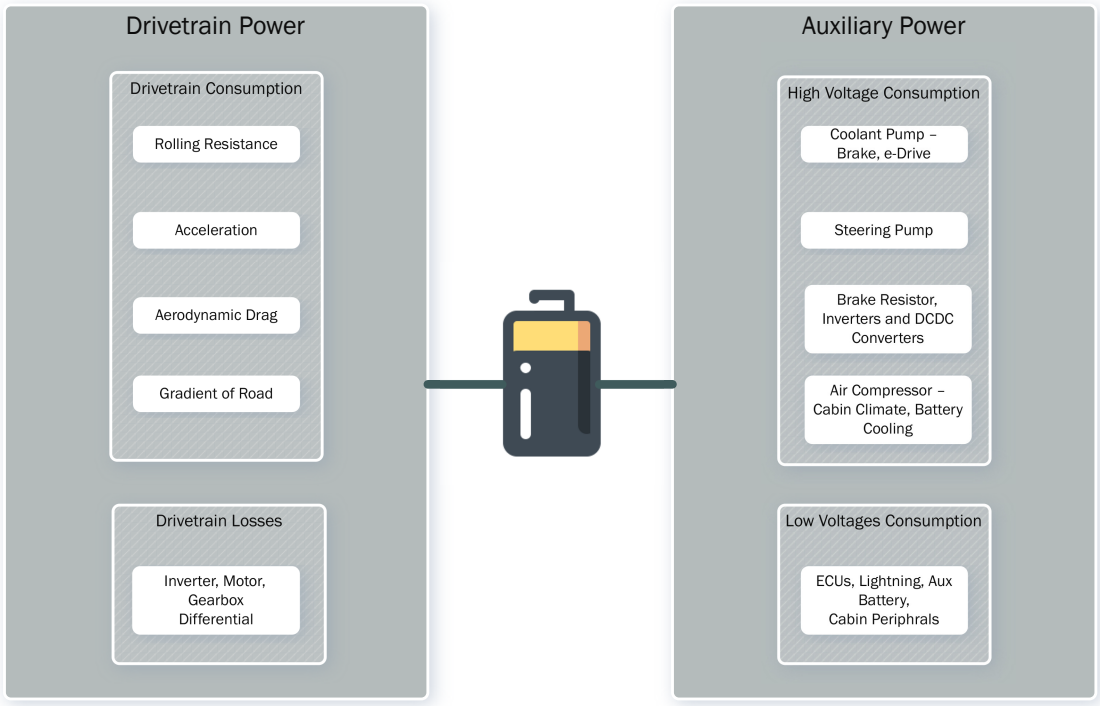


Figure 1.2: Energy Consumption Scheme: Electric Vehicles

- **Traffic Conditions:** Traffic conditions play a significant role in the energy consumption of electric vehicles. The downstream traffic signal status, congestion situations, and vehicle type mix in traffic flow influence EV energy consumption. These conditions can be classified as categorical or interval variables. On the one hand, the categorical variable can determine whether a trip is operated in a certain time or spatial resolution, e.g., time of the day (peak hour vs. non-peak hour), day of the week (weekday, weekend, or holiday), or a month of the year (including seasonal effects). Various techniques are being used to model categorical behavior, such as regression or neural-network models [6]. On another hand, interval variables represent the traffic conditions as a function of continuous vehicle dynamics or overall state of the traffic. To indicate a realistic traffic condition over a trip[1], the ratio of time or number of stops over the travel time can be used as an indicator. This variable is found as a significant estimate for energy consumption in some research [7]. In some other researches efforts were witnessed to

[1]Trip is defined as a journey from start coordinate to the end coordinate of the given route

define the congestion index (i.e., the mean speed of vehicle divided by the standard deviation of the vehicle speed over the route[2]) and was found to be of great significance in the energy consumption models [8].

- **Environment Conditions:** Another important factor influencing the energy consumption of the electric vehicle is the environment. This involves information about road characteristics and meteorological conditions. The most widely used variables in this category are road grade, road type, wind direction, wind speed, ambient temperature, humidity, and lighting conditions [9]. With the advancement in technology, especially the outdoor positioning, road grade information availability with the real-world data, the energy consumption models can be made more accurate. Furthermore, the most commonly used roadway characteristics related variable in the current literature is the road type (freeway vs. arterial). Attributes involving the road infrastructure, including traffic light, speed limit, are used continuously as the independent variables to estimate the consumption [10]. The meteorological variable is the ambient temperature and humidity that affects the auxiliary power for heating or cooling demand of the vehicle. In some studies, the relationship between the meteorological parameters and battery performance was explored by measuring the temperature of the battery cells. In one particular study, the regression model was developed, including a dummy variable representing the day and night time. It was concluded that the lightning conditions were strongly correlated with the EV energy consumption [9]. Moreover, the ambient temperature, and humidity were measured or estimated based on the GPS coordinates of the driving location. This assist in assessing the potential energy consumption for in-cabin heating or cooling. The temperature and humidity has its affects on aerodynamic drag and can influence the energy consumption.

Methodology for Modeling

The methods for modeling the energy consumption behavior of electric vehicles are classified into three categories: rule-based, data-driven, and hybrid. This classification is visualized in Figure 1.3.

- **Rule-Based:** In comparison to the configuration of internal combustion engine powered vehicle, the configuration and energy flow in electric drive-train is less complicated. Also, the individual component-wise energy efficiency of an EV is more constant. The concepts of fuzzy logic were used in some research to model the regenerative braking, while others assumed simply the understanding of the speed of the vehicle [11]. The rule-based models are simpler to implement, but their accuracy might not be satisfactory when applied to a specific vehicle or scenario. This approach further has some challenges when extrapolated to the macroscopic scale. In application to the macroscopic scale, the estimation errors can accumulate if the energy consumption by a fleet of EV is under consideration.
- **Data-Driven:** In the data-driven modeling approach, the data is available from the sensors, automotive electronics, and telematic vehicular technology and can be used to model the energy consumption of the electric vehicle. The most popular statistical method used to date is multivariate linear regression (MVLR). The models built with these methods usually include the instantaneous speed, acceleration, and the interaction terms between them. These interaction terms are generally the independent variables and assume their relationships with a linear predictor function. Other high-level machine learning and high-performance computing algorithms involve artificial neural networks that have been employed to calculate energy consumption

[2]Route is defined as a path followed by vehicle in-between two locations

estimation for electric vehicle. Other useful methods involve unsupervised learning methods (e.g., clustering) for data pre-processing and pattern recognition [12][13][14]. The data-driven approaches can be usefully applied to various data sources, such as vehicle dynamics, traffic information, network profile, and meteorological conditions. There are some limitations of using these models: these models may fail to perform satisfactorily outside their training data sets. Therefore, it is imperative to ensure that the training data is a good representative of the required information about the vehicle. Secondly, these black-box models can provide satisfactory accuracy. Still, it explains a little about the different details of the parameters within and its implications on the energy consumption of the vehicle.

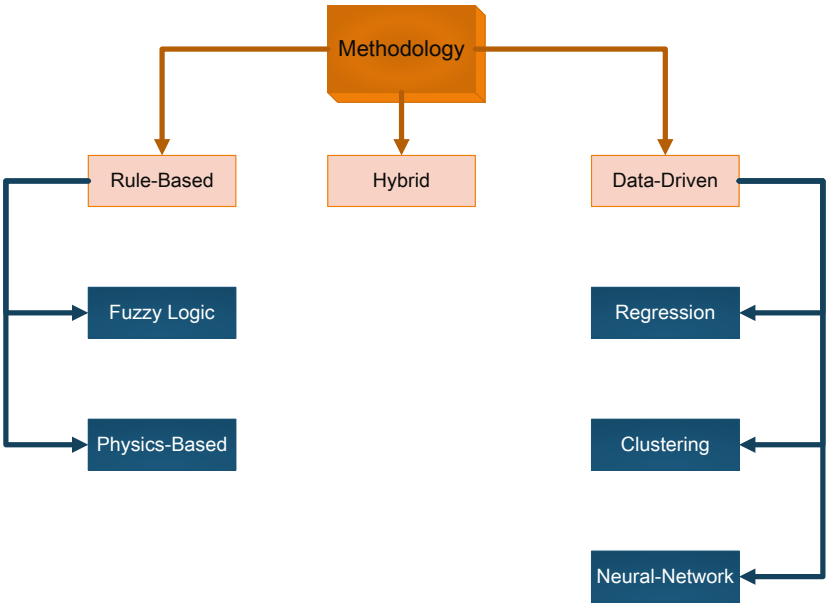


Figure 1.3: Classification scheme for the state-of-the-art EV energy modeling

- **Hybrid Models:** To combine the advantages of both the rule-based methods (simplicity of the model and generalized formulation) and data-driven methods (model accuracy and customization), the hybrid approach can be implemented to have an estimate on the energy consumption of electric vehicles. The feature selection for the data-set is based upon the physical principles, whereas the model parameters are trained to attain the best performance for the selected scenarios [15].

1.4 Project Description

Project Background

Zero-emission transportation is intended to flourish with the earliest movers to be public transport in this field. An increasing number of electric buses are replacing conventional ICE engine buses in the city areas in entire Europe. The bus operators, particularly in the Netherlands, are amongst the pioneers to adapt to the new technology.

The evolution of public transportation from classical ICE buses to electric buses leads to compelling challenges in operational uncertainties, vulnerabilities, and costs. The ambition of the EU funded **Cloud Your Bus** project an EM Europe Research and Innovation Project is to stimulate the transition to zero-emission public transport in Europe. The business goal revolves around reducing the risks and costs of this transition to zero-emission bus operations. The project concentrates on the establishment of an online synergic electric bus (e-Bus) data platform with the intent to establish operational excellence in zero-emission public transportation. This data platform is intended to be kept independent from the original equipment manufacturer (OEM) and will reinforce the operational usage of a fleet of electric city busses.

The transition to electric counterparts in the public transport system turns up to bring the unforeseen challenges with the higher factor of operational uncertainties. A diesel bus, e.g., can drive quickly for a range of about 300 km on a regular shift and can be refueled at the depot overnight. Its electric counterpart, on the other hand, drives significantly less distance on a fully charged battery. It requires special occasions for charging during the operation. Furthermore, the energy usage pattern of electric buses is more uncertain than the diesel buses. This induces more dynamics in operation and tactical planning and hence introduce more vulnerability.

The high operational vulnerability can be resolved by adding more vehicles to the fleet, a cost-inefficient approach. Another approach is to have real-time insights on the status of the electric vehicles and their batteries and the progress of their charging cycles. With this real-time information, the dynamic planning of operations of these buses can be facilitated. Within the project, solutions are being introduced to reduce the risks and costs involved.

The following work packages are defined in this project:

- **eBus data taxonomy:** Streaming a set of standardized BMS/eBus data across different bus types within the context of emerging standards.
- **Live charging data:** Integrating live charge point data in operational planning, integrating the latest OCPP/OCPI protocols.
- **Drive Style optimization :** Influencing drivers to adopt the lowest possible energy profile by providing feedback on drive style specifically tailored to electric buses while driving.
- **Planning optimization :** Adapting line and charge planning in real-time by monitoring set line/charge schedules and performing dynamic re-planning in case of exceptions.
- **Battery monitoring:** Monitoring life-time battery state-of-health by evaluating degradation patterns based on driving and charging patterns. Ensuring that buses are used in conformity with warranty conditions and proactive alerting if this is not the case.
- **Energy consumption modeling:** Optimising the fleet asset base per concession area by developing energy models based on actual energy usage for different road segments, ambient

temperature, load, drive style, traffic intensity.

More information regarding the project is available on https://cloudyourbus.com/

Project Aim

TU/e is responsible for working package **Energy Consumption Modeling** having its impact on working package **Drive Style Optimization**. In the context of energy modeling package, this project aims to develop an energy consumption model for battery-electric buses that can facilitate the dynamic planning of operations of electric buses in real-time.

Project Scope

The project scope evolves around developing and upgrading the energy consumption model for battery electric busses and determining the parameters for this model using real-world data while creating an energy consumption prediction tool for integration in planning systems for urban bus operations.

Project Objective

- Develop and upgrade the generic energy consumption model for battery electric busses.
- Identify the relevant data and its frequency to be recorded from the busses for parameter identification.
- Perform parameter identification on a generic energy consumption model using real-world data.
- Validate the generic energy consumption model using different real-world data.
- Fine-tune the generic energy consumption model.
- Create an energy prediction tool for integration in the planning system for urban bus operations.

Project Contribution

The main contributions of the project are as follows:

- **Cutting-edge model:** The energy consumption model developed in this research use only few parameters hence, avoiding the estimations for a large number of model parameter variables leading to more complexities and non-reliable estimations.
- **Advanced estimation techniques:** In this research an advanced algorithm has been designed, which has the capability of self-learning from the data collected and make corrections for uncertainties during the vehicle's operation. These corrections were made over its predictions throughout the route as the newer data was made available during the trip.
- **Facilitate the Operator:** The output of the real-time energy estimation system allow the user of the system to have more information regarding the energy consumed by the electric city bus over the selected route for any given trip. A reliability index assure that the operator had enough capability to make informed/guided decisions when re-planning the operations schedule.

1.5 Report Outline

The report is organized as follows:

- Chapter 2: This chapter describes the methodology of the working process that was carried out during the project.
- Chapter 3: This chapter gives a detailed description of the architecture developed for the system under development. It also provides an overview of the design decisions taken during the project.
- Chapter 4: This chapter provides a theoretical background on the subjects required for the development of the real-time energy estimation system.
- Chapter 5: This chapter gives an overview of the design implementation work carried out during the prototyping phase of the project.
- Chapter 6: This chapter demonstrates the effectiveness of the proposed methodology through simulations using various test scenarios.
- Chapter 7: This chapter summarizes conclusions and recommendations for future work.

2 Methodology

This chapter describes the methodology followed during the project. It defines the project's different phases, and activities carried out in the subsequent phases.

2.1 Work Phase Plan

The project is divided into certain phases to make the development process streamlined and easy to manage. These phases are as follows:

- **Scope Definition and Planning:** This phase involves understanding the project and stakeholder concerns in order to define the high-level system requirements. The initial project timeline was documented in this phase. This was managed in the form of Gantt Chart using the tool Microsoft Project.
- **System Architecture:** After the scope was defined, the CAFCR framework was used to decompose the architecture description and to set a building block for the design architecture. This was further elaborated in Section 2.2.
- **Design Implementation:** Based upon the V-Model in Section 2.2, the design, development, and testing have been carried out on the defined system.
- **Validation:** This phase was necessary to validate the design implementation. The experiments were done, and the results were analyzed in order to check the functionality of the developed system.

2.2 Development Process

This section discusses the processes used in the development phase of the project.

CAFCR

The "CAFCR" model is used for the design and development of complicated systems. It is a decomposition of system description into five views, as shown in Figure 2.1. The different views can be understood as the customer objectives view (**What** does the customer want to achieve?) and the application view (**How** does the customer realize his goals?) capture the needs of the customer. The needs of the customer (what and how) provide the justification (**Why**) for the specification and the design choices made. The functional view describes what of the product, which includes (despite its name) the non-functional requirements. The how of the product is described in the conceptual and realization views. On the one hand, the conceptual view deals with the conceptual picture of the proposed design, whereas the realization view burrows deeper into how these concepts will be realized

as a design [16].

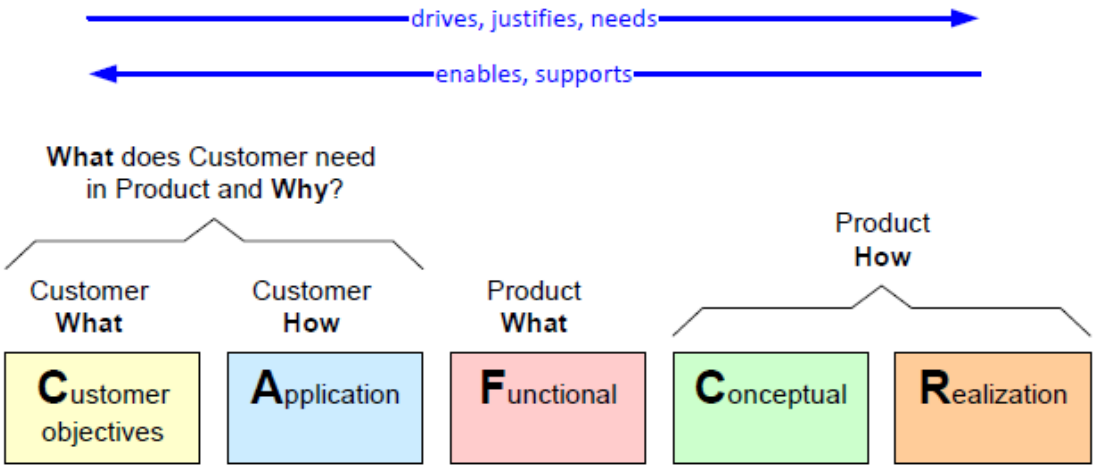


Figure 2.1: CAFCR Model: Different Process Views [16]

The particular approach was applied to the project. The CAFCR structure obtained by gathering all relevant views in a single frame is shown in Figure 2.2

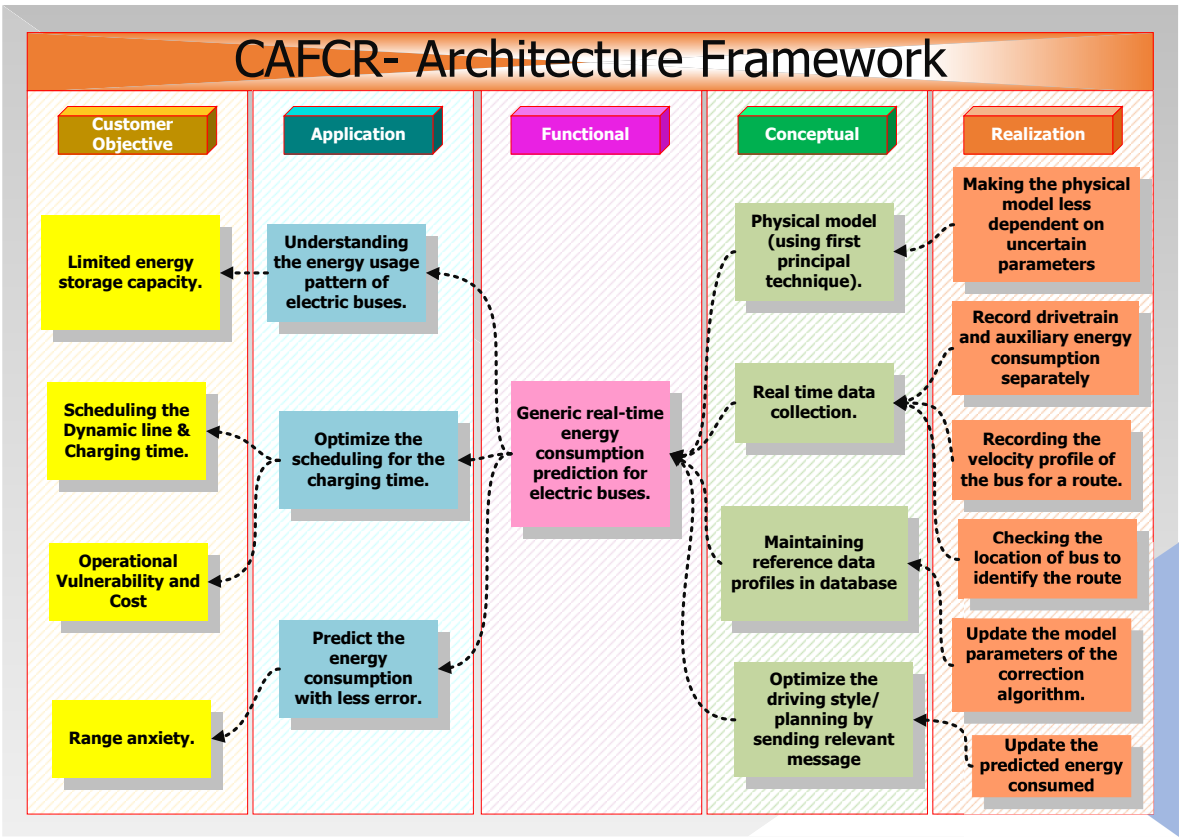


Figure 2.2: CAFCR Model: Real-time Energy Consumption Prediction

The CAFCR framework's supports the development of the main functionality for the system under investigation. It can be clearly seen from Figure 2.2 that the primary function of the project is to create a generic real-time energy consumption prediction system for electric buses. Each of these views is intertwined and can be linked together. The ideas that need to be realized can be traced back to the customers' actual objectives through the intermediate views. The CAFCR framework gives the basis to define further the system requirements which are discussed in Chapter 3.

V-Model

The rational integrated systems development process emphasizes using the classic "V" diagram. The left section of the V-model describes the top-down design approach. In contrast, the right-hand section describes the bottom-up integration aspects from unit testing to the final stage of system development, which is acceptance testing [17].

Using the notation of the *Change Request* on the workflow, the high-level interrupt can be visualized on the process. This will ensure that whenever an interrupt occurs, the process must start from the requirements analysis phase. The harmony process, as shown in Figure 2.3 consists of two closely cohabited sub-processes.

- Harmony of System Engineering
- Harmony of System Development

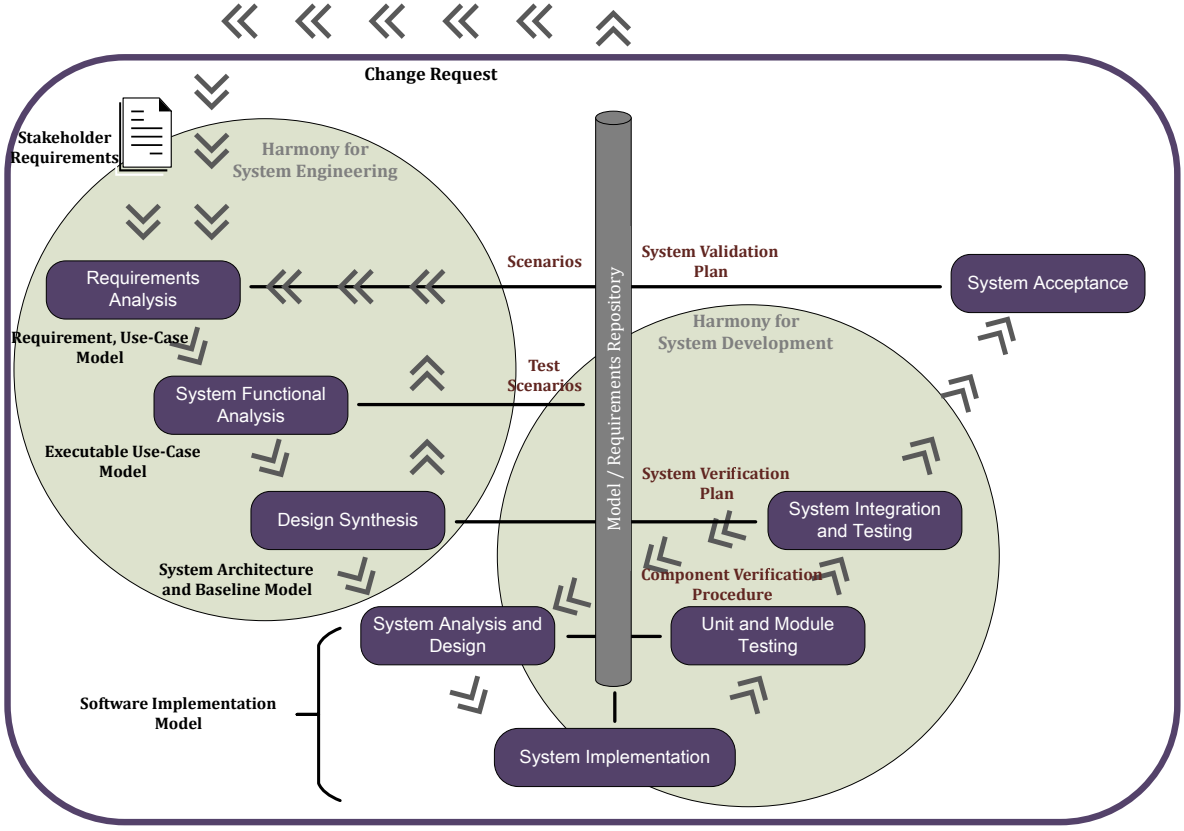


Figure 2.3: V-Model: Harmony of System Engineering and System Development

The system engineering workflow is iterative with incremental cycles through the requirements anal-

ysis, system functional analysis, and design synthesis. These increments are based upon use cases. On the other hand, the system development phase's workflow iterates through system analysis and design phase, implementation phase, and different levels of integration and testing. The system engineering and development are going through the process, providing something demonstrable with each iteration. The creation and use of requirements related test scenarios during the initial phases of the process is useful for assisting the integration and testing during the later phases. It is crucial to understand that a central model-driven approach is useful in creating the correct workflow.

2.3 Project Timeline and Milestones

The project timeline gives a detailed overview of the time for different phases during the project. The initial version of the timeline was proposed during the project's initiation phase and was updated from time to time. A detailed version of the timeline with its corresponding milestones achieved can be seen in Figure 2.4

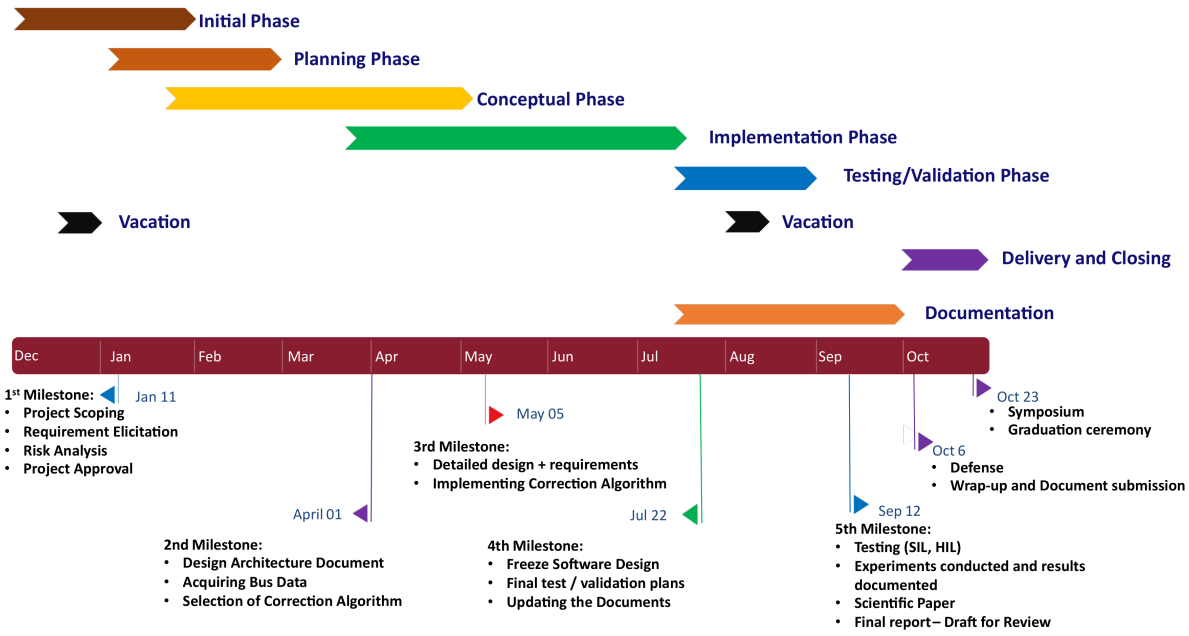


Figure 2.4: Project Timeline

2.4 Summary

In this chapter, the methodology is provided to carry out the development of the project. It explains how the different work phases are planned throughout the project which helps in allowing the formulation of the work breakdown structure. This chapter also gives a foundation on the system thinking concept developed for real-time energy consumption prediction system using the CAFCR model of development. The V-model also allows to develop a realistic timeline for the project.

3 System Architecture and Design Description

Understanding the functionality of the system can be attained through a descriptive architecture of the system. This chapter elaborates and document the architecture and design decisions for the real-time energy consumption prediction system.

3.1 Model-Based Systems Engineering

The objective of model-based system engineering revolves around identification and derivation of desired system functionality. It is also important to identify the associated system modes and states and finally allocating the identified functionality and modes/states to a sub-system structure on a high-level of abstraction with regards to modeling. The attention remains on identification and allotment of required functionality and state-based behavior and not on the detailed functional behavior. Model-based systems engineering begins with analyzing the concerns of the stakeholders. The next section 3.2 describes these concerns in detail.

3.2 Stakeholder Analysis

It is crucial to analyze the concerns of the stakeholders. The two major stakeholders in the project were dr.ir. Igo Besselink (project lead from TU/e) and Sycada (direct client from **Cloud Your Bus** Consortium). Below the primary concerns of these two key stakeholders are listed:

Technical University Eindhoven

- Understanding the energy usage pattern of electric city buses.
- Accounting for the uncertainties such as weather, traffic, road, and payload conditions affecting the energy consumption of electric city buses.
- Identifying the relevant standardized messages to be sent to the cloud.

Sycada

- Logging data in real-time.
- Analysis of status of electric vehicles and charging point infrastructure.
- Facilitate the energy consumption prediction for optimizing dynamic scheduling.

3.3 Requirement Analysis

The intention of performing the requirement analysis is to inspect the process inputs. The stakeholder concerns are the starting point to define the requirements for the system to be developed. These system requirements educate the developers about the functionality of the system and its performance. More concrete functional and performance requirements can then be derived from key system requirements in the developing phase.

The workflow adopted in the requirement analysis phase to make a transition to define the use cases for the system is described in Figure 3.1

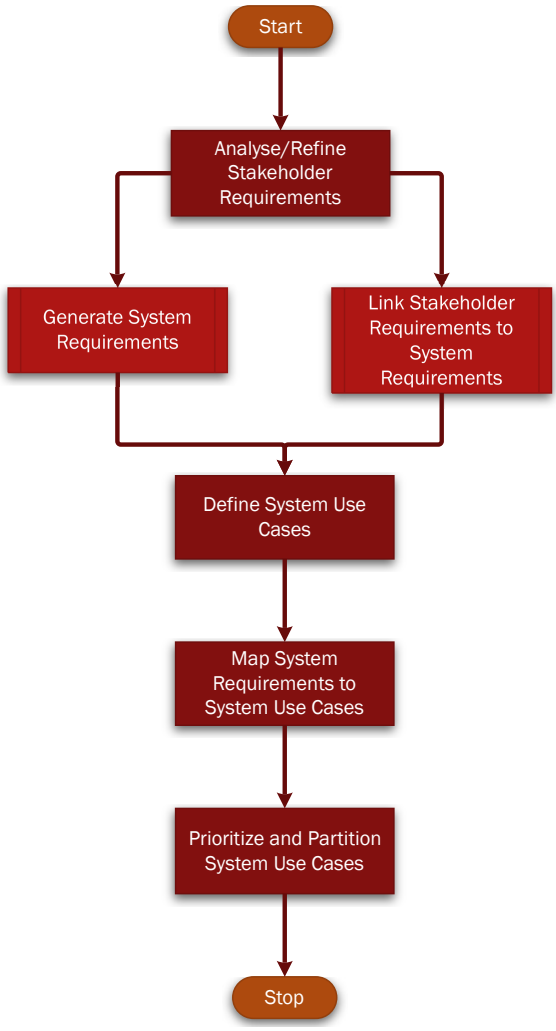


Figure 3.1: Workflow: Requirements Analysis Phase

This process has gone through a few iterations with refinements in the stakeholder requirements before starting the system's design. The requirements were prepared and documented in the Requirements document (See **Appendix A** for a detailed look at functional and performance requirements and how these requirements were mapped with the stakeholder concerns). The most high-level requirements are presented in the Table 3.1. Here GR means that these requirements are contained in General Requirements packages (More details in Appendix A).

High-Level Requirements

Table 3.1: High-Level System Requirements for Real-time energy estimation system

ID	Name	Specification	Status
GR - 1	Predict Energy Consumption	The system must predict the energy consumption needed for a trip of battery-electric busses.	Done
GR - 2	Generic Model	The system must work on vehicles of different OEM's.	Done
GR - 3	Cloud Service	The system must create a relevant message to be sent to cloud service.	Done
GR - 4	Dynamic Scheduling	The system must send a relevant message to the cloud to be used in dynamic scheduling.	Done
GR - 5	Trip Status	The system must be able to identify the start and stop of the trip cycle for the respective route.	Done
GR - 6	Update Parameters	The system must account for uncertainties and update the model parameters periodically to give accurate energy consumption estimates.	Done
GR - 7	Energy	The system must provide as output a pre-determined energy consumption estimate for the given route and corrected predicted/estimated energy for the given route.	Done
GR - 8	Database	The system must have an updated database to store parameters for different routes.	Done
GR - 9	Source Code	The system must run locally on the embedded device.	Done
GR - 10	Reliability	The system must produce reliable estimations of the predicted energy consumption.	Done

3.4 Use Case

After the first set of requirements are detailed, the next step is to identify the use-cases and to make use-case descriptions. A use-case diagram is mandatory to understand the operational aspects of the system. The use-case usually is the means to specify the system's behavior as perceived by the actors/users of the system. The use-cases actors can be a person, another system, a piece of hardware, or another software. The system's internal structure can not be implied or revealed using a use-case diagram.

The use-case diagram for the real-time energy consumption prediction system was made, and is shown in Figure 3.2. It can be seen that the system has two primary use cases, also called base use-cases, which are *"Receive CAN-Bus and GPS Data"* and *"Send information to Sycada Software (SW)"*. These two base use-cases are linked with other sub-use cases with the directed relationships explained next:

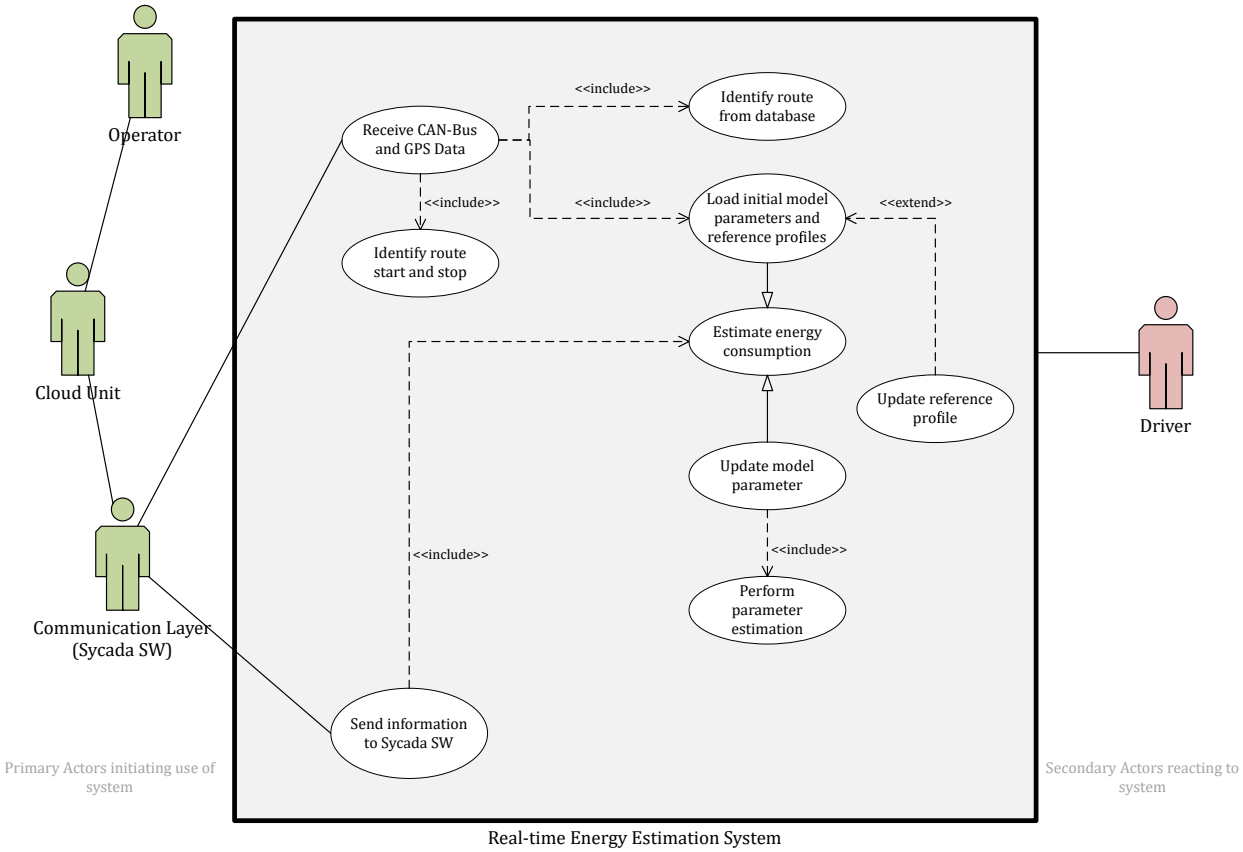


Figure 3.2: Use-Case Diagram

- **Receive CAN-Bus and GPS Data:** The base use-case includes the use-cases such as *"Identify route from database"*, *"Identify route start and stop"* and *"Load initial model parameters and reference profile"*. This «include» relationship allows the use-case to be entertained alone while checking the completeness of the base use-case. On the other hand the *"Update reference profile"* use-case was added with «extend» relationship. This relationship allows conditional usage of the use cases so the base use-case can be used without this extension.
- **Send information to Sycada SW:** This base use-case includes the use-case *"Estimate energy consumption"*. The generalization relationship was used with this use-case since it was allowed to use any other use cases conditionally. The conditional use cases were *"Load initial model parameters and reference profile"* and *"Update model parameter"*. Finally, an include relationship was used amongst the use-case *"Perform parameter estimation"*

Understanding the use-case description is equally important as understanding the directed relationships among them. A use-case description involves an overview of the actions defining the actors' interactions and the system to attain a goal. In the following section, a descriptive understanding of these use-cases is made.

Use-Case Description

- **UC-1 : Receive CAN-Bus and GPS Data (Base Use-Case)**
  1. The data available on the CAN-Bus and GPS data is accessed by the estimation system.

2. The data is pre-processed before being used in the algorithms of the estimation system.
3. The pre-processed data is checked.

- **UC-2 : Send Information to Sycada SW (Base Use-Case)**
  1. The energy estimation data is recorded and sent to Sycada SW.
- **UC-3 : Identify route from database (Sub Use-Case)**
  1. The estimation system receives the route identification number from the Sycada SW.
  2. The estimation system compares the received identification number from the list of identification numbers and its corresponding parameters and profile in the memory.
  3. The estimation system identifies the current route from the set of the available routes.
- **UC-4 : Identify route start and stop (Sub Use-Case)**
  1. The estimation system reads the GPS data available after the pre-processing.
  2. The estimation system compares the current start and stop GPS coordinate parameters from the selected GPS start and stop coordinate route parameters.
  3. The estimation system raises a flag (Start Route) when current GPS coordinates are near the selected start GPS route coordinates.
  4. The estimation system raises a flag (Stop Route) when current GPS coordinates are near the selected stop GPS route coordinates.
- **UC-5 : Load initial model parameters and reference profiles (Sub Use-Case)**
  1. The estimation system checks if the route has started and the *Start Route* flag is raised.
  2. The estimation system loads initial model parameters and reference profiles to be used.
- **UC-6 : Update reference profile (Sub Use-Case)**
  1. The estimation system collects the real-time data from the algorithm.
  2. The estimation system updates the reference profile for better estimation.
- **UC-7 : Estimate energy consumption (Sub Use-Case)**
  1. The estimation system performs the real-time estimation and correction on drive-train energy.
  2. The estimation system performs the real-time estimation and correction on auxiliary energy.
- **UC-8 : Update model parameter (Sub Use-Case)**
  1. The estimation system collects the data in real-time.
  2. The estimation system updates the model parameters to correct for discrepancies in the estimates and actual values.
- **UC-9 : Perform parameter estimation (Sub Use-Case)**
  1. The estimation system performs online-parameter estimation to improve the predictions.

Use-Case and Requirements Tractability

In order to understand if all the functional and performance requirements of the system are covered by the use-cases, it is required to associate the use-cases with the requirements. Here, the association is done with the high-level requirements of the real-time energy consumption prediction system. Further, this association can be linked to all functional, performance, and technical requirements by using the mapping between the requirements from the **Requirements Document** made available in Appendix A

Table 3.2: Use-Case mapping with High-Level Requirements

ID	Name	Satisfied with
GR - 1	Predicted Energy Consumption	UC-7
GR - 2	Generic Model	UC-8
GR - 3	Cloud Service	UC-2
GR - 4	Dynamic Scheduling	UC-2
GR - 5	Trip Status	UC-4
GR - 6	Update Parameters	UC-8, UC-9
GR - 7	Energy	UC-7
GR - 8	Database	UC-3, UC-5, UC-6
GR - 9	Source Code	UC-1, UC-2
GR - 10	Reliability	UC-7, UC-8

3.5 Architecture

After deriving the use-cases and ensuring the completeness of the system functionality within these use-cases, the steps involved in further establishing the system's architecture are the definition of the structural and behavioral diagrams for the system under development. The tasks of creating the structural and behavioral diagrams were handled simultaneously.

Structural Diagrams

The structural diagrams explain the static structure of the system and its parts at different abstraction and application levels. They also describe how these parts are interconnected. The elements in the structural diagram represent meaningful concepts of the system, like implementation in the real world. The first diagrams in this category is the system context diagram.

System Context Diagram: The system context diagram is a high-level view of the system that defines the boundaries of the system and its environment. It shows all the entities present in the environment with which the system interacts. It also gives complete information about the flow of information between the system and the entities in the environment. The entities available in the environment can be sensors, actuators, users, other systems, or software. The system context diagram is described in Figure 3.3. In this figure, it can be seen that the system of interest is a real-time energy consumption prediction system that is in-acting with other systems in the environment. Its direct linkage is with the communication layer which is the software prepared by the stakeholder [Sycada], which further interacts with the CAN-Bus and the cloud unit. The input in this case for the real-time energy estimation system will be the CAN-Bus data, and the output from the system will be the energy consumption

estimates.

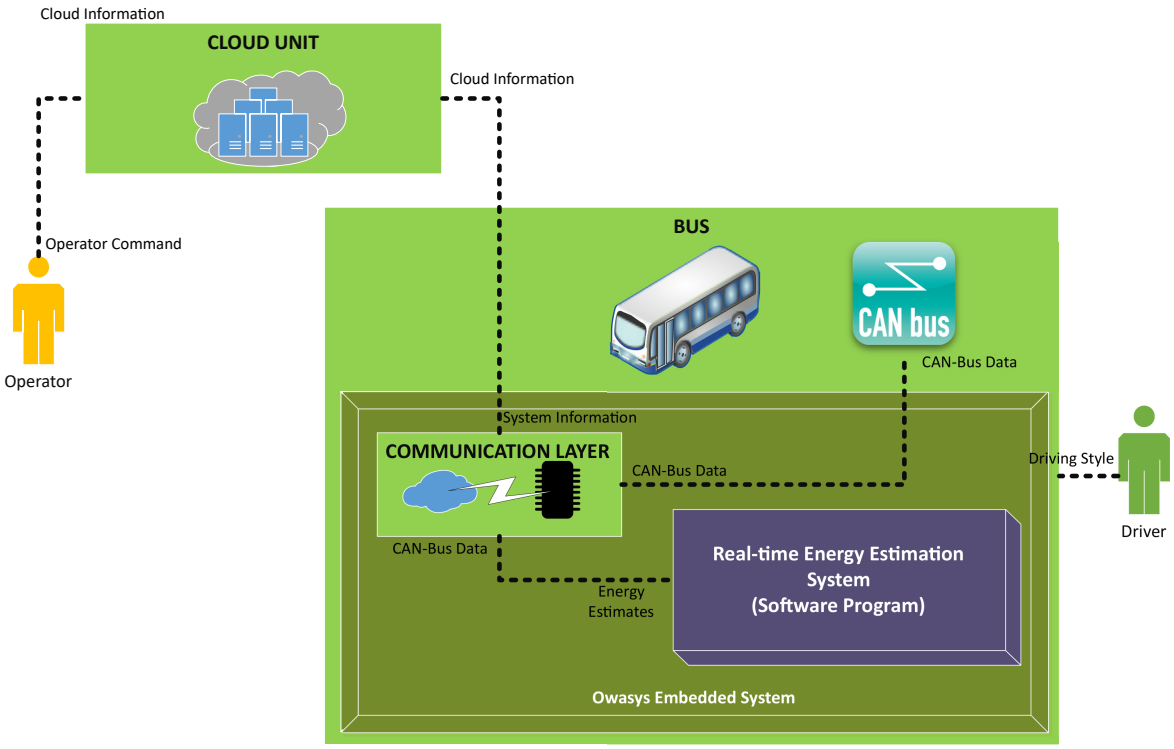


Figure 3.3: System Context Diagram

Block Definition Diagram: The block definition diagram shows the essential structural elements or blocks of the system and their relationships/dependencies. For the real-time energy estimation system, the blocks are defined in Figure 3.4. The detailed functionality of each block is as follows:

- **Cloud Unit:** The block cloud unit is used to send the route information by the operator for the particular bus schedule. The information is received by the communication layer software developed by Sycada from this block. This block is actually out of the system scope and is drawn just to connect the dots. *The functional blocks in scope are the ones contained in the On-board unit section in Figure 3.4.*
- **Receive Data:** The block receiving data is in direct contact with the communication layer software. It receives the relevant information required for a real-time energy estimation system to work, such as CAN-Bus and GPS data and the route information.
- **Lookup Table:** For corresponding route ID's certain parameters that are unique to the route ID are stored in the lookup table block.
- **Database:** The database block stores the reference data profiles both base and updated reference profiles for the specific routes. This database stores information on data profiles to perform the estimations. These data profiles are Road Characteristic Indicator (RCI), velocity, voltages and currents, distance traveled, time, and auxiliary power profile.
- **Online Parameter Estimation:** This block performs the parameter estimations to correct the

estimates as the data becomes available in real-time.

- **Measure Data:** This block performs all the tasks involved with measured data. The tasks include pre-processing and performing health checks.
- **Model Parameter:** This block allows the parameter values to be fed into energy estimation model parameters. It handles both the initial and updated parameters required by the model to provide energy estimates.
- **Energy Consumption Model:** This block contains the model of energy consumption by the electric vehicle.
- **Energy Computation:** Taking its inputs from the energy consumption model, this block provides at any specific moment in the trip, the estimate for the energy consumed by the vehicle by the end of the trip.
- **Send Data:** This block is used to send the relevant information required to the cloud unit using the communication layer software.

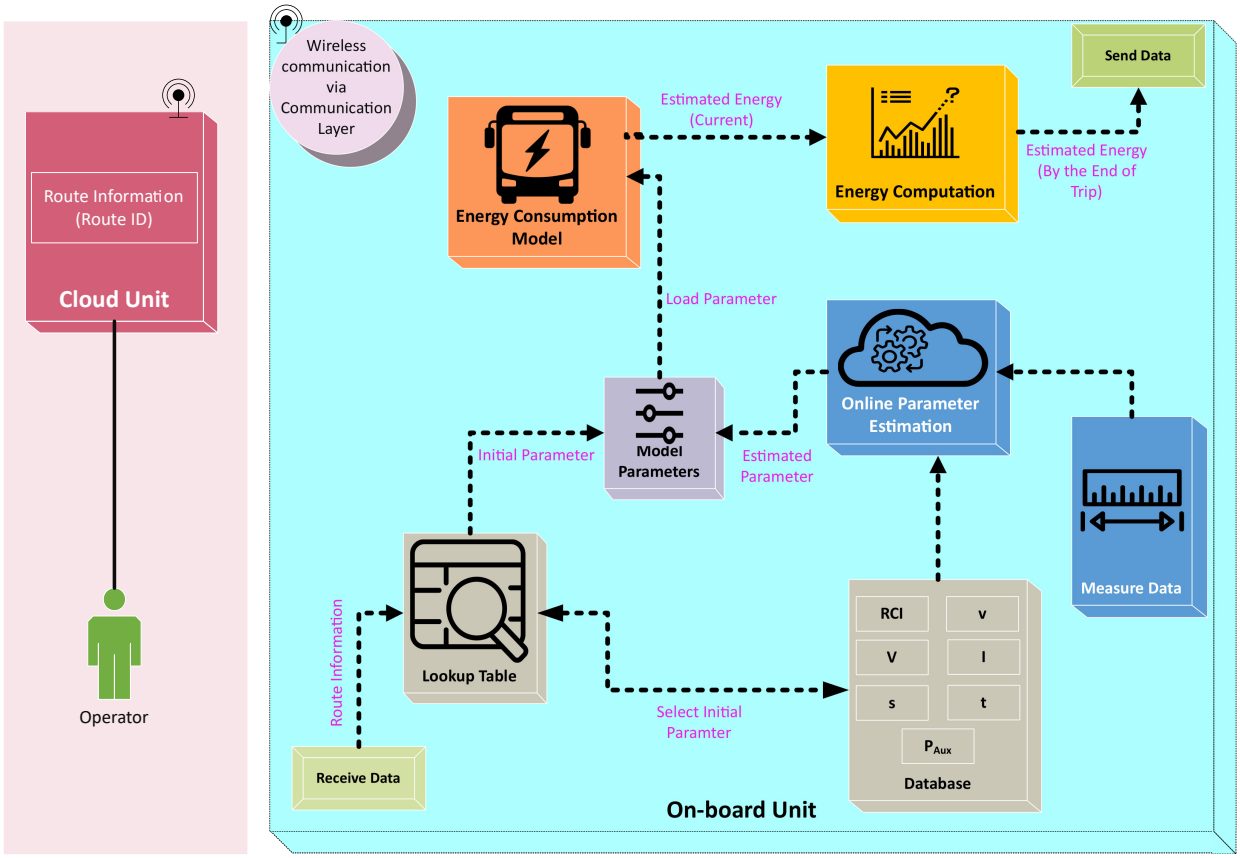


Figure 3.4: Block Definition Diagram

Detailed Block Definition Diagram: A detailed block definition diagram was designed to complement the design architecture. It defines the functions of the system under development in more detail. It was designed, keeping in mind all technical design, functional, and performance requirements captured in the **Requirements Document**. The scope of the system design can be seen in Figure 3.5. The blocks

captured in the bottom left of the figure surrounded by the dotted orange line are out of the scope of the system design.

The diagram captures the internal structure of the system blocks in terms of its properties and connections between these properties. The interactions between the components and the information flow in the system can be observed with the detailed block definition diagram.

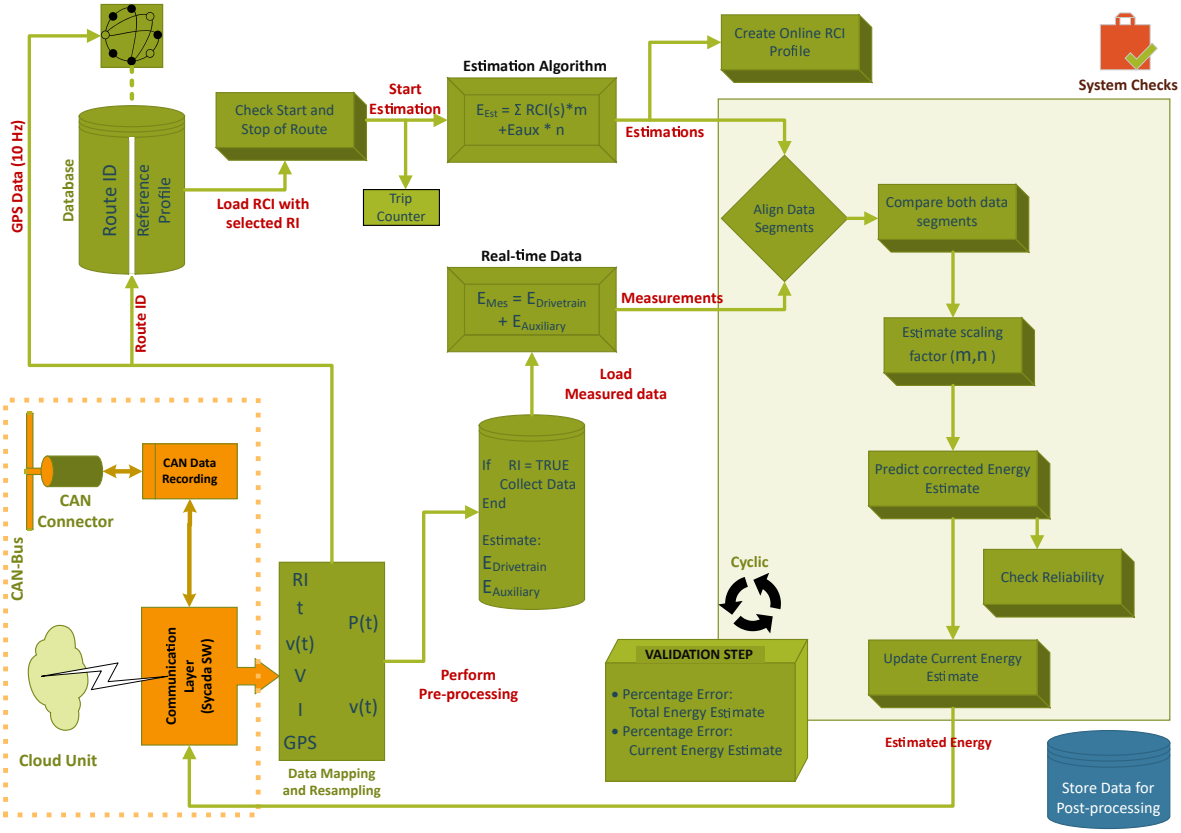


Figure 3.5: Detailed Block Definition Diagram

The system requires a flow of information from the communication layer software to ensure its functioning. *Route Information* (RI) is one of the critical signals needed to identify the actual route of the bus. Using this information, the corresponding reference profile is extracted from the database to allow the estimation algorithm to start predicting the energy consumption. As soon as the new measurements are available, the estimation algorithm tries to improve the estimations by minimizing the error between the measured and estimated value of energy. The estimations are reviewed for their reliability. The estimated energy is sent back to the communication layer software, ensuring the correct information can be made available to the user over the cloud service. Furthermore, during the operation, some data is stored in memory for post-processing and updating the reference profiles in the database. Also, system checks are performed during the whole operation (Described in Section 5.3).

Interface Diagram: The energy consumption prediction system (software) developed has to work in harmony with the software developed by the client. It was essential to understand the interfaces between these two software in order to ensure the overall functionality and deployment of the system.

The interface diagram is shown in Figure 3.6.

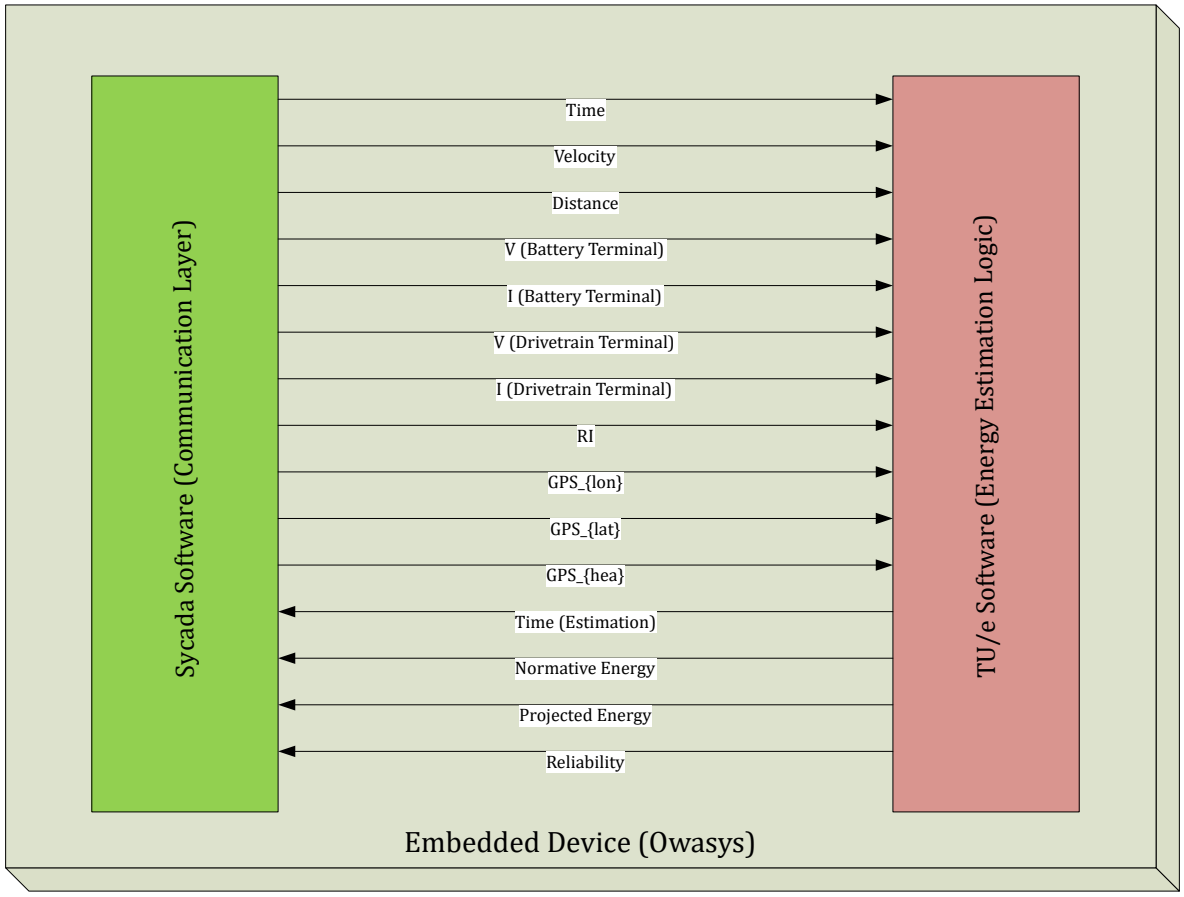


Figure 3.6: Interface Diagram

This diagram is used to understand the information flow between the two software's and in what direction. More details regarding the description, type, resolution and format of the signals is available in the **Interface Documentation** (Refer Appendix B).

Behavioural Diagrams

The behavioral diagrams explain the functional and dynamic behavior of the objects in the system. These behaviors can be described as a series of changes to the system over time. Each behavioral diagram plays a specific role in describing the use-case behavior. The *Activity Diagram* can be used to describe the overall functional flow of the use-case. It allows to group functional requirements in action with the equivalent operations and grants a link in between them. On the other hand, the *Sequence Diagram* elaborates upon the interactions between the actual operations and the actors. A *State-Chart Diagram* assembles the information from the activity and sequence diagram. This diagram allows putting this information into the context of the system states. It adds to it the behavior of the system due to external stimuli of the different priorities [17].

System Activity Diagram: These diagrams show sequence and conditions for coordinating lower-level behaviors and are comparable to the classic flowcharts. It can define a workflow, process, or algorithm

by decomposing the flow of the implementation into a set of actions and sub-activities linked by transitions and various connectors. An activity diagram usually is a simple line order of actions, but it can also be a complex series of parallel actions with conditional branching and concurrency.

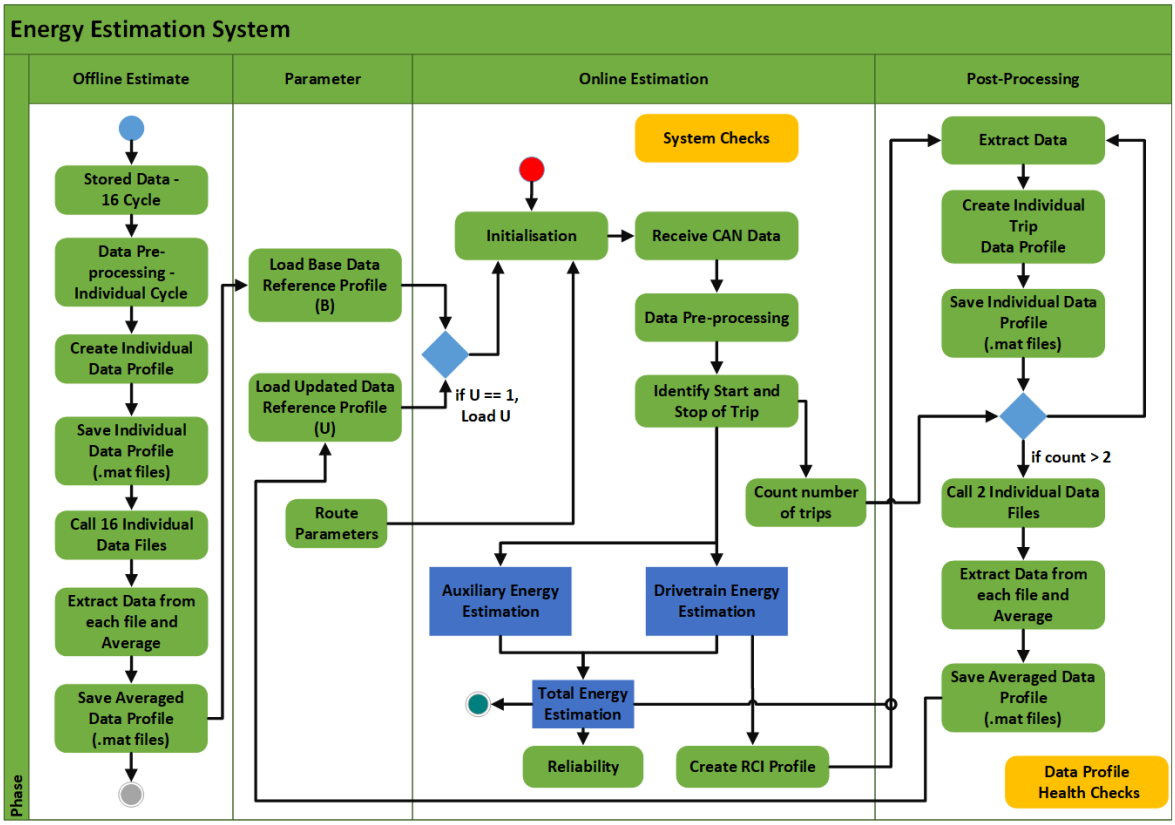


Figure 3.7: System Activity Diagram

The system activity diagram in Figure 3.7 illustrates four features in the developed system in Matlab/Simulink. These features of the system are as follows:

- **Offline Estimate:** *This feature in the system is used to have an energy estimate for the given route as soon as the route has been selected. The energy estimate for the route is made available before the start of the trip on that particular route.*
  For this feature the line of order of actions starts with storing the data required by the system for every trip cycle for the given route. It is then followed with some data pre-processing on the individual trip cycle. This data is further saved in the different data profiles as structured data for each individual trip cycle and saved in the form of files (.mat or CSV). After the 16 data files with data information from individual trip cycles are available, an averaged data profile is created and stored as an individual file. The rationale for the selection of 16 data files is provided in Chapter 5.
- **Parameter:** *This feature in the system exists in order to store certain route parameters, data reference profiles, and other information in memory.*
  Once the averaged data profile is available from the offline estimation tasks performed, the available individual data file is stored as "Base Data Reference Profile" in the memory, which is

later used by online parameter estimations to perform the estimations. Similarly, it also stores the "Updated Data Reference Profile" as the result of the post-processing feature.

- **Online Estimation:** *The most critical feature of the system is performing the estimations in real-time and executing corrections on the previous estimates as the new measurement data becomes available.*
  The action starts with checking if the "Updated Data Reference Profile" is available in the memory. If this file is available, then the online estimations are initialized with the updated profile; otherwise, the estimation algorithm uses the "Base Data Reference Profile". Once the parameters and the data profiles are loaded, and the system receives the CAN-Data, it starts checking if the trip has begun using the GPS coordinates data. Once the information is there that the trip has begun, the energy estimations start. Simultaneously, the Road Characteristics Index (RCI) profile [more explanation in Chapter 5] is created, which is required for updating the reference data profile. Concurrently, it is also ensured that the system checks are being performed by the system.
- **Post-Processing:** *Another important feature of the system is to carry out the processing of data during estimations to update the reference profile with the new incoming data. This allows the system to learn by itself and improve the estimations while accounting for the uncertainties occurring in the operational domain of the vehicle.*
  The activity action starts with the collection of the relevant data during the estimations and saving this data as individual profiles. When the two trip cycles (see Remark 3.5.2) have been finished, and the data from both trips are available in the form of files. This data is averaged in order to create an "Updated Data Reference Profile" which is stored in the system memory. The system also ensures the good health of the data files saved in the memory by performing relevant checks.

Remark 3.5.1 *As shown in Figure 3.7, the **Data Profile** is the data structure in which some data is stored as reference data, which is used by the estimation algorithms. This data structure includes arrays of data of the distance traveled on the route, time taken for the trip cycle, RCI profile estimated during the trip cycle, and auxiliary power consumed during the trip cycle.*

Remark 3.5.2 *The rationale for updating the **Updated Data Reference Profile** after the two trip cycles is because, during the analysis of data collected on for about three weeks from a unique bus. It was observed that the particular bus was scheduled to travel on the same given (selected) route at most two times in a day. It then made more sense to update the reference profile on a daily basis to account for the uncertainties of the weather, traffic, and payload conditions and to adapt the reference profile with these conditions to have a better estimate.*

State Chart Diagram: A state-chart diagram describes the state-based behavior of a block. It is one of the most significant behavioral diagrams as it has the ability to combine information from the activity diagram or sequence diagram and adds to it the event-driven block behavior. A state-chart diagram is mostly composed of a set of states joined by transitions and various connectors. Whenever an event is triggered, it leads to the transition from one state to another state. Also, it is capable of performing actions on transitions and on state entry or exit.

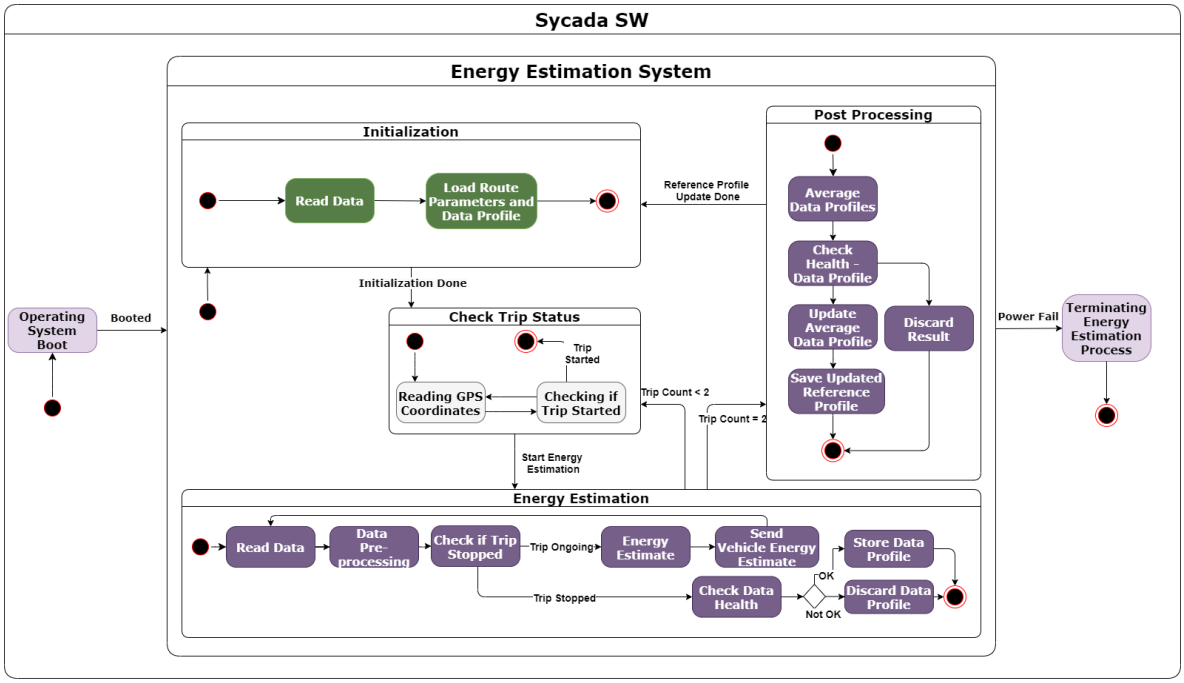


Figure 3.8: State Chart Diagram

In Figure 3.8, it can be observed that upon booting the device, the system enters into the "Energy Estimation System" state. Upon its entry to this state, it is first initialized. Once the initialization is completed with the loading of parameters and data reference profiles, the system jumps into the "Check Trip Status" state. Here, it monitors if the trip has yet started or not. Upon the start of the trip, the system moves to the "Energy Estimation" state, where it performs the desired functions. Upon completing the trip, the system stores the relevant data into memory and checks for the number of trip cycles already done for the selected route. If the number of trip count is less than 2, it enters the "Check Trip Status" state; otherwise, the system enters into the "Post-processing" state. Here, it performs the required activities of updating the reference data profile. As the reference data profile file is updated with the new reference data profile, the system goes to the "Initialization" state and performs the assigned duties to ensure that the "Updated Reference Data Profile" is loaded, and further estimations are occurring using the new profile.

3.6 Summary

In this chapter, the model-based systems engineering approach was used to define the system and make design description more tangible. The concerns of the stakeholders were examined to make a descriptive analysis on the requirements and defining the system specifications. Later, with the help of the descriptive structural and behavioural architectural diagrams a design concept was proposed for the implementation.

4 Theoretical Framework

A crucial aspect in predicting the energy consumed by an electric vehicle is the selection of a suitable model. There exists two categories of energy consumption models: physical models and data-driven models. Physical models are analytical models established on underlying principles of physics. On the other hand, data-driven models are statistical models representing correlations between input parameters and consumed energy using real-world data. Nonetheless, the characteristics of these models is confined to the quality of data. Also, in circumstances when the model is used to anticipate the vehicle behavior outside the range of available data, the accuracy of the model can not be ensured. There are also studies in which the concept of the physical and data-driven model are combined. This idea is adopted for the energy consumption prediction system for electric city buses, where the practiced methodology is principally data-driven but is backed with the model design that is solidly based on physics [18].

This chapter discusses the energy consumption modeling of electric city buses in Section 4.1 and the idea of parameter estimations in Section 4.2.

4.1 Energy Consumption Model

The energy consumption of vehicle is influenced by a number of factors, be it vehicle usage and driving behavior and various environmental factors. Some of the key elements are as follows:

1. Forces acting on the vehicle such as rolling resistance force, aerodynamic force, force due to the road slope, etc.
2. Effective mass of the vehicle and mass due to varying passenger load over the route.
3. Drive-train efficiency.

The equation of motion of a vehicle is used to calculate the energy consumption of the drive-train of the vehicle. The longitudinal forces can be expressed by (4.1).

$$m_{eff} a_x = F_x - F_r - F_{aero} - F_g \sin(\alpha) \tag{4.1}$$

here, a_x is the acceleration of vehicle in (m/s^2); F_x is the propulsion force acting on the wheels in (N). m_{eff} is vehicle effective mass, which is sum of the vehicle mass and equivalent mass of the motor and wheel inertia. The effective mass is commonly 102% of the total mass of the vehicle. F_r is rolling resistance force expressed in (4.2) [19][20].

$$F_r = f_r mg \cos(\alpha) \tag{4.2}$$

here, f_r is coefficient of rolling resistance in $(-)$; g is acceleration due to gravity and α is the road slope. F_{aero} is force occurring due to aerodynamic drag and is expressed in (4.3).

$$F_{aero} = \frac{1}{2}\rho C_d A_f (v - W)^2 \tag{4.3}$$

here, ρ is the air density in (Kg/m^3); C_d is coefficient of aerodynamic drag in $(-)$; A_f is the front area of vehicle in (m^2); v is the vehicle velocity in (m/s) and W is wind speed in the direction of the vehicle in (m/s). The road slope force F_g is expressed as follows:

$$F_g = mg\sin(\alpha) \tag{4.4}$$

Substituting the above equations in (4.1) and rearranging them; the resultant force equation can be used to derive the expression for the power drawn by the drive-train from the battery and is expressed as a function of vehicle velocity (v) and acceleration (a_x) by (4.5).

$$P_{drive} = \frac{v}{\eta}\left[m_{eff}a_x + f_r mg\cos(\alpha) + \frac{1}{2}\rho C_d A_f (v - W)^2 + mg\sin(\alpha)\right] \tag{4.5}$$

here, η is drive-train efficiency.

In most of the research, the vehicle's motion is assumed to be a function of time. Hence, the total energy dissipated can be modeled as the sum of drive-train and auxiliary power integrated over time. In the time domain, the model can be represented by (4.6).

$$\begin{aligned} E(T) = &\int_0^T \frac{v(t)}{\eta}\left[m_{eff}a_x(t) + f_r mg\cos(\alpha)) + \frac{1}{2}\rho C_d A_f (v(t) - W(t))^2 + mg\sin(\alpha)\right] dt \\ &+ \int_0^T P_{aux}(t)\, dt \end{aligned} \tag{4.6}$$

where, $E(T)$ is the total energy consumed at time T for a given route.

The model in (4.6) has a set of vehicle parameters (η,m,C_d,A_f) and other set of environmental parameters (f_r,α,ρ,g). In order to use this model, it is required to accurately determine each of the model parameters, which is a challenging task. It is almost practically impossible; due to the presence of uncertainties in the operation domain of the vehicle, for e.g. variations in weather, road, traffic and payload conditions. Some of the challenges in estimating the parameters is explained below:

1. Rolling resistance coefficient (f_r): This coefficient can vary due to various reasons such as; road type, ambient pressure, ambient temperature, humidity, tire pressures, snow, rain etc.
2. Vehicle mass (m): The mass of the vehicle can change when the nuumber of passengers changes.
3. Road gradient (α): The road gradient depends upon the construction of the road and can be extracted from available online elevation data with additional efforts (like setting up API connections).

After studying the use case of electric city buses, it was observed that making a prediction of vehicle energy consumption for a given route will be difficult if the multi-parameter model was used for estimation. For city public transportation, the route location is always fixed. On the other hand,

driving time taken can vary depending upon traffic conditions or other delays. So, instead of using the driving time based traditional model, it was decided to use a distance-based model in order to produce more reliable energy consumption estimation for a given route with less uncertainties.

Since energy dissipated is estimated with respect to distance traveled, it is logical to implement this method for the estimation of drive-train energy consumption. In this case, only the energy is consumed once the vehicle starts to move. On the other hand, auxiliary peripheral's continue to use the energy from the battery even if the vehicle was standing still. Estimations can thus not be solely based on the distance traveled approach. A hybrid approach will be used in which both aspects of estimation using the distance-based and time-based approach are considered. Energy can be described by forces acting on the body during the displacement or power dissipated over time. Hence, (4.6) was updated as follows:

$$E(T) = \int_0^{s(T)} \frac{1}{\eta} \left[m_{eff} a_x(s(t)) + f_r mg \cos(\alpha(s(t))) + \frac{1}{2} \rho C_d A_f \left(v(s(t)) - W(s(t)) \right)^2 + mg \sin(\alpha(s(t))) \right] ds + \int_0^T P_{aux}(t)\, dt \tag{4.7}$$

the travelled distance $s(t)$ is a function of time t and can be expressed as:

$$s(t) = \int_0^T v(t) dt \tag{4.8}$$

Here, the energy $E(T)$ now depends upon both the distance traveled and time. The velocity and acceleration are interpolated w.r.t. distance. This approach is expected to reduce the uncertainty due to variations in time taken for a given route.

It can be seen in (4.7), all terms except aerodynamic force are mass-dependent. The propulsion force term can be hence normalized with the mass $'m'$, which remains constant during the sections of the trip. This makes (4.7) as:

$$E(T) = \int_0^{s(T)} \frac{m}{\eta} \left[\frac{m_{eff} a_x(s(t))}{m} + f_r g \cos(\alpha(s(t))) + \frac{1}{2m} \rho C_d A_f \left(v(s(t)) - W(s(t)) \right)^2 + g \sin(\alpha(s(t))) \right] ds + \int_0^T P_{aux}(t)\, dt \tag{4.9}$$

The energy consumption prediction is focused on electric city buses and limited to city travel; it is assumed that the impact of aerodynamic drag on the energy consumption estimation is small and can be ignored for this use case. (4.7) can be written in terms of propulsion force F_x and mass terms as follows:

$$E(T) = \frac{m}{\eta} \int_0^{s(T)} \left(\frac{F_x(s(t))}{m} \right) ds + \int_0^T P_{aux}(t)\, dt \tag{4.10}$$

Equation 4.10 standalone can represent the road characteristics with non-vehicular parameters and also the drive-train efficiencies. A new term has been introduced with this equation which is RCI (Route Characteristics Indicator). The RCI is defined as energy consumed per unit distance per unit mass in $(J/(Kg)(m) = Kg(m^2)/(Kg)(m)s^2 = m/s^2)$ giving it the unit of acceleration. So, (4.10) can more easily be expressed as:

$$E(T) = m \int_0^{s(T)} \mathrm{RCI}(s(t))\, ds + \int_0^T P_{aux}(t)\, dt \tag{4.11}$$

The RCI can be calculated from the drive-train power request from the battery and the vehicle velocity for a given route.

$$\mathrm{RCI} = \frac{F_x}{m} = \frac{P_{drive}}{m\, v} \tag{4.12}$$

The drive-train power P_{drive} in the equation can be calculated by measuring the voltages and currents at drive-train terminals. See Figure 4.1.

$$P_{drive} = V_{drive}\, I_{drive} \tag{4.13}$$

The auxiliary power request can be calculated by measuring the voltages and currents at battery and drive-train respectively.

$$P_{aux} = P_{bat} - P_{drive} \tag{4.14}$$

where,

$$P_{bat} = V_{bat}\, I_{bat} \tag{4.15}$$

The initial version of the model was designed by using the real-world data collected from a passenger electric research vehicle (TU/e Lupo EL) [21][22]. The model (4.11) uses the reference RCI for a given route for offline estimations by using the data collected over multiple cycles for the route repeatedly. This part was updated later for making the estimations and corrections over the drive-train energy estimation part for online estimation case (explained in Chapter 5). On the other hand, a passenger electric vehicle withdraws almost a constant amount of auxiliary power over the entire trip for a given route. So a mean value of auxiliary energy measured over multiple cycles was added over the drive-train energy in order to get the overall total energy estimation of the trip for a given route.

After the application of the same model on electric city buses, it was observed from the results that the auxiliary and drive-train power request are not as consistent as passenger electric vehicle over different cycles for a given route. In the case of electric city buses, the drive-train power request is influenced by the weight of passengers being carried for a particular trip on a given route. Several factors influenced the auxiliary request; one of the most dominant is ventilation and air conditioning, which is further dependent upon a number of other factors, such as; number of passengers travelling in the bus and environmental conditions. So, it is proposed to update the mass term for drive-train energy estimation and adding a correction term for the auxiliary energy estimation part that accounts for the above-mentioned influencing factors on auxiliary energy estimation. Both of these terms were subjected to change as the trip progress to make corrections in real-time energy estimation of a given route. Similar to the drive-train energy estimation case, a reference P_{aux} profile was used for a given route [23]. The updated model is described in equation 4.16

$$E(T) = m(s(t)) \int_0^{s(T)} RCI(s(t))\, ds + n(t) \int_0^T P_{aux}(t)\, dt \tag{4.16}$$

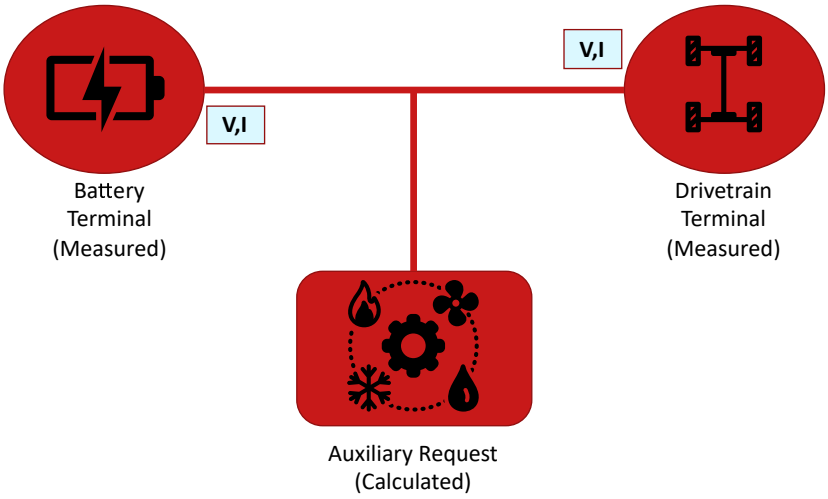


Figure 4.1: Power Measurement Configuration of Electric City Buses

In (4.10) $m(s(t))$ is a coefficient described as a function of travelled distance which is further the function of time, this term not only accounts for the mass variation of the vehicle over the route but also accounts for other environment uncertainties such as rolling resistance, road slope and aerodynamic drag etc. $n(t)$ is coefficient described as function of time that accounts for the variable demand of energy from the auxiliary units.

4.2 Online Identification

Based upon the use-case of the energy estimation model discussed in Section 4.1 on the electric city buses, it is quite useful to have a reference profile to be available on-line when the system is in operation. This on-line available reference model allow the system to make the best predictions for the next outputs and can be regarded as an adaptive prediction method (see Figure 4.2).

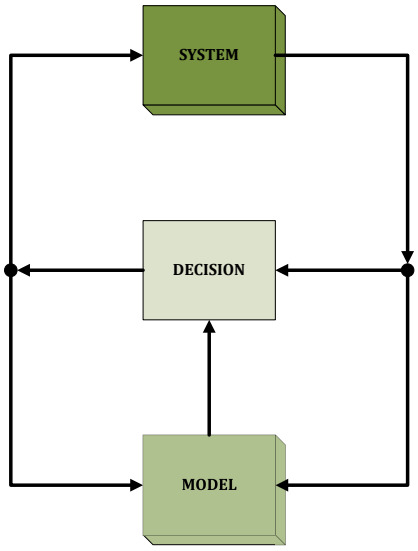


Figure 4.2: Adaptive Methods

The challenge now is concerned with estimating the parameters m and n of the model as the new data is made available during the operation. A typical choice is made, and estimation is performed using recursive algorithms (also known as online, real-time identification, adaptive parameter estimation, or sequential parameter estimation). The algorithm estimates the parameter values at each time step by using currently made observations (measurement data) and using previous parameter estimates. These algorithms are a viable option because they are efficient in terms of memory usage and also posses smaller computational demands [24].

4.3 Online Parameter Estimation

The recursive algorithms used for online parameter estimation can be parted into two categories. The infinite-history algorithm and finite-history algorithm. The infinite-history algorithm aims to minimize the error between the measured and the estimated outputs for all time steps from the beginning. Whereas, the finite-history algorithm aims to minimize the error between the measured and the estimated outputs for a finite number of past time steps. In order to carry out estimation for this particular application; i.e. estimation on mass and auxiliary power correction gain, relevant results were obtained when the complete route data was handled in order to capture different behaviour of energy consumption over the given route

So, the infinite-history recursive estimation algorithm was used for this particular application.

Remark 4.3.1 *The finite-history recursive algorithms are generally easier to tune as compared to the infinite-history recursive algorithms when the parameters have accelerated and possibly considerable alterations over time.*

Recursive Infinite-History Estimation

In the estimations using the least square approach, unknown parameters of a linear model are adapted such that the actual difference between the observed and the computed value stays minimum. In the general form of the recursive estimation algorithm, it follows a set of regression equations that minimizes cost function. For the linear model (4.16) in consideration it translates to finding the parameter that minimizes the cost function (4.17).

$$J(\hat{\theta}, k) = \frac{1}{2} \sum_{k=1}^{k_1} [y(k) - \hat{y}(k)]^2 \tag{4.17}$$

the predicted estimate of the parameter is given by the following equation:

$$\hat{\theta}(k) = \hat{\theta}(k-1) + K(k)[y(k) - \hat{y}(k)] \tag{4.18}$$

where $\hat{\theta}(k)$ is the estimation of parameter at every sample k, $y(k)$ is the observed or measured output at sample k and $\hat{y}(k)$ is the estimation of $y(k)$ based on observations up to samples $k-1$. $K(k)$ gain determines how much the current estimate error $[y(k) - \hat{y}(k)]$ affects the estimate of the parameter. The main idea of the algorithm is to minimize the prediction error term.

The model used to obtain the energy estimation (refer to equation 4.16) is linear, and hence, a linear-regression form of the model is used for online parameter estimation. Thus, the role of the gradient $\psi(k)$ can be expressed by the following equation.

$$y(k) = \psi^T(k)\theta_0(k) + e(k) \tag{4.19}$$

The predicted estimate of the output is given by the following equation:

$$\hat{y}(k) = \psi^T(k)\hat{\theta}(k-1) \tag{4.20}$$

where, $\psi(k)$ is the regressor vector or gradient of the predicted output $\hat{y}(k|\theta)$ with respect to parameters θ, which is computed based upon the previously measured input and output values. $e(k)$ is assumed to be white noise. The definitive form of $\psi(k)$ is determined by the structure of the polynomial model.

The estimation gain K has the following form.

$$K(k) = Q(k)\psi(k) \tag{4.21}$$

The parameters that are needed to be estimated are time-varying, and it is required to keep track of these variations over time. So, the infinite-history recursive estimation algorithm becomes a viable choice. This has different types; forgetting factor, Kalman filter and gradient-based approach to compute the estimation gain $K(k)$ and covariance $Q(k)$ which are discussed further in the next sections.

Forgetting Factor

In some applications where the time-varying parameter estimation techniques are employed, periodic resetting of the estimation schemes can possibly capture the new values of the parameters when the parameter values changes abruptly. When parameters are varying continuously with a slow rate, a different approach can be employed. The concept of a forgetting factor can be a viable approach. In this algorithm, the older data is discarded gradually in favour of the most recent incoming information. In the least square approach, the forgetting factor can be viewed as dispensing less weights to the older data and more the newer data [25]. The cost function takes the following form in this approach:

$$J(\hat{\theta}, k) = \frac{1}{2}\sum_{k=1}^{k_1} \lambda^{k_1-k}[y(k) - \hat{y}(k)]^2 \tag{4.22}$$

λ used in Equation 4.22 is the forgetting factor, with the value in the range $0 < \lambda \leq 1$. This operates as the weight which shrinks for the more outlying data. The difference in using this approach is the update of the correction gain $K(k)$. The equations used in the Forgetting Factor algorithm are summarized below:

$$K(k) = Q(k)\psi(k) \tag{4.23}$$

$$Q(k) = \frac{P(k-1)}{\lambda + \psi^T(k)P(k-1)\psi(k)} \tag{4.24}$$

$$P(k) = \frac{1}{\lambda}\left(P(k-1) - \frac{P(k-1)\psi(k)\psi^T(k)P(k-1)}{\lambda + \psi^T(k)P(k-1)\psi(k)}\right) \tag{4.25}$$

The forgetting factor approach discounts old measurements exponentially such that an observation that is τ samples old carries weight equal to λ^τ times the weight of the current observations. $\tau = \frac{1}{1-\lambda}$ is the *memory horizon* of the algorithm. Typically, the measurements older than $\tau = \frac{1}{1-\lambda}$ carries a weight that is less than 0.3.

Remark 4.3.2 *In the conventional recursive least square approach the covariance matrix $P(k)$ is updated and vanishes to zero with time. This makes the algorithm losing its capability to keep track of the changes in the parameters. However, in the forgetting factor approach, the covariance matrix is divided by λ at every update and allows the slowing down of the fading out of the covariance matrix. Also, the algorithm must ensure that the covariance estimator must be kept bounded in order to assure that the tracking error will be bounded. More rigorous mathematical arguments can be found in [26]*

Remark 4.3.3 *The Forgetting Factor algorithm becomes a Kalman Filter if $\lambda = 1$ with $R_1 = 0$ and $R_2 = 1$*

Kalman Filter

Kalman filter, also called as Linear Quadratic Estimator (LQE), is essentially a set of mathematical equations which implements a predictor-corrector type estimator, these estimations are optimal as it minimizes the error covariance when certain conditions are met. It is a method that uses the sequence of measurements observed over time, that accommodates statistical noise and other inaccuracies. This algorithm yields estimates of unknown variable that conduces more accuracy than those based upon single measurements. These estimates are made by using a joint probability distribution over the variables for each time-frame.

The Kalman filter handles a process by using a feedback: the filter estimates the process parameter at certain sample and obtains feedback in the form of measurements. The related algorithms for the filter fall under two categories: sample update equations and measurements update equations. The sample update equations are usually responsible for projecting ahead in sample, the current parameter and its error covariance estimation to realize $a\ priori$ estimate for the next sample step. The measurement update equations ensures for the feedback - i.e for incorporating a new measurement into the $a\ priori$ estimate to retrieve an improved $a\ posteriori$ estimate. The estimation algorithm hence resembles of a predictor-corrector algorithm.

The equations used in Kalman filter adaptation algorithm are summarized to compute the Kalman gain $K(k)$ as in (4.21).

$$K(k) = Q(k)\psi(k) \tag{4.26}$$

$$Q(k) = \frac{P(k-1)}{R_2 + \psi^T(k)P(k-1)\psi(k)} \tag{4.27}$$

$$P(k) = P(k-1) - \frac{P(k-1)\psi(k)\psi^T(k)P(k-1)}{R_2 + \psi^T(k)P(k-1)\psi(k)} \tag{4.28}$$

In the prediction step the Kalman filter projects the parameter and error co-variance ahead in sample. It then performs a correction step and computes the Kalman gain. It then updates the estimates with measurements $y(k)$ and finally updates the error co-variance $P(k)$. This process is repeated with

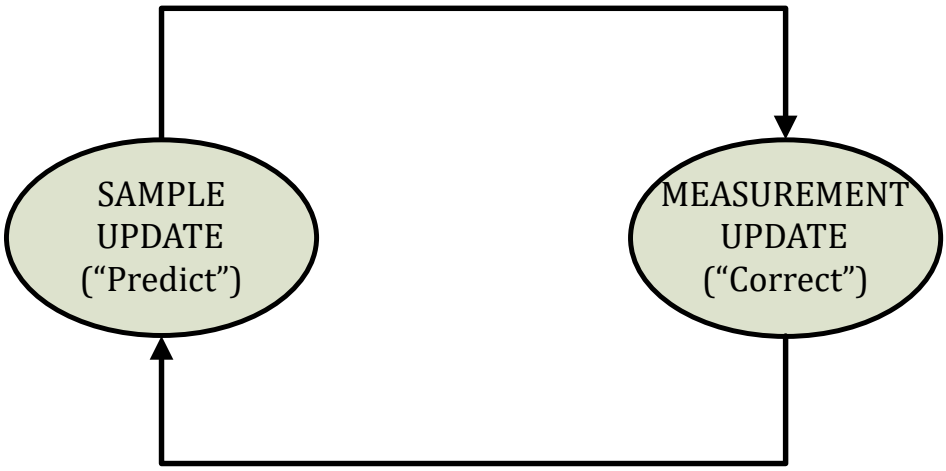


Figure 4.3: Kalman filter cycle. $Sample$ update projects the current state estimate before the actual measurement is available. The projected estimate is later updated by $measurement$ update.

previous $a\ posteriori$ estimate used to project new $a\ priori$ estimate. This recursive nature of Kalman filter makes it useful for practical estimation [27].

It is ensured that the $P(k)$ computed is a positive-definite matrix.

Assumption 1 *$P(k)$ is computed assuming that the residuals (the difference between estimated and measured output) are white noise, and the variance of these residuals is* 1.

The R_1 and R_2 in Kalman filter algorithms are the process error co-variance (co-variance matrix of parameter changes) and measurement error co-variance (variance of residuals).

Assumption 2 *$Q(k)$ computed in Equation 4.27 to calculate the Kalman gain assumes that the parameters $\theta_0(k)$ are described by the random walk.*

$$\theta_0(k) = \theta_0(k-1) + w(k) \tag{4.29}$$

where $w(k)$ is Gaussian white noise with the covariance matrix R_1 as:

$$E[w(k)\,w^T(k)] = R_1 \tag{4.30}$$

R_2 is the variance of the $e(k)$

$$y(k) = \psi^T(k)\theta_0(k) + e(k) \tag{4.31}$$

Assumption 3 *It is assumed that R_1 and $P(0)$ matrices are scaled such that $R_2 = 1$.*

Normalized and Unnormalized Gradient

In the linear regression case, the gradient method is also known as the *least mean square* (LMS) approach. The only variation in this case from the above-discussed cases is in the computation of the gain used for estimation of the parameters. The gain Q used for computation of estimation gain K in Equation 4.21 uses the following form in case of unnormalized gradient case:

$$Q(k) = \gamma \tag{4.32}$$

and in case of normalized gradient case:

$$Q(k) = \frac{\gamma}{|\psi(k)|^2 + Bias} \tag{4.33}$$

The normalized gradient algorithm is used to scale the adaptation gain, γ, at every step by the square of the two-norm of the gradient vector. In case if the gradient vector is closer to zero, a bias term is introduced in the scaling factor to avoid uncertain jumps in the estimated parameters [24].

4.4 Summary

The work established in this chapter provides a novel approach for modeling the energy consumption of the electric vehicles. This involves introduction of a new concept for estimation of the drive-train energy consumption using the Road Characteristics Indicator. An approach which requires the inclusion of minimal number of the parameter variables to be introduced in the energy modeling concept. This ensures the inclusion of self adapting estimation policy feasible to be applied to have predictions from the proposed model.

5 Design Implementation

The key objectives of having a model-based system engineering concept, as discussed in Chapter 3 are as follows:

- Identification and derivation of required system functions.
- Identification of associated system modes and states.
- Allocation of the identified system functions and modes to a subsystem structure.

In Chapter 4, the theoretic framework was developed. Using the knowledge gained from both these subjects, the design of the system was implemented in Matlab/Simulink. This chapter will give the reader a description about the implementation of the proposed design.

5.1 Offline Energy Estimation

In some research, emphasis is put on accurately measuring the energy estimate by using the driving cycle and vehicle parameters (mass, rolling resistance coefficient, air drag coefficient, active frontal area), gradient of the road, and vehicle speed relative to air. Although some of these parameters can be determined relatively easy, others are more strenuous to assess and show variation in the operation domain of the vehicle. This makes the energy estimation for electric vehicle a challenging task. Moreover, for an electric city bus, the load will vary during a daily trip on a given route; this makes it even more difficult to accurately predict the energy consumption by these vehicles. Instead of estimating many parameters, this research is limited to estimate only two correction parameters, as explained. This part of the system development are the building block for the development of a real-time energy consumption prediction system.

In this section the a procedure will be discussed to prepare the reference RCI profile with pre-processing the data stored offline from the multiple trip cycles of a city electric bus running on a given route. The data was collected from the beginning of the route to the end of the route, and a database was established from this historical collection of data. A list of the most important signals required to maintain this database includes; time, GPS (latitude, longitude, and heading), vehicle speed, battery voltages and currents, and the drive-train voltages and current. The data for these signals were collected from the vehicle CAN-Bus. The sampling frequency of the data collected for the establishment of the system feature was 10 Hz. Using the reference RCI and Power profile obtained using the data, the initial estimations on the energy consumption can be made at the offline estimation stage.

Data Pre-processing

The data used in the research and development of the real-time energy estimation system has been collected from a city electric bus, which was in operation in the city transportation setting. To create the profiles from the pool of the data collected over various routes in the city, a particular route data was extracted using the selected route's GPS coordinates. The time-based data was interpolated to the distance-based data.

- **Route Range:** The GPS coordinates of the starting bus stop and the final bus stop was extracted from the Google Maps. These two coordinate points were used to extract the data in between the two locations by using a function "*idxStartStop*" developed in the software library of the system. This function uses the pre-allocated parameters of GPS coordinates (Start and Stop location) to extract the relevant data, See Figure 5.1 for the route description. To ensure the robustness of the functionality of this function, a two-step coordinate verification was developed while giving an uncertainty range around the given GPS coordinates to raise Start Route and Stop Route flags. The uncertainty range was kept to 0.0004 in latitude and longitude (which means a range of about 40 m in the distance). The uncertainty range allows a fail proof functionality of this function. It ensure the precision of the route start and stop identification by the system. The range was selected after performing the tests on the available data.

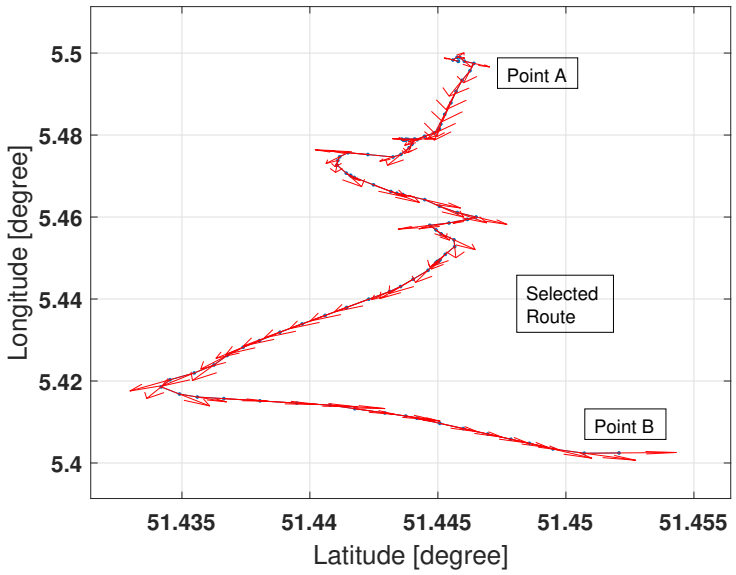


Figure 5.1: Selected Test Route: GPS coordinate map

- **Re-sampling Data:** The data collected from the CAN-Bus has the total distance travelled values varying from segment to segment (for different trip cycles) due to GPS operating errors or driver behaviour of using the bus. An averaged total route distance was calculated from the collected data. Next an equi-sampled distance data array was created with which the other data signals were interpolated. This technique ensured the identical number of sampling points for the stored data arrays in the reference data structure.

Reference RCI and Auxiliary Power Profile

For a given route, the RCI can be determined for every trip using (4.12). The RCI profile obtained was re-sampled to the travelled distance domain. Similar actions were employed to create the power profile for the auxiliary estimations. Averaging of multiple individual RCI and power profile has been done to obtain the averaged reference profiles.

To create the averaged reference profile, a minimum number of trip is required for the selected route to capture different dynamics. The determination for minimum trip cycle size came from the evolution of root-mean-square error (RMSE) of the latest profile with the chosen baseline profile. The primary deciding factor is the RCI profile and not the auxiliary power profile due to the dominance of the contribution from the RCI profile in the total energy consumption. The RMSE between the newest RCI profile and the baseline RCI profile is described by (5.1).

$$RMSE_{RCI} = \sqrt{\frac{\sum_{i=1}^{n}(RCI_{New} - RCI_{Base})^2}{n}} \tag{5.1}$$

where n is number of data points in the RCI profile, RCI_{New} is the averaged RCI profile created using multiple data profiles. RCI_{Base} is the baseline profile which can be defined as the selected route trip reference profile.

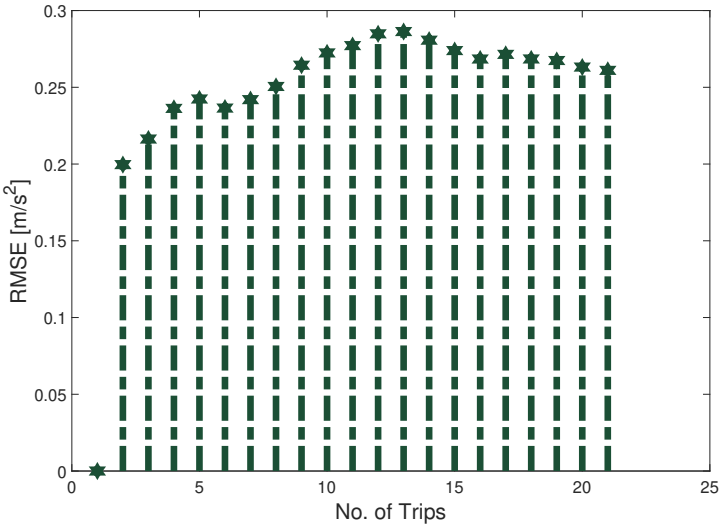
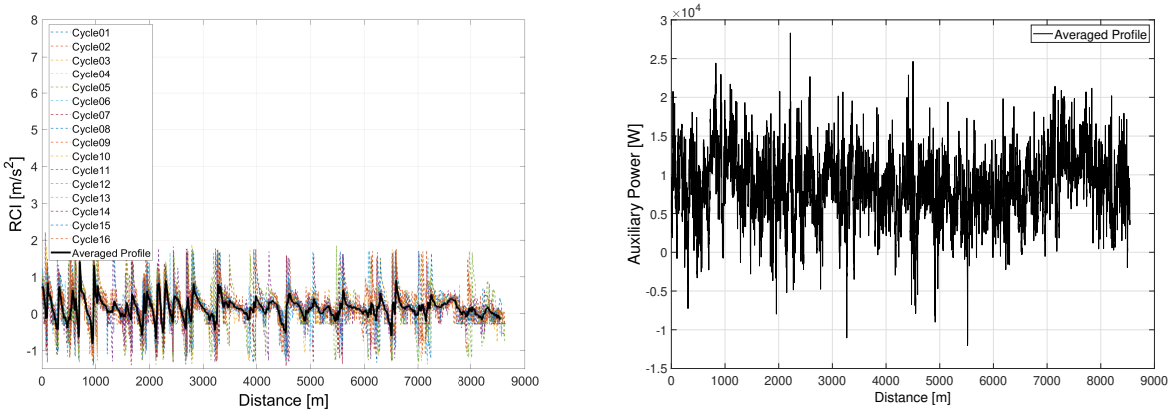


Figure 5.2: Evolving of root-mean squared error (RMSE) with the progression of data from No. of trips

It can be observed from the Figure 5.2 that as the number of trips increases the RMSE value goes up. It can also be observed that it became fairly steady after the 16 trip cycle data, which can be considered as the inflection point for the RMSE tendency for variation in the value. Therefore, as the RMSE becomes consistent after 16 trip cycles, it was concluded that 16 trip data cycle would be sufficient to create the offline reference profile (AKA Base Data Reference Profile, which include both the averaged RCI and auxiliary power profile and other relevant data (check Remark 3.5.1)). The averaged reference data profile will be sufficient to capture the different dynamics of the model and will be suitable to perform estimations.

Offline Energy Estimate

With the availability of the averaged reference profile as shown in Figure 5.3, the offline energy consumption can be estimated.



(a) Averaged RCI Profile

(b) Averaged Auxiliary Power Profile

Figure 5.3: Base Data Reference Profile computed offline using 16 trip data cycles.

The total energy estimation for the selected route using the base data reference profile is shown in Figure 5.4 with a varying range of ± 2.5 kWh

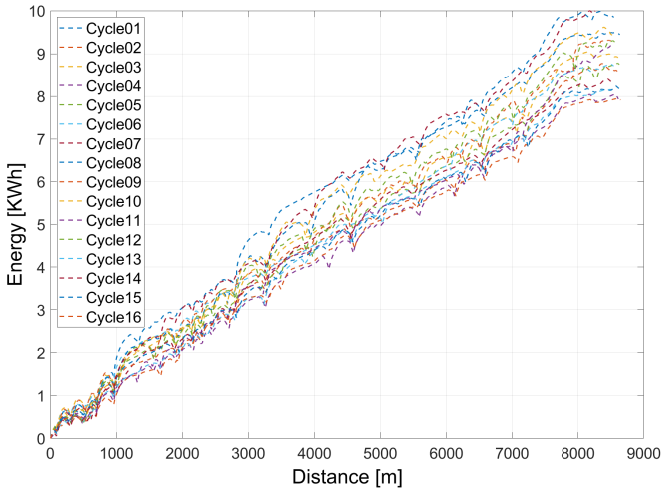


Figure 5.4: Total energy consumption from 16 cycle data

While estimating offline, the correction term, which is the mass of the bus, remains constant. This forces the estimated total energy consumption for the investigated route to stay fixed for the entire route.

5.2 Online Energy Estimation

To improve the accuracy of the energy consumption prediction, it was required to update the model parameters in real-time as the bus is driving along the route. The parameters of the model in real-time (online) estimation algorithms were updated as the new measurements are available. Two parameters needed to be updated in real-time were mass-estimate for drive-train estimation and auxiliary power correction gain-estimate for auxiliary estimation. (refer to equation 4.16). All other influencing factors on energy estimation were considered as perturbations [28].

In Section 4.3, various methods are discussed to perform the online estimation. These algorithms were developed and deployed for this application. The linear Kalman filter approach was chosen for the final deployment because of its superiority and its fit with the application. The rationale for choosing this approach is provided in the Table 5.1 below:

Table 5.1: Rationale for selection of type of Online Parameter Estimation

Algorithm Type	Pros	Cons
Gradient based	Simple implementation	Correction gain remain constant
	Less memory and processing required	Less Robust
	Absolute error is ≈ 5%	
Forgetting factor	Adaptive gain estimation based upon error co-variance	Gain estimation is sensitive. Better for system with quick changing dynamics in which last few measurements are critical
		Absolute error is ≈ 8%
Kalman filter	Adaptive gain estimation based upon error co-variance	Relatively complex to implement
	Absolute error is < 2%	
	More robust results (tested on limited data)	
	Better estimation on overall historical data	

The Kalman Filter uses various equations which are summarized in Section 4.3

Remark 5.2.1 *Applying the terminology of the Kalman Filter to this application in discrete time, the regressor vector* $\psi(k)$ *is* $\sum_0^k RCI(k)$ *and parameter vector* θ *is* $m(k)$ *for drive-train estimation and correction. While for auxiliary estimation and correction, the regressor vector* $\psi(k)$ *is* $\sum_0^k P_{aux}(k)$ *and parameter vector* θ *is* $n(k)$.

To use the Kalman recursive algorithms, an initial value is required. In the drive-train energy consumption estimation and correction algorithm, the parameter vector $\hat{\theta}(0)$ is initialized with the unladen mass of the vehicle (which is 19000 Kg) and covariance matrix $P(0)$ that indicates the parameter error was initialized with the identity matrix. Furthermore, in the auxiliary energy consumption estimation and correction algorithm, the parameter vector $\hat{\theta}(0)$ was initialized with 0.1 (which was sufficient for the slowly varying linear energy demand) and covariance matrix $P(0)$ that indicates the parameter error

was initialized with the identity matrix. The algorithm always ensure that the covariance matrix $P(k)$ was a positive-definite matrix by using a square root algorithm to update it. [29]

This allows to tune the algorithm with only one tuning factor, R_1. In the case of drive-train estimation, it was observed that the consumption of the energy over different cycles could vary with the range of uncertainty of 2.5 kWh, the value of the R_1 was set to $2.5 * 10^3$ to giving an uncertainty range to the algorithm for the predictions. It also meant that the process variance allowed was larger, and actual measurement will be trusted more than the predicted value. This means that the estimates will deviate away less from the actual measured value. On the other hand, this value cannot be kept too large to prevent the algorithm's trust from shifting significantly towards the noisy measurements. The auxiliary energy consumption profile remain relatively constant and linear for different trip cycles except for some exceptional trips. So, the process variance allowed was kept smaller; in this case, $R_1 = 10^{-5}$, and the algorithm was pushed to trust the predicted value more than the actual measurements.

Online Energy Estimates: Flow Chart of Algorithm

In Figure 5.5, a flow chart of the implemented algorithm is shown to explain the functionality of the estimation algorithm itself. While the system software was receiving the CAN-Bus data, it was being processed by the algorithm. Once the GPS coordinates come closer to the start coordinates, the *startRoute* flag was raised, acknowledging the start of the trip. The algorithm reset the estimation signals and starts predicting the new estimates. When the bus has traveled 100 m, the mass estimations start to update using the recursive algorithm approach. Once the bus was closer to the stop coordinate and the *stopRoute* flag was raised or the data samples in the reference profile end, then the algorithm stops the estimation.

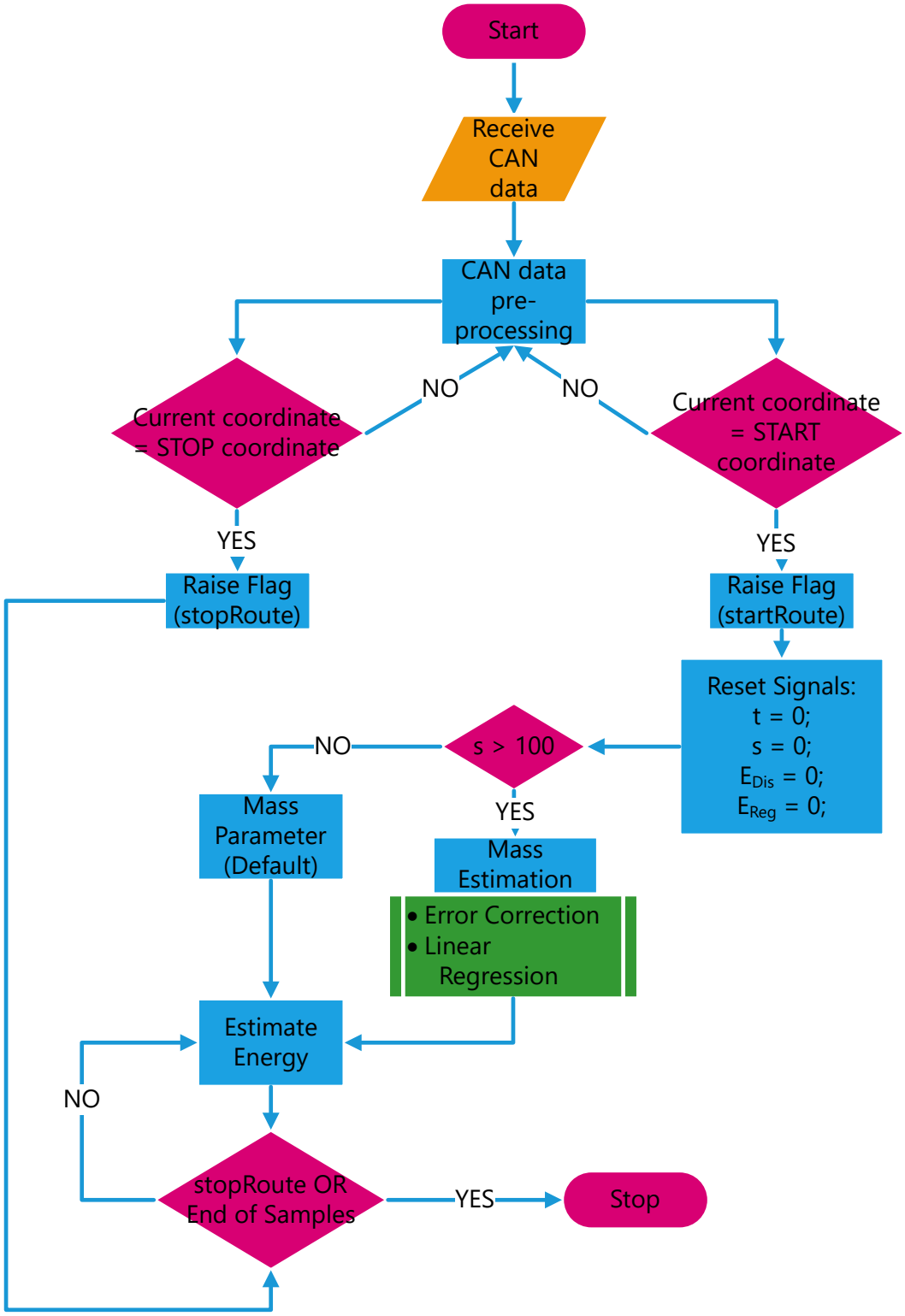


Figure 5.5: Flow Chart: Online Energy Estimation Software Algorithm

5.3 System Checks

To ensure reliability and debug system errors it was decided to accommodate the energy estimation system with some checks. These checks are as follows:

- **CAN-Bus Data Check:** The CAN-Bus data check ensure that the data received is in good health. Once the defined conditions for this check are met, *"CANdataReceivigOK"* check flag is raised.
- **Data Pre-processing Check:** This check ensures that the pre-processed data before going to estimation algorithm remains in good health. When the design criteria is met, *"DataPreprocessingOK"* check flag is raised in the system software.
- **Estimations Check:** Once the system has started doing estimations. A criteria check is done to ensure that estimates are in good health and respective flags (*"EstimationActiveOK"*, *"EstimationDrivetrainOK"*,*"EstimationAuxOK"*) are raised in the system software.
- **Reliability Check:** On meeting the defined specification set for reliability (> 80%) the *"reliabilityOK"* check flag is raised.
- **Post-processing Checks:** Various conditions are introduced to check health of the reference profiles generated by the system software. These profiles when only in good health are stored in the system memory. The check flags raised in software are *"postProcessingOK","tripDataHealth"*

These conditions and criteria are such as the data signals must not be NaN (Not a Number) as the estimations are calculated, the data signal can not be zero, if the vehicle is moving ($v_x > 0$) and the frequency of these data signals must be $\approx$ 10 Hz. [More specific information on these conditions and criteria is available in Appendix A and Appendix E]

5.4 Summary

This chapter discusses the implementation of the concept proposed in Chapter 3 using the algorithms discussed in Chapter 4. In this chapter a detailed description was given to prepare the reference data profiles for the proposed algorithms and descriptive rationale for some choices made for the design during the implementation phase. This chapter also includes very first results from the offline algorithm produced, which later complemented the functionality of the online algorithms in the energy estimation system.

6 Validation and Results

The focus of this chapter is on the validation of the system design and ensuring a working prototype by the means of simulations. The test scenarios and results are discussed in this chapter.

6.1 Simulation Software Setup

To start developing the real-time energy consumption prediction system for the electric vehicles, the development was started at the prototype level using simulations. The system functionality requires some pre-collected data from the vehicle over the given route. For the purpose of collecting data from the vehicle, and an ARM-based embedded device is installed on the vehicle. This device has access to CAN-Bus and GPS data of the vehicle. The device logged the data of the vehicle's entire motion for a couple of weeks, which includes all relevant charging and discharging data. From the data logged, the appropriate segments are extracted associated with the selected route. It was observed that 21 trips of the selected route could be extracted from the logged data, which was sufficient to begin with the development. Since this application require the estimation of parameters using the data, it is necessary to categorize the available data into training data and validation data.

Remark 6.1.1 *The training data has been used to build the base reference model and tuning the parameters of the algorithm. Whereas on the other hand, validation data was used to provide an unbiased evaluation of the final results from the estimation and correction algorithm. The validation data has never been used in the training of the algorithms. By the rule of thumb, 70 % of total data collected was allocated to training, which was approximately 16 cycle data, this fortunately matches up with the decision of choosing 16 data cycles for preparing base data reference profile using RMSE criteria as was explained in Section 5.1. The rest of the 30 % data, which was the remaining 5 cycle data, was used as validation data.*

The Matlab scripts are written to further process this data. A dedicated library and a Simulation setup are prepared to emulate real-time simulations of the scenarios. The final estimation results from the emulated scenarios were called back into the Matlab script to perform the post-processing and data visualization and analysis. The visual description of this software setup can be seen in Figure 6.1

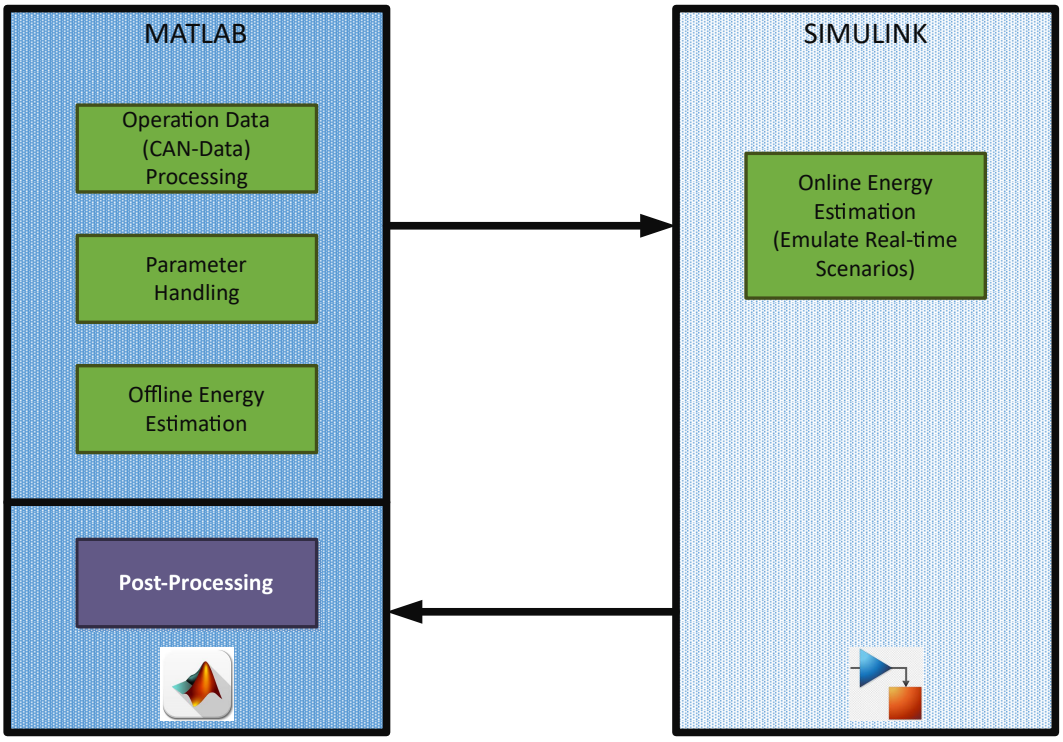


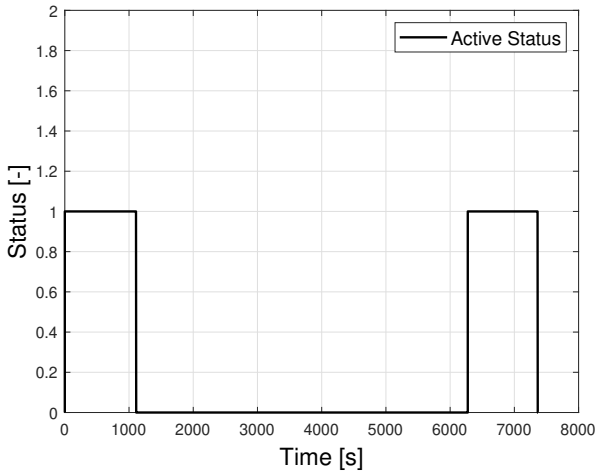
Figure 6.1: Simulation Software Setup: MATLAB-Simulink

6.2 Test Scenarios

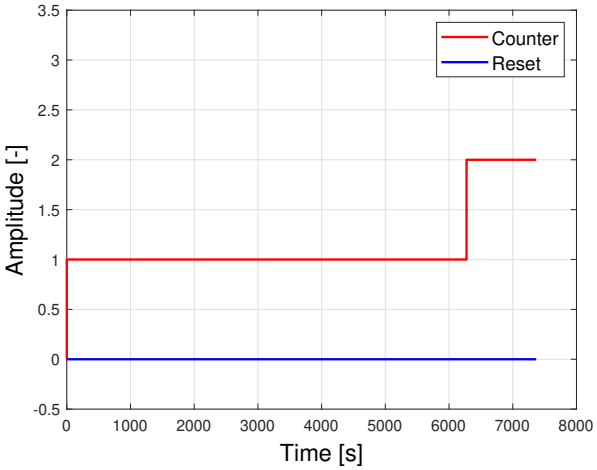
Three test scenarios were designed to check the overall functionality of the real-time energy consumption prediction system.

Test Scenario 1: Estimations using Training Data Set 1

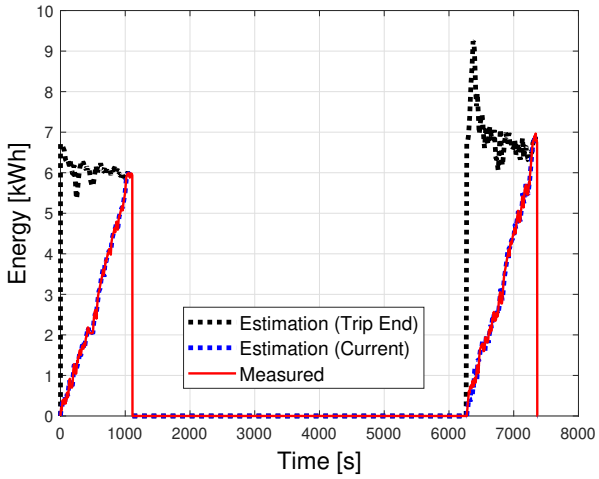
In this test scenario, the real-time energy consumption prediction system was simulated with the first two cycles of the training data (called as training data set 1). The base reference profile was used for estimation and corrections. It can be observed from figure 6.2a, that the system identifies when to activate the real-time estimation. Here 1 means the system is active (calculate energy estimates) and 0 means it is inactive. The two-cycle data was simulated together which means that electric city bus travels on the selected route two times. This scenario is captured by trip counter in Figure 6.2b This has been clearly identified by the system, and energy estimations are made. The system does the drive-train and auxiliary energy estimation separately, which can be seen in Figure 6.2c and 6.2d respectively. It can be observed that the estimations are tracking the actual measurements. The corresponding correction parameters evolution (m, mass estimation and n, auxiliary power correction gain estimation) can be observed in Figure 6.2e and 6.2f respectively. From these results it can be noticed that the estimations algorithms are capable to perform the tracking of the actual measurements efficiently and with high accuracy.



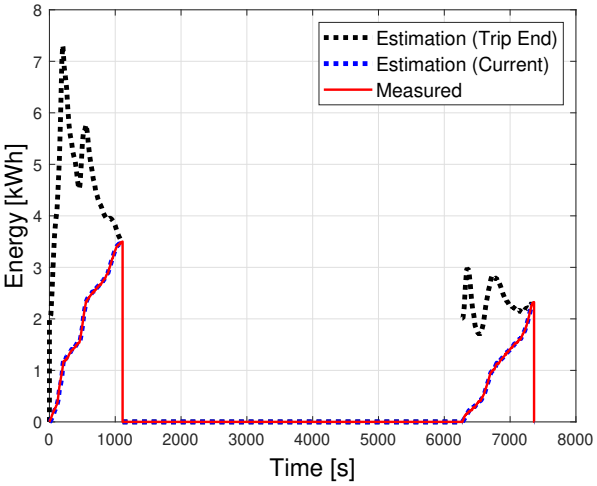
(a) Active Status



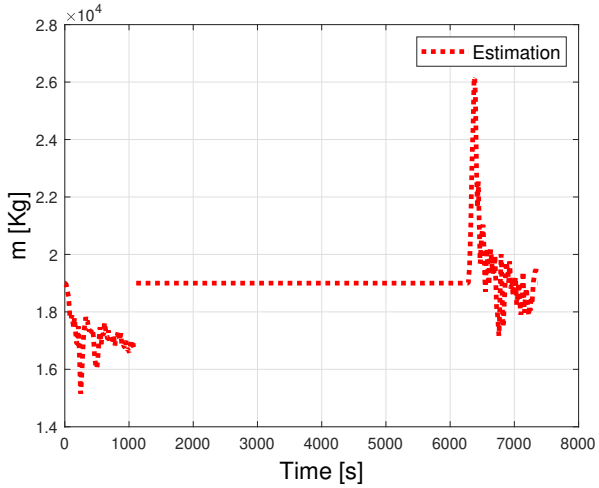
(b) Route Count



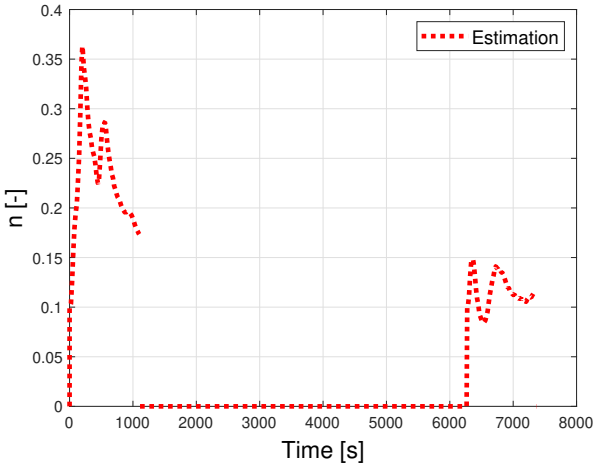
(c) Drive-train Online Energy Estimation



(d) Auxiliary Online Energy Estimation



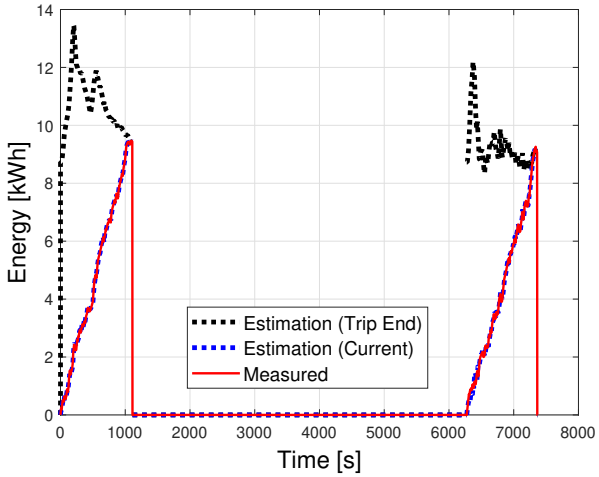
(e) Mass Estimation



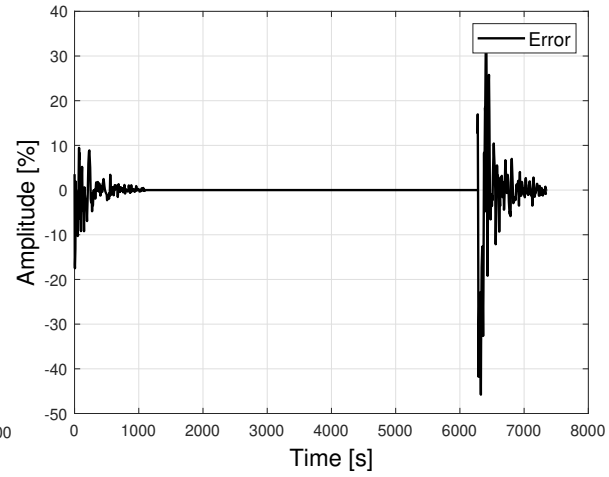
(f) Correction Gain Estimation

Figure 6.2: Simulation Results: Test Scenario 1 (Estimations using Training Data Set 1)

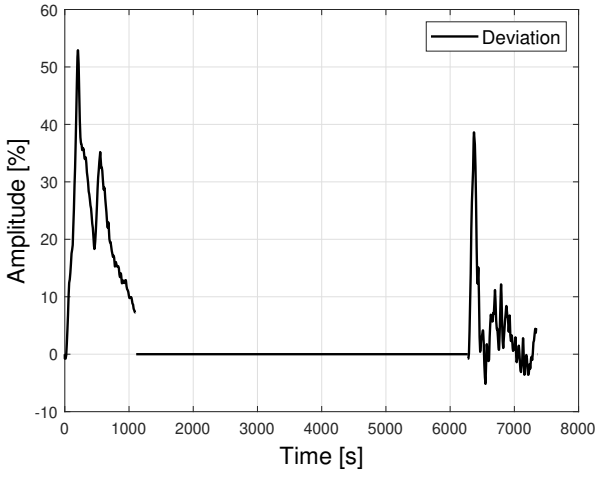
The total energy consumption estimation from drive-train and auxiliary estimation & correction algorithms can be observed in figure 6.3a. The error between the estimated value and actual measured values can be observed in figure 6.3b; it can be seen that as the more data was getting available, the resulting error starts decreasing and eventually becomes bounded around zero. It was also important to observe the reliability of the energy estimation and correction algorithm, which was based upon the evolution of the error. To trust the predicted energy estimates, it was important to know if the estimation's reliability (See Remark 6.2.1) was high or not. It can be observed that the reliability stays above 80% for most part of the trip, especially at the later part of the trip when more data was available. This can be observed in figure 6.3d. This means that more trusted energy estimates can be expected from the system as the trip progresses. Another important aspect was to analyze the performance of real-time estimation compared to the the offline estimations for a particular trip cycle of the selected route. The importance of real-time estimation in this particular application can be measured by using a term *deviation from Offline Estimation*, which at any particular time gives an idea about the deviation of the current estimates from the initial estimate, made in the beginning of the trip.



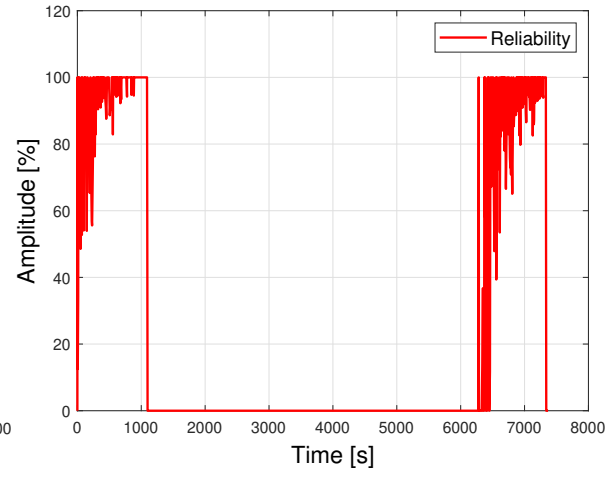
(a) Total Energy Estimation



(b) Error in Estimation



(c) Deviation from the Offline Estimate



(d) Reliability of Estimation

Figure 6.3: Simulation Results: Test Scenario 1 (Estimations using Training Data Set 1)

Remark 6.2.1 *The reliability of the estimations are calculated based upon the assumption that the maximum allowable instantaneous error between estimations and measurements is 20 %. If the estimation error is greater than 20 % than the reliability is assumed to be zero. Otherwise, the reliability is scaled with respect to error using the formula:*

$$Reliability(\%) = 100 - |Error| * 5 \tag{6.1}$$

The error is calculated using the following formula:

$$Error(\%) = \frac{Estimation - Measurement}{Measurement} * 100 \tag{6.2}$$

where, Estimation refers to calculated current energy estimates form the algorithm and Measurement refers to the actual energy measurements from the CAN-Bus

Remark 6.2.2 *The Deviation from the offline estimate is calculated using the following formula:*

$$Deviation(\%) = \frac{Estimation_{Online} - Estimation_{Offline}}{Estimation_{Offline}} * 100 \tag{6.3}$$

The online and offline estimations are total predicted energy estimation for the entire trip during the trip and before the trip respectively.

It is also essential to calculate and record the updated (real-time) *RCI* profile using the power, velocity, and mass estimates of the drive-train estimation (shown in figure 6.4) using the Equation 4.12. This profile will later be processed in the post-processing step and will be used to generate a new updated reference profile for the energy estimation system.

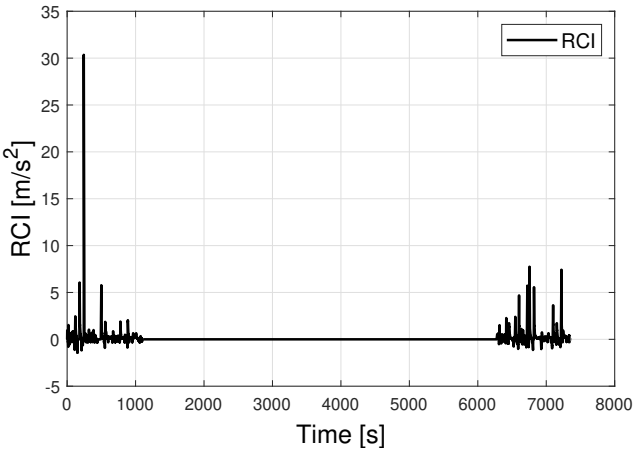
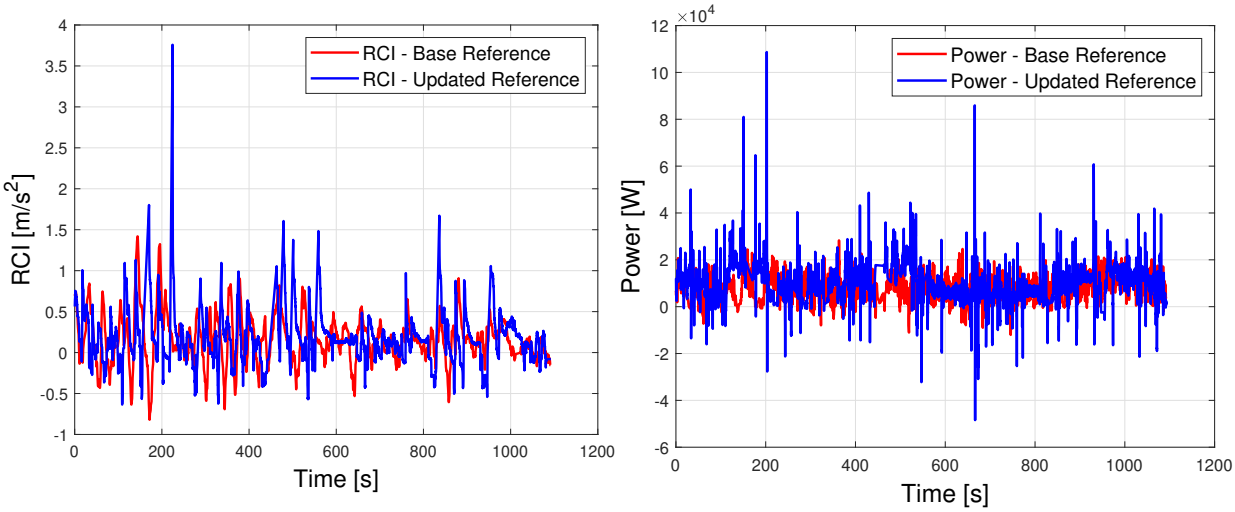


Figure 6.4: RCI Profile generated in real-time as mass estimation is updated.

The result of *Test 1* generate an updated reference profile (averaged profile). Now, when the real-time energy estimation system runs next time over the selected route, then instead of using the base reference profile, the updated reference profile is used. This will ensure the adaptability of the energy estimation system towards changes in the environment, weather, traffic, road conditions, and also the commuters traveling trend, etc.



(a) RCI Profile

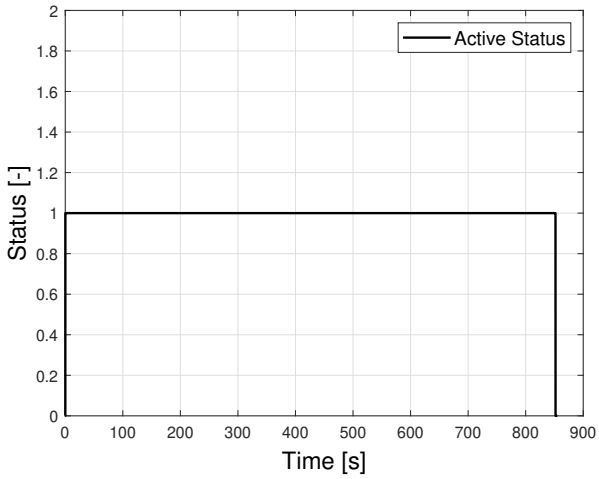
(b) Power Profile

Figure 6.5: Comparison between base and updated RCI and Power Profile

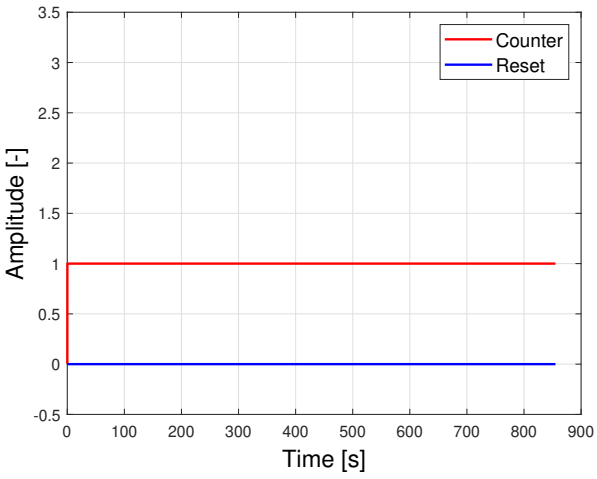
Test Scenario 2: Estimations using Training Data Set 2

In this test scenario the real-time energy consumption prediction system has been simulated with the different cycles of the training data. The updated reference profile was used for estimation and corrections. It can be observed from figure 6.6a, that the system identifies when to activate the real-time estimation and correction function. The system does the drive-train and auxiliary energy estimation separately which can be seen in figure 6.6c and 6.6d respectively. The corresponding model parameters evolution can be observed in 6.6e and 6.6f respectively.

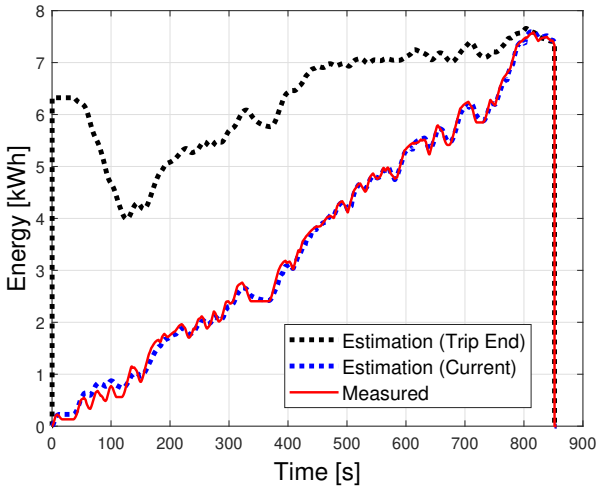
The combined energy consumption estimation from drive-train and auxiliary estimation & correction algorithms can be observed in figure 6.7a. The error between the estimated value and actual measured values can be observed in figure 6.7b; it can be seen that as more data was getting available, the error starts decreasing and eventually becomes bounded around zero. The reliability can be observed in figure 6.7d. The deviation from the initial estimate can be observed in figure 6.7c



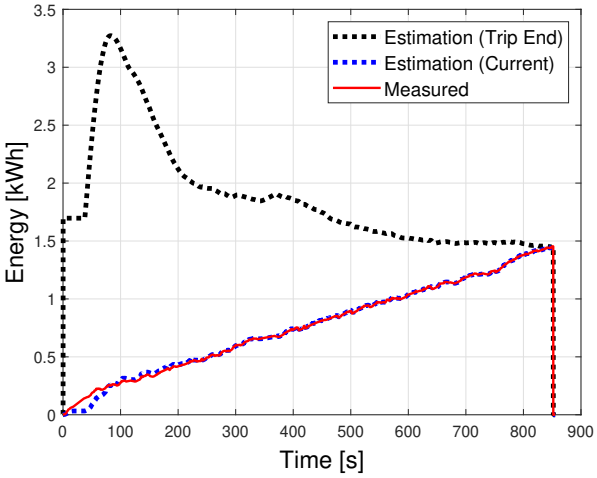
(a) Active Status



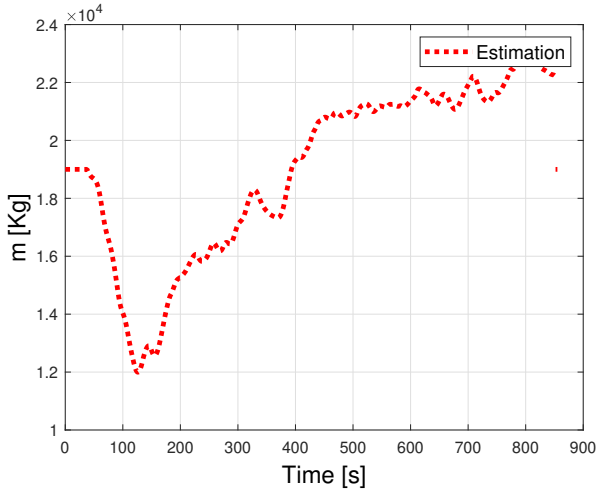
(b) Route Count



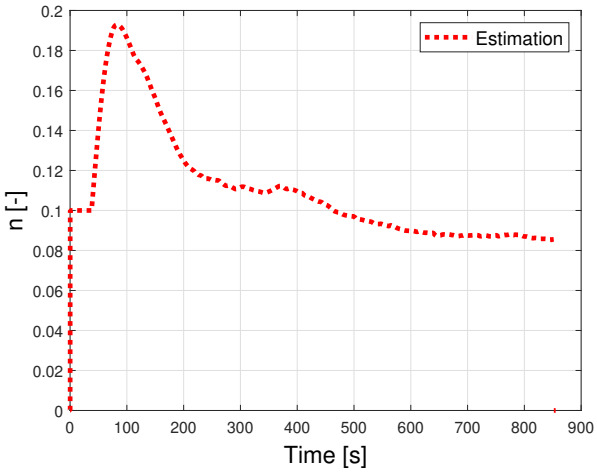
(c) Drive-train Online Energy Estimation



(d) Auxiliary Online Energy Estimation

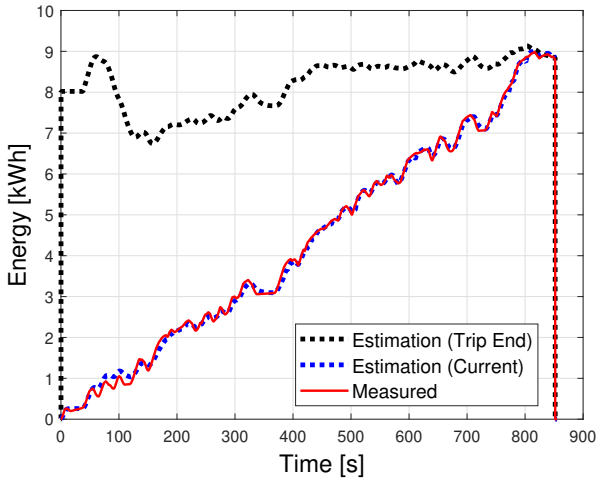


(e) Mass Estimation

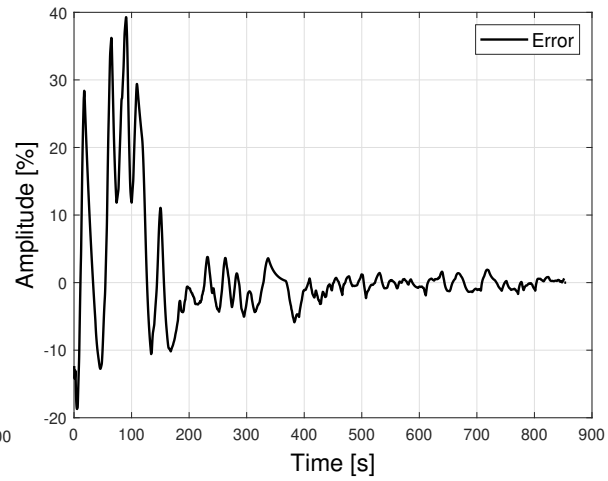


(f) Correction Gain Estimation

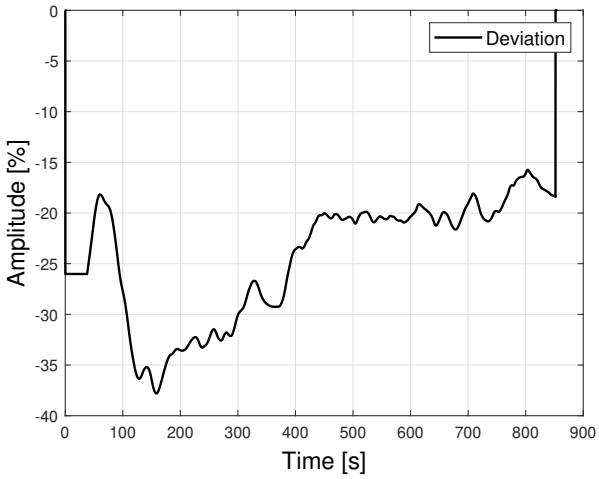
Figure 6.6: Simulation Results: Test Scenario 2 (Estimations using Training Data Set 2)



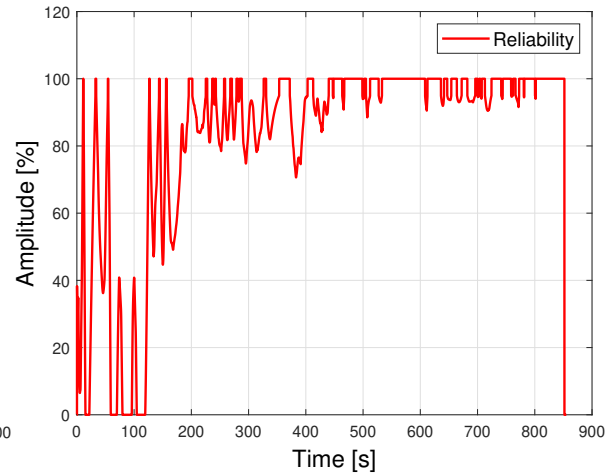
(a) Total Energy Estimation



(b) Error in Estimation



(c) Deviation from the Offline Estimate



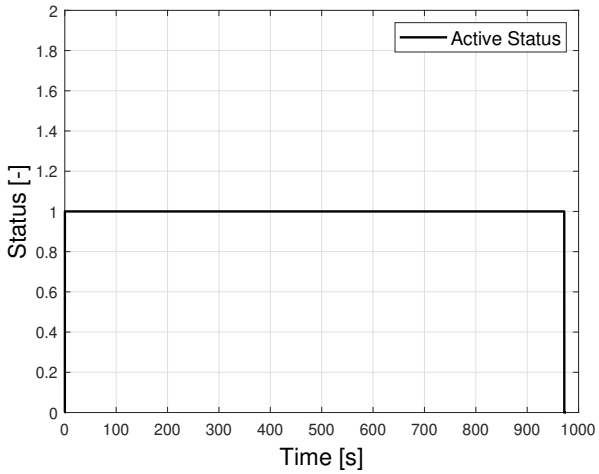
(d) Reliability of Estimation

Figure 6.7: Simulation Results: Test Scenario 2 (Estimations using Training Data Set 2)

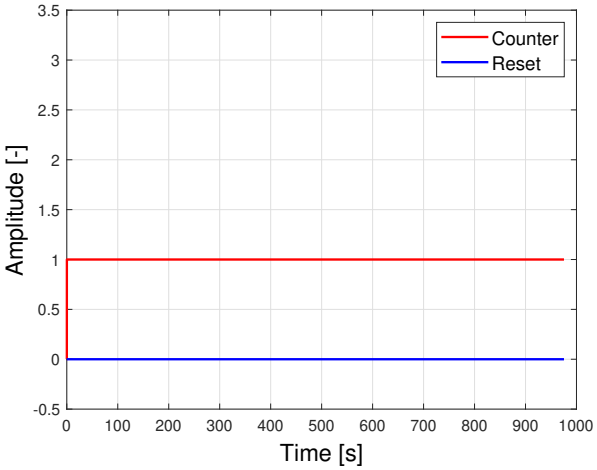
Test Scenario 3: Estimations using Validation Data Set 1

In this test scenario, the real-time energy estimation system was simulated with the validation data. These data cycles have not been seen by the estimation and correction algorithm during the development of the system and were kept solely to perform the software-in-the-loop (SIL) testing. The updated reference profile was used for estimation and corrections. The active system status can be observed in figure 6.8a. The system does the drive-train and auxiliary energy estimation separately which can be seen in figure 6.8c and 6.8d respectively. The corresponding correction parameters evolution can be observed in 6.8e and 6.8f respectively.

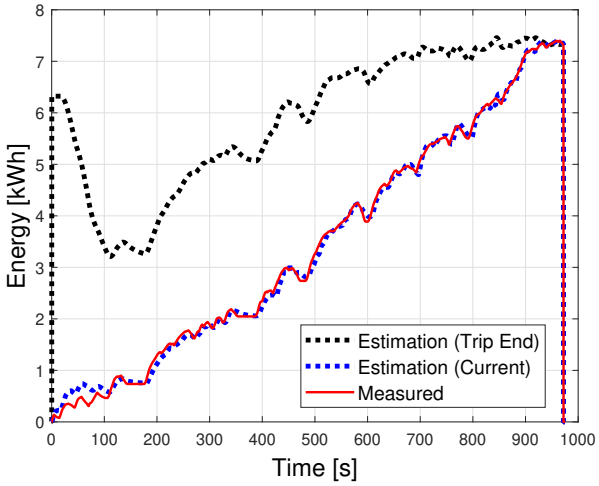
The total energy estimation from drive-train and auxiliary estimation & correction algorithms can be observed in figure 6.9a. The error can be observed in figure 6.9b. The reliability can be observed in figure 6.9d. The deviation from the initial estimate can be observed in figure 6.9c. The result appear to follow the similar trends as from the previous tests ensuring the robustness of the estimation algorithms towards different data sets representing different operating and environmental conditions.



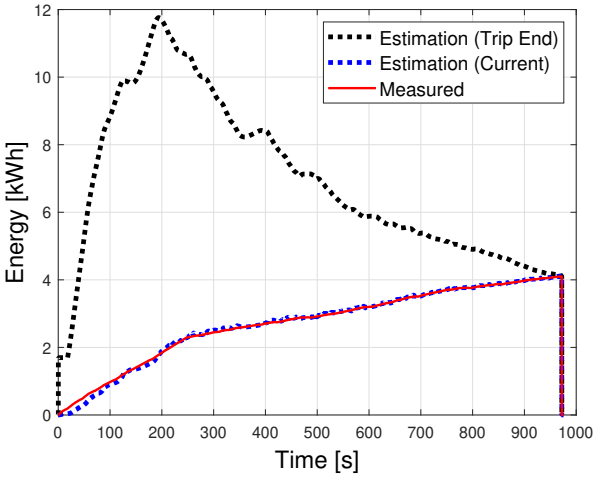
(a) Active Status



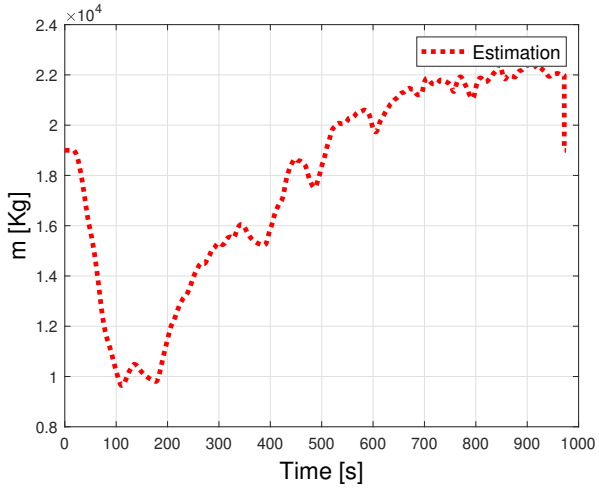
(b) Route Count



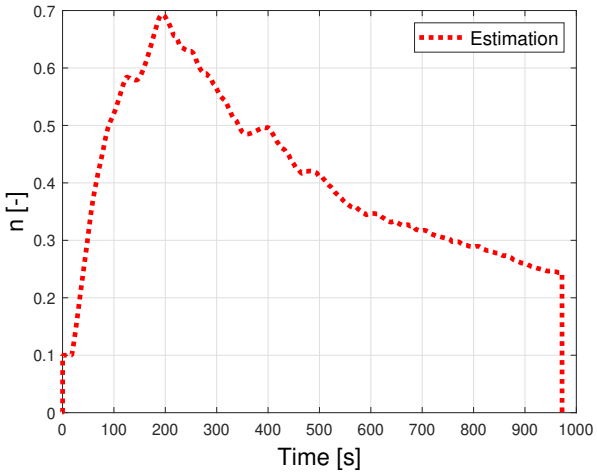
(c) Drive-train Online Energy Estimation



(d) Auxiliary Online Energy Estimation

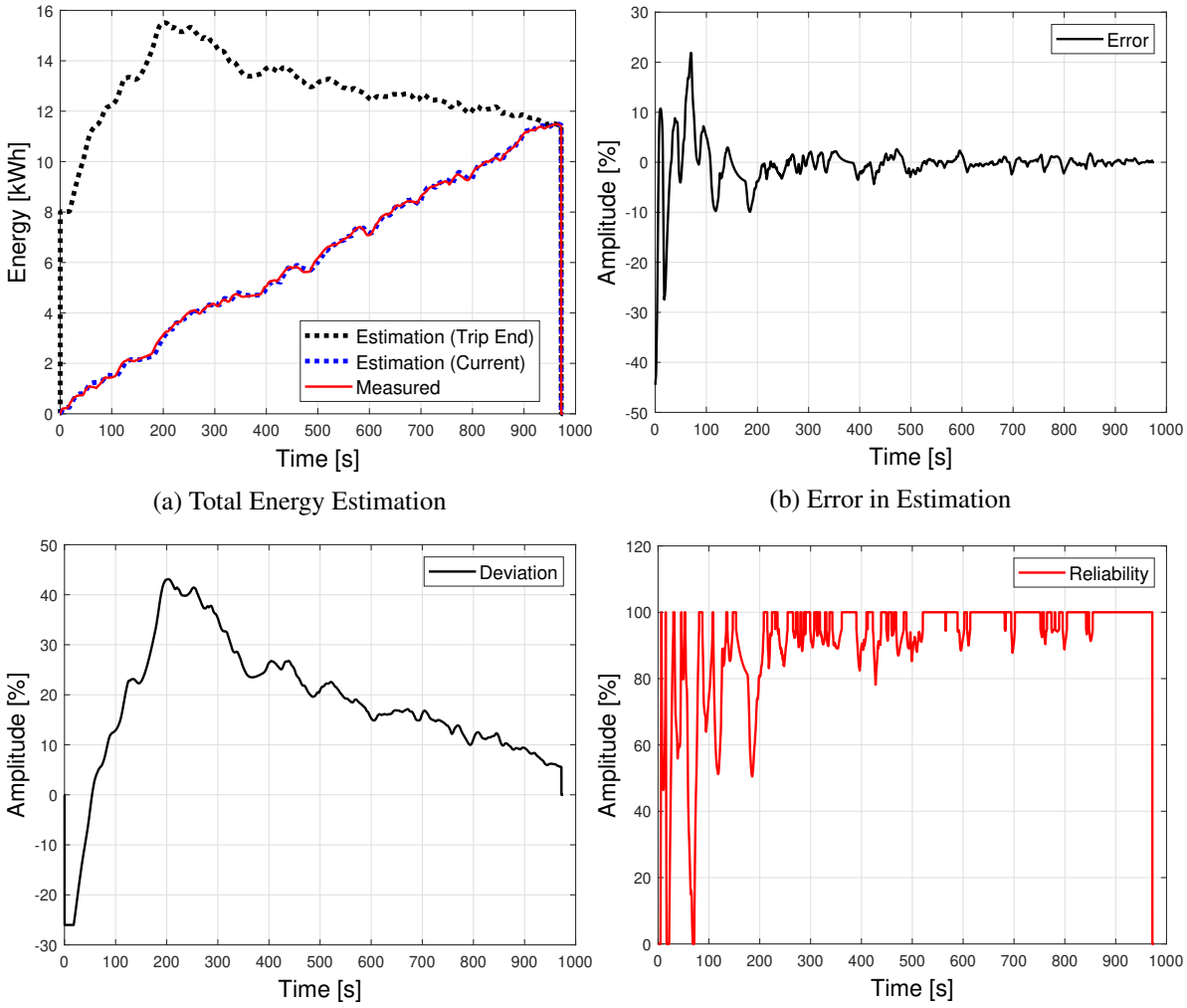


(e) Mass Estimation



(f) Correction Gain Estimation

Figure 6.8: Simulation Results: Test Scenario 3 (Estimations using Validation Data Set 1)



(a) Total Energy Estimation

(b) Error in Estimation

(c) Deviation from the Offline Estimate

(d) Reliability of Estimation

Figure 6.9: Simulation Results: Test Scenario 3 (Estimations using Validation Data Set 1)

6.3 Results

From the simulation test results in section 6.2, it can be clearly seen that the energy consumption prediction system is an advanced system capable of estimating the approximate energy consumed by the electric city bus over the given route well in time, and is also producing the robust results for different data cycles. A detailed analysis is also made on the system's accuracy and precision by observing the absolute accumulative error (see Remark 6.3.1) while using the base reference profile.

Remark 6.3.1 *The absolute accumulative error computed in the Figure 6.10 is calculated for the estimations done over the trip cycle distance travelled using the following formula:*

$$Error_{Absolute}(\%) = \frac{|\sum_0^{s(T)} Estimation - \sum_0^{s(T)} Measurement|}{\sum_0^{s(T)} Measurement} * 100(\%) \tag{6.4}$$

In Figure 6.10, it can be clearly observed that the performance of real-time (online) energy estimation

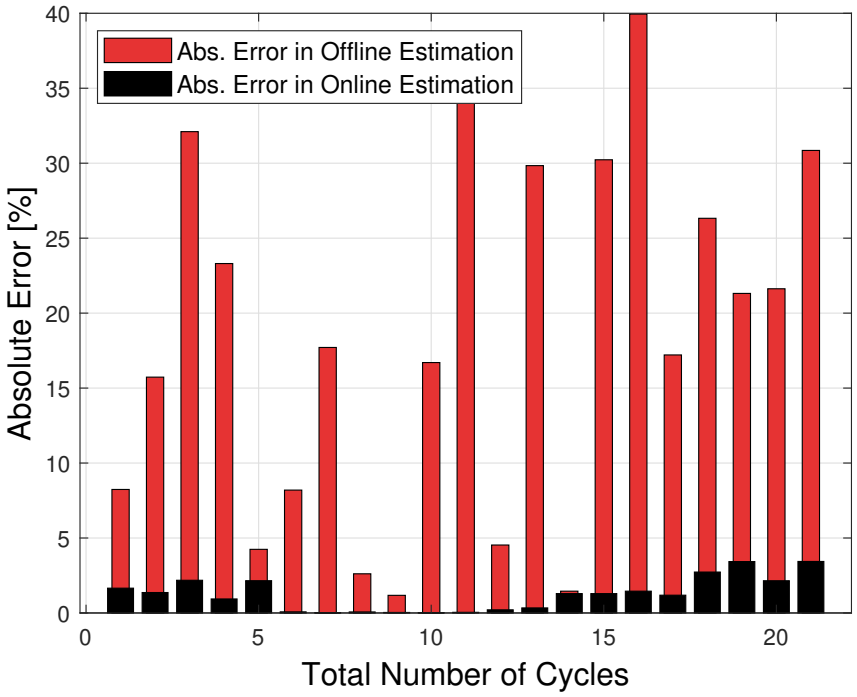


Figure 6.10: Comparison of absolute accumulative error in estimations for offline and online energy predictions done over base reference profile.

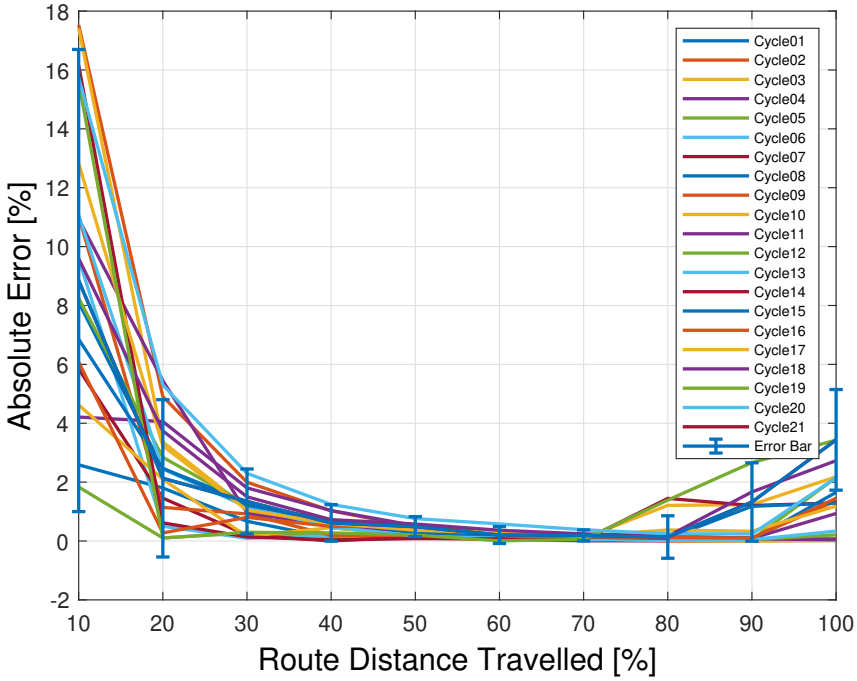


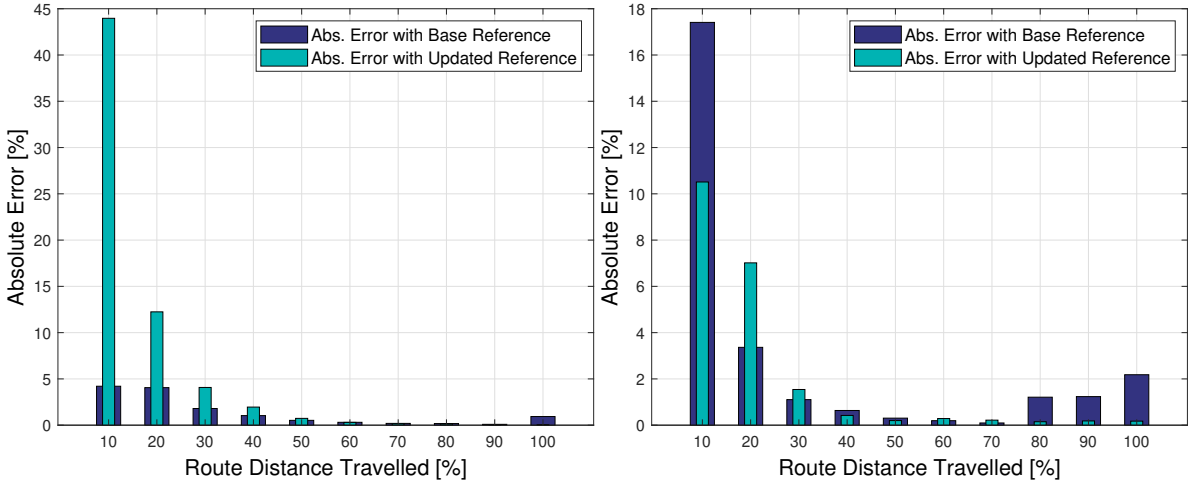
Figure 6.11: Progression of absolute accumulative error for online energy estimation w.r.t % of distance travelled over the route.

system is more accurate than the offline estimations. In offline estimations, the absolute accumulative error over some cycles can go as high as 40 % and, on an average, remains at 18.5%.

On the other hand, in real-time (online) energy estimations, the absolute accumulative error is under 4 % and on average remains 1.2%.

This illustrates the superior performance of the developed real-time energy consumption prediction system.

In Figure 6.11, the error results are compiled as a function of travelled distance for all available data cycles with reference profile used as base data profile. It can be seen that the performance of the real-time energy consumption prediction is good in the region from 30% to 80% of the traveled distance. Outside this region, the performance remains reasonably good, and the system tries to bound the error. The reason behind the difference in this performance is that the electric city bus does not have a dedicated path to complete the trip, especially outside the above-mentioned range. Another possible reason is the abrupt fluctuations in the number of passengers around some very busy bus stops, where a lot of passengers board or de-board the vehicle.



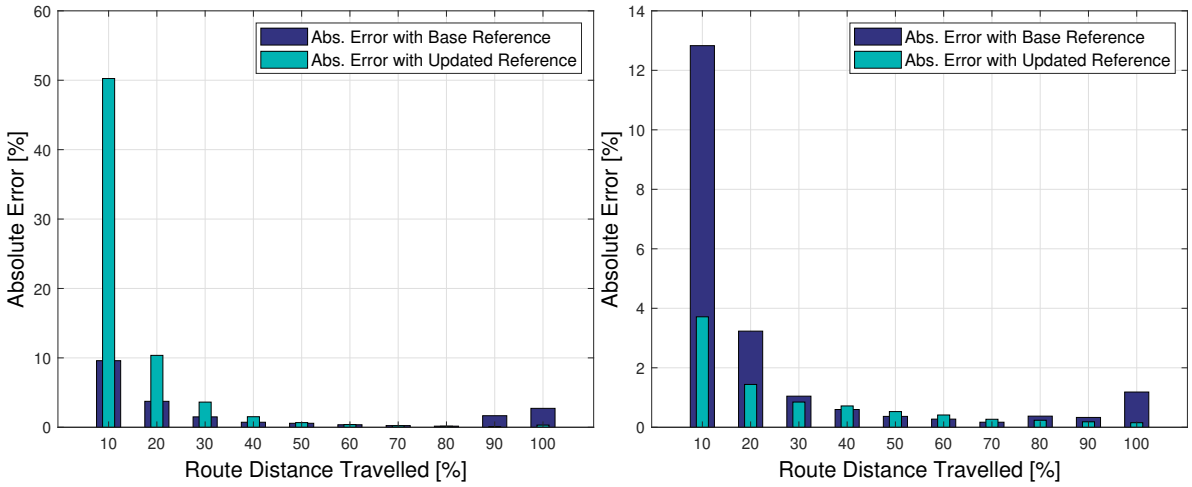
(a) Comparison: Trip Data Cycle A

(b) Comparison: Trip Data Cycle B

Figure 6.12: Comparison of absolute error in estimations for online energy predictions done over base and updated reference profile.

In Figure 6.12, the training data cycles were used to calculate the accumulative absolute error in energy estimations using the base and updated reference profiles respectively. It can be observed while the base data reference profile was used, the overall absolute accumulative error remains consistently small throughout the trip. In the case when updated reference data profile was used, the absolute accumulative error was relatively high in the beginning; it can be because of two reasons; one was not using the dedicated path at the beginning of the trip and the other being the updated reference data profile was a bit volatile as it uses the averaged data from a smaller number of trips (two in this case). It can also be observed that very quickly, the absolute accumulative error tends to zero, and the trend was almost exponential (negative) in nature; this happens due to the availability of most recent data from capturing the operating domain characteristics of the electric city buses. The absolute accumulative error by the end of the trip in the case where updated reference profile was used was much less than the one compared to the base reference profile. It can be concluded that the predictions only get better when using the updated reference data profile with the online estimation algorithms.

A similar phenomenon was observed in the case when the validation data cycle was used, see Figure 6.13.



(a) Comparison: Trip Data Cycle A

(b) Comparison: Trip Data Cycle B

Figure 6.13: Comparison of absolute accumulative error in estimations for online energy predictions done over base and updated reference profile.

6.4 Summary

In this chapter, simulations were carried out using the proposed energy consumption model with the real-time operation data collected, from the devices installed on the bus. The parameter estimation and correction algorithm was tested using various test scenarios designed to evaluate the functionality of all features of the system. These tests gave robust results while minimizing the error in the energy predictions for the trip cycles on given route. The proposed online technique was seen to perform better than the offline estimation technique. Also, the self learning and adapting feature of the online algorithm allows the predictions to improve when subsequent trips were made on the selected route.

7 Conclusion and Recommendation

In section 7.1, the research conclusions are summarized. In section 7.2, the recommendations and future research objectives are given.

7.1 Conclusion

Current research in energy estimation suffers from the shortcomings of estimating many parameters required for the models to capture the characteristics of the electric vehicle. This makes it difficult to update these parameters when the vehicle is subjected to disturbances in its operational domain. To tackle this problem, this dissertation aims to design a real-time energy consumption prediction system that can work in a more general way on different types of electric vehicles. In public transportation route remains fixed, and the propelling force can be characterized by using the historical data collected. It allows the estimation model to pack all the parameter variables in traditional modeling techniques altogether. This allows omitting the necessity to upgrade all conventional parameters individually. The proposed scheme uses the Kalman filter regression algorithm to correct the model parameters that allows the system to be more adaptable to the changes. Overall the real-time energy estimation system has the capability of self-learning by updating the reference profile by itself to make better predictions on estimations.

The following conclusions can be drawn:

- The presented system is able to identify the start and stop of a particular trip cycle on a selected route without failing using GPS coordinates matching. This ensures that estimations will be available from the system every time the trip cycle starts.
- The estimation and correction scheme achieves asymptotic tracking of the actual measurements and ensures that the tracking error decayed to zero.
- The estimation and correction scheme is capable to produce reliable predictions in the presence of disturbances occurring due to the weather, traffic, payload conditions, and driving style behavior.
- The system is able to successfully update the reference profiles used by the estimation and correction algorithm and is able to adapt and learn with the availability of new data. This allows the system to produce better estimations on the subsequent trips and ensures high accuracy in predictions.
- The system could ensure a good reliability index number for the estimations during the trip, and the reliability increases by the end of a trip.

7.2 Recommendations

In this section some of the recommendations are presented as follows:

- **Improving offline model:** The offline model has the potential to be improved. This model can further be optimized by including relevant factors not only from the vehicle itself but also from the environmental aspect. For instance, the temperature varies from season to season or from day to night, which also influences the energy consumption. The time of the operation determines the level of the variation of passenger load, which is also a significant component in energy usage. Considering more factors can improve the estimating accuracy.
- **Increase the amount of memory of the embedded device:** More memory is required if at some point the infrastructure is needed to be expanded to store more number of reference profiles based upon various operating conditions such as weather dependent conditions, time of the day travel conditions, off-peak and peak hours travel conditions and seasonal based conditions are considered. Also, if the architecture will be expanded to consider multiple route estimations on the same device, then multiple such reference profiles have to be stored, which requires more memory space on the device.
- **Expansion of the design:** The current infrastructure can be expanded to accommodate the additional functionalities, such as performing the estimations on multiple routes and saving conditional based reference profiles used for estimations.
- **Autonomy in the generation of reference profile:** The current infrastructure for the energy estimation system requires a base reference data profile pre-loaded in the memory. This data profile is computed offline. Future steps can accommodate the creation of this base reference profile more autonomously on the embedded device itself.
- **Geo-fencing:** Future versions of the software can also use the mapping of the energy estimations with the GPS coordinates. This will further assist in improving the estimation algorithm by helping in aligning the data concerning the location and analyzing the correlations between the energy consumed and the road characteristics. This feature could not be achieved due to the limitation of the memory available on the device and due to time constraints.
- **Route ID:** Route identification is required to load the relevant data profiles and parameters into the memory, which are used by the estimation and correction algorithms to make the predictions on the energy. In the current infrastructure this is hard-coded and limited to a single route. When the software infrastructure will be expanded to account for energy estimations on multiple routes, than system will require the route updates during the operation to identify the correct route. An Application Programming Interface (API) based solution can be used to receive this information on a specific embedded device.
- **Cloud computing:** Another recommendation would be changing the design entirely from a more local based device to a global cloud computing solution. These database and estimation algorithms can be maintained and run on the cloud service while just transferring the vital information between bus and cloud.
- **Expanding HMI:** The HMI developed for the prototype version can be extended to the real-time browser-based or application-based service while expanding it for more features.

Bibliography

[1] IEA, *Global Electric Vehicle Outlook 2019.* International Energy Agency, Paris,France, 2019.

[2] Y. Chen and Y. Fan, "Transportation fuel portfolio design under evolving technology and regulation: a california case study," *Transportation Research Part D*, vol. 24, pp. 76–82, 2013.

[3] N. Lutsey, "Transition to a global zero-emission vehicle fleet: A collaborative agenda for governments," *International Council on Clean Transportation*, 2015.

[4] Y. Chen, G. Wu, R. Sun, A. Dubey, A. Laszka, and P. Pugliese, "A review and outlook of energy consumption estimation models for electric vehicles,"

[5] W. Li, P. Stanula, P. Egede, S. Kara, and C. Herrmann, "Determining the main factors influencing the energy consumption of electric vehicles in the usage phase.," vol. 48, no. 352-7, 2016.

[6] M. Masikos, K. Demestichas, E. Adamopoulou, and M. Theologou, "Mesoscopic forecasting of vehicular consumption using neural networks.," *19*, vol. 1, no. 145-56, 2015.

[7] J. Felipe, J. Amarillo, J. Naranjo, F. Serradilla, and A. Díaz, "Energy consumption estimation in electric vehicles considering driving style.," *IEEE 18th International Conference on Intelligent Transportation Systems*, pp. 101–106, 2015.

[8] L. Kessler and K. Bogenberger, "Forecast of the energy consumption of bev based on dynamic traffic information," *International Scientific Conference on Mobility and Transport Technologies*, 2015.

[9] K. Liu, J. Wang, T. Yamamoto, and T. Morikawa, "Modelling the multilevel structure and mixed effects of the factors influencing the energy consumption of electric vehicles," *Applied energy*, vol. 183, no. 1351-60, 2016.

[10] J. Wang, I. Besselink, and H. Nijmeijer, "Electric vehicle energy consumption modelling and prediction based on road information.," *World Electric Vehicle Journal*, vol. 7, no. 3, pp. 447–458, 2015.

[11] R. Maia, M. Silva, R. Araújo, and U. Nunes, "Electrical vehicle modeling: A fuzzy logic model for regenerative braking," *Expert Systems with Applications*, vol. 42(22), no. 8504-19, 2015.

[12] S. Sun, J. Zhang, J. Bi, and Y. Wang, "A machine learning method for predicting driving range of battery electric vehicles.," *Journal of Advanced Transportation*, 2019.

[13] J. Wang, I. Besselink, and H. Nijmeijer, "Battery electric vehicle energy consumption prediction for a trip based on route information.," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 232(11), no. 1528-42, 2018.

[14] K. Hu, J. Wu, and T. Schwanen, "Differences in energy consumption in electric vehicles: An exploratory real-world study in beijing.," *Journal of Advanced Transportation*, 2017.

[15] F. Ye, G. Wu, K. Boriboonsomsin, and M. Barth, "A hybrid approach to estimating electric vehicle energy consumption for ecodriving applications.," *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 719–724, 2016.

[16] G. Muller, *CAFCR: A Multi-view Method for Embedded Systems Architecting; Balancing Genericity and Specificity*. PhD thesis, Delft University of Technology, 2004.

[17] H. P. Hoffmann, "System engineering best practices with rational solution for systems and software engineering.," Feb. 2011. Deskbook Release 3.1.2.

[18] C. Beckers, I. Besselink, J. Frints, and H. Nijmeijer, "Energy consumption prediction for electric city buses," *13^{th} ITS European Congress, The Netherlands*, 2019.

[19] J. Wang, I. Besselink, and H. Nijmeijer, "Battery electric vehicle energy consumption modelling for range estimation," *International Journal of Electric and Hybrid Vehicles*, vol. 9, no. 2, pp. 79–102, 2017.

[20] W. Jiquan, *Battery Electric Vehicle Energy Consumption Modelling, Testing and Prediction - A Practical Case Study*. PhD thesis, Eindhoven University of Technology, 2016.

[21] J. Wang, I. Besselink, and H. Nijmeijer, "Evaluating the tu/e lupo el bev performance," *EVS27 International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium*, 2013.

[22] I. Besselink, H. Nijmeijer, P. Oorschot van, and E. Meinders, "Design of an efficient, low weight battery electric vehicle based on a vw lupo 3l," *The 25th World Battery, Hybrid and Fuel Cell Electric Vehicle Symposium and Exhibition*, 2010.

[23] Y. Ma, D. Jagga, I. Besselink, and H. Nijmeijer, "Real-time energy consumption prediction for electric city buses by characterizing the route locations with real-world data," *Publication Awaited*.

[24] L. Lennart, *System Identification: Theory for the User*. Pearson, 2nd ed., 1999.

[25] A. Vahidi, A. Stefanopoulou, and H. Peng, "Recursive least squares with forgetting factor for online estimation of vehicle mass and road grade: theory and experiments," *Vehicle System Dynamics*, vol. 43, no. 1, pp. 31–55, 2005.

[26] M. Campi, "Journal of mathematical systems, estimation and control." 4,1-25, 1994.

[27] G. Welch and G. Bishop, *An Introduction to the Kalman Filter*. Chapel Hill, NC 27599-3175, university of north carolina at chapel hill department of computer science ed., 2001.

[28] M. Khalil, S. Kelouwani, K. Agbossou, Y. Dube, and N. Henao, "Long-trip optimal energy planning with online mass estimation for battery electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 11, pp. 4929–4941, November 2015.

[29] N. Carlson, "Fast triangular formulation of the square root filter," *AIAA Journal*, vol. 11, no. 9, pp. 1259–1265, 1973.

[30] D. @Google, "Protocol buffers." `https://developers.google.com/protocol-buffers`, 2001.

[31] Owasys, "Technical specifications document, owa3x platform." https://www.owasys.com/en/products/owa3x, 2019.

A Appendix: Requirements Documentation

CLOUD YOUR BUS

Requirements Document

Abstract

This document contains relevant system requirements for the project Cloud Your Bus

Jagga, D.

d.jagga@tue.nl

Document Number		2019-CYB-REQ
Document Name		Requirements Document
Process Responsibility		TU/e, Sycada
Prepared By		Dhruv Jagga
Affected Product(s)		ICRON, Owasys hardware, OEM Busses
Start Date		15-12-2019
Completion Date		15-07-2020
Revision		V 1.1

The project is carried out with industrial partners

Table of Contents

TU/e

Version Control			
Version	Author	Date	Description/Updates
V 1.0	DJ	19/12/2019	Initial Version of the requirements table.
V 1.1	DJ	07/04/2020	• Project requirements, assumptions, constraints added. • System requirements are updated.

List Of Abbreviations

ID	Category Name
C	Concern
PR	Project Requirements
PC	Project Constraints
PAS	Project Assumptions
GR	General Requirements
DR	Data Requirements
MR	Model Requirements
PER	Performance Requirements
TR	Technical Requirements
HW	Hardware
CAN	Controller Area Network
OEM	Original Equipment Manufacturer
TBD	To Be Discussed

Stakeholder Concerns:

In this section, the concerns of the key stakeholders of the project are captured. The two major stakeholders in the project were dr.ir. Igo Besselink (project lead from TU/e) and Sycada (direct client from **Cloud Your Bus** Consortium). Below are mentioned the major concerns:

ID	Name	Specification
C – 1	Energy Usage	Understanding the energy usage pattern of the electric city buses
C – 2	Uncertainties	Accounting for uncertainties affecting energy consumption of the electric city buses.
C – 3	Messages	Identifying the relevant standardized messages to be sent to the cloud.
C – 4	Data	Logging Data in real-time.
C – 5	Planning	Facilitate the energy consumption prediction for optimizing dynamic scheduling.

Requirements Packages:

The requirements package has been divided into two major sections concerning the project and system respectively. It also includes the relevant assumptions and constraints linked to the project and system.

1. Project:

This section includes the relevant requirements, assumptions, and constraints linked to the project.

1.1 Project Requirements:

This section includes the project requirements.

ID	Name	Specification
PR – 1	Energy Prediction	The energy prediction model should be generic and must be deployable on electric buses from various operators.
PR – 2	Data Collection	The data from the electric buses have to be collected by Sycada over various trips for the selected route.
PR – 3	Data Type	The data must be delivered in the format usable with MATLAB (text or CSV files).
PR – 4	Run Locally	The system is needed to be run locally on the HW installed in Bus.
PR – 5	Functions	The system functions must be made relevant to the Linux operating system.

1.2 Project Constraints:

This section includes the constraints related to the project.

ID	Name	Specification
PC – 1	Data Collection	Data can not be collected over the 10 Hz frequency due to limitations of CAN bus bandwidth.
PC – 2	Network	The network used to receive/transmit data from/to the bus is on a 2G network.
PC – 3	Timeline	The implementation of testing on the HW level on the bus can take up to 3 months.
PC – 4	Diversions	The energy consumption model might not produce viable results in case of any diversions (different route) taken on the route.

1.3 Project Assumptions:

This section includes the assumptions related to the project.

ID	Name	Specification
PAS – 1	Data Collection	It is assumed that the data delivered by Sycada is of good health.
PAS – 2	Route Identifier	It is assumed that a route identifier/operator is always there to monitor the bus route.
PAS – 3	City Travel	It is assumed that the bus is traveling in the city all time.
PAS – 4	Intended Route	The bus is always intended to use the fixed-route on the selection of routes before the journey begins.
PAS – 5	Bus-lanes	The buses are assumed to be running on dedicated bus-lanes and hence, the traffic flow is assumed to be smooth.
PAS – 6	Ambient Temperature	The ambient temperature inside the bus is maintained when delivered at the first bus-stop.

2. System:

This section includes the relevant requirements, assumptions, and constraints linked to the system.

2.1 System Requirements:

This section discusses the system requirements, which are divided into five major packages including two major types of requirements; functional requirements and performance requirements.

- General Requirement Package.
- Data Requirement Package.
- Model Requirements Package.
- Technical Requirements Package.
- Performance Requirements Package.

The general requirements package is derived from stakeholder concerns. General, data, model, and technical requirements packages include the functional requirements for the given system. These requirements are required to ensure the operational capability of the system. On the other part, the performance requirements are required to understand the extent to which a function must be executed. The columns ‘mapping’ and ‘status’ in the table below show the source of the requirement from where it is derived and the current status of the requirement respectively.

ID	Name	Specification	Mapping	Status
GR – 1	Predict Energy Consumption	The system must predict the energy consumption of battery-electric busses.	C – 1	Done
GR – 2	Generic Model	The system must work on vehicles from different OEM's.	C – 1	Done
GR – 3	Cloud Service	The system must have a relevant message to be sent to the cloud service.	C – 3	Done
GR – 4	Dynamic Scheduling	The system must send a relevant message to the cloud to be used in dynamic scheduling.	C – 5	Done
GR – 5	Trip Status	The system must be able to identify the starting and stopping of the trip cycle for the respective route.	C – 4	Done
GR – 6	Update Parameters	The system must account for uncertainties and update the model parameters periodically (10 Hz) to give accurate energy consumption estimates.	C – 2	Done
GR – 7	Energy	The system must provide (as output) a pre-determined energy estimate for the given route and corrected predicted/estimated energy for the given route by the end of the selected route.	C – 3	Done
GR – 8	Database	The system must have an updated database to store initial parameters for different routes.	C – 1	Done

GR – 9	Source Code	The system source code must run locally on the embedded device.	C – 1	Done
GR – 10	Reliability	The system must produce reliable estimations on the predicted energy.	C – 1	Done
DR – 1	Velocity Profile	The system must record/collect velocity data of the vehicle from CAN-Bus at a sample rate of 10 Hz over respective routes.	GR – 7	Done
DR – 2	Acceleration Profile	The system must record/collect acceleration data of the vehicle from CAN-Bus at a sample rate of 10 Hz over respective routes.	GR – 7	Done
DR – 3	Mass Profile	The system must record/collect bellow pressure data of the vehicle from CAN-Bus over respective routes.	GR – 7	Done
DR – 4	State of Charge	The system must record/collect the state of charge of battery data of the vehicle from CAN-Bus at a sample rate of 10 Hz over respective routes.	GR – 7	Done
DR – 5	Distance Profile	The system must record/collect distance traveled data of the vehicle from CAN-Bus at a sample rate of 10 Hz over respective routes.	GR – 7	Done
DR – 6	Voltage	The system must record/collect voltage data at the battery and drive-train terminal of the vehicle from CAN-Bus at a sample rate of 10 Hz over respective routes.	GR – 7	Done
DR – 7	Current	The system must record/collect current data at the battery and drive-train terminal of the vehicle from CAN-Bus at a sample rate of 10 Hz over respective routes.	GR – 7	Done
DR – 8	GPS Coordinates	The system must record/collect GPS data (latitude, longitude, and heading) of the vehicle from CAN-Bus at a sample rate of 10 Hz over respective routes.	GR – 5	Done
MR – 1	Road Load	The model must account for energy consumption by the number of passengers/load.	GR – 6	Done
MR – 2	Auxillary System	The model must account for energy consumption by auxiliary systems including Ventilation and Air Cooling system.	GR – 6	Done
MR – 3	Road Topography	The model must account for energy consumption by road topography (height, slope).	GR – 6	Done
MR – 4	Ambient Temperature	The model must account for energy consumption due to changes in ambient weather (temperature).	GR – 6	Done
MR – 5	Aerodynamic drag	The model may account for energy consumption due to aerodynamic drag.	GR – 6	Done
MR – 6	Gradient	The model must account for energy consumption due to the gradient of the road for the route.	GR – 6	Done
MR – 7	Mass Estimate	The model must estimate the mass profile from available bellow pressure data.	GR – 6	Not Done
TR – 1	Trip Start [Range]	The system must identify the start of the trip when GPS coordinates are in the range of 0.0004 in latitude and longitude and 50° in heading angle.	GR – 5	Done
TR – 2	Trip Stop [Range]	The system must identify the stop of the trip when GPS coordinates are in the range of 0.0004 in latitude and longitude and 50° in heading angle.	GR – 5	Done
TR – 3	Output Message	The system must send an output message to Sycada software at a sample rate of 10 Hz.	GR – 4	Done

TR – 4	Database (1)	The system must store a base reference profile in memory for estimations.	GR – 8	Done
TR – 5	Database (2)	The system must be able to perform the estimation for a given route with the base reference profile.	GR – 8	Done
TR – 6	Database (3)	The system must be able to create its own base reference profile for the given route.	GR – 8	Not Done
TR – 7	Database (4)	The system must be able to perform the estimation for a given route with the updated reference profile.	GR – 8	Done
TR – 8	Database (5)	The system must create a new updated reference profile after every second trip cycle for the given route.	GR – 8	Done
TR – 9	Database (6)	The system must ensure that estimations are done using the updated reference profile when available.	GR – 8	Done
TR – 10	Database (7)	The system must ensure that estimations are done using the base reference profile when an updated reference profile is not available or is not a healthy data profile.	GR – 8	Done
TR – 11	Multiple Route	The system must be able to perform estimations for multiple routes.	GR – 1	Done (Simulation)
PER – 1	Error (1)	The system should not have more than 20 % instantaneous error in energy estimation using the energy consumption model and correction algorithm.	GR – 10	Done
PER – 2	Error (2)	Thes system should not have more than 5 % absolute error (for the entire route) in energy estimation using an energy consumption model and correction algorithm.	GR – 10	Done
PER – 3	Deviation	The allowed deviation in estimated energy and actual energy should be less than 0.1 KWh.	GR – 10	Done
PER – 4	Reliability	The reliability of estimations must be more than 80 %.	GR – 10	Done
PER – 5	Checks	The system must have the warning/failure checks to identify the fault during operation.	GR – 10	Done

References

[1] I. 24765:2010(E), *ISO/IEC/IEEE International Standard - Systems and software engineering.*, 2010, pp. 1-418.

[2] *29148-2011 - ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes -- Requirements engineering.*

[3] *Systems Engineering Tutorial for Rational Rhapsody, IBM Rational Rhapsody, 7.4,* 2009.

B Appendix: Interface Documentation

CLOUD YOUR BUS

Interface and Data Specifications Document

Abstract

This document contains the interface and data specifications required for interfacing the software developed by TU/e with software developed by Sycada

Jagga, D.

d.jagga@tue.nl

Document Number		2020-CYB-INT
Document Name		Interface Document
Process Responsibility		TU/e, Sycada
Prepared By		Dhruv Jagga
Affected Product(s)		ICRON, Owasys hardware, OEM Busses
Start Date		24-01-2020
Completion Date		17-05-2020
Revision		V 1.0

The project is carried out with industrial partners

Table of Contents

List of Figures

List of Tables

Version Control			
Version	Author	Date	Description/Updates
V 1.0	DJ	24/01/2020	Initial version of interface documents.

Interface:

The software developed by TU/e for estimating the energy consumption of electric busses has an interface with the software developed Sycada, which is acting as a communication layer in between the energy consumption logic developed by TU/e and data available on the hardware level (See Figure 1). The interface parameters are selected and are available in Table 1. For these interface parameters, the data types, the data specifications is also available in Table 1. While collecting the data as described, each variable's value is captured along with the time stamp. This is required to run the algorithm and make desired calculations from energy consumption logic.

Table 1: Data Specifications

S.No	Name	Description	Type	Resolution	Unit	Format	
1	Time	Time stamp	i/p	0.1 (10 Hz)	sec	double	RD
2.	Velocity	Average longitudinal velocity	i/p	0.1 (10 Hz)	m/s [km/h]	double	RD
3.	Distance	Travelled distance	i/p	0.1 (10 Hz)	m [km]	double	RD
4.	V (Battery)	Voltage [Battery terminal]	i/p	0.1 (10 Hz)	V	double	RD
5.	I (Battery)	Current [Battery terminal]	i/p	0.1 (10 Hz)	A	double	RD
6.	V (Drivetrain)	Voltage [Drivetrain terminal]	i/p	0.1 (10 Hz)	V	double	RD
7.	I(Drivetrain)	Current [Drivetrain terminal]	i/p	0.1 (10 Hz)	A	double	RD
8.	RI	Route information	i/p	0.1 (10 Hz)	-	int	RD
9.	GPS_{lon}	GPS longitudinal coordinate	i/p	0.1 (10 Hz)	Deg	double	RD
10.	GPS_{lat}	GPS latitudinal coordinate	i/p	0.1 (10 Hz)	Deg	double	RD
11.	GPS_{hea}	GPS Heading angle	i/p	0.1 (10 Hz)	Deg	double	RD
12.	Time (Est)	Time stamp (Estimations)	o/p	0.1 (10 Hz)	sec	double	RD/RW
13.	Normative E	Offline predicted energy estimate	o/p	0.1 (10 Hz)	kWh	double	RD/RW
14.	Projected E	Online predicted energy estimate	o/p	0.1 (10 Hz)	kWh	double	RD/RW
15.	Reliability	Reliability of estimations	o/p	0.1 (10 Hz)	%	double	RD/RW

Interface Diagram:

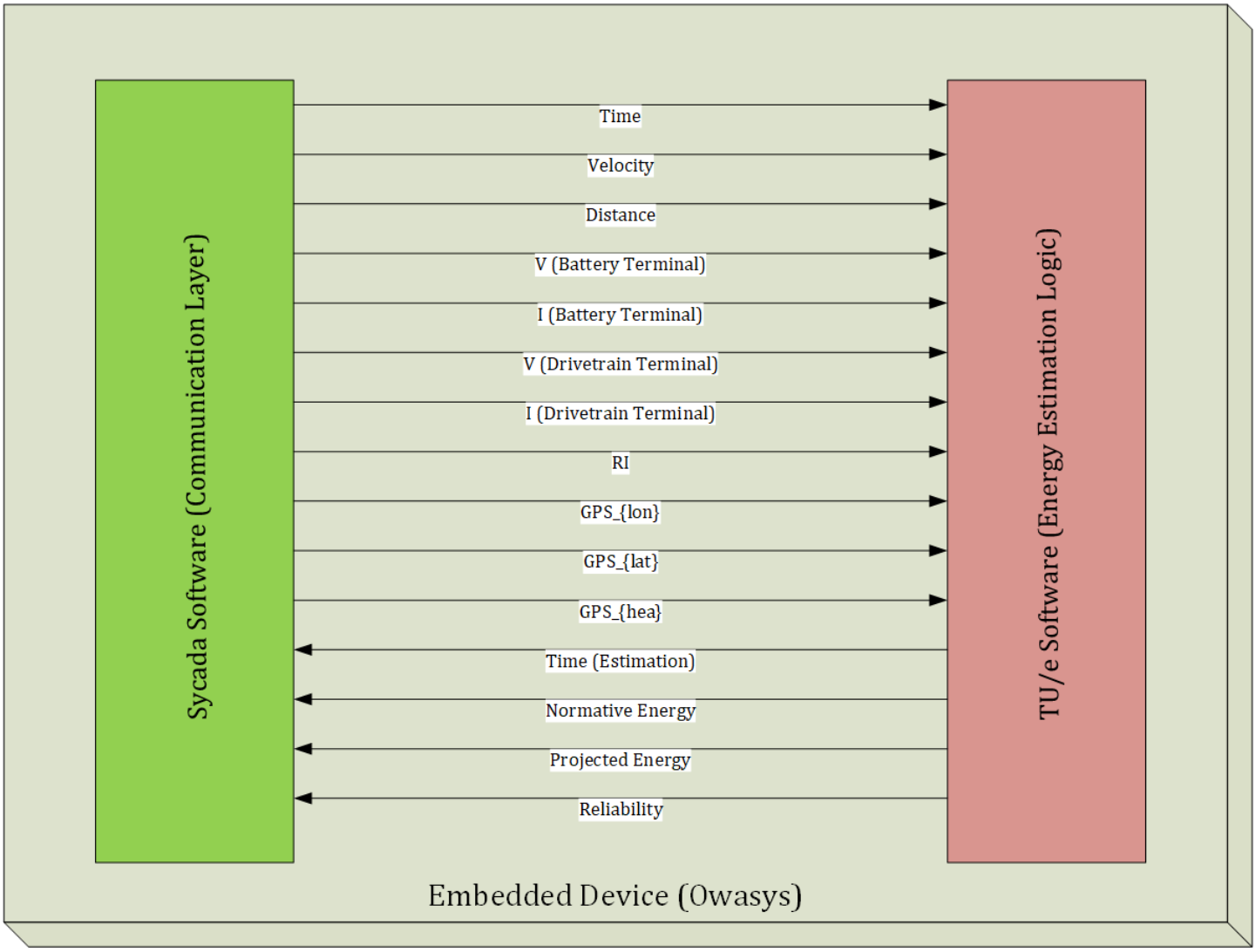


Figure 1: Interface between Software's

C Appendix: Hardware Setup

The hardware setup made for doing the experiments (Hardware In the Loop - HIL testing) is shown in Figure C.1. A developer's PC was used on which the real-time estimation system software was written. This software was compiled into executable file in binary format which was transferred to the embedded device (using RS-232 and USB connector) and was run on the device. The CAN data was loaded as the re-playable data on the embedded device using a GSM afaik connection. This data is available on the embedded device in form of csv files. While the energy estimation system software is running on the embedded device it requires the CAN data information from these csv files. The data is being received via an intermediate software prepared by the client; which made available the CAN data in readable form on the protobuf queues. Protobuf (Protocol Buffers) is a method of serializing the structured data. This method is used in developing programs to communicate with one another over the wired connection or for storing the data. The protobuf involves and interface description language describing the structure of the data and a program which is capable of generating a source code from the given description for generating or parsing a stream of bytes that represents the structured data. [30]

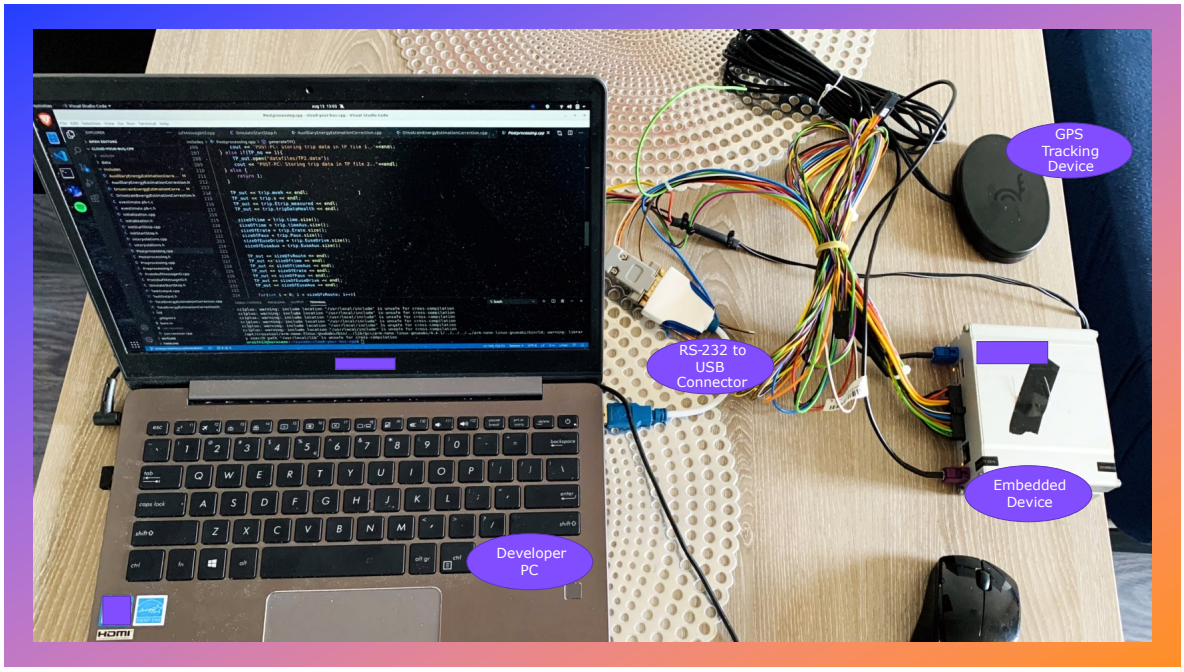


Figure C.1: Hardware Setup

The hardware used, especially the embedded device is prepared by one of the partners in the consor-

tium of *Cloud Your Bus* project. [31] The specification of this device is available in Table C.1

Table C.1: Embedded Device Specifications

Features	Device Specifications
Processor / MHz	ARM9 / 400
Linux Kernel (OS)	2.6.36
RAM	32 MB / 64 MB
Flash	64 MB + 32 GB with uSD
Micro-SD Card	Available
GNSS	u-blox
GSM / GPRS	Gemalto
Digital I/O	10
Analog Inputs	4
RS-232	Available
RS-485	Available
Accelerometer	Available
CAN	Available
GSM / GPS Connector Type	FAKRA
USB Host	2
Ethernet	Available
Internal Battery	1800 mAh
Protection	IP30

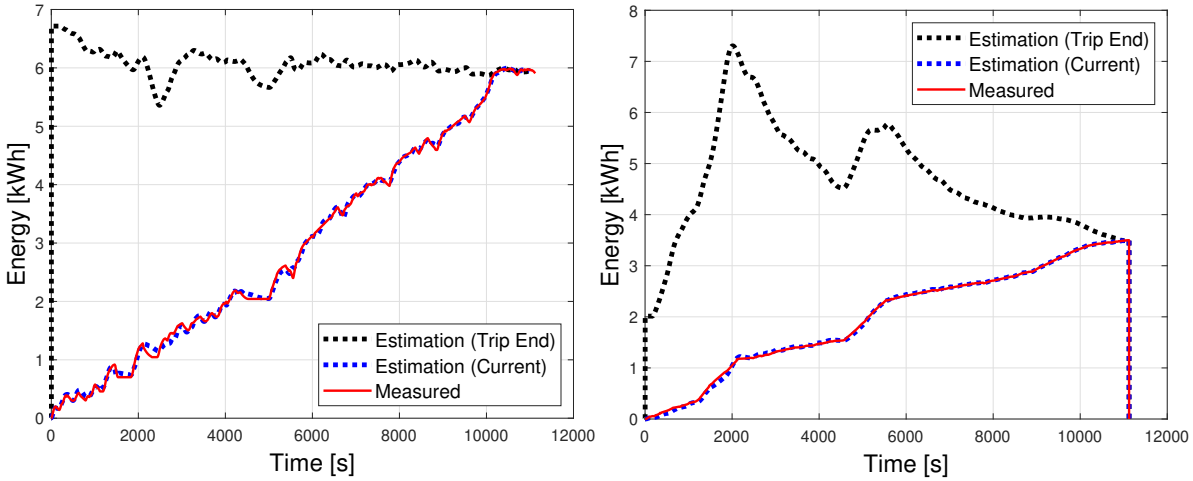
D Appendix: Additional Results from Hardware-in-the-Loop Test

In the testing phase of the project. The three level testing approach was used. The extensive Software in the Loop (SIL) testing was done first. The results of these tests are shown in Chapter 6. Moving one step further from here the experiments were conducted on the hardware. The results from the Hardware in the Loop (HIL) testing is shown in this Appendix D. The third level of testing was Factory Acceptance Test (FAT) and the results of these tests are shown in Appendix E using a test plan.

The hardware in the loop test was done on the embedded device with a re-playable data. The base reference profile was used for estimations and corrections. It clearly identified that the system does the drive-train and auxiliary energy estimation separately which can be seen in Figure D.1a and D.1b respectively. The corresponding correction parameters evolution can be observed in Figure D.1c and D.1d respectively.

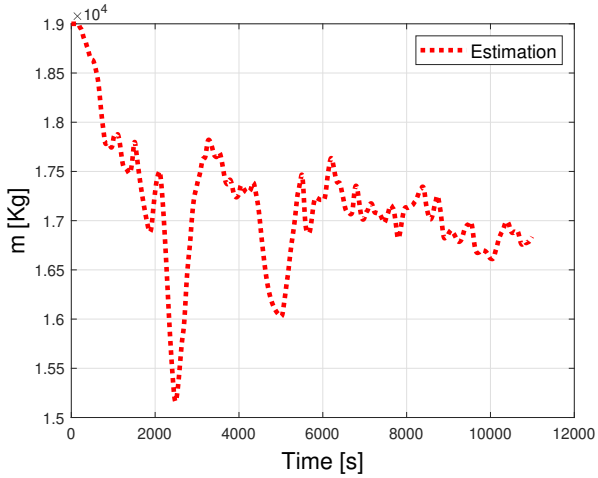
The combined total energy estimation from drive-train and auxiliary estimation-correction algorithms can be observed in Figure D.2a. The error resulting in between the estimated value and actual measured values can be observed in Figure D.2b, it can be seen that as the more data is getting available the resulting error starts decreasing and eventually becomes bounded around zero. It was also crucial to observe the reliability of the energy estimation and correction algorithm which is based upon the evolution of the error. To trust the predicted energy estimates it was important to know if the reliability of the estimation was higher or not. This can be observed in Figure D.2d

The *Updated Reference Data Profile* generated by the system after completion of two trips cycles on the selected route is shown in Figure D.3

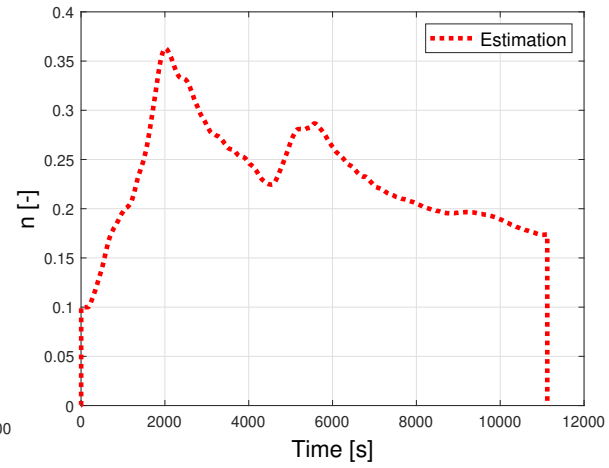


(a) Drive-train Online Energy Estimation

(b) Auxiliary Online Energy Estimation

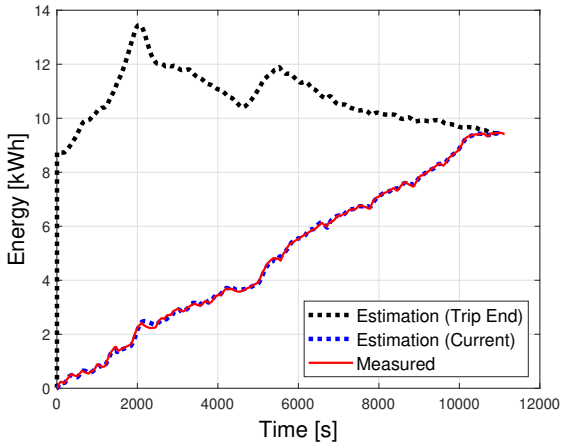


(c) Mass Estimation

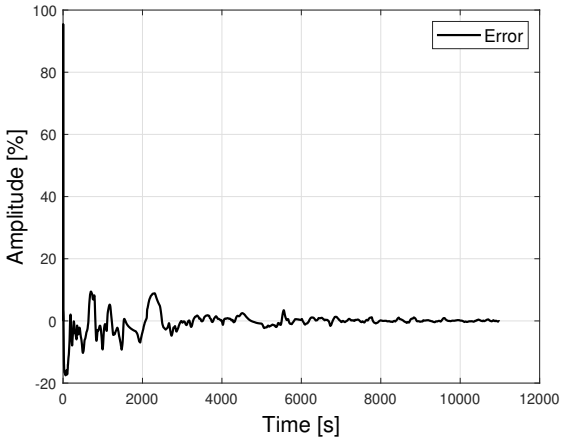


(d) Correction Gain Estimation

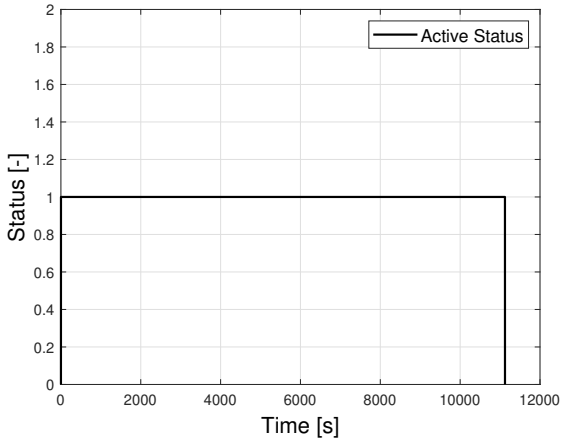
Figure D.1: Hardware in the Loop Test Results



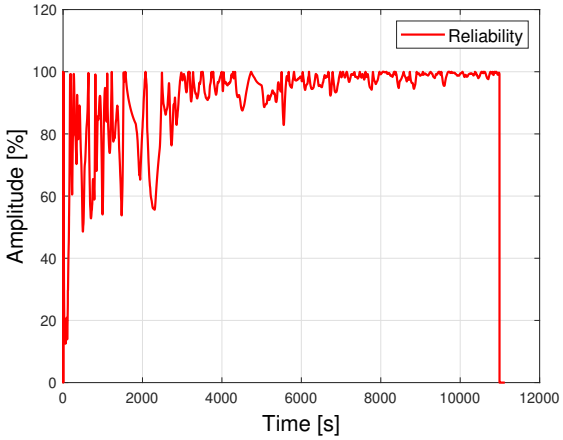
(a) Total Energy Estimation



(b) Error in Estimation

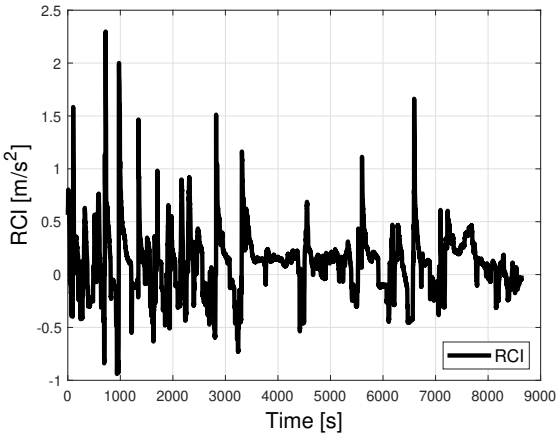


(c) Active Status

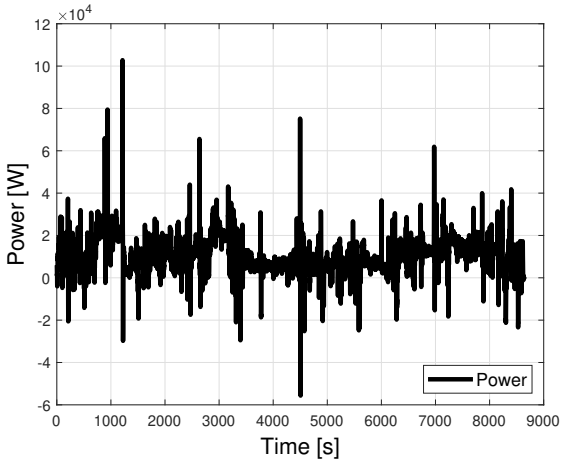


(d) Reliability in Estimation

Figure D.2: Hardware in the Loop Test Results



(a) Averaged RCI Data Profile



(b) Averaged Auxiliary Power Data Profile

Figure D.3: Hardware in the Loop Test Results

E Appendix: Test Plan and Test Results


CLOUD YOUR BUS

CLOUD YOUR BUS

Test Plan

Abstract

This document contains the test plan required for verifying and validating the software developed by TU/e for its designed functionalities.

Document Number		2020-CYB-Test Plan
Document Name		Test Plan
Process Responsibility		TU/e, Sycada
Prepared By		Dhruv Jagga
Reviewed By		Dan, Yuzhe
Affected Product(s)		ICRON, Owasys hardware, OEM Busses
Start Date		03-07-2020
Completion Date		01-09-2020
Revision		V 2.0

The project is carried out with industrial partners

Contents

Functional Testing

Functional Testing is the process that ensures the quality of the system developed. It is usually considered as black-box testing that bases its test cases on the requirements or specifications generated for the component under testing. Functionality testing of the system under investigation is achieved by feeding the inputs to the functions and auditing the output it generates. The interior functionality is rarely considered. Functional testing is usually done to evaluate the compliance of the system/component/function with respect to its specified functional requirements. It also allows to understand what exactly the system is doing. [1]

The process of functional testing can be conducted without knowing the internal working of the software. This allows to reduce the developers-bias during the process of testing.

The major tasks involved in Functional Testing are as follows [2]:

1. Identifying all the functionality needed to be performed by the system.
2. Creation of input data based upon the specification of the functionality.
3. Determining the output based upon the specification of the functionality.
4. Executing the test cases.
5. Analyzing and comparing the actual outputs to the expected outputs and recording the observations.
6. Analyzing and performing checks whether the system functionality works as per the client's need.

In the next section of this document, the functional test tables are available. Each table includes the process of testing the functionality and the results of the tests.

Test Case ID	CYB_001	Test Case Description	Test Functionality – Initialization/Checking Connections		
Created By	Dhruv	Reviewed By		Version	1.3

QA Tester's Log

Tester's Name	Dan	Date Tested	15-08-2020	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	The device power is on and the operating system is booted.
2	
3	
4	

S #	Test Data		
1	Base Data		
2	Schedule Data		
3			
4			

Test Scenario Verify that the connection is made with Sycada Software/Device and information exchange is taking place.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Check Message in Log file – Base Data Read	True	Expected	Pass
2	Check Message in Log file – Schedule Data read	True	Expected	Pass

Test Case ID	CYB_002	Test Case Description	Test Functionality – Database (Base Data Profile Check) / Parameter		
Created By	Dhruv	Reviewed By		Version	1.3

QA Tester's Log

Tester's Name	Dan	Date Tested	27-08-2020	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	Pre-saved/offline data structure file
2	
3	
4	

S #	Test Data		
1	Route distance	6	Route auxiliary power
2	Mass of vehicle	7	Estimated drivetrain energy
3	Max route distance traveled	8	Estimated auxiliary energy
4	Route time	9	Total Estimated energy
5	Route Erate	10	

Test Scenario Verify that the pre-loaded database is available for the estimation algorithms to work for the selected route

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Check GPS parameters (Manually)	GPS start and stop coordinates present	Expected	Pass
2	'ErateAvgRoute401_A', check the file in memory	The file must be available	Expected	Pass
3	Check the file data structure	The Test Data mentioned in this sheet must be available	Expected	Pass
4	Check System State	System State – Check Trip Start	Expected	Pass

Test Case ID	CYB_003	Test Case Description	Test Functionality – Receiving Operation Data		
Created By	Dhruv	Reviewed By		Version	1.3

QA Tester's Log

Tester's Name	Dan	Date Tested	27-08-2020	Test Case (Pass/Fail/Not Executed)	Suspended

S #	Prerequisites:
1	The device power is on and the operating system is booted.
2	The operation data sender is running
3	System State – Check Trip Start
4	

S #	Test Data		
1	Time (Operation Data)	5	Vehicle velocity (Operation Data)
2	GPS coordinates (Operation Data)		
3	Battery voltage and Current (Operation Data)		
4	Drivetrain voltage and current (Operation Data)		

Test Scenario Verify that the data received is in good order

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Access the Protobuf Queue	Receive first input data	Expected	Pass
2	Check the frequency at which data is received (for say 100 samples)	The frequency ~ 10 Hz	The frequency is ~20Hz. Need to ask Sycada to lower it.	Suspended
3	Access the data as described in the test data section in this test sheet	The test data must not be NaN	Expected	Pass
4	Check Message in Log File - CANdataReceivingOK	True	Expected	Pass
5	Check System State	System State – Energy Estimation	Expected	Pass

Test Case ID	CYB_004	Test Case Description	Test Functionality – Initiate start of estimation/route		
Created By	Dhruv	Reviewed By		Version	1.3

QA Tester's Log

Tester's Name	Dan	Date Tested	27-08-2020	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	Defined specific start and stop GPS coordinates (parameters)
2	System State – Check Trip Start
3	
4	

S #	Test Data
1	Start Route
2	
3	
4	

Test Scenario Verify that the system is able to identify the starting of the specifically selected route

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Check System State	If the trip has not started (eState is 0), the system state is Check Trip Start	Expected	Pass
2	Check Log file – for eState Value	eState flag is active (1) once the trip has started	Expected	Pass
3	Check Log file – for printed GPS coordinates when the trip has started (Manual Test)	The start flag of the route will be raised at the instance when actual GPS coordinates are in the range of 0.0004 in lat and long and 50° in heading to GPS parameters	Expected	Pass
4	Check System State	System State – Energy Estimation	Expected	Pass

Test Case ID	CYB_005	Test Case Description	Test Functionality – Data Pre-processing		
Created By	Dhruv	Reviewed By		Version	1.3

QA Tester's Log

Tester's Name	Dan	Date Tested	27-08-2020	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	System State – Check Trip Start
2	
3	
4	

S #	Test Data
1	Battery Power and Energy
2	Drivetrain Power and Energy
3	Auxiliary Power and Energy
4	Route distance

Test Scenario Verify that data pre-processing is done before it can be used further in the system functionality

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Check System State	System State – Energy Estimation	Expected	Pass
2	Check Log File – (for first 100 samples) Access the data as described in the test data section in this test sheet	The test data must not be NaN.	Expected	Pass
3	Check Log file message - DataPreprocessingOK	True	Expected	Pass

Test Case ID	CYB_006	Test Case Description	Test Functionality – Resetting of state signals		
Created By	Dhruv	Reviewed By		Version	1.3

QA Tester's Log

Tester's Name	Dan	Date Tested	27-08-2020	Test Case (Pass/Fail/Not Executed)	Suspended

S #	Prerequisites:
1	System State – Check Trip Start
2	
3	
4	

S #	Test Data
1	Time Route (timeRoute)
2	Distance traveled for the Route (sRoute)
3	Energy consumption at battery terminal (EbatRoute)
4	Energy consumption at drivetrain terminal (EdriveRoute)
5	Energy consumption by auxiliaries (EAuxRoute)

Test Scenario Verifying that signals specified in the test data section are reset to zero as the route starts.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Test CYB_001 to CYB_005 must be performed and passed	Expected results of these steps must be verified	3 is Suspended	Suspended
2	Check System State	System State – Energy Estimation	Expected	Pass
3	Check the Log file	The test data mentioned in this test sheet should start from zero.	Expected	Pass
4	Check log file message - DriveTrainValuesZeroOK	True	Expected	Pass

Test Case ID	CYB_007	Test Case Description	Test Functionality – Estimations from drivetrain		
Created By	Dhruv	Reviewed By		Version	1.3

QA Tester's Log

Tester's Name	Dan	Date Tested	27-08-2020	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	Start Route Flag is active
2	
3	
4	

S #	Test Data
1	Drivetrain energy estimate by the end of the trip (energyDriveTrip)
2	Current drivetrain energy estimate (energyDriveTripCurrent)
3	Actual drivetrain energy estimate (energyDriveActual)
4	Correction Gain (massEstimate)

Test Scenario Verifying the drivetrain energy estimation and correction algorithm

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Check if the estimation function is active	eState flag is active (eState = 1)	Expected	Pass
2	Check System State	System State – Energy Estimation	Expected	Pass
3	Check the log file (for first 300 samples) Observe the energyDriveTripCurrent	The estimated signal must track the energyDriveActual signal (absolute error is < 0.1 kwH)	Deviation current - actual: Avg: 0.018093 WC: 0.037628	Pass
4	Check the log file (for first 300 samples) Observe the correction gain	The massEstimate must not be a constant signal	Expected	Pass
5	Check log file message– EstimationDrivetrainOK	True	Expected	Pass
6	Frequency of estimated signal	~ 10 Hz	Expected	Pass

Test Case ID	CYB_008	Test Case Description	Test Functionality – Estimations from auxiliary		
Created By	Dhruv	Reviewed By		Version	1.3

QA Tester's Log

Tester's Name	Dan	Date Tested	27-8-2020	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	System State – Check Trip Start
2	
3	
4	

S #	Test Data
1	Auxiliary energy estimate by the end of the trip (energyAuxTrip)
2	Current drivetrain energy estimate (energyAuxTripCurrent)
3	Actual drivetrain energy estimate (energyAuxActual)
4	Correction Gain (corrGain)

Test Scenario Verifying the auxiliary energy estimation and correction algorithm

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Check if the estimation function is active	eState flag is active (eState = 1)	Expected	Pass
2	Check System State	System State – Energy Estimation	Expected	Pass
3	Check the log file (for first 300 samples) Observe the energyAuxTripCurrent	The estimated signal must track the energyAuxActual signal (absolute error is < 0.1 kwH)	Deviation current - actual: Avg: 0.022183 WC: 0.039699	Pass
4	Check the log file (for first 300 samples) Observe the correction gain	The corrGain must not be a constant signal	Expected	Pass
5	Check log file message – EstimationAuxiliaryOK	True	Expected	Pass
6	Frequency of estimated signal	~ 10 Hz	Expected	Pass

Test Case ID	CYB_009	Test Case Description	Test Functionality – Total estimations		
Created By	Dhruv	Reviewed By		Version	1.3

QA Tester's Log

Tester's Name	Dan	Date Tested	27-08-2020	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	System State – Check Trip Start
2	
3	
4	

S #	Test Data
1	Total energy estimate by the end of the trip (energyTrip)
2	Current total energy estimate (energyTripCurrent)
3	Actual total energy estimate (energyActual)
4	Error in between estimation and actual measure (error)

Test Scenario Verifying the total energy estimation and correction algorithm

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Check if the estimation function is active	eState flag is active (eState = 1)	Expected	Pass
2	Check System State	System State – Energy Estimation	Expected	Pass
3	Check the log file (for first 300 samples) Observe the energyTripCurrent	The signal must track the energyActual signal (the absolute error is < 0.1 kWh).	Deviation current - actual: Avg: 0.017663 WC: 0.041299	Pass
4	Observe the Error signal	The error must always be bounded (20 %). Exception for the first 10 estimation loops.	Expected	Pass
5	Check log file message – EstimationTotalOK	True	Expected	Pass
6	Frequency of estimated signal	~ 10 Hz	Expected	Pass

Test Case ID	CYB_010	Test Case Description	Test Functionality – Reliability Function		
Created By	Dhruv	Reviewed By		Version	1.3

QA Tester's Log	Test for Analysis

Tester's Name	Dan	Date Tested	27-08-2020	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	The estimation data from the total energy estimate and correction function
2	System State – Energy Estimation
3	
4	

S #	Test Data
1	reliability
2	
3	
4	

Test Scenario	Verify the functionality of reliability function – Reliability of estimation must improve as more data is being available

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Check if the estimation function is active	eState flag is active (eState = 1)	Expected	Pass
2	Check System State	System State – Energy Estimation	Expected	Pass
3	Check the value of reliability after estimation has stopped.	Reliability is above 80	Expected	Pass
4	Check log file message – reliabilityOK	True	Expected	Pass

Test Case ID	CYB_011	Test Case Description	Test Functionality – Erate Profile		
Created By	Dhruv	Reviewed By		Version	1.3

QA Tester's Log	Test for Analysis

Tester's Name	Dan	Date Tested	27-08-2020	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	The data required is drivetrain power, velocity, and the mass estimate of the entire route
2	
3	
4	

S #	Test Data
1	Erate
2	
3	
4	

Test Scenario	Verify that the updated Erate is generated according to mass variations over the entire route

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Check if the estimation function is active	eState flag is active (eState = 1)	Expected	Pass
2	Check System State	System State – Energy Estimation	Expected	Pass
3	Check the log file (for first 300 samples) Observe the Erate data signal	The signal values must be non-nan when the estimations are happening	Expected	Pass
4	Check log file message – EprofileDataOK	True	Expected	Pass

Test Case ID	CYB_012	Test Case Description	Test Functionality – Initiate stop of estimation/route		
Created By	Dhruv	Reviewed By		Version	1.3

QA Tester's Log

Tester's Name	Dan	Date Tested	27-08-2020	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	Defined specific start and stop GPS coordinates (parameters)
2	System State – Energy Estimation
3	
4	

S #	Test Data
1	Start Route
2	
3	
4	

Test Scenario Verify that the system is able to identify the stopping of the specifically selected route

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Check Log file – for eState Value	eState flag is not active (eState = 0) once the trip has stopped	Expected	Pass
2	Check Log file – for printed GPS coordinates when the trip has stopped	The stop flag of the route will be raised at the instance when actual GPS coordinates are in the range of 0.0004 in lat and long and 50° in heading to GPS parameters	Expected	Pass
3	Check System State	System State – Energy Estimation. Note: the state should not be Check Trip Start or Post Processing immediately after the trip end. The program must first store TP data, check the number of TPs, and optionally post process.	Expected	Pass

Test Case ID	CYB_013	Test Case Description	Test Functionality – Output Message		
Created By	Dhruv	Reviewed By		Version	1.3

QA Tester's Log

Tester's Name	Dan	Date Tested	27-08-2020	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	System State – Energy Estimation System
2	
3	
4	

S #	Test Data
1	Normative Energy
2	Projected Energy
3	Reliability
4	

Test Scenario Verify that the output data is being available by the system

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Check System State	System State – Energy Estimation	Expected	Pass
2	Normative Energy	Should be a non-zero value throughout the trip (estimation state is 1)	Expected	Pass
3	Projected Energy	Should be a non-zero value throughout the trip (estimation state is 1)	Expected	Pass
4	Reliability	Should be a non-zero value throughout the trip (estimation state is 1)	Expected	Pass
5	Check Message: EstimationSendOK		Expected	Pass

Test Case ID	CYB_014	Test Case Description	Test Functionality – Trip Data Storage		
Created By	Dhruv	Reviewed By		Version	1.3

QA Tester's Log

Tester's Name	Dan	Date Tested	28-08-2020	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	System State – Energy Estimation
2	
3	
4	

S #	Test Data
1	Erate
2	time
3	s (distance)
4	Paux (Trip)
5	energyActual (Trip)

Test Scenario Verify that the relevant data is being stored during the estimations over the route

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Check if the estimation function is inactive	eState flag is inactive (eState = 0)	Expected	Pass
2	Check System State	System State – Energy Estimation	Expected	Pass
3	The test data mentioned in this test sheet must be saved/stored. Files must be manually checked.	Check if files are created in the memory with names: - TP_sRoute[i].data - TP_time[i].data - TP_timeAux[i].data - TP_ERate[i].data - TP_Paux[i].data - TP_singleVals[i].data Where i = {1,2}	Expected	Pass

4	Check log file message – ErateProfileDataOK (test 11)	True	Expected	Pass
5	Check log file message – TripDataHealthOK (I.e. Vector values are not NaN and vector sizes are non-zero. Etrip_measured is non-zero and not NaN)	True	Expected	Pass

Test Case ID	CYB_015	Test Case Description	Test Functionality – Post Processing (1)		
Created By	Dhruv	Reviewed By		Version	1.3

QA Tester's Log

Tester's Name	Dan	Date Tested	28-07-2020	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	The complete trip data available
2	Check Counter: No. of trip == 2
3	
4	

S #	Test Data		
1	Erate	5	energyTrip (Actual consumed)
2	time		
3	s (distance)		
4	Paux		

Test Scenario Verify the data used in post-processing and the updated individual data structure file are available in the memory

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Check System State	System State – Post Processing	Expected	Pass
2	Check log file message – tripDataHealthOK	True	Expected	Pass
3	- TP_sRoute1.data - TP_time1.data - TP_timeAux1.data - TP_ERate1.data - TP_Paux1.data - TP_singleVals1.data	The file must be available	Expected. Must be checked manually (logging "ls" command fails).	Pass
4	- TP_sRoute2.data - TP_time2.data - TP_timeAux2.data - TP_ERate2.data - TP_Paux2.data - TP_singleVals2.data	The file must be available	Expected. Must be checked manually (logging "ls" command fails).	Pass

Test Case ID	CYB_016	Test Case Description	Test Functionality – Post Processing (2)		
Created By	Dhruv	Reviewed By		Version	1.3

QA Tester's Log

Tester's Name	Dan	Date Tested	28-08-2020	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	Check Counter: No. of trip == 2
2	
3	
4	

S #	Test Data
1	
2	
3	
4	

Test Scenario Verify that the averaged data structure file is available in the memory

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Check System State	System State – Post Processing	Expected	Pass
2	Check log file message – tripDataHealthOK	True	Expected	Pass
3	EP_sRoute.data EP_time.data EP_timeAux.data EP_ERate.data EP_Paux.data EP_singleVals.data	The file must be available	Expected	Pass

Appendix

List of Abbreviations/Definition

Abbreviation	Description
GPS	Global Positioning System
Trip	One trip cycle is journey covered from Station (Eindhoven) to Airport (Eindhoven)
Erate	Energy consumed by the vehicle per unit mass per unit distance traveled
NaN	Not a Number
eState	System Status
WC	Worst Case
Base EP	Base Erate Profile: Database reference profile created offline with 16 trip cycle data
EP	Erate Profile: Database reference profile created online after every trip cycle
TP	Trip Profile: Averaged database reference profile created online after every second trip cycle

References

[1] D. K. Prasad, *ISTQB Certification Study Guide*, ISBN 978-81-7722-711-6 ed., Wiley, 2008.

[2] I. 24765:2010(E), *ISO/IEC/IEEE International Standard - Systems and software engineering.*, 2010, pp. 1-418.

F Appendix: HMI Aspects

A Human-Machine Interface (HMI) (or user interface or dashboard) was designed in order to connect a user with the system. This allows the user to interact with a system in many ways while giving important information about the system. Some of these ways in which an HMI can be used are as follows:

- Visually inspect the data on the display.
- Tracking the trends in system performance.
- Overseeing the Key Performance Index's (KPI's) of the system.
- Monitoring system input and outputs.

The HMI designed for the application of real-time energy estimation is shown in Figure F.1.

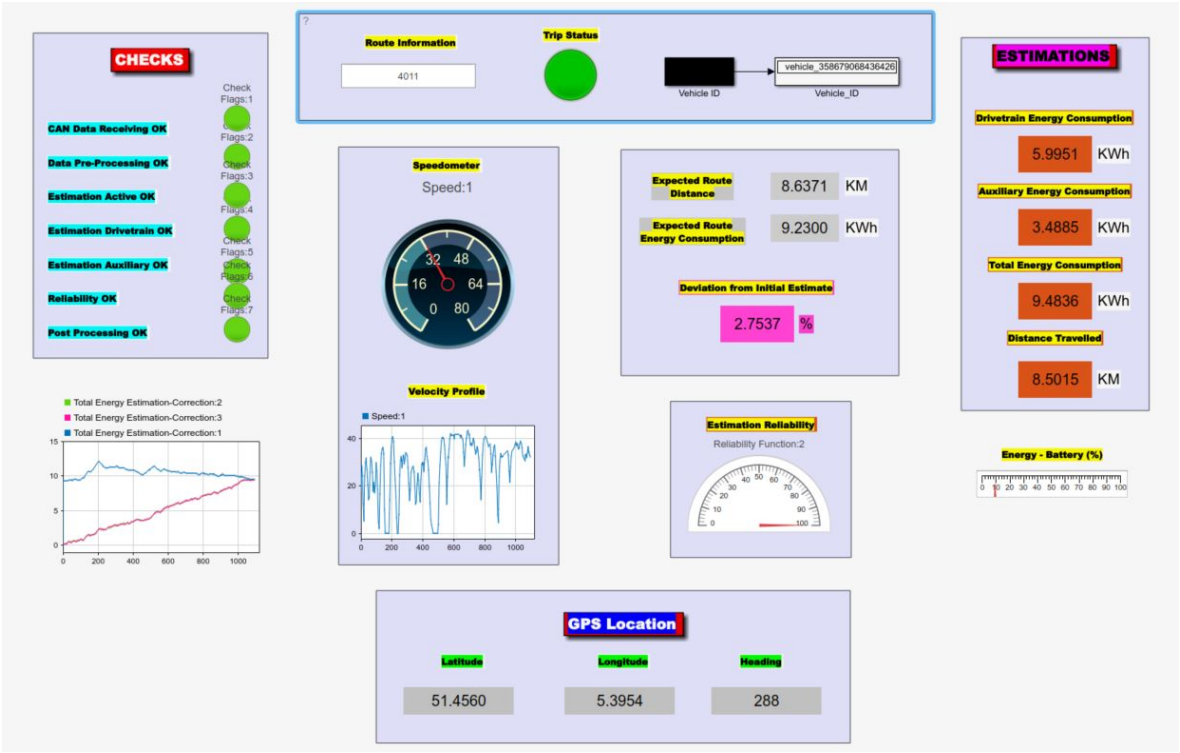


Figure F.1: HMI: Real-time Energy Estimation System

This particular HMI has the following capabilities:

- **Route and Vehicle Information:** In the top (middle) of the HMI can be seen the the informa-

tion regarding the route on which bus is travelling. This comes in the form of Route ID. Also unique vehicle ID is displayed in the same panel.

- **System Status:** In the same panel an LED is available with name *Trip Status* giving the information regarding the activation status of the real-time energy estimation system. When it is *green* the energy estimation system is switched *ON* and is switched *OFF* when *red*.
- **System Functionality Checks:** In the top (left) part of HMI, a panel is made available keeping a check on the various functionalities of the system. A warning is raised with LED turning *red* when something goes wrong.
- **Offline Estimations:** In the middle (right) of the HMI, expected pre-determined (offline) estimates and distance to be travelled by bus are made available
- **Online Estimations:** In the top (right) part of HMI, a panel is made available keeping the track of the estimated energy during the trip cycle and also about the distance travelled during that trip cycle. A visualization of these estimations are also made available in the bottom (left) part of the HMI.
- **Location Tracking:** In the bottom of the HMI the live GPS location of the bus can be tracked.
- **Velocity Profile Tracking:** In the middle (left) of the HMI the current velocity profile of the bus can be traced.
- **Reliability Meter:** Reliability meter is made available in order to built the trust of the user on the estimated energy numbers. The system is capable of checking the reliability of the estimates based upon the errors occurring during estimations.

G Appendix: Project Management Documentation

G.1 Stakeholder Register

Stakeholder Register
PDEng Project - Cloud Your Bus

Name	Title	Organization	Role	Power (H/L)	Interest (H/L)	Communication Requirement	Expectations	Concerns
Igo Besselink	Professor	TU/e	Project Supervisor	H	H	(bi)Weekly progress meeting, Project Steering Group Meeting, Minutes of meeting, Monthly Progress Report	develop energy consumption model of electric bus	Understand the energy usage pattern of electric busses. Accounting the uncertainties affecting energy consumption. Identify the relevant standardized messages to be sent to the cloud.
Kristian Winge	CEO	Sycada	Project Supervisor	H	H	Project Steering Group Meeting, Minutes of meeting, Monthly Progress Report	create an interface between energy consumption model and Sycada wireless embedded system	Real time data collection. Analysis of status of electric vehicle and batteries, charging point infrastructure. Develop driving coaching techniques to optimize the drive style.
Rogier Mulder	CTO	Sycada	Project Supervisor	H	H	Project Steering Group Meeting, Minutes of meeting, Monthly Progress Report		
Peter Heuberger	Professor	TU/e	Project Guide	M	M	Project Steering Group Meeting, Minutes of meeting, Monthly Progress Report	trainee successfully meet the concerns of stakeholders	
		ICRON Technologies	Develop planning, scheduling and optimization solutions	L	L		use energy consumption model for advanced planning, scheduling and optimization	To improve the dynamic planning and avoid uncertain issues.
		Owasys	Provide hardware solutions	L	L		use the inputs to develop technologies to upgrade the advanced wireless devices	
		Bus OEM's		L	L		receive inputs to improve design and testing of smart electric busses	
		Bus Operators		L	M		improve the process of dynamic planning	facilitate planning of operations of electric busses in real time
		Driver's		L	L			easy to use system

G.2 Project Development Process

Project Development Process
PDEng Project - Cloud Your Bus

Project Process: Agile + V-Model

AGILE:

- The project process method employed is agile. The work will be carried out throughout the project in the sprints (time based).
- The sprints are two weeks long and tasks carried out in these sprints are measured in days.
- The tasks in the sprint are associated with specific acceptance criteria and will be reviewed by the stakeholders.
- The scrum board is maintained throughout the project in all sprints. The tasks will be updated with the start of the sprint.
- The tool used for scrum is Microsoft based Azure DevOps and can be accessed via < https://dev.azure.com/djagga0346/Cloud%20Your%20Bus >
- Classification of Scrum tasks:
    - ► Backlog: Collection of tasks to be done in the sprint. This will be populated before starting the sprint.
    - ► In Progress: Tasks from the backlog which are taken up during the sprint falls under this category.
    - ► In Review: After the task is completed it is reviewed by the stakeholder.
    - ► Done: After the review is completed and necessary actions are taken the task will be moved to section Done.
    - ► Blocked: The tasks that do not receive the stakeholder support in time, the tasks that run into technical dead ends, and the tasks that have a low priority (nice to have but not necessary) that can't be taken by anyone in the team ends in Blocked section.

V-MODEL:

- The V-Model describes the system development process . It describes the activities that has to be performed and the results that have to be produced during the product development.
- The left side of the V-Model summarize the decomposition of the requirements and creation of the system specifications, whereas the right side of the model represents integration of parts and validation using testing.

G.3 Risk Management - Risk Register

Risk Register
PDEng Project - Cloud Your Bus

Risk Type	Description	I	P	Risk Priority Number	Impact on Project	Mitigation Plan	Contengency Plan
Technical	Handling bugs while implementation of the project on Linux based operating system.	5	3	15	Delay in project progress	Testing of the code simultaneously on the linux based virtual machine	Debugging help from Sycada or Tue colleagues
Technical	Handling bugs while implementation of model in C++.	5	4	20	Delay in project progress	Generating C++ code using Matlab C compiler application	Debugging help from Sycada or Tue colleagues
Organizational	Delay in testing on system level.	4	2	8	Delay in project progress	Planning accordingly	No contengency plan
Scope	Extension of scope (Implementing a more complex model).	4	1	4	Extension of project	Freezing scope and requirements in early phase of the project	No contengency plan
Technical	Delay in data availability for learning model parameters.	5	3	15	Without data, the implementation is difficult and can lead to delay in process	Use of data from LUPO vehicle (TU/e)	No contengency plan
Technical	Not enough memory space availability on HW for saving data and processing the algorithm.	4	2	8	Failure of HW	Using compact but sufficient data, Hardware in the loop testing in early stages of testing	
Technical	Under performing system due to generic model parameters.	4	4	16	Performance Requirement not met	Use large data for better learning	Negotiate on the performance requirement

Risk Scoring Matrix
PDEng Project - Cloud Your Bus

Scoring matrix: as defined by the formulas in the threats

			Impact				
			Very low	Low	Moderate	High	Very high
		Score	1	2	3	4	5
Probability	Highly likely	5	5	10	15	20	25
	Likely	4	4	8	12	16	20
	Moderate	3	3	6	9	12	15
	Unlikely	2	2	4	6	8	10
	Highly unlikely	1	1	2	3	4	5

Reference: < A risk matrix for risk managers, National Patient Safety Agency (NHS) >
< BSI British Standards. Risk Managemnt Vocabulary - Guidelines of Use in Standards. (2002). PD ISO/IEC guide 73:2002 >

Green = 1-4 Amber = 5-14 Red = >=15

G.4 Quality Management Process

Quality Management Process
PDEng Project - Cloud Your Bus

Quality Management	It ensures that the system is consistent. It does not only check the quality of the end-product but also ensures that the process is smooth. It directly measures the fitness or the performance of the intended functionality of the system.
Best Coding Practices	To ensure the quality in this project the SW is maintained by using the best coding practices as described in this section.
Commenting	The following comments can be used as a description: ● Name of the module ● Purpose of the module ● Description of the module ● Original Author ● Modifications with date and reason to modify
Naming Conventions	Usage of proper naming conventions. Example: ● TruckWeight
Simple	The code must be kept as simple as possible so it is easily understood by the next coder using it.
Portability	Program code must not have "hard-coded" values refereeing to environmental parameters, this can include absolute file paths, file names, user names, IP Addresses, URLs, UDP/TCP ports. Otherwise, the application will not run on a host that has a different design than anticiapted. Parameterize these variables and configure them for the hosting environment outside of application.
Testing	It is an integral part of Software development. Test cases must be planned before the coding starts, and test cases are developed while the application is being designed and coded. Unit testing can be performed to check the functionality of the individual block in the code.
Documents	To ensure the quality of documents in this project. The documents are reviewed by stakeholder and project supervisior.

Reference: < Meek, Brian; Heath, Patricia (1980), Guide to Good Programming Practice, Ellis Horwood, Wiley, p. 15 >

G.5 Version Control Management Process

Version Control
PDEng Project - Cloud Your Bus

Term	Description
Version Control	● A system is made to record all necessary changes made in a file or set of files over time, so things can be recalled to specific versions later on <if and when necessary>. For this project a MATLAB code repository is made on **git** on **TU/e** server. The GitLab is kept private and is not shared with anyone. ► The link to the GitLab repository is: < https://gitlab.tue.nl/20184719/cloud-your-bus > ● The other Documents are backed-up on the TU/e drive <Surf Drive>. The access to this folder on the drive is also kept to be private, but limited access can be given to stakeholders upon request. ► The link to the Surf Drive repository is: < https://surfdrive.surf.nl/files/index.php/apps/files/?dir=/Final%20Project%20-%20Module%205&fileid=6242474450 >

H Appendix: Software Manual and Function Library

This manual describes the steps for building and running the provided software package. The Matlab prototype version software can be built and run on any computer with Matlab/Simulink installed on it. It does not require any other external Matlab toolbox package to compile this software library.

H.1 Pre-Requisites

1. 1 laptop with sufficient memory and processing power. (Recommended Specification > 8GB RAM)
2. Clone of the repository. (Is available on TU/e Github and TU/e Research Drive)
3. MATLAB/Simulink. (This package is developed with version 2020a)

H.2 Building and Running

Unzip the provided software package on the local drive on your computer. The system software package unzips a folder named *"Cloud Your Bus - Energy Estimation for EV"*. This folder have several sub-folders in it and some Matlab scripts and Data (.mat) files in the main folder. The run script for the **Real-time Energy Estimation System** is *"EnergyEstimationMain"*. This file can be run directly to simulate the test scenarios directly. To save files in the directory during the simulation it is required to update the directory name in the run script to your local computer directory name.

```
% This file is the 'Run Script' file to prepare data for the Simulink ...
    (Energy Estimation System) file to run

%   Author: Dhruv Jagga
%   Project: Source Code - Cloud Your Bus
%   email: d.jagga@tue.nl
%   Date: 10-12-2019;
%   Revised: 06-08-2020

% Clear Workspace
clc;
clear;
close all;

% Create path in the current directory
directory.current = pwd;                    % returns path of current ...
    directory
addpath(genpath('./Offline Estimate'));     % add the folder and ...
```

```
    subfolders to the working path (workspace)
addpath(genpath('./Parameter'));            % add the folder and ...
    subfolders to the working path (workspace)
addpath(genpath('./Post Processing'));      % add the folder and ...
    subfolders to the working path (workspace)

% Load test data profile
load('testDataRoute401_A_2Cycle.mat');
%load('validationDataRoute401_A_2Cycle.mat');

% Vehicle ID
vehicleID = sym(358679068436426);

% Establishing a new data (Just for Simulation - Has to be extracted from ...
    Bison API)
% Route Information
routeInfo  = 4011;

% Extracting data from the data structure

time        = m.data(:,1)-m.data(1,1);
dt          = [0;diff(time)];                           % used as ...
    weighted sample space

lat         = m.data(:,2);                              % degree
lon         = m.data(:,3);                              % degree
head        = m.data(:,4);                              % degree

v           = m.data(:,5);                              % kmph
v           = v * (5/18);                               % mps

vin         = m.data(:,6);

VoltBat     = m.data(:,7);                              % V
AmpBat      = m.data(:,8);                              % A

VoltDrive   = m.data(:,9);                              % V
AmpDrive    = m.data(:,10);                             % A

PressFA     = m.data(:,11);                             % KPa
PressRA     = m.data(:,12);                             % KPa
PressAA     = m.data(:,13);                             % KPa

% Pre-processing the extracted data
s           = cumsum(v.*dt);                            % m

PowerBat    = (VoltBat.*AmpBat);                        % W
EnergyBat   = cumsum(PowerBat.*dt)/(3.6e6);             % KWh

PowerDrive  = (VoltDrive.*AmpDrive);                    % W
EnergyDrive = cumsum(PowerDrive.*dt)/3.6e6;             % KWh

PowerAux    = PowerBat - PowerDrive;                    % W
EnergyAux   = EnergyBat - EnergyDrive;                  % KWh

% Convert the loaded data as timeseries data
time       = timeseries(m.data(:,1)-m.data(1,1),m.data(:,1)-m.data(1,1)); ...
```

```
                          % [s]
Vx          = timeseries(v,m.data(:,1)-m.data(1,1)); ...
                                                  % [m/s]
VoltBat     = timeseries(VoltBat,m.data(:,1)-m.data(1,1)); ...
                                                % [V]
AmpBat      = timeseries(AmpBat,m.data(:,1)-m.data(1,1)); ...
                                                 % [A]
VoltDrive   = timeseries(VoltDrive,m.data(:,1)-m.data(1,1)); ...
                                               % [V]
AmpDrive    = timeseries(AmpDrive,m.data(:,1)-m.data(1,1)); ...
                                                % [A]
PressFA     = timeseries(PressFA,m.data(:,1)-m.data(1,1)); ...
                                                % [KPa]
PressRA     = timeseries(PressRA,m.data(:,1)-m.data(1,1)); ...
                                                % [KPa]
PressAA     = timeseries(PressAA,m.data(:,1)-m.data(1,1)); ...
                                                % [KPa]

s           = timeseries(s,m.data(:,1)-m.data(1,1)); ...
                                                  % [KPa]
PowerBat    = timeseries(PowerBat,m.data(:,1)-m.data(1,1)); ...
                                                % [KPa]
EnergyBat   = timeseries(EnergyBat,m.data(:,1)-m.data(1,1)); ...
                                               % [KPa]
PowerDrive  = timeseries(PowerDrive,m.data(:,1)-m.data(1,1)); ...
                                              % [KPa]
EnergyDrive= timeseries(EnergyDrive,m.data(:,1)-m.data(1,1)); ...
                                             % [KPa]
PowerAux    = timeseries(PowerAux,m.data(:,1)-m.data(1,1)); ...
                                                % [KPa]
EnergyAux   = timeseries(EnergyAux,m.data(:,1)-m.data(1,1)); ...
                                               % [KPa]


% Preprocessing the GPS data (Replace the NaN value with the last recorded ...
    value)
for k = find(isnan(lat))
    lat(k) = lat(k-1);
end

for l = find(isnan(lon))
    lon(l) = lon(l-1);
end

for n = find(isnan(head))
    head(n) = head(n-1);
end

GPSlat  = timeseries(lat,m.data(:,1)-m.data(1,1)); ...
                                                % [deg.]
GPSlon = timeseries(lon,m.data(:,1)-m.data(1,1)); ...
                                                 % [deg.]
GPShead = timeseries(head,m.data(:,1)-m.data(1,1)); ...
                                                % [deg.]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load relevent trip data
```

```
% Load Erate and other necessary data profiles for energy estimation and ...
    correction
trip = routeData(routeInfo);
trip = trip.trip;

% Load start and stop coordinates for the relevent trip
[start,stop] = routeParameter(routeInfo);

% Perform the simulation
 out=sim('EnergyEstimationCorrection.slx');

% Post processing of data and updating database
% Step 1 - Extract Data segments of the different cycles of the route
[ErateExt,timeExt,sExt,PauxExt,energyTripExt,checkflag] = extractData(out);

% Step 2 - Make individual data structures for these subsequent trips
[tripPost] = dataStruct(ErateExt,timeExt,sExt,PauxExt,energyTripExt,out);

% Save the trip data structure into the Output Data Profile Folder in ...
    directory
savdir = 'C:\Users\20184719\Documents\Final Project - Module 5\Source ...
    Code\MATLAB\Cloud Your Bus - Energy Estimation System for EV\Post ...
    Processing\DataProfile-Route_401';

% Save trip data - Route 401 - Cycle 1
if tripPost.trip1.tripDataHealth == true
    trip = tripPost.trip1;
    save(fullfile(savdir,'ErateProfile401_A_01'),'trip');
end

% Save trip data - Route 401 - Cycle 2
if tripPost.trip2.tripDataHealth == true
    trip = tripPost.trip2;
    save(fullfile(savdir,'ErateProfile401_A_02'),'trip');
end

% Step 3 - Do averaging of these data structures to make a unique data ...
    structure
tripUpdate = avgDataStruct(out,tripPost);

savdirUp = 'C:\Users\20184719\Documents\Final Project - Module 5\Source ...
    Code\MATLAB\Cloud Your Bus - Energy Estimation System for ...
    EV\Parameter\UpdatedProfile';

if tripUpdate.tripDataHealth == true
    trip = tripUpdate;
    save(fullfile(savdirUp,'UpdatedErateProfile401_A'),'trip');
end

% Checking for Absolute error of respective cycles of the trips made (For ...
    Analysis purpose only)
absError = absoluteError(out);
```

Besides the run script there are some other scripts used for data visualization and plotting and can be used once the entire run is completed. Other than the script files this folder also contains already prepared data files, these data files emulate the CAN-Bus data. The test and validation data cycles

are made available in this folder. There exist then the folders in the main folder. The content of each folder is explained below:

- **Results - Online Estimation:** This folder contains the saved work-spaces of the runs conducted for the test scenarios (Software-in-the-Loop Testing) explained in Chapter 6.
- **HIL Test:** This folder contains the saved files form the embedded device to analyze the results of runs available from the Hardware-in-the-Loop Testing
- **Plots:** This folder contains the plot figures from these results for data visualization and analysis.

Now apart from these 3 folder, there are some folders which are necessarily required to run the simulations for the real-time energy estimation system and are regarded as important folders which compose the software toolbox package (functional library)

- **Offline Estimate:** This folder is used to make the Offline estimations on the selected route and is important to create the *"Base Data Reference Profile"*
- **Parameter:** This folder contains the parameter information for the selected route. The *"Base Data Reference Profile"* and *"Updated Data Reference Profile"* are stored in this folder for the given route. This folder also contains two function scripts. *"routeData"* It stores the information of the data profiles for the specific routes and load subsequent data related to given route ID. In simulation the 2 different routes can be identified and selected based upon this signal route ID information.

```
function trip = routeData(routeInfo)

%   Author: Dhruv Jagga
%   Project: Source Code - Cloud Your Bus
%   email: d.jagga@tue.nl
%   Date: 10-12-2019;
%   Revised: 06-08-2020

% Available routes

route401_A = 4011;
route401_B = 4012;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Creating an Empty trip data structure

trip0.mveh   = 19000;                                          % Kg
trip0.s      = 0;
trip0.sRoute = zeros(10000,1);

trip0.time      = zeros(10000,1);

% Saving updated Erate profile
trip0.Erate = zeros(10000,1);

% Saving updated Paux profile
trip0.Paux = zeros(10000,1);

% Creating Drivetrain trip profile
trip0.EuseDrive = zeros(10000,1);
```

```
32
33 % Creating Auxiliary trip profile
34 trip0.EuseAux   = zeros(10000,1);
35 trip0.timeAux   = zeros(10000,1);
36
37 trip0.Etrip_measured = 0;
38 trip0.tripDataHealth = false;
39
40 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
41 % Operating directory to look for relevant files
42 opdir = 'C:\Users\20184719\Documents\Final Project - Module 5\Source ...
       Code\MATLAB\Cloud Your Bus - Energy Estimation System for ...
       EV\Parameter\UpdatedProfile';
43
44 % Check the presence of relevent files in the directory
45 statusUpdatedDataFile = ...
       isfile(fullfile(opdir,'UpdatedErateProfile401_A.mat'));
46
47 % Load the relevent data set
48
49 switch(routeInfo)
50     case route401_A
51         if statusUpdatedDataFile == true
52             trip = load('UpdatedErateProfile401_A.mat');
53         else
54             trip = load('ErateAvgRoute401_A.mat');
55         end
56     case route401_B
57         trip = load('ErateAvgRoute401_B.mat');
58     otherwise
59         trip = trip0;
60 end
61
62 end
```

Similarly there exist another function *"routeParameter"* which has the information regarding the starting and stopping coordinates of the respective routes.

```
1  function [start,stop,routeState] = routeParameter(routeInfo)
2
3  %   Author: Dhruv Jagga
4  %   Project: Source Code - Cloud Your Bus
5  %   email: d.jagga@tue.nl
6  %   Date: 10-12-2019;
7  %   Revised: 06-08-2020
8
9  route401_A = 4011;
10 route401_B = 4012;
11
12 routeState = routeInfo;
13
14 switch(routeInfo)
15     case route401_A
16
17     % Define starting point and ending point of the route using GPS ...
           coordinates
```

```
18      % [The below entered route coordinates is for line 401 moving in ...
            between Eindhoven Station and Airport]
19      % Direction - Station to Airport
20
21      start.lat = 51.4442;                                        % ...
            Physically/manually entered GPS latitude
22      start.lon = 5.4788;                                         % ...
            Physically/manually entered GPS longitude
23      start.head = 278;                                           % ...
            Physically/manually entered GPS heading
24
25      % Allowing variation/deviation with following range [in degrees] ...
            around GPS coordinate
26      start.latRange = 0.0004;                                    % ...
            Physically/manually entered
27      start.lonRange = 0.0004;                                    % ...
            Physically/manually entered
28      start.headRange= 50;                                        % ...
            Physically/manually entered
29
30      stop.lat = 51.4567;                                         % ...
            Physically/manually entered GPS latitude
31      stop.lon = 5.3933;                                          % ...
            Physically/manually entered GPS longitude
32      stop.head = 309;                                            % ...
            Physically/manually entered GPS heading
33
34      % Allowing variation/deviation with following range [in degrees] ...
            around GPS coordinate
35      stop.latRange = 0.0004;                                     % ...
            Physically/manually entered
36      stop.lonRange = 0.0004;                                     % ...
            Physically/manually entered
37      stop.headRange= 50;                                         % ...
            Physically/manually entered
38
39      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
40
41      case route401_B
42
43      % Direction - Airport to Station
44
45      start.lat = 51.4501;                                        % ...
            Physically/manually entered GPS latitude
46      start.lon = 5.4027;                                         % ...
            Physically/manually entered GPS longitude
47      start.head = 154;                                           % ...
            Physically/manually entered GPS heading
48
49      % Allowing variation/deviation with following range [in degrees] ...
            around GPS coordinate
50      start.latRange = 0.0003;                                    % ...
            Physically/manually entered
51      start.lonRange = 0.0003;                                    % ...
            Physically/manually entered
52      start.headRange= 5;                                         % ...
            Physically/manually entered
```

```
53
54      stop.lat = 51.4442;                                          % ...
            Physically/manually entered GPS latitude
55      stop.lon = 5.4788;                                           % ...
            Physically/manually entered GPS longitude
56      stop.head = 72;                                              % ...
            Physically/manually entered GPS heading
57
58      % Allowing variation/deviation with following range [in degrees] ...
            around GPS coordinate
59      stop.latRange = 0.0004;                                      % ...
            Physically/manually entered
60      stop.lonRange = 0.0004;                                      % ...
            Physically/manually entered
61      stop.headRange= 5;                                           % ...
            Physically/manually entered
62
63      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
64
65      otherwise
66      start.lat = 0;                                               % ...
            Physically/manually entered GPS latitude
67      start.lon = 0;                                               % ...
            Physically/manually entered GPS longitude
68      start.head = 0;                                              % ...
            Physically/manually entered GPS heading
69
70      % Allowing variation/deviation with following range [in degrees] ...
            around GPS coordinate
71      start.latRange = 0;                                          % ...
            Physically/manually entered
72      start.lonRange = 0;                                          % ...
            Physically/manually entered
73      start.headRange= 0;                                          % ...
            Physically/manually entered
74
75      stop.lat = 0;                                                % ...
            Physically/manually entered GPS latitude
76      stop.lon = 0;                                                % ...
            Physically/manually entered GPS longitude
77      stop.head = 0;                                               % ...
            Physically/manually entered GPS heading
78
79      % Allowing variation/deviation with following range [in degrees] ...
            around GPS coordinate
80      stop.latRange = 0;                                           % ...
            Physically/manually entered
81      stop.lonRange = 0;                                           % ...
            Physically/manually entered
82      stop.headRange= 0;                                           % ...
            Physically/manually entered
83  end
84  end
```

- **Online Estimation:** This feature contains the functions that are built in Simulink. It start with the, Identification of the start and stop of the trip cycle. The function used to perform this identification based upon the stored parameter is *"InitiateStartStop"*

```
function [startRoute, stopRoute] = ...
    fcn(time,GPSlat,GPSlon,GPShead,start,stop)

%   Author: Dhruv Jagga
%   Project: Source Code - Cloud Your Bus
%   email: d.jagga@tue.nl
%   Date: 10-12-2019;
%   Revised: 06-08-2020

% Define and Initializing the variables
% Initiated as persistent variables for retaining values of these ...
    variables
% to be used in the next time cycle

persistent routeStart routeStop xStart xStop
if isempty(routeStart)
    routeStart = false;                                              % ...
        vector which checks if current GPS coordinate is closer to ...
        start coordinate
end

if isempty(routeStop)
    routeStop = false;                                               % ...
        vector which checks if current GPS coordinate is closer to ...
        stop coordinate
end

if isempty(xStart)
    xStart = false;
end

if isempty(xStop)
    xStop = true;
end

global counterStart counterStop

startRoute = false;                                                  % ...
    checking the number of times the xStart state is active
stopRoute = false;                                                   % ...
    checking the number of times the xStop state is active

% Storing the values in variable to be used as previous value
routeStart_prev = routeStart;
routeStop_prev = routeStop;

% Compare current GPS coordinate with the specified GPS coordinates ...
    of Starting point
if abs(GPSlat - start.lat) < start.latRange && abs(GPSlon - ...
    start.lon) < start.lonRange && abs(GPShead - start.head) < ...
    start.headRange
    routeStart = true;
    routeStop = false;

    if (routeStart_prev == true && routeStart == true) && xStart == ...
        false && xStop == true                 % check if 2 ...
        consecutive points are closer to start coordinate
```

```
45          xStart = true; ...

                % then raise the flag that route has started
46          xStop = false;
47          startRoute = xStart;
48
49          if xStart == true
50              counterStart = counterStart + 1; ...
                                                                                    ...
                    % update the route counter
51              counterStop = counterStop;
52          end
53      end
54
55  % Compare the current GPS coordinate and heading with the Fixed ...
        stopping point GPS coordinate
56  elseif abs(GPSlat - stop.lat) < stop.latRange && abs(GPSlon - ...
        stop.lon) < stop.lonRange && abs(GPShead - stop.head) < stop.headRange
57
58      routeStart = false;
59      routeStop = true;
60
61      if (routeStop_prev == true && routeStop == true) && xStart == ...
            true && xStop == false                  % check if 2 ...
            consecutive points are closer to stop coordinate
62          xStart = false; ...
                                                                                    ...
                % then raise the flag that route has stopped
63          xStop = true;
64          stopRoute = xStop;
65
66          if xStop == true
67              counterStop = counterStop + 1; ...
                                                                % update ...
                    the route counter
68              counterStart = counterStart;
69          end
70      end
71
72  else
73  % No changes are made if both the conditions are not met
74
75      routeStart = false;
76      routeStop = false;
77  end
78
79  % Reset counter
80  if counterStart ≥ 100 && counterStop ≥ 100
81      counterStart = 0;
82      counterStop = 0;
83  end
84
85  end
```

The energy estimation algorithm comes into play after the system identifies the start of the route. The drive-train and auxiliary energy estimations are done separately but simultaneously using the Kalman Filter approach as discussed in Chapter 4. The function works similar so only

the one with drive-train energy estimation is shown in this manual.

```
%   Author: Dhruv Jagga
%   Project: Source Code - Cloud Your Bus
%   email: d.jagga@tue.nl
%   Date: 10-12-2019;
%   Revised: 06-08-2020

% Define and Initializing the variables
% Initiated as persistent variables for retaining values of these ...
    variables
% to be used in the next time cycle

persistent timeStamp sStamp EbatStamp EdriveStamp P massEstimatePrev

if isempty(timeStamp)
    timeStamp = 0.0;
end

if isempty(sStamp)
    sStamp = 0.0;
end

if isempty(EbatStamp)
    EbatStamp = 0.0;
end

if isempty(EdriveStamp)
    EdriveStamp = 0.0;
end

if isempty(P)
    P = eye(1);
end

if isempty(massEstimatePrev)
    massEstimatePrev = trip.mveh;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variable time stamp resetting

if startRoute == true && stopRoute == false
    timeStamp = time;
    sStamp = s;
    EbatStamp = Ebat;
    EdriveStamp = Edrive;
    massEstimatePrev = trip.mveh;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define States
eState1 = 0;
eState2 = 1;

persistent currentState
```

```
if isempty(currentState)
    currentState = eState1;
end

eState = currentState;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Switch to new state based upon value state register
switch(currentState)

    case eState1                            % State when the trip has not ... yet started
        timeRoute = 0;
        sRoute    = 0;
        EbatRoute = 0;
        EdriveRoute = 0;
        energyDriveTrip= 0;
        energyDriveTripCurrent = 0;
        energyDriveActual= 0;
        massEstimate = trip.mveh;
        y = 0;
        y_Estimate = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % State Check

        if startRoute == true && stopRoute == false
            currentState = eState2;
        else
            currentState = eState1;
        end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    case eState2                            % State when the trip has started
        timeRoute = time - timeStamp;
        sRoute    = s - sStamp;
        EbatRoute = Ebat - EbatStamp;
        EdriveRoute = Edrive - EdriveStamp;

        % Energy estimate powertrain
        EuseEstimate = cumsum(trip.Erate)*massEstimatePrev/3.6e6; ...
                                      % Energy estimate in KWh
        EuseFinal     = EuseEstimate(length(EuseEstimate)); ...
                                 % Final energy estimated needed for trip ... using the previous avg profiles
        EuseCurrent  = interp1(trip.sRoute,EuseEstimate,sRoute); ...
                           % Energy usage interpolated with respect to ... distance travelled

        % Total energy consumption estimate for the entire trip
        energyDriveTrip = EuseFinal; ...
                                                   % Energy consumtion ... Estimate given at the start of the trip for entire route
        energyDriveTripCurrent = EuseCurrent;

        % Actual energy consumption
        energyDriveActual = EdriveRoute; ...
```

```
                                                % Actual total energy ...
                consumed over the trip
103
104
105             % METHOD 4 - MASS ESTIMATE (Kalman Filter Approach)
106
107             y = EdriveRoute;
108
109             phi = cumsum(trip.Erate)/3.6e6;
110             phi = phi';
111
112             yEstimate = phi'*massEstimatePrev;
113             yEstimate = interp1(trip.sRoute,yEstimate,sRoute);
114
115             y_Estimate = yEstimate;
116
117             for i = 1:length(trip.Erate)
118                 R1 = 2.5*1e3;
119                 R2 = 1;
120                 pPrev = P;
121
122                 P = pPrev + R1 - ((pPrev*(phi(i)*phi(i)')*pPrev)/(R2 + ...
                        (phi(i)'*pPrev*phi(i))));
123                 Q = pPrev/(R2 + (phi(i)'*pPrev*phi(i)));
124
125                 K = Q * phi(i);
126             end
127
128             if sRoute > 100
129                 massEstimate = massEstimatePrev + K * (y - yEstimate);
130             else
131                 massEstimate = massEstimatePrev;
132             end
133
134             massEstimatePrev = massEstimate;
135
136     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
137             % State Check
138
139             if  startRoute == false && stopRoute == true
140                 currentState = eState1;
141             else
142                 currentState = eState2;
143             end
144
145     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
146
147         otherwise
148             timeRoute = time;
149             sRoute    = s;
150             EbatRoute = Ebat;
151             EdriveRoute = Edrive;
152             energyDriveTrip= 0;
153             energyDriveTripCurrent= 0;
154             energyDriveActual= 0;
155             massEstimate = trip.mveh;
156             y = 0;
```

```
157         y_Estimate = 0;
158
159 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
160
161
162 end
```

With the estimations going on, the RCI profile (called as Erate in the Software) is being calculated by the function *"ErateProfile"*

```
1  function Erate = ErateProfile(Pdrive,Vx,eState,massEstimate)
2
3  %   Author: Dhruv Jagga
4  %   Project: Source Code - Cloud Your Bus
5  %   email: d.jagga@tue.nl
6  %   Date: 10-12-2019;
7  %   Revised: 06-08-2020
8
9  if eState == 1
10
11     if Vx == 0
12         ForceDrive = 0;
13         Erate = ForceDrive/massEstimate;
14     else
15         ForceDrive = Pdrive/Vx;
16         Erate = ForceDrive/massEstimate;
17     end
18
19 else eState == 0
20     Erate = 0;
21 end
22
23
24 end
```

Also the reliability function works concurrently to give information about the reliability of the predictions made. The function is *"ReliabilityFunction"*

```
1  function [reliability, dispReliability] = fcn(eState,error)
2
3  %   Author: Dhruv Jagga
4  %   Project: Source Code - Cloud Your Bus
5  %   email: d.jagga@tue.nl
6  %   Date: 10-12-2019;
7  %   Revised: 06-08-2020
8
9  if eState ==  1
10     if abs(error) ≤ 20
11         reliability = 100 - abs(error) * 5;
12     else
13         reliability = 0;
14     end
15
16 else
```

```
    reliability = 0;
end

if reliability > 95
    dispReliability = 100;
else
    dispReliability = reliability;
end

end
```

- **Post-Processing:** Once the real-time scenarios are emulated using the simulations. The data is pre-processed and the reference data profiles are created using the averaging of the individual trip cycle data. There are three functions used to perform the pre-processing. *"extractData"* function extracts the data segments from the simulation results. The individual data structures are prepared using the function *"dataStruct"*. Finally these individual data structures are used to create averaged data structure using the function *"avgDataStruct"* which is stored as *"Updated Data Reference Profile"*

```
function [tripUpdate] = avgDataStruct(out,tripPost)

%   Author: Dhruv Jagga
%   Project: Source Code - Cloud Your Bus
%   email: d.jagga@tue.nl
%   Date: 10-12-2019;
%   Revised: 06-08-2020

% Averaging 'n' number of cycles;
n = 2;

% Operating directory to look for relevant files
opdir = 'C:\Users\20184719\Documents\Final Project - Module 5\Source ...
    Code\MATLAB\Cloud Your Bus - Energy Estimation System for EV\Post ...
    Processing\DataProfile-Route_401';

% Check the presence of relevent files in the directory
statusDataFile1 = isfile(fullfile(opdir,'ErateProfile401_A_01.mat'));
statusDataFile2 = isfile(fullfile(opdir,'ErateProfile401_A_02.mat'));

% Checking when to activate averaging (Average when the counter is ...
    set to desired number)
for z = 1:length(out.tout)
    if out.postProcess.counter.Data(z) == n
        activateAvg = true;
    else
        activateAvg = false;
    end
end

% Load the files if the files are present in the directory
if statusDataFile1 == true
    m1 = load("ErateProfile401_A_01.mat");
end

if statusDataFile2 == true
```

```
    m2 = load("ErateProfile401_A_02.mat");
end

switch activateAvg
    case true

        e1 = m1.trip.Erate;
        e2 = m2.trip.Erate;

        % Creating averaged Erate Profile
        e = min([length(e1),length(e2)]);
        s = min([m1.trip.s,m2.trip.s]);

        ErateAvg = zeros(e,1);
        timeAvg = zeros(e,1);

        % Averaging the Erate and corresponding time array
        for y = 1:e
            ErateAvg(y) = (m1.trip.Erate(y)+m2.trip.Erate(y))/n;
            timeAvg(y)  = (m1.trip.time(y)+m2.trip.time(y))/n;
        end

        mveh = 19000;

        %         ErateAvg = ErateAvg';
        %         timeAvg  = timeAvg';

        % Creating time varying Auxiliary Power Profile
        f = min([length(m1.trip.time),length(m2.trip.time)]);

        % Extracting the time array from the data structure
        timeAux1 = m1.trip.timeAux;
        timeAux2 = m2.trip.timeAux;

        % Extracting the Power array from the data structure
        Paux1 = m1.trip.Paux;
        Paux2 = m2.trip.Paux;

        % Interpolating the Power according to the distance-time ...
            interpolation
        Paux1 = interp1(timeAux1,Paux1,m1.trip.time);
        Paux2 = interp1(timeAux2,Paux2,m2.trip.time);

        % Averaging the Auxiliary Power and corresponding time array
        for j = 1:f
            PauxAvg(j)    = (Paux1(j) + Paux2(j))/n;
            timeAuxAvg(j) = (m1.trip.time(j) + m2.trip.time(j))/n;
        end

        PauxAvg     = PauxAvg';
        timeAuxAvg  = timeAuxAvg';
        dt          = [0;diff(timeAuxAvg)]; ...
                                        % used as weighted sample space

        sRoute      = [1:1:s]'; ...
                                                % distance vector ...
            with sample of '1m' for particular route
```

```
87          ds          = [0;diff(sRoute)]; ...
                                                % used as weighted sample ...
                space
88
89          EuseDrive   = cumsum(ErateAvg.*ds)*mveh/3.6e6;
90          EuseAux     = cumsum(PauxAvg.*dt)/3.6e6;
91
92          Etrip_measured = mean([m1.trip.Etrip_measured ...
                m2.trip.Etrip_measured]);
93
94          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
95
96          % Making a new trip parameter structure with average Erate (+ ...
                all averaged profiles)
97          trip.sRoute         = sRoute;
98          trip.mveh           = mveh;
99
100         trip.s              = s;
101         trip.time           = timeAvg;
102
103         trip.Erate          = ErateAvg;
104         trip.Paux           = PauxAvg;
105
106         trip.Euse_estimate_drive  = EuseDrive;
107         trip.Euse_estimate_aux    = EuseAux;
108
109         trip.timeAux        = timeAuxAvg;
110         trip.Etrip_measured = Etrip_measured;
111
112         % Checking health status of the stored data
113         trip.tripDataHealth = all(¬isnan(trip.Erate(:))) && ...
                all(¬isnan(trip.Paux(:))) && (trip.Etrip_measured ≠0 || ¬...
                isnan(trip.Etrip_measured));
114
115         tripUpdate = trip;
116
117     otherwise
118
119         tripUpdate = tripPost.trip0;
120 end
121
122 end
```

I Appendix: Scientific Paper

Real-time energy consumption prediction for electric city buses by characterizing the route locations with real-world data

Yuzhe Ma, Dhruv Jagga, Igo Besselink, and Henk Nijmeijer, *Fellow, IEEE*

***Abstract*—Current energy consumption prediction methodologies for electric vehicles suffer several practical limitations. In particular, many vehicle-specific parameters should be known and estimated in many designed systems, which limits the practical applicability. Meanwhile, many existing prediction models rely on the timeliness, which means the driving time should be estimated as well. That further increases the challenge for dealing with uncertainties. There are also some approaches requiring a huge database to find out the characteristics of the driving behavior, such as the machine learning method. But in some cases that will be also limited by the on-board storage space if a real-time prediction will be done. This research explores an energy consumption prediction approach for electric buses, which is based on limited measurement data for a given bus route. The approach will mainly rely on the domain of distance instead of time, which means the energy consumption can be estimated as long as the travelled distance is known at the start. Moreover, buses have the characteristics that the mass is changing with passenger load and the auxiliary power increases over time, which can not be predicted before the trip. Therefore, an online correction algorithm is designed, which is based on recursive algorithm and Kalman filter to improve the estimate accuracy in real-time. Results show that the offline model can give a rough prediction before the trip and the online model can improve the estimate to a promising accuracy.**

***Index Terms*—Energy consumption prediction, electric vehicles, real-time, real-world data**

I. Introduction

ELECTRIC vehicles (EV) have been more rapidly developed in recent years due to their contribution to reduce greenhouse gases (GHG) emissions and fossil energy use. In particular, the prediction of the energy consumption for coming trips has been an important technique, which aims to alleviate the range anxiety [1]. These possibilities can also be conducive to build better infrastructures, which could help achieve a higher rate of adoption of electric vehicles.

In the public transport sector, the energy consumption prediction is helpful for optimizing the charging strategies and the fleet management [2]. Compared to increasing the available battery capacity, an accurate range estimation system is less costly and it can avoid adding extra weight and space. Various energy consumption estimate methods have been investigated by researchers. Current prediction systems are designed from two main aspects: One is the physical modelling approach [3]−[5], the other is data analyzing approach [6]−[8]. The physical model usually requires a complete set of vehicle-specific parameters, which are usually not given on the brochure. For obtaining those parameters, additional tests should be designed. For instance, the rolling resistance coefficient need to be determined by an extra coastdown test [9]. Meanwhile, a motor efficiency map is also necessary for an accurate physical model, which should be determined by the dynamometer experiment. However, these kinds of tests are costly and not always possible in practice unless you have an automotive lab. Meanwhile, the environmental parameters are also necessary for a physical model, such as the road gradients and the weather conditions. For projects that no equipment can be provided for completing the parameter-determination tests, many current prediction models might be stuck with insufficient inputs.

The data-driven model has the advantage of estimating the realistic energy consumption, and less relies on the vehicle dynamics. Especially for public transport, the data-driven approach is more suitable because of the fixity of routes. Data can be repeatedly collected from the bus on one route and characterized. In literature, the simulation of most existing data-driven models is based on the time series [10]−[12], which always needs a time estimation as well. However, it is usually not easy to do an accurate time prediction for the future driving because the events on the road are randomly happening. As the locations on the route is alwasy fixed, therefore the prediction will be more reliable if the model is based on the locations that the vehicles are driven. Research [13] has a distance-based simulation program, but many vehicle-specific parameters need to be estimated.

An offline data-driven prediction model is always not enough to guarantee the accuracy in real time. To solve this, some researchers also design online correction to adjust the predicting results from time to time. However, many models still need some vehicular parameters even if the authors thought those parameters were easy to obtain. For instance, the real time prediction model in research [14] requires the information of air density, aerodynamic drag coefficient, frontal area and even a prediction of rolling resistance coefficient. The complexity of the model is still very high, which might be limited in practice.

This paper aims to design an prediction algorithm for electric city buses, which does not rely on plenty of vehicle parameters and time series. Meanwhile, an online correction

This work was supported by the China Scholarship Council under Grant 201606170074.

The authors are with the Department of Mechanical Engineering, Eindhoven University of Technology, 5612AZ Eindhoven, The Netherlands (e-mail: y.ma1@tue.nl; d.jagga@tue.nl; I.J.M.Besselink@tue.nl; H.Nijmeijer@tue.nl).

model which only requires a prediction for one parameter is designed to keep the complexity low. This proposal considers to characterize the chosen route in location domain by analyzing the historical data. The algorithm is divided into two parts, the offline algorithm and the online algorithm. The offline model uses the historical data to generate an initial estimate of the energy consumption. The online model will correct the prediction result by adjusting the vehicle mass value. Additionally, the online approach is based on recursive algorithm to adjust only two parameters, which simplifies the practical complexity during the operation.

The outline of this paper is as follows. In Section II, an energy consumption prediction model is proposed where the algorithm is mainly simulated in the distance domain. Route characteristic indicator (RCI) is defined for the use of offline estimate. In Section III the procedures for generating a reference RCI is discussed. An offline estimate of the energy consumption can be made from this procedure as well. Section IV introduces the online correction algorithm with using the offline estimate and the recursive algorithm. In Section V, the testing scenarios and the simulation results are presented. The conclusions and recommendations are given in Section VI.

II. Methodology

The energy consumption for an electric vehicle can be simulated by a model in time domain, which can be expressed by

$$\begin{aligned} E(t_1) &= \int_0^{t_1} [P_t(t) + P_{aux}(t)]dt \\ &= \int_0^{t_1} \{\frac{v(t)}{\eta(t)}[f_r mg\cos(\alpha(s(t))) + \frac{1}{2}\rho C_d A_f (v(t) \\ &- v_w(t))^2 + mg\sin(\alpha(s(t))) + m_{eff}a_x(t)] \\ &+ P_{aux}(t)\}dt \end{aligned} \quad (1)$$

where $E(t_1)$ is the energy consumption at any time t_1 on the trip, $P_t(t)$ is the power request as a function of time in the drivetrain for propelling the vehicle, f_r is the rolling resistance coefficient, m is the vehicle mass, g is the gravitational acceleration, $\alpha(s(t))$ is the road gradient as a function of distance $s(t)$, the travelled distance $s(t)$ is a function of the time t, which can be expressed by

$$s(t_1) = \int_0^{s(t_1)} ds = \int_0^{t_1} v(t)dt \quad (2)$$

ρ is the air density, C_d is the aerodynamic drag coefficient, A_f is the frontal area, $v(t)$ is the vehicle speed as a function of t, $\eta(t)$ is the drivetrain efficiency as a function of time, $v_w(t)$ is the wind speed as a function of t, m_{eff} is the vehicle effective mass, $a_x(t)$ is the acceleration as a function of t, $P_{aux}(t)$ is the auxiliary power as a function of t and t_e is the time at the end of the trip. For the purpose of validating such a model, all those parameters should be known. Similarly, these parameters should also be given if this model will be used for predicting the future trips. However, in reality, not all the parameters can be provided or measured due to practical limitations. And a part of parameters are always changing randomly and they might be also correlated with each other, it would be very complicated to estimate them separately. For example, the rolling resistance coefficient f_r changes when the road type, ambient pressure and ambient temperature change. The vehicle mass m changes when the load in the vehicle changes. The road gradient α is dependent on the road structure, you should give additional efforts on extracting the available online elevation data. The air density is changing with diverse ambient pressure and ambient temperature. The vehicle speed v is changed by various factors, which can not be known precisely before a trip. Meanwhile, when estimating the future driving time, the accuracy will be limited by different kinds of uncertainties the vehicle could encounter on the road. Changing average driving speed, uncertain waiting time for traffic lights and unexpected accidents can all affect driving time. Therefore, predicting an electric vehicle's energy consumption would be not easy if such a time-based and parameter-based method is adopted.

For a given route, the location is always fixed there. Compared to predicting the driving time, using the fixed location to determine the energy usage is more reliable. A vehicle can only consume energy when it begins moving, which means the energy consumption is actually dependent on the travelled distance. An exception is the auxiliary system, which will still cost energy when the car is standstill. With combing the two aspects, the energy consumption calculation formula can be divided into two terms. One is a distance-based term, and the other is a time-based term. We define the expression as follows

$$E(t_1) = \int_0^{s(t_1)} F_s(s)ds + \int_0^{t_1} P_{aux}(t)dt \quad (3)$$

where $E(t_1)$ is the energy consumption at any time t_1 on the trip, $s(t_1)$ is the travelled distance at t_1, $F_s(s)$ is the propelling force in the drivetrain as a function of distance $s(t)$. Compared with Equation 1 which is fully dependent on time, Equation 3 reduces the impact of the time uncertainty. When the travelled distance is s_1, F_s can be expressed by

$$\begin{aligned} F_s[s(t_1)] =& f_r mg\cos[\alpha(s(t_1))] + \frac{1}{2}\rho C_d A_f [v(s(t_1)) \\ &- v_w(s(t_1))]^2 + mg\sin[\alpha(s(t_1))] \\ &+ m_{eff}a_x[s(t_1)] \end{aligned} \quad (4)$$

This research aims to avoid using too many modelling inputs but Equation 4 also involves a group of parameters. Meanwhile, the proposed approach will be a data-driven model which is based on the average of the historical data. Averaging parameters that are dramatically changing will make errors. For a bus case on the fixed route, the gravitational acceleration g, the aerodynamic drag coefficient C_d, the frontal area A_f and the effective vehicle mass m_{eff} are constant parameters. The road gradient α is fixed with the route locations. The rolling resistance coefficient f_r can be a function of the ambient temperature [15]. The ambient temperature is usually not changing dramatically. Therefore, f_r is relatively steady during driving. The air density ρ is dependent on the ambient temperature and air pressure, and the air pressure is not changing heavily so that ρ is mostly steady. The wind speed v_w is small in the city area, and the bus driving speed is also low. So the influence of the wind speed is not changing too

much. The driving speed v can show a similar characteristic of the profile between different cycles on the same route. The vehicle mass m has variations due to the changing passenger load. The passenger number from each stop is random and cannot accurately predict accurately. Meanwhile, the mass variation can have a significant influence on the propelling force because the rolling resistance force, the road slope force and the acceleration force are all related to the vehicle mass m. The passenger number is also not easy to obtain in practice. Therefore, averaging the mass m will be difficult and bring inaccuracy. Above all, except for the vehicle mass m, all other parameters can be characterized by averaging the historical data. For excluding m from the model, F_s can be normalized by the mass m. With considering the drivetrain efficiency η, the normalized force from the motor can be expressed by

$$\begin{aligned}\frac{F_s[s(t_1)]}{m\eta[s(t_1)]} =& \frac{1}{\eta[s(t_1)]}[f_r g\cos(\alpha(s(t_1)))\\ &+ \frac{1}{2m}\rho C_d A_f[v(s(t_1)) - v_w(s(t_1))]^2\\ &+ g\sin[\alpha(s(t_1))] + \frac{m_{eff}}{m}a_x(s(t_1))]\end{aligned} \tag{5}$$

We define $\frac{F_s}{m\eta}$ as the route characteristic indicator (RCI). The unit for RCI can be m/s^2 because $\frac{F_s}{m\eta}$ can be physically an acceleration. It is not practical to directly measure the RCI, F_s can be calculated by the drivetrain power request P_d from the motor and vehicle speed v, which can be given as

$$F_s[s(t_1)] = \frac{P_d[s(t_1)]\eta[s(t_1)]}{v[s(t_1)]} \tag{6}$$

Then RCI can be calculated by

$$RCI[s(t_1)] = \frac{F_s[s(t_1)]}{m\eta[s(t_1)]} = \frac{P_d[s(t_1)]}{mv[s(t_1)]} \tag{7}$$

P_d can be calculated by the product of the voltage and the current of the drivetrain, which is

$$P_d[s(t_1)] = U_d[s(t_1)]I_d[s(t_1)] \tag{8}$$

where U_d and I_d are the voltage and the current of the drivetrain, respectively. Assume the mass is constant for the whole trip, the Equation 3 becomes

$$E(t_1) = m\int_0^{s(t_1)} RCI(s)ds + \int_0^{t_1} P_{aux}(t)dt \tag{9}$$

The auxiliary power request P_{aux} is the difference between the battery power and the drivetrain power, which can be calculated by

$$P_{aux}[s(t_1)] = P_b[s(t_1)] - P_d[s(t_1)] \tag{10}$$

where P_b is the battery power and it can be determined by the product of the battery voltage and battery current

$$P_b[s(t_1)] = U_b[s(t_1)]I_b[s(t_1)] \tag{11}$$

in which U_b and I_b are the voltage and the current of the battery, respectively.

However, in case of electric city buses the drive-train power request is significantly influenced by the weight of passengers being carried for a particular trip on a given route. The

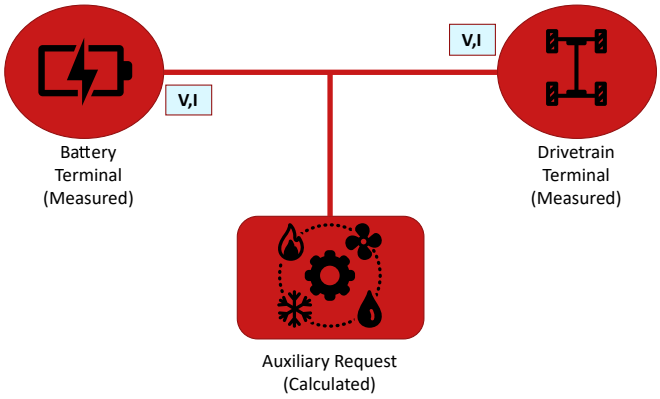


Fig. 1: Power Measurement Configuration of Electric City Buses

auxiliary request can be also influenced by several factors, one of the most dominant is ventilation and air conditioning, which is further dependent upon number of other factors, such as; number of passengers travelling in the bus, environmental conditions etc. Therefore, a new update was proposed in this model, which was to update the mass term for drive-train energy estimation and adding a correction term for the auxiliary energy estimation part that accounts for the above mentioned influencing factors on auxiliary energy estimation. Both of these terms are subjected to change as the trip progresses to make relevant corrections in real time on energy estimation of a given route. Similar to the drive-train energy estimation case a reference auxiliary power P_{aux} profile was used for a given route. The updated model is described in Equation 12

$$\begin{aligned}E(t_1) =& m[s(t_1)]\int_0^{s(t_1)} RCI(s)\,ds\\ &+ n(t_1)\int_0^{t_1} P_{aux}(t)\,dt\end{aligned} \tag{12}$$

In Equation 12 $m[s(t_1)]$ is a coefficient described as function of travelled distance which is further the function of time, this term not only accounts for the mass variation of the vehicle over the route but also accounts for other environment uncertainties such as rolling resistance, road slope and aerodynamic drag etc. The detailed explanation of the chosen algorithm for achieving the updating goal will be discussed in Section IV, which is a re-calculation of the total energy consumption from the start to the end of the given route. $n(t)$ is the gain for correcting the auxiliary energy estimate over real-time.

III. Offline Estimate Based on Historical Data

As a preparation for the online prediction, this section will discuss the procedure to properly generate an offline reference RCI as defined in Section II with processing multiple collected runs from a given bus driving route. For a given route from place A to place B, the data can be collected repeatedly to establish a historical database. In this research, the essential signals consist of the time, the GPS latitude,

the GPS longitude, the vehicle speed, the battery voltage, the battery current, the drivetrain voltage and the drivetrain current. Normally, these signals can be easily collected by the CAN bus on-board. Assuming the sampling frequency is identical for all signal channels and the data lengths for all channels are identical, the reference RCI profile and reference auxiliary power profile can be obtained. Using the profiles, the initial energy consumption estimate can be done on the offline stage.

A. Data preprocessing

The data used in the research is from one of the buses in operation of the city. Prior to bring the data to simulate the model, some steps should be taken to process the raw data. First, the locations of the start and end of the route should be accurately selected. Second, a fixed-step travelled distance data should be made by re-sampling.

1) Determine the range of route: According to the GPS coordinates of the first and the last stops from Google Map, the data between the two locations can be extracted from the raw data. On the one hand, this step can remove points which were collected when the bus was in standstill before or after the trip. On the other hand, making the start and end locations the same can be helpful to align the data size more easily.

2) Re-sample the travelled distance: In the raw data, the distance values between sampling points vary from segment to segment. That would be difficult to align the same location from different cycles just by directly searching it with the sequence of the corresponding sample. To deal with this, the travelled distance in the measurement can be divided into fixed intervals. The value of the fixed interval can be determined by averaging the historical actual intervals between samples. Assume the route lengths based on the calculation for all cycles are the same, the amount of points for all cycles will be equal after re-sampling by this fixed interval. But in practice, measured cycles may have slightly different lengths due to the GPS operating errors or various driver behaviors. The influence of this difference is neglected in this research because it has been proven to be too small to have an effect (just a few meters). A fixed total travelled distance is also defined by averaging the total travelled distances of all cycles. Consequently, we have the identical number of sampling points and equal travelled distance for all measured cycles, which means multiple cycles can be programmed synchronously.

B. Reference RCI and auxiliary power

As defined in Section II, the term RCI can be determined for a single cycle (Equation 7). With the re-sampled distance domain determined in Section III-A2, a RCI profile can be obtained. For the auxiliary power, the number of sampling points are different from the number of re-sampled points. Therefore, a further alignment should be done for the generation of the auxiliary power profile. Once the data sizes are aligned, the averaging can be done to generate the reference RCI profile and auxiliary power profile. The sections also gives a simple algorithm to determine the minimum number of RCI profiles that are enough to be used to prepare the offline estimate, which has the advantage on the memory saving in case a practical limitation is considered.

1) RCI and auxiliary power for individual cycles: Equation 7 and Equation 10 are respectively for obtaining the RCI profile and auxiliary power profile for a single cycle.

2) Interpolate sample size to match the size of re-sampled distance: Due to the difference between the raw distance data and the re-sampled distance data, RCI profile should also be re-sampled. This can be achieved by MATLAB for an interpolation. For the auxiliary power $P_{aux}(t)$, the time can be also re-sampled by interpolating the time-distance relation in the original data to the re-sampled distance. Then, $P_{aux}(t)$ can be obtained by interpolating the time-P_{aux} relation in the original data to the re-sampled time.

3) Average multiple RCI profiles and auxiliary power profiles: RCI profiles and auxiliary power profiles then can be made for all measured cycles. Meanwhile, the sample size for all cycles are now identical. That is easy to make an averaging for all RCI and auxiliary power profiles, which are thus the reference RCI and reference auxiliary power profile.

4) Minimum sample size determination: Although it will be more accurate with collecting the data as much as possible, a huge amount of data might be not very necessary in some case. Actually when the data is much enough, the averaged profile will become steady. The determination of the minimum cycle size can be operated by comparing the root mean square error (RMSE) of the latest profile with a chosen baseline profile. We only take RCI profile into account rather than auxiliary power profile, which is because the RCI plays a bigger role in the total energy consumption. RMSE of between the newest RCI profile and the baseline RCI profile can be expressed by

$$RMSE_{RCI} = \sqrt{\frac{\sum_{i=1}^{n}(RCI_{new} - RCI_{base})^2}{n}} \tag{13}$$

where n is the number the data points, RCI_{new} is the averaged RCI profile with multiple RCI profiles and RCI_{base} is the baseline RCI which can be defined by a random single cycle. With increasing the amount of the cycles used for generating the reference RCI, namely RCI_{new}, $RMSE_{RCI}$ will go up dramatically when the amount of cycles is small and then become steady after a specific amount of cycles.

Figure 2 illustrates the investigation of the inflection point in the tendency. The change becomes much smaller after the point that has 16 cycles involved. Therefore, in this case 16 cycles are sufficient to generate a reference RCI profile with an acceptable accuracy. The plots of RCI and auxiliary power profiles for all cycles are shown in Fig. 3 and Fig. 4, respectively. The similarity among different RCI profiles can be seen, which makes the averaging method work. In the auxiliary power profiles, the time varies from cycle to cycle. This further explains the advantage of using distance domain for prediction rather than time domain.

C. Off-line energy consumption estimate

With the reference RCI profile and reference auxiliary power profile, the energy consumption is estimated by Equation 9. Fig. 5 shows the energy consumption calculated from

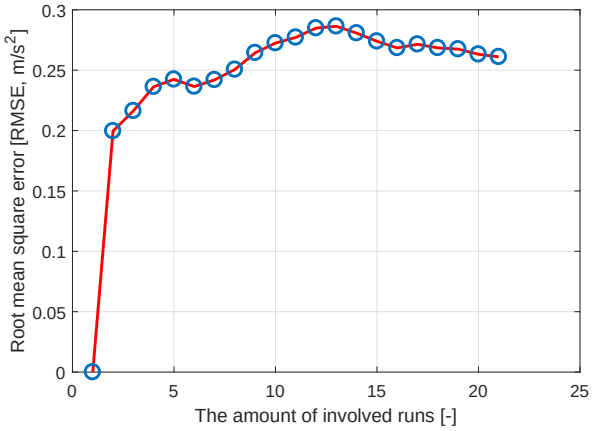


Fig. 2: The tendency for RMSEs of averaged RCI profiles with increasing the involved cycle amount

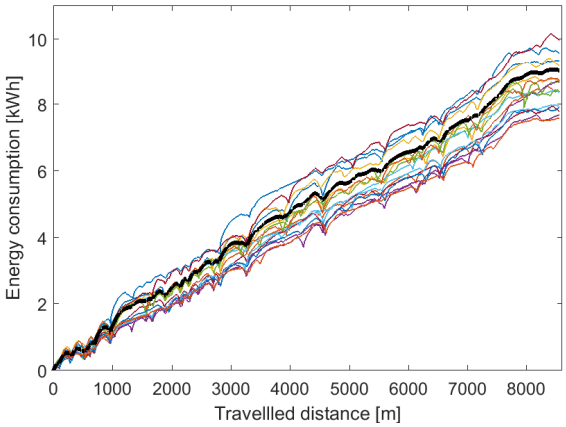


Fig. 5: Energy consumption from 16 measured cycles and the estimate based on the reference RCI and auxiliary power profiles (in black)

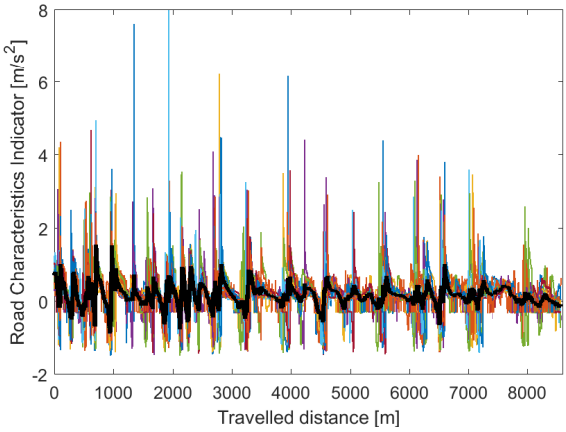


Fig. 3: RCI profiles from 16 cycles and the reference RCI (in black)

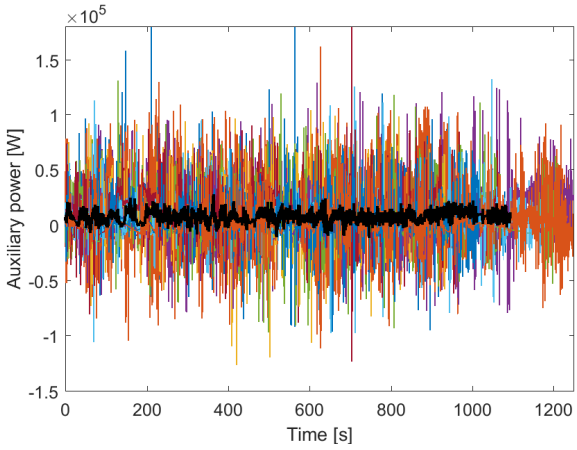


Fig. 4: Auxiliary power profiles from 16 cycles and the reference auxiliary power profile (in black)

the measured data for 16 cycles and the estimated energy consumption based on the reference RCI and auxiliary power profiles. Because the bus mass is assumed as a constant, and the time for operating the auxiliary system is fixed, the estimated total energy consumption for the investigated route is fixed as well.

IV. Online Correction with Tuning Parameters

To improve the accuracy of the energy consumption prediction over the given route, it is required to update the model parameters in real-time. The parameters of the model in real-time (online) estimation algorithms are updated as the new measurements are available during the operation of the physical system. Now, the two major parameters needed two be updated in real-time are mass-estimate for drive-train estimation and correction gain-estimate for auxiliary estimation. (refer to Equation 12). All other influencing factors on energy estimation are considered as perturbation [17].

A. Online Identification

In this particular case, it is quite useful to have a reference profile to be available online when the system is in operation. This online available reference model allows the system to make best predictions for next outputs and can be regarded as the an adaptive prediction method (see Fig. 6).

Since, the challenge now is concerned with estimating the parameter of the model as the new data is made available during the operation. A typical choice is made and estimation is performed using recursive algorithms (also known as online, real-time identification, adaptive parameter estimation, or sequential parameter estimation). This algorithm estimates the parameter values at each time step by using currently made observations (measurement data) and using previous parameter estimates. These algorithms are a viable option because they are efficient in terms of memory usage and also possess smaller computational demands [18].

B. Recursive Algorithm: Online Parameter Estimation

The recursive algorithms used for online parameter estimation can be parted into two categories. The infinite-history

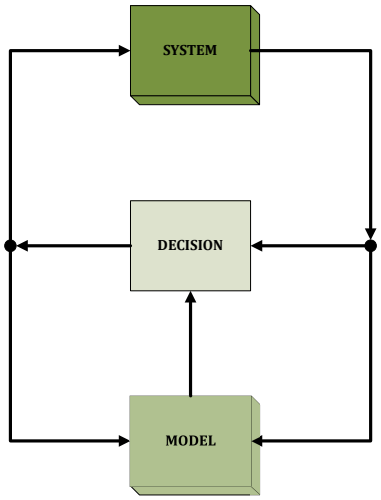


Fig. 6: A basic procedure for the adaptive method

algorithm and finite-history algorithm. The infinite-history algorithm aims to minimize the error between the measured and the estimated outputs for all time steps from the beginning. Whereas, finite-history algorithm aims to minimize the error between the measured and the estimated outputs for a finite number of past time steps. In order to carry out estimation for this particular application; i.e. estimation on mass and correction gain, relevant results are obtained when the complete route data is handled in order to capture different behaviours of energy consumption over the given route. Therefore, the infinite-history recursive estimation algorithm was used.

1) Recursive Infinite-History Estimation: In the general form of the recursive estimation algorithm, it follows a set of regression equations that minimizes the following cost function.

$$J(\hat{\theta}, k) = \frac{1}{2}\sum_{k=1}^{k}[y(k) - \hat{y}(k)]^2 \tag{14}$$

the predicted estimate of the parameter is given by the following equation:

$$\hat{\theta}(k) = \hat{\theta}(k-1) + K(k)[y(k) - \hat{y}(k)] \tag{15}$$

where, $\hat{\theta}(k)$ is the estimation of parameter at every sample k. $y(k)$ is the observed or measured output at sample k and $\hat{y}(k)$ is the estimation of $y(k)$ based on observations up to samples $k-1$. $K(k)$ gain determines how much the current estimate error $[y(k) - \hat{y}(k)]$ affects the estimate of the parameter. The main idea of the algorithm is to minimize the prediction-error term.

The model used to obtain the energy estimation (refer to Equation 12) is linear in relationship and hence, a linear-regression form of model is used for online parameter estimation. Thus, the role of the gradient $\psi(k)$ can be visualised by the following equation.

$$y(k) = \psi^T(k)\theta_0(k) + e(k) \tag{16}$$

The predicted estimate of the output is given by the following equation:

$$\hat{y}(k) = \psi^T(k)\hat{\theta}(k-1) \tag{17}$$

where, $\psi(k)$ is the regressor vector or gradient of the predicted output $\hat{y}(k|\theta)$ with respect to parameters θ, which is computed based upon the previously measured input and output values. $e(k)$ is assumed to be white noise. The definitive form of $\psi(k)$ is determined by the structure of polynomial model.

The estimation gain has the following form.

$$K(t) = Q(k)\psi(k) \tag{18}$$

Further, the infinite-history recursive estimation algorithm has different types; forgetting factor, kalman filter and gradient based approach. All these algorithms were developed and deployed for this application, but kalman filter approach was chosen for the final deployment because of its superiority and its fit with the application. The rationale for choosing this approach is also provided in the Table I below:

TABLE I: Rationale for selection of type of Online Parameter Estimation

Algorithm Type	Advantages	Disadvantages
Gradient based	Simple implementation; Less memory and processing; Absolute error is $\approx 5\%$.	Correction gain remain constant; Less Robust.
Forgetting factor	Adaptive gain estimation based upon error co-variance	Gain estimation is sensitive; Better for system with quick changing dynamics in which last few measurements are critical; Absolute error is $\approx 8\%$.
Kalman filter	Adaptive gain estimation based upon error co-variance; Absolute error is $< 2\%$; More robust results tested on limited data; Better estimation on overall historical data.	Relatively complex to implement.

2) Kalman Filter: Kalman filter, also recognised as Linear Quadratic Estimator (LQE), is essentially a set of mathematical equations which implements a predictor-corrector type estimator, these estimations are optimal as it minimizes the error co-variance; as certain conditions are met. It is a method that use sequence of measurements observed over time, that accommodates statistical noise and other inaccuracies. This algorithm yields estimates of unknown variable that conduces more accuracy then those based upon single measurements. These estimates are made by estimating a joint probability distribution over the variables for each time-frame.

The Kalman filter handles a process by using a feedback control: the filter estimates the process parameter at certain sample and obtains feedback in the form of measurements. The related algorithms for the filter falls under two categories: sample update equations and measurements update equations. The sample update equations are usually responsible for projecting ahead in sample, the current parameter and its error co-variance estimation to realize $a\ priori$ estimate for the next sample step. The measurement update equations ensures for the feedback - i.e for incorporating a new measurement into the $a\ priori$ estimate to retrieve an improved $a\ posteriori$

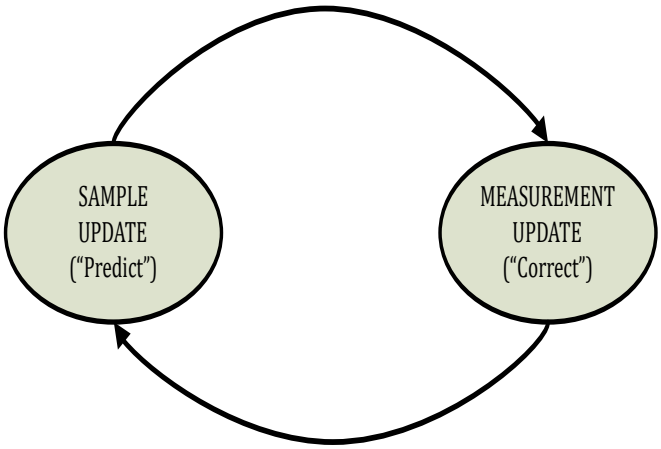


Fig. 7: Ongoing Kalman filter cycle. $Sample$ update projects the current state estimate before the actual measurement is available. The projected estimate is later updated by $measurement$ update.

estimate. The estimation algorithm hence, resembles of a predictor-corrector algorithm. The equations used in Kalman filter adaptation algorithm are summarized to compute the Kalman gain as in Equation 18

$$K(k) = Q(k)\psi(k) \tag{19}$$

$$Q(k) = \frac{P(k-1)}{R_2 + \psi^T(k)P(k-1)\psi(k)} \tag{20}$$

$$P(k) = P(k-1) + R_1 - \frac{P(k-1)\psi^T(k)P(k-1)}{R_2 + \psi^T(k)P(k-1)\psi(k)} \tag{21}$$

In the prediction step the Kalman filter projects the parameter and error co-variance ahead in sample. It then jumps to the correction step and first computes the Kalman gain. It then updates the estimates with measurements $y(k)$ and finally update the error co-variance $P(k)$. This process is repeated with previous $a\ posteriori$ estimate used to project new $a\ priori$ estimate. This recursive nature of Kalman filter makes it useful for practical estimation [19].

Applying the terminology in this application in discrete time, the regressor vector $\psi(k)$ is $\sum_0^k RCI(k)$ and parameter vector θ is $m(k)$ for drive-train estimation and correction. While for auxiliary estimation and correction, the regressor vector $\psi(k)$ is $\sum_0^k P_{aux}(k)$ and parameter vector θ is $n(k)$. To use the Kalman recursive algorithms, an initial value is required for its start-up. In the drive-train estimation and correction algorithm the parameter vector $\hat{\theta}(0)$ is initialised with the unladen mass of the vehicle and co-variance matrix $P(0)$ that indicates the parameter error is initialised with identity matrix. Furthermore, in auxiliary estimation and correction algorithm the parameter vector $\hat{\theta}(0)$ is initialised with 0.1 and co-variance matrix $P(0)$ that indicates the parameter error is initialised with identity matrix. Also, the algorithm always ensures that the co-variance matrix $P(k)$ is a positive-definite matrix by using a square root algorithm to update it [20].

$P(k)$ is computed assuming that the residuals (difference between estimated and measured output) is white noise, and the variance of these residuals is 1. The R_1 and R_2 in Kalman filter algorithms are the process error co-variance (co-variance matrix of parameter changes) and measurement error co-variance (variance of residuals). It is assumed that R_1 and $P(0)$ matrices are scaled such that $R_2 = 1$. This allows the designer to tune the algorithm with only one tuning factor in this case which is R_1. In case of drive-train estimations, it was observed that the consumption of the energy over different cycles can vary with the uncertainty of $2.5\ KWh$, the value of the R_1 was set to $2.5 * 10^3$ to giving an uncertainty range to the algorithm for the predictions. Also, it means that the process variance allowed is larger and actual measurement will be trusted more than the predicted value. This means that the estimates will deviate away less from the actual measured value. Also, this value cannot be kept too large in order to prevent the trust of algorithm to shift significantly towards the noisy measurements. On the other hand the auxiliary energy profile remains fairly similar and linear for different trip cycles except some exceptional trips. So, the process variance allowed is kept smaller; in this case $R_1 = 10^{-5}$, and algorithm is pushed to trust the predicted value more than the actual measurements.

V. Testings and simulation results

To start developing the novel real-time energy estimation system for the electric vehicles; more focused on the electric city buses, the development was made at the prototype level using simulations. The system functionality required some pre-collected data from the vehicle over the given route. For the purpose of collecting data from the vehicle and ARM based embedded device was installed on the vehicle. This device has access to all CAN-Bus and GPS data of the vehicle for the given route. During the experiment for the collection of data, the device logged the data for the entire motion of the vehicle for a couple of weeks, which includes all relevant charging and discharging data. From this ocean of data logged; the relevant data segments were extracted associated with the selected route (From the city train station to the city airport, 8.6 km). It was observed that 21 such cycles of the selected route can be extracted from the total logged data, which was sufficient to begin with the development. Since, this application requires the estimation of certain parameters using the data, it was necessary to categorize the available data into training data and validation data.The training data was used to build the base reference model and tuning the parameters of the algorithm. Whereas, on the other hand validation data was used to provide and unbiased evaluation of the final results from the estimation and correction algorithm. The validation data has never been used in the training of the algorithms. By the rule of thumb, 70 % of total data collected was allocated to training which is approximately 16 cycle data and the rest of the 30 % data which was the remaining 5 cycle data was used as validation data.

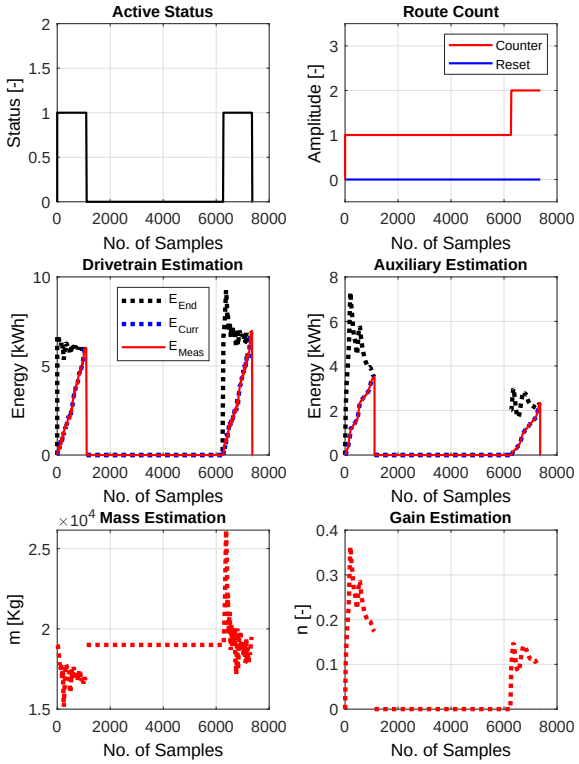


Fig. 8: Simulation Results: Test Scenario 1

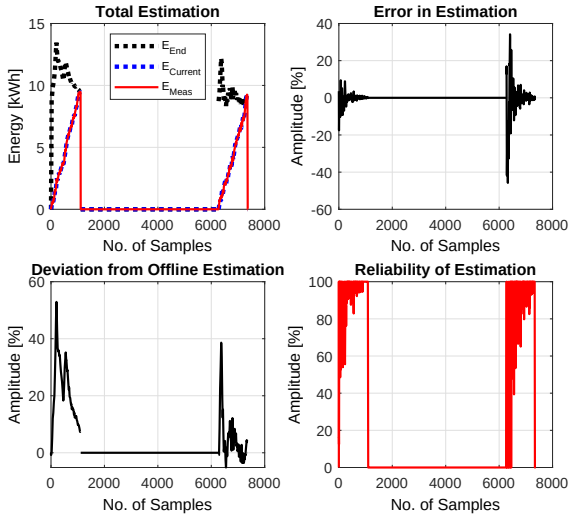


Fig. 9: Simulation Results: Test Scenario 1

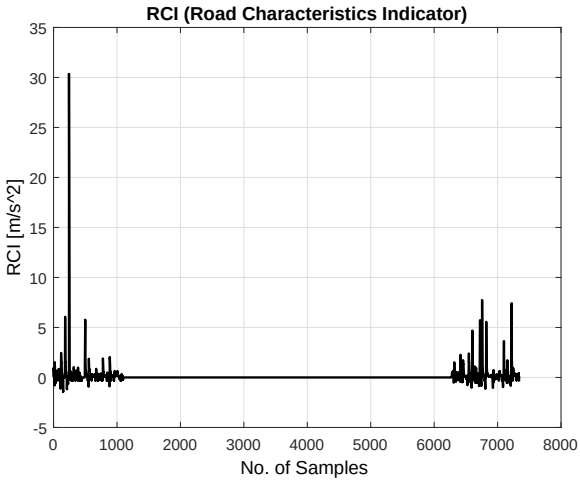


Fig. 10: RCI Profile generated in real-time as mass estimation is updated.

A. *Testing*

To carry out the testing to check the overall functionality of the real-time energy estimation system; three test scenarios were designed.

1) Test Scenario 1: In this test scenario the real-time energy estimation system was simulated with the first two cycles of the training data. The base reference profile was used for estimation and corrections. It can be observed from Fig. 8, that the system identifies when to activate the real-time estimation and correction function. Since two cycle data was simulated together (which means that electric city bus travels on the selected route two times). This has been clearly identified by the system and estimations were made. The system does the drive-train and auxiliary energy estimation separately which can be seen in Fig. 8. The corresponding correction parameters evolution can be observed in Fig. 8 as well.

The combined total energy estimation from drive-train and auxiliary estimation & correction algorithms can be observed in Fig. 9. The error resulting in between the estimated value and actual measured values can be observed in Fig. 9, it can be seen that as the more data is getting available the resulting error starts decreasing and eventually becomes bounded around zero. It was also crucial to observe the reliability of the energy estimation and correction algorithm which is based upon the evolution of the error. To trust the predicted energy estimates it was important to know if the reliability of the estimation is higher or not. This can be observed in Fig. 9.

Also, another important aspect of real-time estimation was to find its superiority from the offline estimations for a particular trip cycle of the selected route. The importance of real-time estimation in this particular application can be measured by using a term *deviation from Offline Estimation* which at any particular time gives an idea about the deviation of the current estimates from the initial estimate made in the beginning of the trip cycle for the given route.

While the estimations were happening, it was also important to calculate and record the updated (real-time) RCI profile using the power, velocity and mass estimates of the drive-train estimation (shown in Fig. 10). This data profile will later be processed in the post-processing step and will be used to generate a new updated reference profile for the energy estimation system.

The result of *Test 1* generates the updated reference profile. Now, when the real-time energy estimation system runs next

time over the selected route, then instead of using the base reference profile for estimations and corrections; the updated reference profile is used. This will ensure the adaptability of the energy estimation system towards changes in the environment, weather, traffic, road conditions and also the commuters travelling trend etc.

2) Test Scenario 2: In this test scenario the real-time energy estimation system was simulated with the different cycles of the training data. The updated reference profile was used for estimation and corrections. It can be observed from Fig. 11, that the system identifies when to activate the real-time estimation and correction function. The system does the drive-train and auxiliary energy estimation separately. The corresponding correction parameters evolution can be observed in Fig. 11 also.

The combined total energy estimation from drive-train and auxiliary estimation & correction algorithms can be observed in Fig. 12. From the error resulting in between the estimated value and actual measured values, it can be seen that as the more data is getting available the resulting error starts decreasing and eventually becomes bounded around zero. The reliability and the deviation from the initial estimate can be also observed in Fig. 12.

3) Test Scenario 3: In this test scenario the real-time energy estimation system was simulated with the different cycles of the validation data. These data cycles are never seen by the estimation and correction algorithm during the development are kept solely to perform the software in the loop (SIL) testing. The updated reference profile was used for estimation and corrections. The active system status can be observed from Fig. 13. The system does the drive-train and auxiliary energy estimation separately. The corresponding correction parameters evolution can be also observed in Fig. 13.

The combined total energy estimation from drive-train and auxiliary estimation & correction algorithms can be observed in Fig. 14. The error, the reliability and the deviation from the initial estimate can be also observed in Fig. 14.

B. Results

From the simulation test results in Section V-A, it can be clearly seen that the real-time estimation system is an advanced system capable of estimating the approximate energy consumed by the electric city bus over the given route well in time, and is also producing the robust results over different data cycles. A detailed analysis was also made on the accuracy and precision of the system by observing the absolute error in estimation over all these trip data cycles while using the base reference profile. The absolute error computed in the Figure 15 is calculated for the estimations done over the trip cycle distance travelled using the following formula:

$$e_{abs}(\%) = \frac{\mid \sum_0^{s(t)} E_{est} - \sum_0^{s(t)} E_{mea} \mid}{\sum_0^{s(t)} E_{mea}} * 100(\%) \qquad (22)$$

where E_{est} is the estimated energy consumption, E_{mea} is the measured energy consumption. In Fig. 15 it can be clearly observed that the performance of real-time (online) energy estimation system is a way more than the offline estimations.

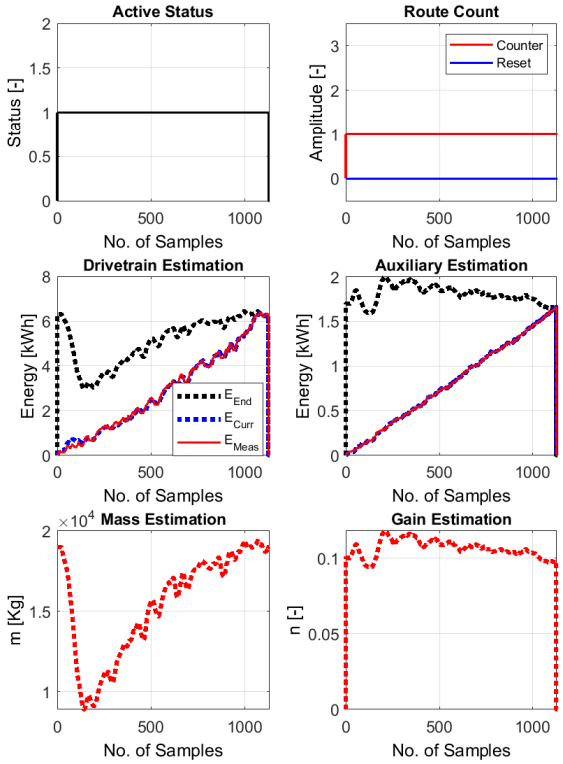


Fig. 11: Simulation Results: Test Scenario 2

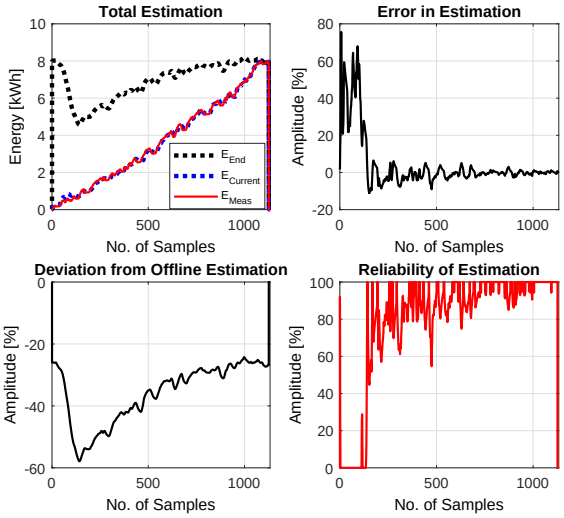


Fig. 12: Simulation Results: Test Scenario 2

In offline estimations the absolute error over some cycles can go as high as 40 % and on an average remains ≈ 18.5%. Whereas, on the other hand in real-time (online) energy estimations the absolute error is under 4 % and on an average remains ≈ 1.2%. This assures the superior performance of real-time energy estimation system. In Fig. 16 the results are compiled for the progression of absolute error in estimation

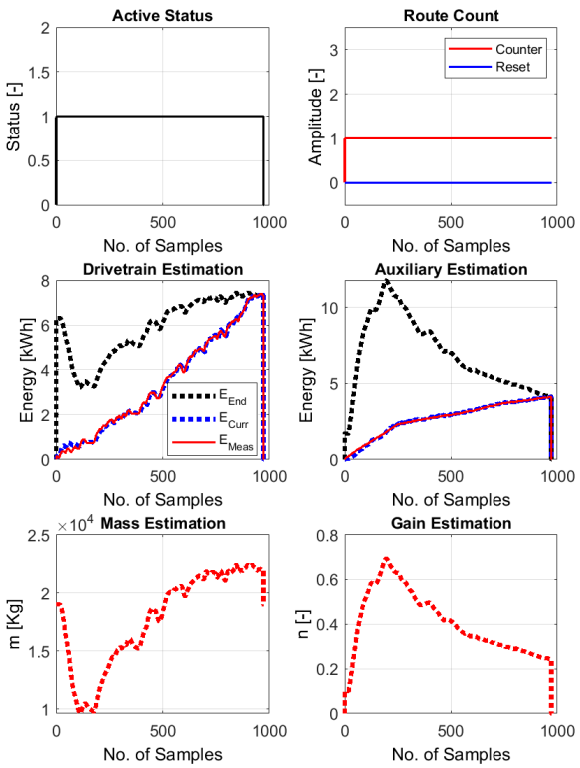


Fig. 13: Correction Gain Estimation

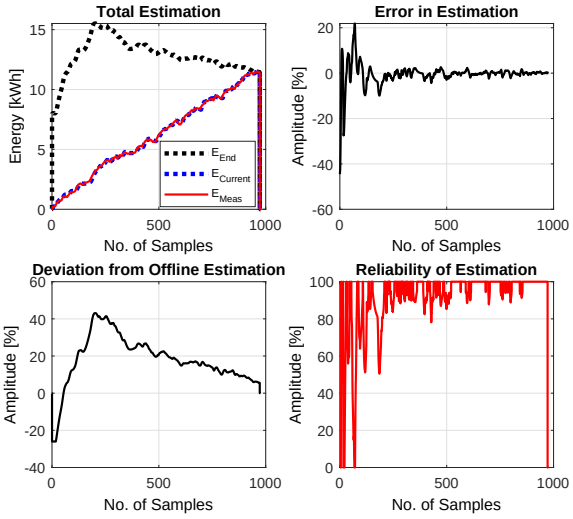


Fig. 14: Simulation Results: Test Scenario 3

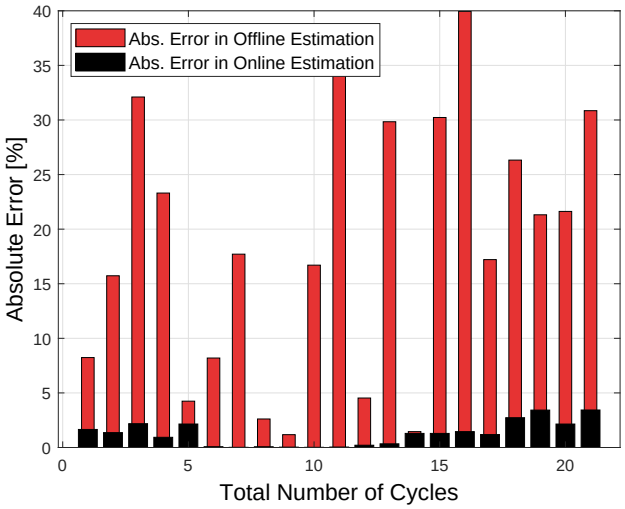


Fig. 15: Comparison of absolute error in estimations for offline and online energy predictions done over base reference profile.

for all available data cycles with reference profile used as base data profile. It can be seen that the performance of the real-time energy estimation is exceptionally good in the region from 30% to 80% of the travelled distance. Outside this region the performance remains reasonably good and the system tries to bound the error. The reason behind the difference in this performance was due to the electric city bus does not have the dedicated path to complete the selected route journey especially outside the above mentioned range. It only uses a patch of this route as a dedicated path lying in between the region. In Fig. 17 results are compiled in order to make a comparison between absolute error in estimation for a trip data cycle when the base and updated reference profiles are used for similar data trip cycle. Particularly in the case of Fig. 17 the training data cycles were used. It can be observed while the base data reference profile is used the overall absolute error remains consistently small throughout the trip. In the case when updated reference data profile is used the absolute error is quite high in the beginning, it can be because of two reasons; one being not using the dedicated path in the beginning of the trip and other being the updated reference data profile is a bit volatile as it uses the averaged data from smaller number of trips. It can also be observed that very quickly the absolute error tends to zero and the trend is almost exponential in nature; this happens due to the availability of most recent data from capturing the operating domain characteristics of the electric city buses. The absolute error by the end of trip in the case where updated reference profile is used is much less than the one compared to the base reference profile. The similar phenomenon was observed in the case when the validation data cycle was used (see Fig. 18).

VI. Conclusion and Future Work

The typical prediction method for electric vehicles are based on simulating a physical model in time domain. However, a physical model is not easy to build due to the complexity on the parameters determination. Meanwhile, the time usage is not stable for different individual cycles for a given route. Alternatively, the energy consumption estimating model can be built in distance domain, which is calculated by the propelling force and the travelled distance. Especially for the public transport, the travelled distance for a selected route is fixed and the propelling force generated over the road can be characterized

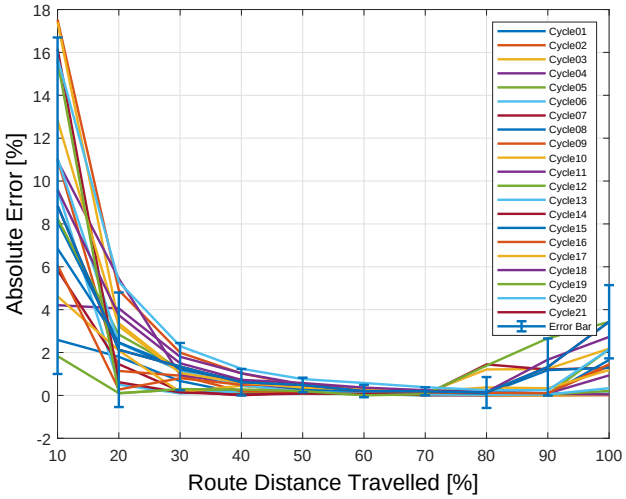


Fig. 16: Progression of absolute error for online energy estimation w.r.t % of distance travelled over the route.

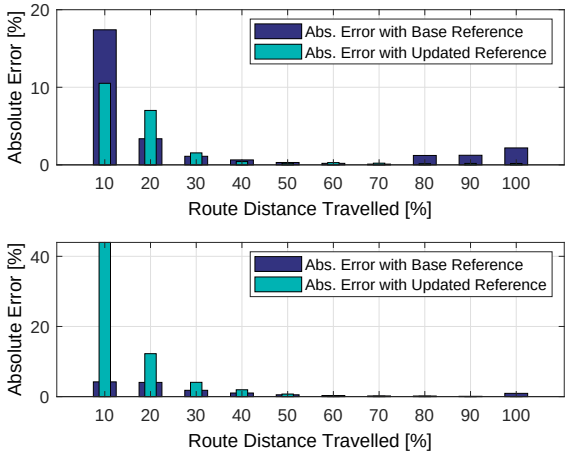


Fig. 17: Comparison of absolute error in estimations for online energy predictions done over base and updated reference profile.

as long as sufficient data was collected. The advantage of the model in the paper is that the model can just pack all the parameters together, without any necessity to determine individual parameters. Unlike the passenger cars, the city buses have varying passenger load during the trip. Therefore, the model excludes the mass from the propelling force by normalization. The mass for normalization is the curb weight of the vehicle. Based on this concept, the road characteristics indicator is defined. The total energy consumption for an electric vehicle also involves the auxiliary power as a function of time. With the data repeatedly collected from one selected route, the reference RCI profile and auxiliary power profile can be determined. This is for the offline estimate, which is only a constant value for a given route. The advantage of the offline algorithm is that the estimate is from the actual data which characterizes the route, while the disadvantage

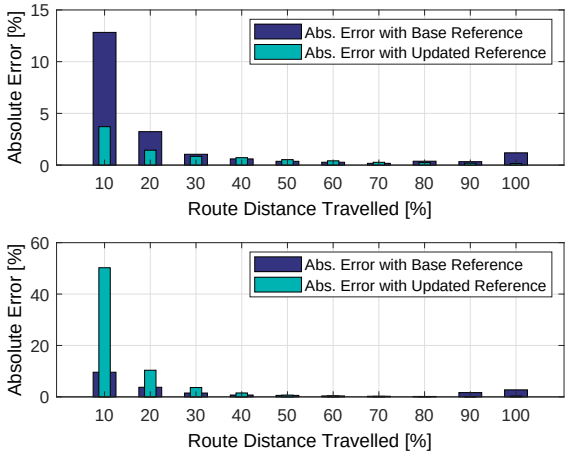


Fig. 18: Comparison of absolute error in estimations for online energy predictions done over base and updated reference profile.

is that the estimate is unchangeable for different situations. For enhancing the flexibility, a correction can be considered during the operation. Recursive algorithm and Kalman filter are chosen as the best choices in this case. Different testings were implemented with different data cycles and results show that the real-time correction can further improve the prediction accuracy significantly. For future work, the offline model can be further optimized by developing more relevant factors not only from the vehicle itself but also the environmental aspect. For instance, the temperature varies from season to season or from day to night, which also has an influence on the energy consumption. The time of the operation determines the level of the variation of passenger load, which is also a big component in the energy usage. Considering more factors can improve the estimating accuracy, but in the meantime will increase the complexity of the model.

REFERENCES

[1] Nilsson M., Electric Vehicle: The phenomenon of range anxiety, Lindholmen Science Park, Sweden, June 2011.

[2] Beckers, C. J. J., Besselink, I. J. M., Frints, J. J. M., and Nijmeijer, H. (2019, June). Energy consumption prediction for electric city buses. In 13th ITS European Congres.

[3] J. G. Hayes, R. P. R. de Oliveira, S. Vaughan, andM. G. Egan, Simplified electric vehicle power train models and range estimation, in Proc. Veh. Power Propulsion Conf., Chicago, IL, USA, Sep. 2011, pp. 1-5.

[4] C. Fiori, K. Ahn, and H. A. Rakha, Power-based electric vehicle energy consumption model: Model development and validation, Appl. Energy, vol. 168, pp. 257-268, 2016.

[5] K. N. Genikomsakis and G.Mitrentsis, A computationally efficient simulation model for estimating energy consumption of electric vehicles in the context of route planning applications, Transp. Res. D, Transp. Environ., vol. 50, pp. 98-118, 2017.

[6] P. Ondruska and I. Posner, Probabilistic attainability maps: Efficiently predicting driver-specific electric vehicle range, in Proc. Intell. Veh. Symp., 2014, pp. 1169-1174.

[7] B. Zheng, P. He, L. Zhao, and H. Li, A hybrid machine learning model for range estimation of electric vehicles, in Proc. Global Commun. Conf., Washington, DC, USA, Dec. 2016.

[8] C. De Cauwer, J. Van Mierlo, and T. Coosemans, Energy consumption prediction for electric vehicles based on real-world data, Energies, vol. 8, no. 8, pp. 8573-8593, 2015.

[9] Karlsson, R., Hammarstrm, U., Srensen, H., and Eriksson, O. (2011). Road surface influence on rolling resistance: coastdown measurements for a car and an HGV. Statens vg-och transportforskningsinstitut.
[10] Zhou, Y., Li, H., Ravey, A., and Pra, MC (2020). An integrated predictive energy management for light-duty range-extended plug-in fuel cell electric vehicle. Journal of Power Sources , 451 , 227780.
[11] Rhode, S., Van Vaerenbergh, S., and Pfriem, M. (2020). Power prediction for electric vehicles using online machine learning. Engineering Applications of Artificial Intelligence , 87 , 103278.
[12] Yi, Z., and Bauer, PH (2017). Adaptive multiresolution energy consumption prediction for electric vehicles. IEEE Transactions on Vehicular Technology , 66 (11), 10515-10525.
[13] Deschłnes, A., Gaudreault, J., Rioux-Paradis, K., and Redmont, C. (2020). Predicting Electric Vehicle Consumption: A Hybrid Physical-Empirical Model. World Electric Vehicle Journal, 11(1), 2.
[14] Kim, E., Lee, J., and Shin, KG (2013, April). Real-time prediction of battery power requirements for electric vehicles. In 2013 ACM / IEEE International Conference on Cyber-Physical Systems (ICCPS) (pp. 11-20). IEEE.
[15] Wang, J. (2016). Battery electric vehicle energy consumption modelling, testing and prediction: a practical case study.
[16] Broeksteeg, K. (2011). Parallel regenerative braking control for the TU/e Lupo EL. DC Report 2012. 066.
[17] Maalej, K., Kelouwani, S., Agbossou, K., Dub, Y., and Henao, N. (2014). Long-trip optimal energy planning with online mass estimation for battery electric vehicles. IEEE Transactions on Vehicular Technology , 64 (11), 4929-4941.
[18] Lennart, L. (1999). System identification: theory for the user. PTR Prentice Hall, Upper Saddle River, NJ , 1-14.
[19] Bishop, G., Welch, G. (2001). An introduction to the kalman filter. Proc of SIGGRAPH, Course, 8(27599-23175), 41.
[20] Carlson, NA (1973). Fast triangular formulation of the square root filter. AIAA journal , 11 (9), 1259-1265.

Yuzhe Ma received the M.Sc. in mechanical engineering from the College of Automotive Engineering, Jilin University, Changchun, China, in 2016. He is currently working toward the Ph.D. degree in the energy consumption prediction of electric vehicles with the Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands. His interests focus on modelling, data processing and testing on electric passenger vehicles and electric public transport.

Dhruv Jagga has received a bachelors degree in Instrumentation Engineering from Kurukshetra University, India in 2014. He later came to, The Netherlands to expand his knowledge base and received a masters degree in Systems and Control from the Delft University of Technology in 2017. After that period, he joined MariLogic-Marine Automation and worked as a Control System Engineer and became responsible for modelling and control of the gangway systems. He also handled various other tasks involving commissioning and testing of these safety-critical systems. In 2018, he gets associated with Eindhoven University of Technology and pursued his post masters studies (PDEng) in Automotive Systems Design.

Igo Besselink received the graduation in mechanical engineering from the Delft University of Technology, Delft, The Netherlands, in 1990, and received the Ph.D. degree from the TU Delft in 2000. He is an Associate Professor with the Eindhoven University of Technology, Eindhoven, The Netherlands. Thereafter he joined the Fokker Aircraft and became responsible for the analysis of landing gear design loads and stability. In 1996, he joined the TNO automotive, Delft, and was responsible for tyre simulation software development and various projects related to vehicle dynamics. In 2002, he was a part-time Lecturer in the field of vehicle dynamics in the Dynamics and Control group of Prof. Henk Nijmeijer. In 2008, he became full-time employed as an Assistant Professor with the Eindhoven University of Technology. In January 2016, he was appointed as Associate Professor, while still being a member of the Dynamics and Control group. His research interests include tyre modeling, dynamics of commercial vehicles, vehicle control, and battery electric vehicles.

Henk Nijmeijer received the M.Sc. and Ph.D. degrees in mathematics from the University of Groningen, Groningen, The Netherlands. He is a Full Professor with the Eindhoven University of Technology, Eindhoven, The Netherlands, and chairs the Dynamics and Control group. He has published a large number of journal and conference papers, and several books. Dr. Nijmeijer is/was a member of the editorial board of numerous journals. He is an Editor of Communications in Nonlinear Science and Numerical Simulations. He was awarded the IEE Heaviside premium in 1990. He is appointed Honorary Knight of the golden feedback loop (NTNU) in 2011. Since 2011, he has been an IFAC Council Member. From January 2015, he has been a Scientific Director of the Dutch Institute of Systems and Control (DISC). He is recipient of the 2015 IEEE Control Systems Technology Award. He is program leader of the Dutch research program Integrated Cooperative Automated Vehicles (i-CAVE). Since April 2017, he has been the Director of the Graduate Program Automotive Technology of the Eindhoven University of Technology.

PO Box 513
5600 MB Eindhoven
The Netherlands
tue.nl

PDEng AUTOMOTIVE SYSTEMS DESIGN
Track AUTOMOTIVE SYSTEMS DESIGN

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY