TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Andres Saal 204763IVEM

Driver Behavior Classification Application Based on Massive Machine Type Communication Technology

Master's thesis

Supervisor: Muhammad Mahtab Alam
PhD

Priit Roosipuu
MSc

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Andres Saal 204763IVEM

Massiivsel masinkommunikatsioonitehnoloogial põhinev juhi käitumist klassifitseeriv rakendus

Magistritöö

Juhendaja: Muhammad Mahtab Alam
PhD

Priit Roosipuu
MSc

Tallinn 2022

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Andres Saal

09.05.2022

Abstract

Vehicles are nowadays equipped with a lot of sensors that output data which could be used for driver behavior analysis. Classifying drivers based on their driving style has multiple applications and such a system can be very useful for rental companies, car insurance providers, or fleet management.

In this thesis vehicle sensor data was used to build a driver behavior classification system. The prototype device consists of hardware that is able to request data from car onboard sensors and from additional sensors which is sent to the cloud server through LTE Cat-M1 communication technology.

The data was then used to build machine learning models for driver behavior classification. Two different approaches were used: a time-based dataset where a driver was labeled as a bad driver for a set period of time and an event-based dataset where the driver was labeled as a bad driver for only the moments where thresholds were exceeded. The classification overall accuracies from the experiments, were respectively 89% and 99%.

This thesis is written in English and is 58 pages long, including 5 chapters, 35 figures and 12 tables.

Annotatsioon

Massiivsel masinkommunikatsioonitehnoloogial põhinev juhi käitumist klassifitseeriv rakendus

Tänapäeva sõidukid on varustatud paljude anduritega, mille andmeid saab kasutada juhi käitumise analüüsimiseks. Sõidustiilil põhineval juhi klassifikatsioonil on mitmeid rakendusi - selline süsteem võib olla väga kasulik rendiettevõtetele, autokindlustuse pakkujatele või autopargi haldamisel.

Käesolevas magistritöös kasutati sõiduki andurite andmeid juhi käitumist klassifitseeriva süsteemi loomiseks. Prototüüpseade koosneb riistvarast, mis on võimeline koguma andmeid nii auto sisemistelt anduritelt kui ka lisa anduritelt, et saata need seejärel läbi LTE Cat-M1 sidetehnoloogia virtuaalserverisse (cloud server).

Kogutud andmeid kasutati juhi käitumist klassifitseeriva masinõppe (machine learning) mudelite koostamiseks. Hindamiseks kasutati kahte erinevat lähenemisviisi – ajapõhist andmestikku, kus juht märgiti etteantud ajavahemikus halvaks juhiks ning sündmuste põhist andmestikku, kus juht märgiti halvaks juhiks ainult nendel hetkedel, mil ta ületas lävendid. Katsete põhjal oli klassifitseerimise üldine täpsus vastavalt 89% ja 99%.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 58 leheküljel, 5 peatükki, 35 joonist, 12 tabelit.

List of abbreviations and terms

3GPP	3rd Generation Partnership Project
4G	Fourth generation
5G	Fifth generation
ACC	Adaptive Cruise Control
AI	Artificial Intelligence
API	Application Programming Interface
AT	ATtention
ECU	Electronic Control Unit
GNSS	Global Navigation Satellite System
GPIO	General-Purpose Input/Output
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HO	HandOver
I2C	Inter-Integrated Circuit
IoT	Internet of Things
KNN	K-Nearest Neighbor
LPWAN	Low Power Wide Area Network
LTE	Long Term Evolution
MEMS	Micro Electro-Mechanical System
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
NB-IoT	Narrowband Internet of Things
NMEA	National Marine Electronics Association
OBD	On-Board Diagnostics
PDP	Packet Data Protocol
RF	Random Forest
RMC	Recommended Minimum Data
RRC	Radio Resource Control

RX	Receive
SBC	Single-Board Computer
SSH	Secure Shell Protocol
SVM	Support Vector Machines
TX	Transmit
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus

Table of contents

List of figures

List of tables

1 Introduction

As decades have gone by and the technologies have evolved, the number of sensors and electronics in vehicles has risen. Nowadays when buying a car, it is normal for the car to have adaptive cruise control (ACC), lane assist, speed sign detection, driver fatigue detection, etc. The amount of data available in a vehicle is huge. This data could be used for many things for example driver behavior analysis [1]. In Figure 1 the general scheme of driver behavior research is given. The components of the scheme are as follows:

- Vehicle – Source of data from the sensors
- Reader – Device that will gather the data and send it to a database
- Data – Data in the database
- Pre-processing – Modifying and preparing the data for analyzing and classification
- Behavior Classification – Classifying the behavior

Vehicle
Reader
Data
Pre-processing
Behavior classification

Figure 1. General scheme of driver behavior research

Driver behavior analysis and classification is a topic that is gaining attention in many applications. Companies that have a large fleet of cars could monitor and understand how their vehicles are being used. For example, they can see if the drivers are economic or using more fuel than needed. Insurance companies can provide new insurance models where drivers only pay for the insurance according to when they use the car. Drivers could also be monitored to give personalized insurance rates. Another user of driver classification could be car rental companies. Customers can usually rent cars with fixed fee rates and the style of usage itself is not directly monitored. If the car is returned visibly in the same condition as it was before the rental period, then no additional fees will be added for the customer. However, this can lead to situations where the car is taken to a parking lot for drifting or off-road driving which can cause excessive wear on the car that is not possible to detect just by looking at it from the outside. As a result, the driver who misused the car will still pay the same amount as a customer that used the car only to drive to work and back without causing excessive wear on the car.

If the car was used by a driver with bad driving behavior, the vehicle might have nonvisible damage and could need costly repairs and might need to be away at the workshop for an extended period. These unavailable cars are then not able to be rented out and will not make money for the rental companies or might even lose them money. Rental companies need the cars to always be available and in good mechanical condition.

It is also necessary for public safety to keep the aggressive drivers away from the road as they tend to break safety rules and will also be dangerous to other drivers and pedestrians.

1.1 Problem Statement

The purpose of this thesis is to develop a driver behavior classification system that is working using cellular IoT technology and is classifying the driver behavior.

1.2 Background

1.2.1 LTE Cat-M1 and LTE NB-IoT

LTE Cat-M1 and LTE NB-IoT are both cellular communication protocols that can be used for Internet of Things (IoT) devices to send or receive data between the device and the Internet. They are part of fourth-generation (4G) cellular network technology. Although they were introduced as a part of 4G-LTE specifications, they are also fully recognized as fifth-generation (5G) technologies which means that they can be used even after 4G will be upgraded to 5G. The previous statement means that all the developed IoT systems can be used for a long time without the fear of becoming disconnected from the Internet [2].

The good thing about cellular IoT technologies is that they have very good coverage. Everywhere where cell towers are located there will be a reception for the IoT devices. Other technologies on the other hand need special infrastructure to be built [3].

More information about the LTE Cat-M1 and LTE NB-IoT technologies can be seen in the following Table 1 [3].

Table 1. Information about NB-IoT and Cat-M1 [3]

Technology	LTE NB-IoT	LTE Cat-M1
Bandwidth	180 kHz	1.4 MHz
Spectrum	Licensed	Licensed
Frequency Bands	700-2100 MHz	700-2100 MHz
Standardization	3GPP Release 13	3GPP Release 13
Uplink	250 kbit/s, 20 kbit/s	1 Mbit/s
Downlink	250 kbit/s	1 Mbit/s
Latency	1.6s-10s	10ms-15ms
Duplex mode	Half Duplex	Full or Half Duplex
Battery life	10+ years	10 years

LTE NB-IoT had upgrades in 3rd Generation Partnership Project (3GPP) release 14 in which the mobility of NB-IoT was improved. It brought Radio Resource Control (RRC) connection re-establishment which allows the devices to make cell handovers without re-negotiating a new connection [4].

The most important requirements for the technologies for this thesis are the ability to have a country-wide area of coverage and reliability. They both have the coverage of LTE cell towers, and they can handle moving users. The LTE Cells are widely deployed and allow building IoT networks fast without needing to change infrastructure.

1.2.2 OBD-II

On-Board Diagnostics II (OBD-II) is the second generation of On-Board Diagnostics systems. It is a standardized system for vehicles that allows connections with the vehicles Electronic Control Unit (ECU). From the OBD port in the vehicle, different parameters can be accessed like speed, engine data, and emissions data [5]. Electric vehicles are not required to be comply the OBD-II standard [6].

1.3 Overview of the Thesis

This thesis is structured as follows:

In chapter 2, the state of the art is given about driving behavior classification with machine learning. Related work about driving behavior analysis, classification, and measuring is also described with a comparison table.

In chapter 3, the proposed device is described. Hardware data with the initialization information is given. In the second part, the software is also explained together with the cloud server. In the third part, the prototype device is shown with all its modules and wiring diagram.

In chapter 4, the experiments with results are explained, analyzed, and graphically compared.

In chapter 5, the conclusion with the future work of the thesis is discussed.

2 State of the Art

In this chapter, in the first subchapter, the driver behavior classification with machine learning is explored. In the second subchapter, related works and previous works are reviewed.

2.1 Driver behavior classification with machine learning

Data could be analyzed through different machine learning algorithms for behavior classification, but according to the chapter 2.2.1 in this thesis in the conclusion of related work, it is found that some of the most commonly used algorithms are Support Vector Machines (SVM), K-means, and Random Forest (RF).

SVM is a supervised machine learning algorithm that can be used both for classification and regression tasks. In SVM a hyperplane is found in the data that best separates data classes. It is also necessary to have maximized margin between the hyperplane and nearest data points. The more dimensional data is used, the more dimensional the hyperplane will be [7]. In the case of 2 input features in the data, the support vector classifier will be a line. In Figure 2 the hyperplane can be seen with such data.

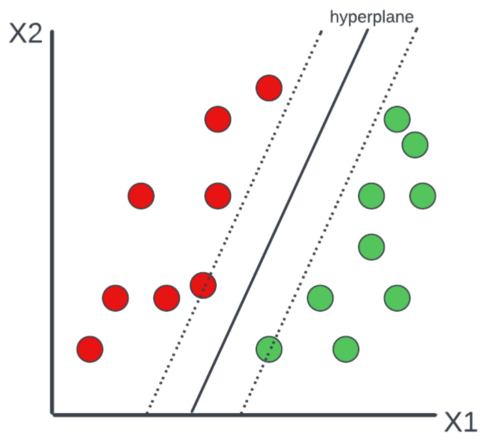


Figure 2. SVM hyperplane with two input features

SVM model can be tuned with multiple parameters for example C and Gamma. These both are hypermeters that are set before training the model. C is a hypermeter to control the error of the model, but the low error does not mean better boundaries for decisions. Gamma is a hypermeter that gives curvature for the decision boundary. Tuning these both and testing can improve the model output. For each dataset different hypermeter values could be best which means that there are no universal values for these [8].

K-means is a popular unsupervised machine learning algorithm. It groups similar data points to find similarities and patterns. The user will have to choose the K number which defines the number of pre-defined clusters. If the number will be 2, then 2 clusters will be made. A ready K-means machine learning model will show in which cluster test data points will belong to [9]. In Figure 3 a K-means plot can be seen where K=3 and 3 clusters are made.

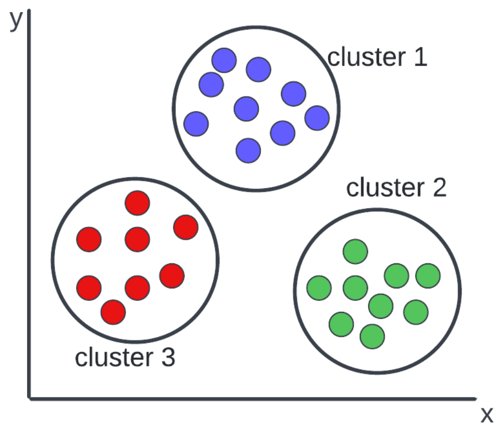


Figure 3. K-means data plot with 3 clusters

Random Forest is a supervised machine learning algorithm that can be used both for classification and regression problems. It combines multiple decision trees for a result. Before training, three parameters must be set. These are node size, number of trees, and the number of features. In the case of regression, the decision trees will be averaged for the result, but for classification, the most selected class selected by the trees will be the result [10]. In Figure 4 the diagram of the Random Forest Classifier is shown.

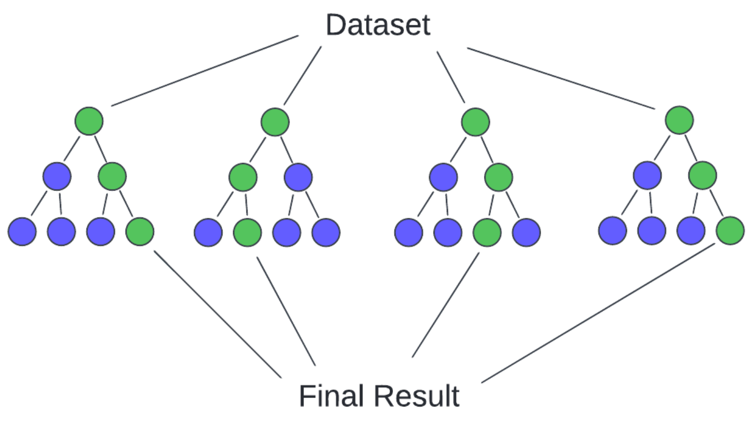


Figure 4. Diagram of Random Forest Classifier [10]

2.1.1 Classification metrics

The confusion matrix gives us a matrix output that describes the complete performance of the machine learning classification model (Figure 5). To understand the matrix better, the machine learning classification classes need to be thought of as a binary result 1 or 0. The confusion matrix consists of 4 important terms [11]:

- TP – True Positives where the prediction was 1 and the actual output was also 1.
- TN – True Negatives where the prediction was 0 and the actual output was also 0.
- FP – False Positives in which the prediction was 1 and the actual output was 0.
- FN – False Negatives in which the prediction was 0 and the actual output was 1.

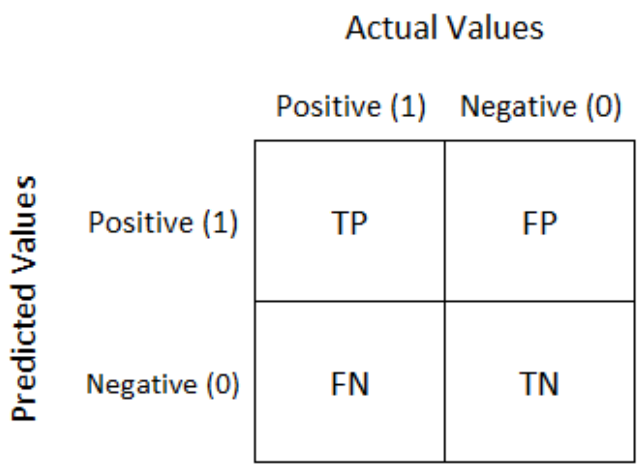


Figure 5. Confusion Matrix [12]

From the Confusion Matrix, multiple other metrics can be calculated [12]:

- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $F1\ score = \frac{2*Precision*Recall}{(Precision+Recall)}$

2.2 Related Work

Digital Matter is an Australian company that is one of the most widely used Low Power Wide Area Network (LPWAN) asset tracking hardware producers. The devices are designed for companies that have to manage a fleet of vehicles. All the vehicles can be tracked, and the drivers can be analyzed to see if vehicles receive harsh treatment while driving and cornering and if fuel is used too much by driving style or idling. They have a wide portfolio of different products with different connectivity technologies. Multiple devices are using NB-IoT and Cat-M1. The products also have accident and rollover detection so in case of an accident the help can reach drivers faster [13].

MyOrien has a product where the user connects the On-Board Diagnostics II (OBD-II) reader to a car and then afterward the data from the car is logged and sent to the server. Then the data is analyzed, and the user is given a score depending on how the car was driven. Vehicle speed, harsh braking, acceleration, and idling are the data that are being analyzed [14].

In [15] a product called AutoCoach is presented. It is an Artificial Intelligence (AI) agent that classifies vehicle drivers depending on how their driving personalities are. They have made a cloud-based Android application that collects, analyzes, and learns from driver's driving data to provide feedback. The Graphical User Interface (GUI) interface provides real-time feedback on the results either giving warnings or rewards. In their study, an on-the-road pilot user study was conducted. The study results show that a personality-based driving agent is more accurate than a non-personality-based. They use embedded motion sensors from phones to detect motion and events while driving. For Machine Learning (ML) algorithms, SVM and K-Nearest Neighbor (KNN) were chosen [15].

This research proposes a new driving behavior research method based on SVM and oversampling. In their work, the data was collected with an OBD-II reader. Then the noise of the data was effectively removed with the Kalman filter. As the positive and negative samples were unbalanced, some of the samples were oversampled by the Random-SMOTE algorithm [16].

In [17] a novel approach was given to profile drivers using Machine Learning. In the method, the OBD-II interface was used to collect data from vehicles. Vehicle speed, engine speed, throttle position, and calculated engine load were used. Then the driver is profiled visually using the K-means clustering algorithm as well as the Elbow method. For the determination thresholds, they have been taken from [18]. The Driver Score is described as the percentile of bad driving behaviors over the observed period.

In [19], vehicle speed was taken from OBD-II but a microcontroller with multiple additional sensors was added like a shock sensor, digital multiplexer, camera module, GPS location detector, and GSM module. The prototype monitored vehicle speed and when the speed gets too high the device sent an SMS with the speed and location. In case of collision, the device will take a picture of the interior of the car and send it as well.

Last year in Taiwan, there was a development of a driving behavior analysis system that used a Dual USB webcam, NVIDIA Jetson Nano, Raspberry PI 3B, and PiCAN2 CAN-Bus board. This system could collect road condition data, drivers' faces, and car data from ECU. The developed system could identify four types of driving patterns: turning without using turn signals, turning without looking into the rearview mirror, distracted driving, and fatigued driving. The accuracy was good, being between 79%-89% [20].

In this research [21] a prototype device was built using Arduino Uno and MPU6050 accelerometer. They gathered driving data by driving around and measuring accelerations. An algorithm was proposed for detecting aggressive driving behavior. In their research, the accuracy of detection is 97% which can be used for future driver aggression detection.

In this [22] analysis, the authors have developed a mobile application that uses smartphone cameras and built-in sensors. The developed application monitors driver behavior to determine unsafe driving, distraction, and drowsiness. The application is available in Google Play Store for everyone interested. The application was tested by several volunteers from different countries who confirmed that the application determines dangerous activities correctly in most cases.

In this [23] paper, a device is designed that monitors vehicles. It consists of an OBD scan tool, Raspberry Pi, and a cloud monitoring application. OBD-II port is used to gather data and then Raspberry Pi uploads it to the cloud through a cellular connection. The data is accessed from the cloud with a monitoring application. With this application, it is possible to read and discover issues with cars and get data from car onboard sensors.

In [24] this work a system is proposed where different parameters are taken from car OBD-II port like speed, acceleration, deceleration, jerk and compared with standard limits and deviations. Driver behavior is divided into 3 levels to show how good he/she is driving.

In this [25] research a novel driver performance model is proposed. For driver performance modeling two nonlinear regression machine learning algorithms are used: Artificial Neural Network and Adaptive neuro-fuzzy inference system. Data is collected from 18 different drivers on a vehicle simulator. The algorithms were designed to predict a driver's ability to keep in the middle of the driving lane and respect the current speed limit. In the end, the Artificial Neural Network was slightly more accurate.

In this paper [26], research was conducted on real drivers on a driving simulator. They developed a system to detect drunk driving. To detect drunk driving behavior a method with a Random Forest algorithm is proposed. In testing, they used 25 drivers and got 265 valid samples.

This paper [27] proposes a method to analyze and classify driver behavior on data from the CAN bus. Data were collected from 64 people who drove over 2000 trips with 10 cars. In the paper, they propose unsupervised learning techniques that will divide drivers into different groups. They also calculate the minimum required amount of data to preserve the dividing ability.

In this [28] study, a driver profiling system was developed. They used data obtained from accelerometer and gyroscope and then used different machine learning algorithms. The algorithms included Decision Tree, Random Forest, Artificial Neural Network, Support-Vector Machine, K-Nearest Neighbor, Naive Bayes, and K-Star. In their experiments, the K-Star algorithm was the most accurate with 100% accuracy.

In [29], a method is proposed to detect drunk driving. They use vehicle onboard sensors, accelerometer, and GPS coordinates that are provided by the diagnostics dongle. With the gathered data, a window-based approach was used for data smoothing and feature extraction. For machine learning, they use the Logistic Regression algorithm which achieves an accuracy of 82%.

Table 2. Comparison of related work

Work	**Topic**	**Algorithm**	**Data**	**Data origin**
[13]	Digital Matter devices	No information	Real data	OBD-II
[6]	MyOrien device	No information	Real data	OBD-II
[15]	Driver behavior feedback	SVM KNN	Real data	Smartphone
[16]	Dangerous driving behavior	SVM	Real data	OBD-II
[17]	Driver profiling	K-means	Real data	Previous study
[19]	Car driver monitoring	Comparing with set thresholds	Real data	OBD-II and GNSS module
[20]	Driving behavior analysis	Comparing with set thresholds	Real data	OBD-II and cameras
[21]	Detecting aggressive driving behavior	Dynamic time warping	Real data	Accelerometer and gyroscope
[22]	Driver behavior analysis	Comparing with set thresholds	Real data	Smartphone

[23]	Vehicle state monitoring	Comparing with set thresholds	Real data	OBD-II
[24]	Driving behavior analysis	Comparing with set thresholds	Real data	OBD-II and additional sensors
[25]	Driver performance model	ANN ANFIS	Simulated	Simulator
[26]	Drunk driving recognition	Random Forest	Simulated	Simulator
[27]	Driving behavior analysis	K-means	Real data	Previous study without information
[28]	Driver risk assessment	Decision Tree Random Forest SVM K-Star	Real data	Accelerometer and gyroscope
[29]	Drunk driving detection	Logistic Regression	Simulated	OBD II and additional sensors

2.2.1 Conclusion

It can be concluded from the literature review, that it is possible to use different algorithms for this type of analysis. The most common algorithms used in the studied literature were SVM, K-means, and Random Forest. From the data origin side, the data is mostly taken from the OBD-II reader. Additional sensors can be added to get more features. Thresholds and behavior classes are different between works and have no common rules.

3 Proposed device

In this chapter, an overview of the needed hardware and software is given. The simple diagram of the device and its connections is given in Figure 6. The proposed device will need to fulfill the following two requirements:

- acquire all needed data of the vehicle
- send all the data to a server for analyzing

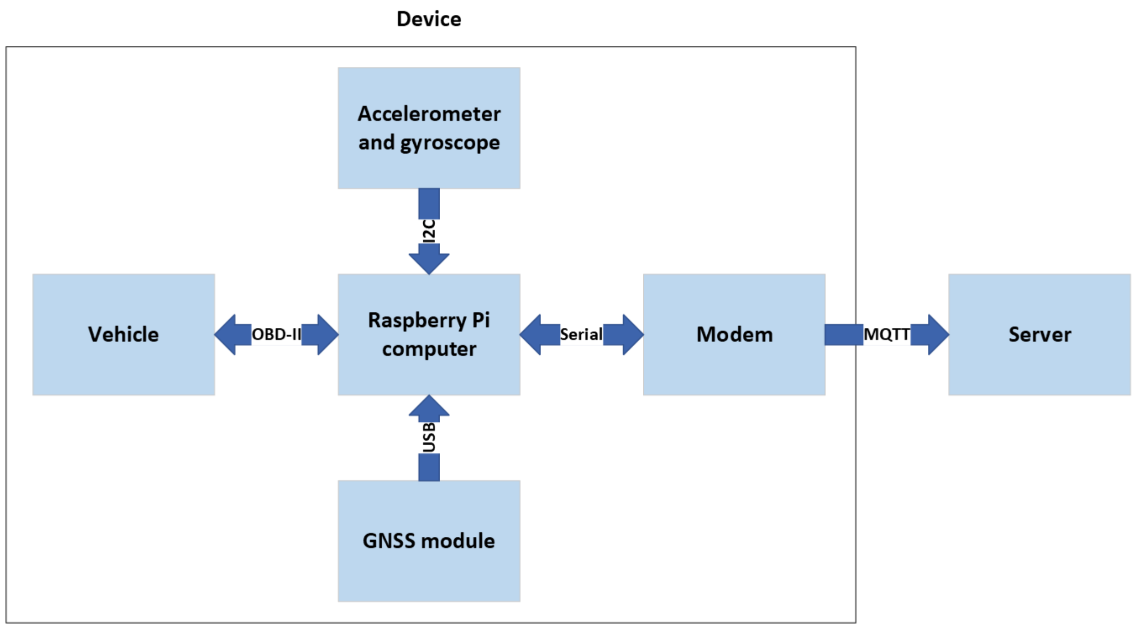


Figure 6. Simple diagram of the device and its connections

3.1 Hardware

The chosen hardware consists of 6 modules:

- The Raspberry Pi computer which contains the software and controls all the connected devices
- The Waveshare SIM7070G modem that connects to the Internet and uploads data
- G-Mouse Universal Serial Bus (USB) Global Navigation Satellite System (GNSS) module that gets the device location and speed
- ELM327 USB OBD-II reader that gets data from car onboard sensors
- ITG/MPU GY-521 accelerometer and gyroscope sensor module that gets the car linear acceleration and angular velocity

- Hama "slim" 1:4 USB hub that gives the possibility to connect multiple USB devices to Raspberry Pi computer

3.1.1 Computer

Raspberry Pi Zero W is a very small-sized computer that is built for IoT applications. It is one of the smallest products in the Raspberry Pi single-board computer (SBC) line-up. With their small size and high capabilities, they are perfect for different IoT projects [30].

For connectivity, the Raspberry Pi Zero W has multiple possibilities. For wireless, we can have Wi-Fi and Bluetooth connections. For mechanical connections, one USB port, HDMI, and 40 pins called General Purpose Input/Output (GPIO). In Figure 7 is the board with its GPIO pinout. In Table 3 the main characteristics of the board are given [30].

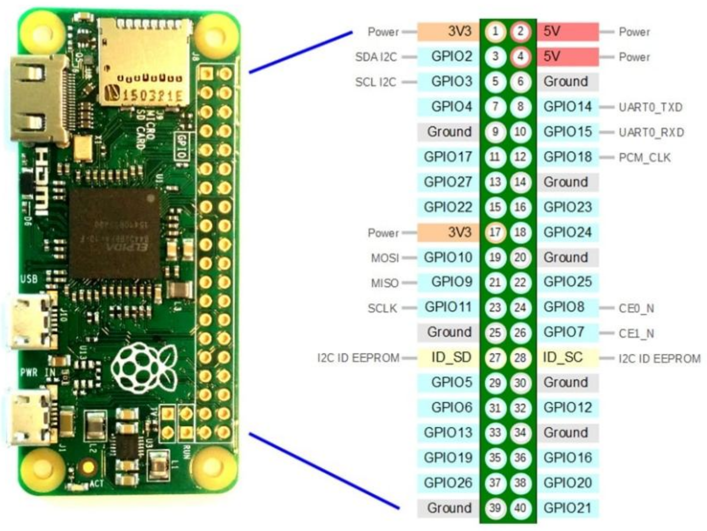


Figure 7. Raspberry Pi Zero W with its GPIO pins [31]

Table 3. Raspberry Pi Zero W main characteristics [30]

CPU	1 GHz, Broadcom BCM2835
RAM	512 MB
Wireless	802.11n / Bluetooth 4.1 / LE
Ports	Micro USB, mini HDMI
I/O	40 GPIO Pins, CSI camera connector
Size	66.0mm x 30.5mm x 5.0mm
Weight	9 grams

The Raspberry Pi Zero W was used without display and keyboard through a Secure Shell Protocol (SSH) connection from a laptop. SSH enables to connect to the Raspberry Pi and use the terminal for settings and software running. The software that will control the whole device will be on the Raspberry Pi computer itself and will be started and monitored through the SSH connection.

3.1.2 Modem

Waveshare SIM7070G HAT modem was chosen for transmitting data from Raspberry Pi as it was available at the university (Figure 8). This is a low-power narrowband cellular IoT communication module for Raspberry Pi. The modem supports LTE NB-IoT, LTE Cat-M, and General Packet Radio Service (GPRS) connectivity. For additional functions, it has GNSS positioning capabilities. It has a standard 40-pin Raspberry Pi GPIO header so that it can be connected on top of the Raspberry Pi board. It also comes with a USB interface [32].

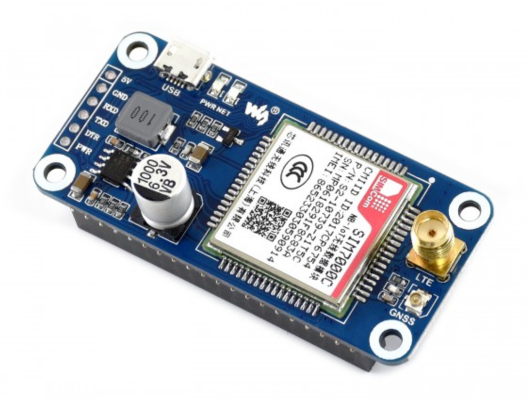


Figure 8. Waveshare SIM7070G modem [32]

The setting up and controlling of the modem is done via AT commands that can be sent either through a USB connection or Universal Asynchronous Receiver-Transmitter (UART) connection. The modem supports a variety of communication protocols as follows: TCP, UDP, HTTP, HTTPS, TLS, DTLS, PING, LWM2M, COAP, and MQTT [32].

It was discovered during the first tests that the device is not working very well in NB-IoT mode during a car movement because of no support of HandOvers (HO). When the device is moving and changing cells, the modem will lose connection, although the device is based on 3GPP release 14 which should support it. At around the same time, the service provider activated Cat-M1 networks fully in Estonia, which meant that the technology settings could be easily changed to start using the modem. Cat-M1 is designed more for devices that need mobility and during the testing, in this stage, no issues were found.

Setting up the modem was done through a serial connection to the Raspberry Pi. In the written software, the necessary AT commands were added so that the modem could be configured correctly. The modem was configured to work in only LTE Cat-M1 mode. MQTT settings were changed according to the server IP and device ID to enable the modem to send data to the cloud server. Configuration commands that were used can be seen in the following Table 4.

Table 4. SIM7070G AT commands [33]

Command	**Description**
AT	Test command
ATE0	Turn off echo for sent messages
AT+CFUN	Set phone functionality
AT+SMCONF	Set MQTT parameters
AT+CGDCONT	Define PDP Context
AT+CNACT	APP network active
AT+CGNSPWR	GNSS power
AT+SMCONN	MQTT connection
AT+SMPUB	Send MQTT packet
AT+SMDISC	Disconnect MQTT
AT+COPS	Operator selection
AT+CMNB	Preferred selection between Cat-M and NB-IoT
AT+CNMP	Preferred mode selection

3.1.3 GNSS module

Although the SIM7070G Modem has built-in capabilities for GNSS, it turned out that on this specific modem the GNSS cannot be used at the same time as the other modem communication functionalities are used and therefore additional GNSS module had to be added. The added GNSS device is G-Mouse USB, which contains a U-Blox Neo-7 module inside.

The connection can only be done with a USB. The module sends automatically National Marine Electronics Association (NMEA) messages from which different kinds of data can be read. In this thesis, the Recommended Minimum Data (RMC) message is used [34].

In the RMC message, the GNSS module sends data about time, status, latitude, longitude, speed, course, date, and magnetic variation. In this thesis, the latitude, longitude, and speed are used for analyzing the driver. In the software, the data is modified to change the data default units. For example the RMC message will look as follows: $GPRMC,173755,A,3216.22,N,02321.42,W,100.5,060.5,270795,008.3,E*62 and from this message we can find the following information [35]:

- 173755 – the time 17:37:55 UTC
- A – receiver warning, either A which is OK, or V which is a warning
- 3216.22,N – latitude 32 degrees 16.22 minutes North
- 02321.42,W – longitude 23 degrees 21.42 minutes West
- 100.5 – speed in knots
- 060.5 – true course
- 270795 – date 27 July 1995
- 008.3,E – magnetic variation 8.3 degrees East
- *62 - checksum

3.1.4 OBD-II reader

ELM327 USB OBD-II reader was chosen for reading car sensor values. This is one of the most widely used inexpensive readers, that can be used on different computers with different software. It is also possible to write your own code to use the reader in a more personally specified way. If using Python3, there is also a python-OBD library that can be imported to make the implementation of your own software quicker.

Python-OBD is a library that can in real-time handle the data from the vehicles OBD-II port. It is possible to get sensor data and perform diagnostics. OBD connections work in request-reply mode, if the user wants to get information from the sensor, it is needed to send a query. In Python-OBD this is done with query() function [36].

3.1.5 Accelerometer and gyroscope

ITG/MPU GY-521 module is used for accelerometer and gyroscope data (Figure 9). It contains an MPU6050 sensor. MPU6050 sensor is a Micro-Electro-Mechanical System (MEMS) that has a 3-axis accelerometer and 3-axis gyroscope inside it [37].

With the module, it is possible to measure acceleration, velocity, orientation and other motion detected values. Mostly these kinds of sensors are used in drones, self-balancing needs, and robots [37].

To use the module on Raspberry Pi, the Inter-Integrated Circuit (I2C) protocol has to be configured. For physical connection, only 4 pins are needed: Vcc, GND, SDA, and SCL. The communication will be done through I2C. For Python3 the Smbus library is also needed to import. The values of the gyroscope will be in g units and the accelerometer will be in degrees per second [38].

As the sensor measures movement, it is important to have it in a fixed position in the car facing the correct direction. As it measures every slight movement, the road surface and engine vibration could also affect the measurement results. During testing, it was found that the sensor does not need extra calibration before each test.

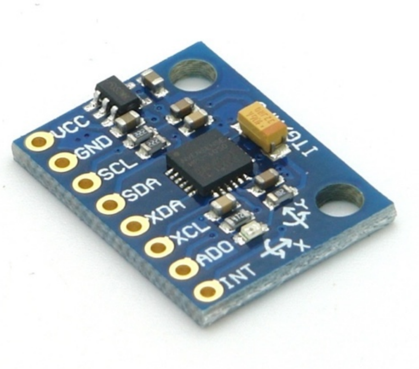


Figure 9. GY-521 MPU6050 module [39]

3.1.6 USB Hub

As the Raspberry Pi Zero W has only one USB port for communications, but we need USB connections for both the GNSS module and OBD reader, then a USB Hub is also needed. For this, a simple Hama “slim” 1:4 USB 2.0 Hub was added (Figure 10). With this USB hub, all of the needed devices were connected with Raspberry Pi without any issues.

Figure 10. Hama USB Hub [40]

3.2 Software

3.2.1 Code

Software for the device to acquire and send data was written in Python3. The communication module of the code is from previous Master thesis work by Priit Kullerkupp. [41] The code was used to develop a prototype for predictive vehicle maintenance working in NB-IoT. The communication part of the code has been modified to fit the needs of this thesis. The modem communication type was changed to Cat-M1. Software needs manual startup from the user and for that, a laptop was needed in the vehicle to see the Raspberry Pi interface through SSH and give commands. Figure 11 shows the basic flow of the software.

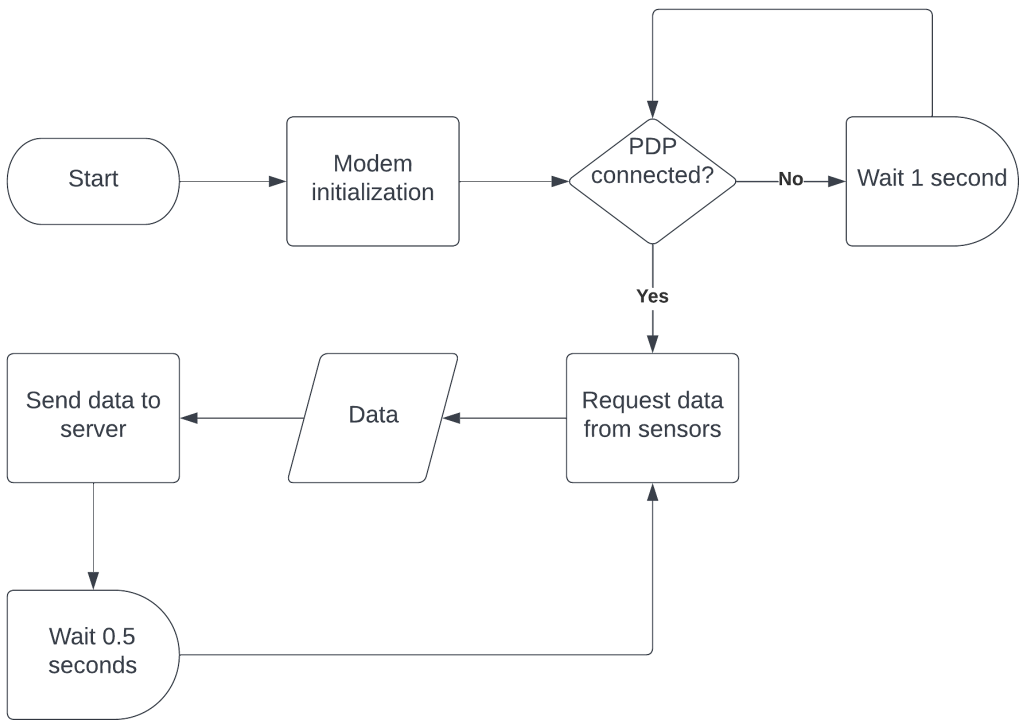


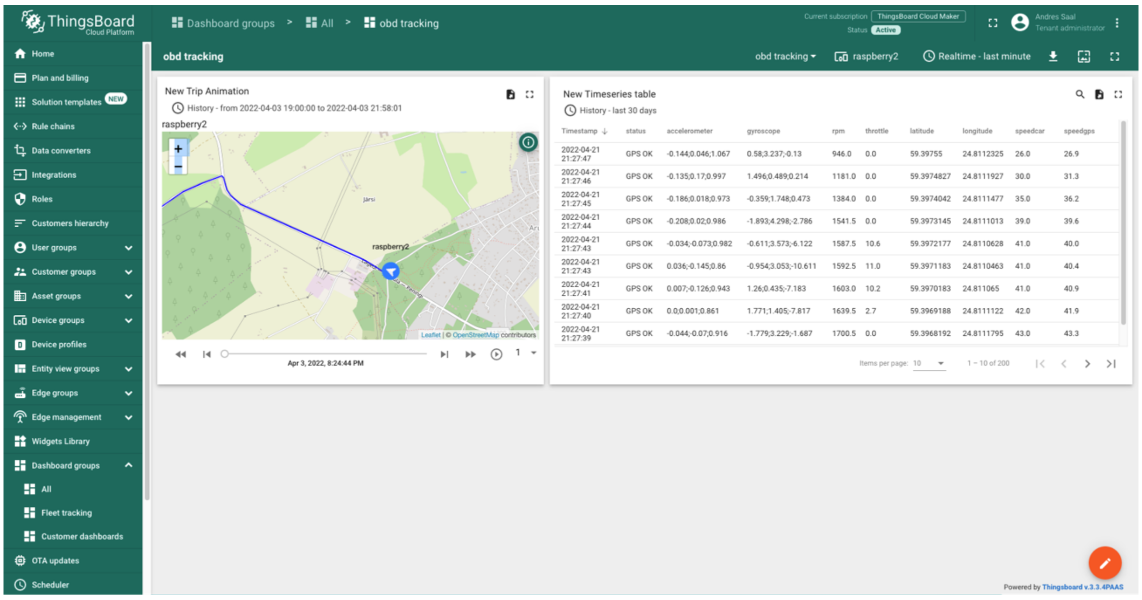
Figure 11. Flowchart of the software

The software starts by initializing the modem – once it is booted up and settings set the software moves to checking the Packet Data Protocol (PDP) connection. If a PDP connection is available, this means that the device has a connection to the internet. Then the software requests data from the car and sensors. The timestamp will be added to the data and then the data will be formatted as a JSON string and sent to the server using the MQTT protocol. After sending the data, there is a 0.5-second delay to keep the dataset interval as 1 second and after that, the loop will start again by requesting new data from sensors and vehicle.

3.2.2 Server

For the server, a ThingsBoard Cloud-based server was used. This meant that no personal server was needed to be set up. ThingsBoard offers cloud-based IoT platform for fast and easy set-up of service. In ThingsBoard Cloud the device was registered as raspberry2, then the server IP and device access token were copied for the needed MQTT connection. In Figure 12 the environment can be seen opened on a PC. In the environment dashboard, all kind of visual information can be shown with the received data. In this example, the device location and received data are shown.

MQTT is a very lightweight data transmission protocol that is very popular in IoT applications. MQTT uses a publish/subscribe architecture which reduces bandwidth by 95% compared with traditional protocols [42]. This protocol is also supported by ThingsBoard Cloud and is easy to set up. To send messages to the server, the device posts messages to specific telemetry topic **v1/devices/me/telemetry**.



New Timeseries table

History - last 30 days

Timestamp	status	accelerometer	gyroscope	rpm	throttle	latitude	longitude	speedcar	speedgps
2022-04-21 21:27:47	GPS OK	-0.144;0.046;1.067	0.58;3.237;-0.13	946.0	0.0	59.39755	24.8112325	26.0	26.9
2022-04-21 21:27:46	GPS OK	-0.135;0.17;0.997	1.496;0.489;0.214	1181.0	0.0	59.3974827	24.8111927	30.0	31.3
2022-04-21 21:27:45	GPS OK	-0.186;0.018;0.973	-0.359;1.748;0.473	1384.0	0.0	59.3974042	24.8111477	35.0	36.2
2022-04-21 21:27:44	GPS OK	-0.208;0.02;0.986	-1.893;4.298;-2.786	1541.5	0.0	59.3973145	24.8111013	39.0	39.6
2022-04-21 21:27:43	GPS OK	-0.034;-0.073;0.982	-0.611;3.573;-6.122	1587.5	10.6	59.3972177	24.8110628	41.0	40.0
2022-04-21 21:27:43	GPS OK	0.036;-0.145;0.86	-0.954;3.053;-10.611	1592.5	11.0	59.3971183	24.8110463	41.0	40.4
2022-04-21 21:27:41	GPS OK	0.007;-0.126;0.943	1.26;0.435;-7.183	1603.0	10.2	59.3970183	24.811065	41.0	40.9
2022-04-21 21:27:40	GPS OK	0.0;0.001;0.861	1.771;1.405;-7.817	1639.5	2.7	59.3969188	24.8111122	42.0	41.9
2022-04-21 21:27:39	GPS OK	-0.044;-0.07;0.916	-1.779;3.229;-1.687	1700.5	0.0	59.3968192	24.8111795	43.0	43.3

Items per page: 10 1 – 10 of 200

Figure 12. ThingsBoard Cloud dashboard where receiver device data and location can be seen

3.2.3 Data analyzing

Data for data analysis is manually exported from the ThingsBoard Cloud environment as a .csv file. This data then can be modified and analyzed with various programs and languages. In this work, the data analysis is done in Microsoft Visual Studio Code in a Jupyter Notebook environment. Data preparation, analyzing, and machine learning were done in Python3 language using the following modules:

- pandas
- overpy
- scikit-learn

3.3 Developed device

The built prototype consists of multiple components. The Raspberry Pi, modem, and the sensor board with accelerometer and gyroscope are connected using jumper cables. The modem could be connected straight on top of Raspberry without cables, but due to the need to connect the sensor module, the GPIO pins need to be accessible. In Figure 14 all the needed connections are shown in the wiring diagram. In Figure 13 the prototype device with all the modules is shown.

The Raspberry Pi and SIM7070G modem are connected using 5 jumper cables. 5V and GND for power supply, RX and TX for communications through UART. The GPIO 4 pin is needed to wake up and shut down the modem through the python script.

Raspberry Pi and Accelerometer & gyroscope sensor module are connected with 4 jumper cables. 3V and GND for power supply, SCL, and SDA for I2C communications to get data from the sensors. In the vehicle, the sensor was attached firmly to the car dashboard facing the correct direction.

Additionally, there is a connected USB cable to the Raspberry Pi from the car to power all the devices and a USB Hub to have connections to additional devices for additional data. There are GNSS module and OBD-II reader connected to the USB Hub.

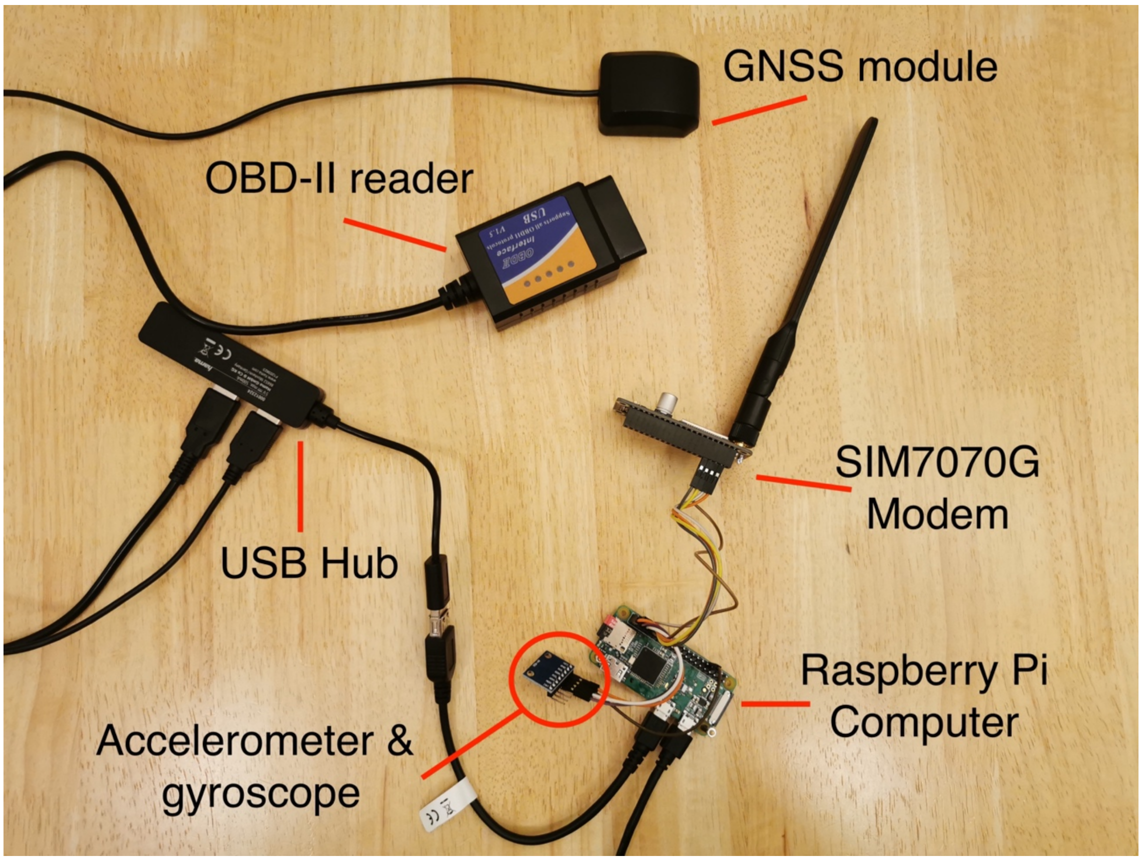


Figure 13. Prototype device with all its modules

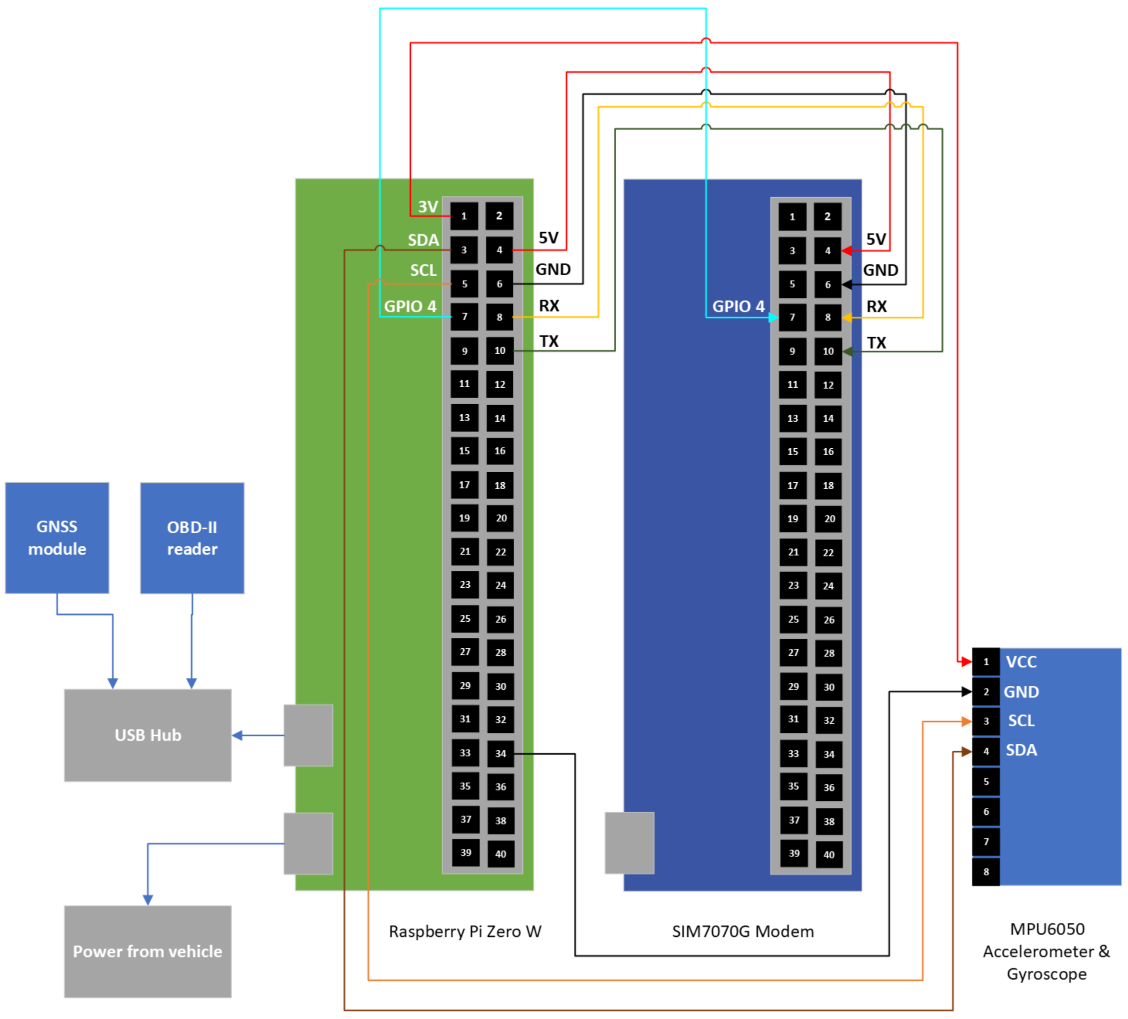


Figure 14. Wiring diagram of full device

4 Experiments

In this chapter, the final two experiments using the developed prototype are described. In the first experiment, driving behavior is classified with a time-based bad driving dataset and in the second experiment, the driving behavior is classified with an event-based bad driving dataset. In the third chapter, the communication between server and device is also investigated with packet loss rates obtained from different tests.

4.1 Experiment 1 – Driving behavior classification with time-based bad driving

In this experiment, machine learning was used to classify bad driving in a dataset. The goal was to evaluate how machine learning will perform. For this experiment, various sensors and data were used which can be seen in Table 5.

Table 5. Experiment 1 sensors and data

Sensors	**Data**
Accelerometer	Acceleration on X, Y and Z-axis
Gyroscope	Angular velocity on X, Y and Z-axis
GPS module	Vehicle latitude and longitude Vehicle speed
Engine speed sensor	Engine speed
Vehicle speed sensor	Vehicle speed
Accelerator pedal position sensor	Position of acceleration pedal

4.1.1 Data collection

Data for this experiment was collected during driving which lasted around 30 minutes. In this driving, both urban and rural roads were used. During the driving, various speeds were achieved. In Figure 15 the track of the experiment is shown. The experiment started on rural roads and ended in an urban environment. In the end part, a lap was chosen which was driven both normally and then aggressively. The car speed data was afterward altered to add more speed to the aggressive driving part. The lap section of the track is zoomed in in Figure 16. From this figure, it is also possible to see that during one lap a section of data is missing from the top left part of the lap. This indicates that a packet was lost at that moment.

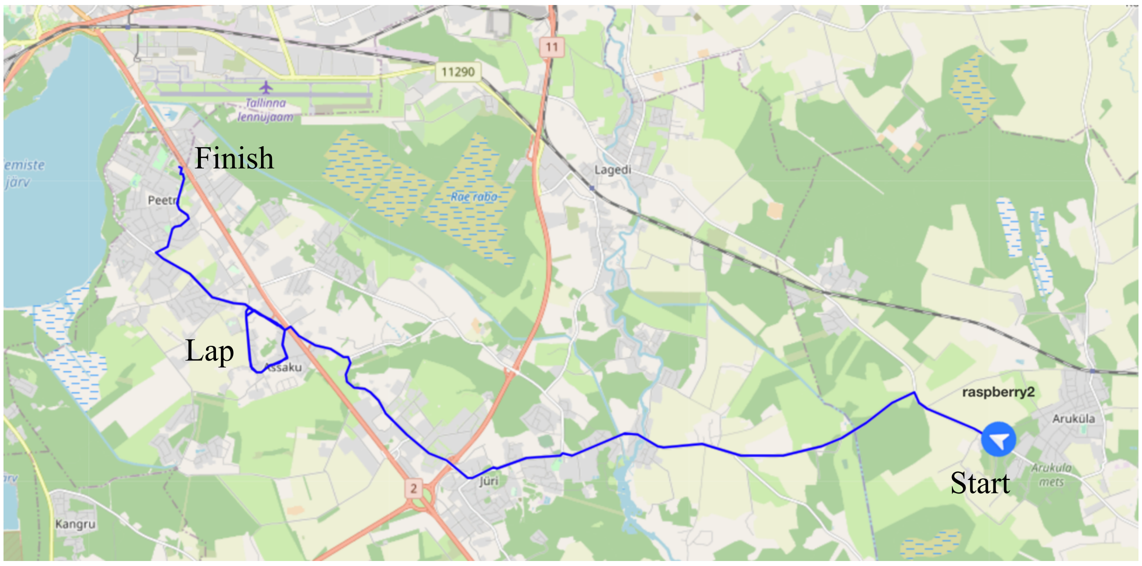


Figure 15. Experiment 1 track, 30 minutes total with lap section which was driven twice

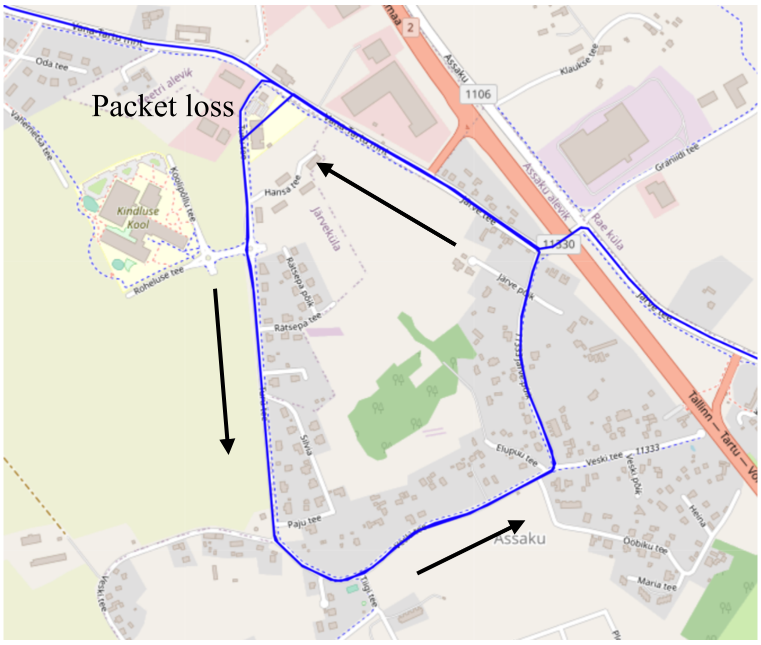
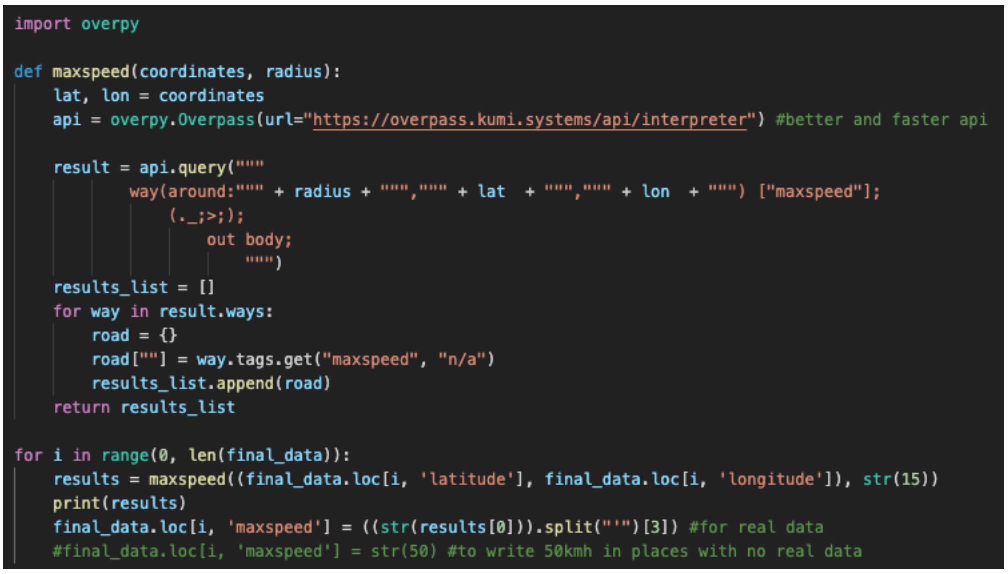


Figure 16. Zoomed lap section of the track

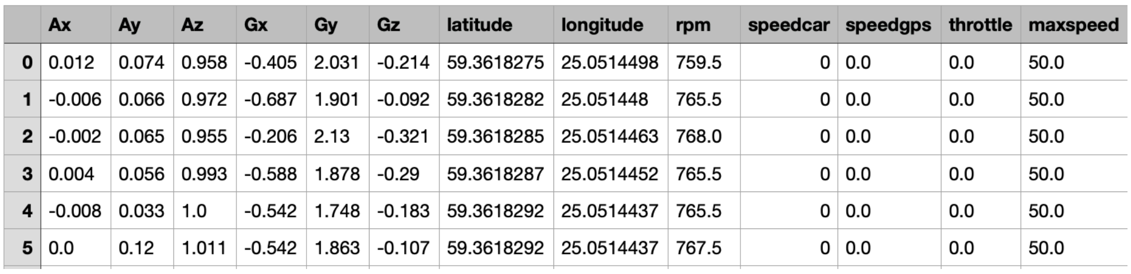
To analyze driver better, additional data such as the speed limit of any location was needed. Adding this should help improve the classification model and make car speed data more important. For this, a script snapshot is shown in Figure 17 was used to get the speed limit of a location based on the latitude and longitude from OpenStreetMap Overpass Application Programming Interface (API). The script base is from user jacohend from his Github and is modified for experiment needs [43].



```python
import overpy

def maxspeed(coordinates, radius):
    lat, lon = coordinates
    api = overpy.Overpass(url="https://overpass.kumi.systems/api/interpreter") #better and faster api

    result = api.query("""
            way(around:""" + radius + """,""" + lat  + """,""" + lon  + """) ["maxspeed"];
                (._;>;);
                    out body;
                        """)
    results_list = []
    for way in result.ways:
        road = {}
        road[""] = way.tags.get("maxspeed", "n/a")
        results_list.append(road)
    return results_list

for i in range(0, len(final_data)):
    results = maxspeed((final_data.loc[i, 'latitude'], final_data.loc[i, 'longitude']), str(15))
    print(results)
    final_data.loc[i, 'maxspeed'] = ((str(results[0])).split("'")[3]) #for real data
    #final_data.loc[i, 'maxspeed'] = str(50) #to write 50kmh in places with no real data
```

Figure 17. Overpass API query script for adding speed limit data from device location

In Figure 18 all the acquired data from the device is shown including the additional "maxspeed" column which shows the speed limit of the location.



	Ax	Ay	Az	Gx	Gy	Gz	latitude	longitude	rpm	speedcar	speedgps	throttle	maxspeed
0	0.012	0.074	0.958	-0.405	2.031	-0.214	59.3618275	25.0514498	759.5	0	0.0	0.0	50.0
1	-0.006	0.066	0.972	-0.687	1.901	-0.092	59.3618282	25.051448	765.5	0	0.0	0.0	50.0
2	-0.002	0.065	0.955	-0.206	2.13	-0.321	59.3618285	25.0514463	768.0	0	0.0	0.0	50.0
3	0.004	0.056	0.993	-0.588	1.878	-0.29	59.3618287	25.0514452	765.5	0	0.0	0.0	50.0
4	-0.008	0.033	1.0	-0.542	1.748	-0.183	59.3618292	25.0514437	765.5	0	0.0	0.0	50.0
5	0.0	0.12	1.011	-0.542	1.863	-0.107	59.3618292	25.0514437	767.5	0	0.0	0.0	50.0

Figure 18. Snapshot of Experiment 1 dataset

4.1.2 Data processing

For machine learning a column called "behavior" was added having binary values as 1 and 0. 1 for good driving and 0 for bad driving. In Experiment 1 the behavior was added manually. The whole lap which occurred during the bad driving phase was classified as bad driving.

First, the feature importance was analyzed to see which features are the most relevant to the output variable. In Figure 19 we can see that the biggest score is on the vehicle speed and maximum speed limit of location. Gyroscope values and accelerometer Z-axis have the least importance.

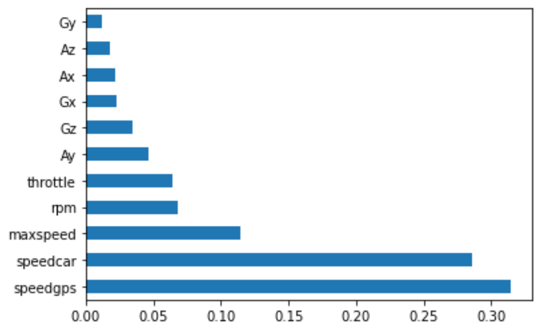


Figure 19. Most important features in data

Various data are plotted against each other to show the differences in the two driven laps and the correlation to driving behavior. First, the gyroscope values for all three axes are plotted. From Figure 20 we can see that there are not many differences in the values of the gyroscope axis between the two laps. There can be seen a slight difference in the gyroscope y-axis (Gy) values.

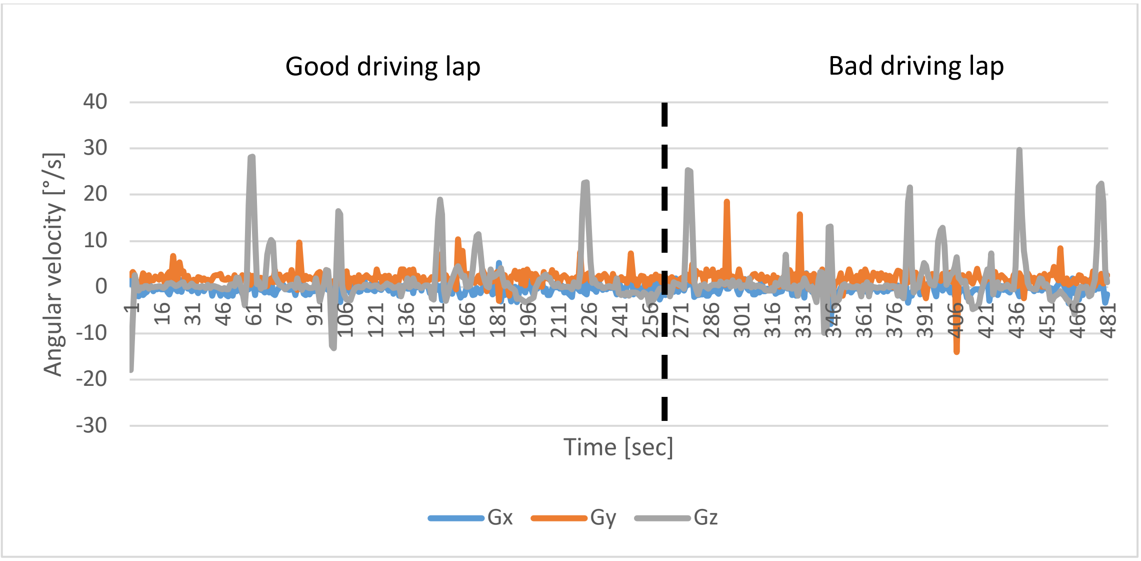


Figure 20. Gyroscope correlation to driving behavior

In Figure 21 the car speed is shown together with the location speed limit. It is seen that during a bad driving lap the driver crossed the speed limit threshold more times than during a good driving lap.

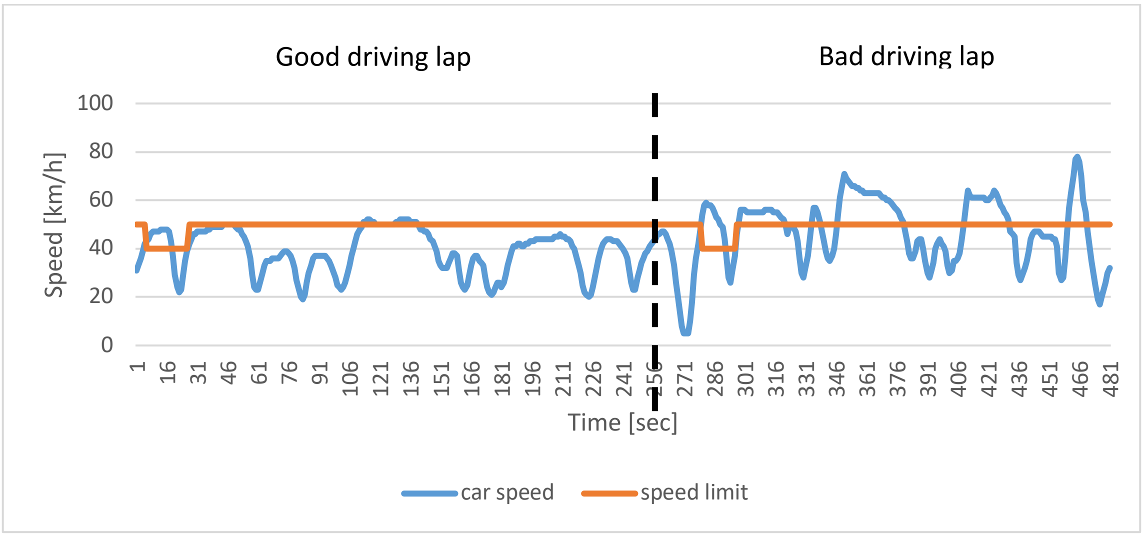


Figure 21. Car speed and speed limit correlation to driving behavior

In Figure 22 it is also clearly seen that the throttle pedal position is different from the normally driven lap. Engine speed is also having its peaks significantly higher. These both are clear indicators of bad driving. Both of their values are in correlation to each other too as the higher the throttle position value is, the higher engine speed will be.

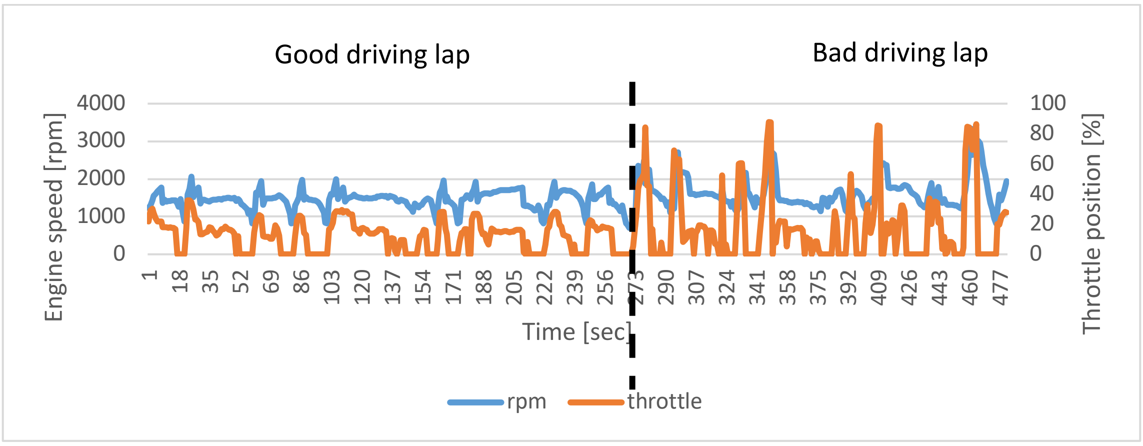


Figure 22. Rpm and throttle pedal position correlation to driving behavior

From Figure 23 it is seen that the accelerometer has slightly higher peaks during the bad lap but not significantly. From Ax data, the sudden accelerations and decelerations can be seen and in Ay the hard cornering.

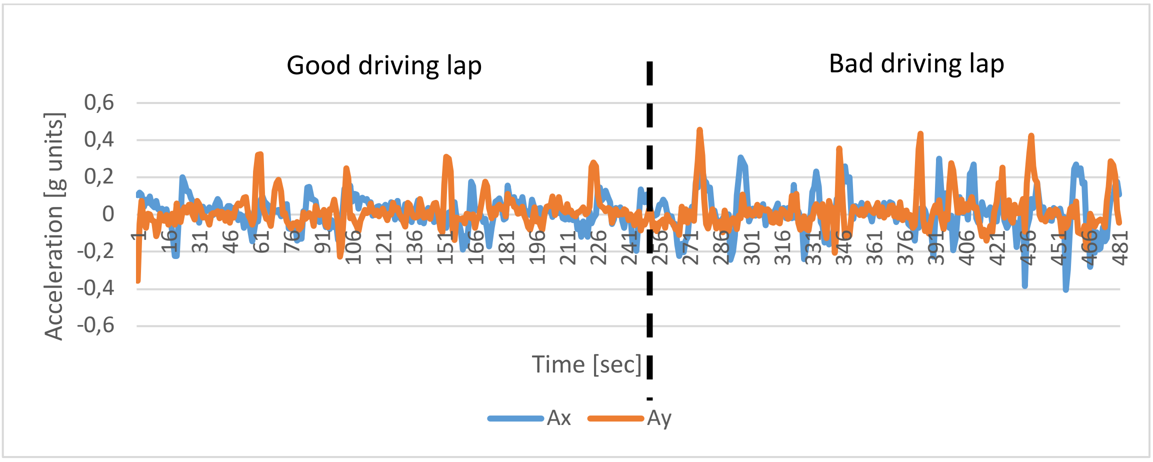


Figure 23. Accelerometer X-Axis and Y-Axis correlation to driving behavior

4.1.3 Modeling results

Data was split as 75% for training and 25% for testing. In total, 1966 data points were available. An SVM classification model was created with the training data and then the confusion matrix was plotted with test data. In Figure 24 the confusion matrix can be seen. Out of the 441 data points where good driving behavior was, all 441 (100%) were correctly classified. But in bad behavior data points out of 51 only 10 were correctly classified as bad behavior which makes accuracy 19,6%. The overall accuracy of the model is 92%.

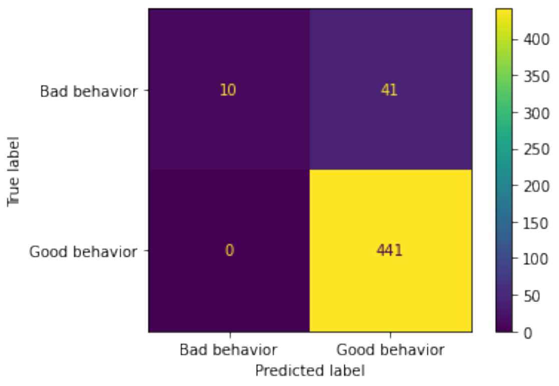


Figure 24. Confusion matrix for Experiment 1 model

After the first matrix, the model parameters were optimized by choosing C=10 and gamma=0.1. In Figure 25 the optimized confusion matrix shows that 13 additional data points were correctly classified as bad behavior (45%) and 7 more incorrectly classified as good behavior (98,4%). The overall accuracy of the model gained from 92% to 93%.

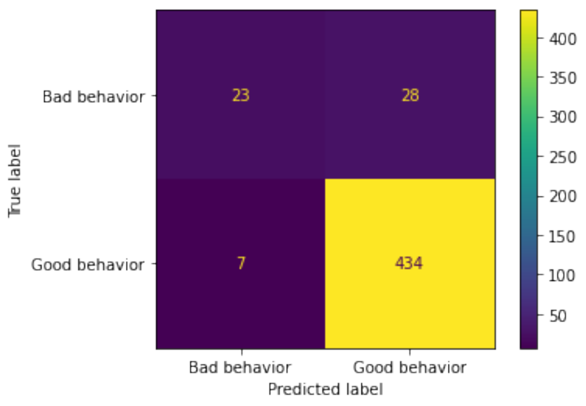


Figure 25. Optimized confusion matrix of Experiment 1

4.1.4 Conclusion of Experiment 1

In conclusion, the good driving behavior classification was excellent at 98% but the bad driving classification was poor at 45%. The overall accuracy of the model was 93%. To improve the accuracy of future experiments, a new method of classification should be proposed. In this experiment, the driver was manually classified as a bad driver during the full bad driving lap, but in a real scenario, the driver cannot cross all the bad driving thresholds 100% of the time.

4.2 Experiment 2 – Driving behavior classification with event-based bad driving

This experiment aimed to build an event-based driver behavior classification model that could classify driver behavior in both urban and rural environments. Same sensors and data types were used as in Experiment 1.

4.2.1 Data collection

Data for this experiment were collected on three different trips. The first was collected in urban, the second was collected in rural, and the third was collected in an environment consisting of both. The first two trips were going to be used to teach the model and the third for testing it.

In Figure 26 the urban track can be seen. The route started and finished from Taltech campus parking lot. The track consisted of two laps, each 6.5km long that took to various places near the campus The first lap was good driving and the second bad driving.

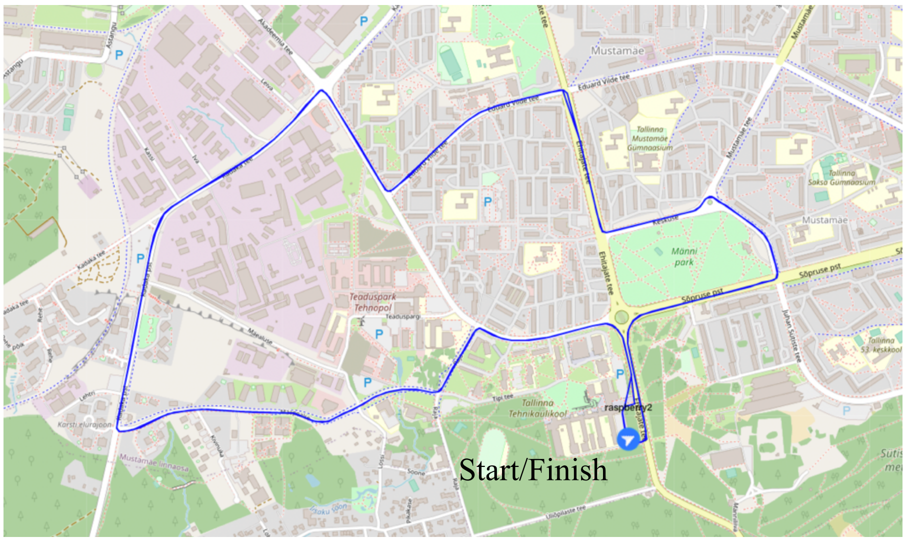


Figure 26. Experiment 2 urban track

In Figure 27 the rural track can be seen in which a highway was driven. The location of the road was from Nõmme to Saku. One direction was driven with good driving and the other direction was with bad driving. As there does not happen much on highway roads other than speeding, accelerating, and braking then highway driving was not done twice. The length of one direction is approximately 8km.

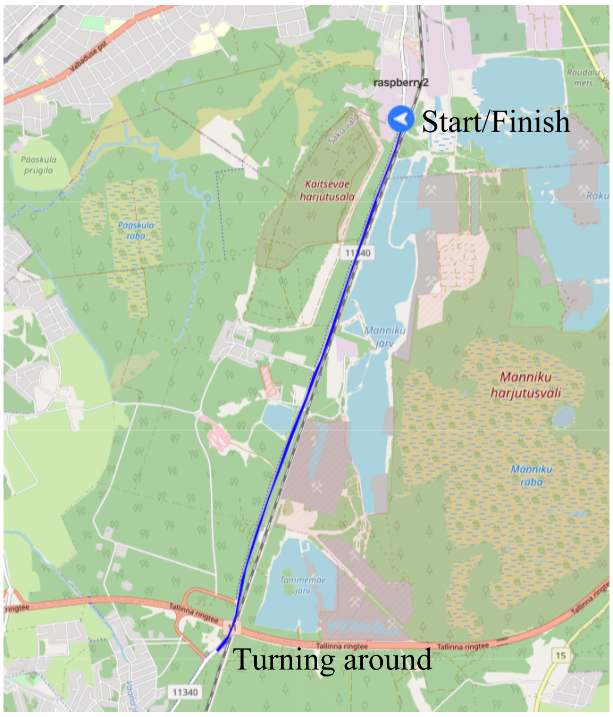


Figure 27. Experiment 2 rural track

The final test track that took place from Luige to Peetri can be seen in Figure 28. This track started with good rural driving, then changed to bad rural driving after which the urban part started. The urban part started and ended with good driving with bad urban driving in between. The length of the test track was 12km.

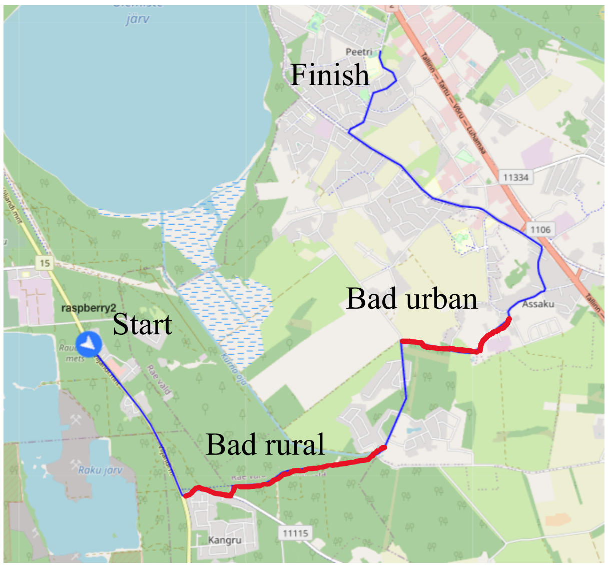


Figure 28. Experiment 2 testing track consisting of both urban and rural track

4.2.2 Data processing

In this experiment instead of classifying whole sections as bad driving the classifications were done instead when any thresholds were crossed. In Figure 29 can be seen that even though most of the bad behavior happens in the clearly marked second part of driving it is not filled completely with bad behavior events. This happens mostly because of stopping behind junctions and traffic lights. Also, a few events could happen with a good driver also as seen by the spikes in the left part of Figure 29.

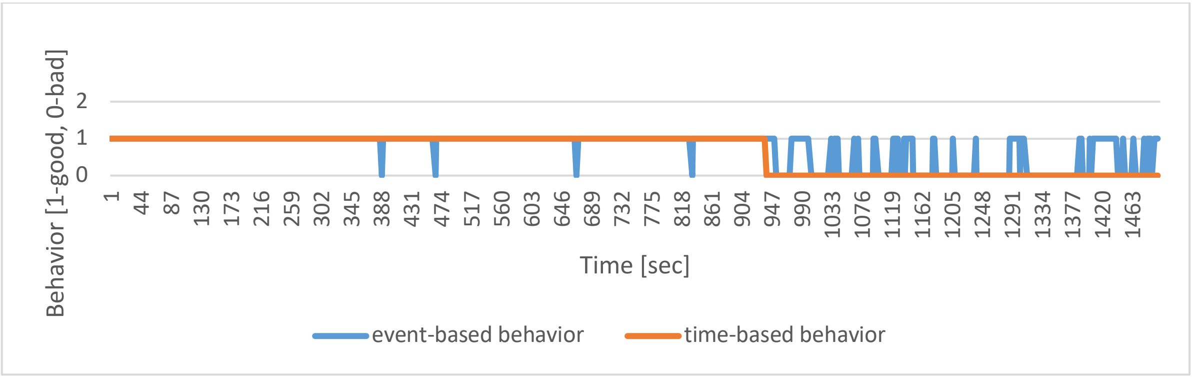


Figure 29. Differences when classifying behavior as time-based or event-based

In the beginning, thresholds found in different papers [18] [24] were investigated and calculated. There was no scientifically proven threshold that all papers would use. Each paper used its own rules and vision. The thresholds from related work did not fit our data. This can be due to differences in sensors but is also depending on the car type and car power. Because of that, the thresholds were chosen by analyzing the experiment data. Thresholds were chosen based on Experiment 1 plots where a good driving lap was compared with a bad driving lap. Chosen thresholds are shown in Table 6.

Table 6. Thresholds for driver classification

	Bad	**Good**
Accelerometer X-axis	> 0.25 < -0.25	< 0.25 > -0.25
Accelerometer Y-axis	> 0.4 < -0.4	< 0.4 > 0.4
Throttle	> 50	< 50
Over speeding	> speed limit + 5	< speed limit +5
RPM	> 3000	< 3000

4.2.3 Modeling results

First, urban track SVM classification models were trained and tested (70% training, 30% testing). Both event-based model and time-based model were created to compare the differences. In Figure 30 can be seen both confusion matrixes and in Table 7 both model metrics. From this can be seen that the event-based model accuracy is 10% higher than time-based, being at 99%. From the confusion matrixes it is possible to see that the time-based model makes mistakes on both good driving and bad driving classification whereas event-based makes only few wrong classifications.

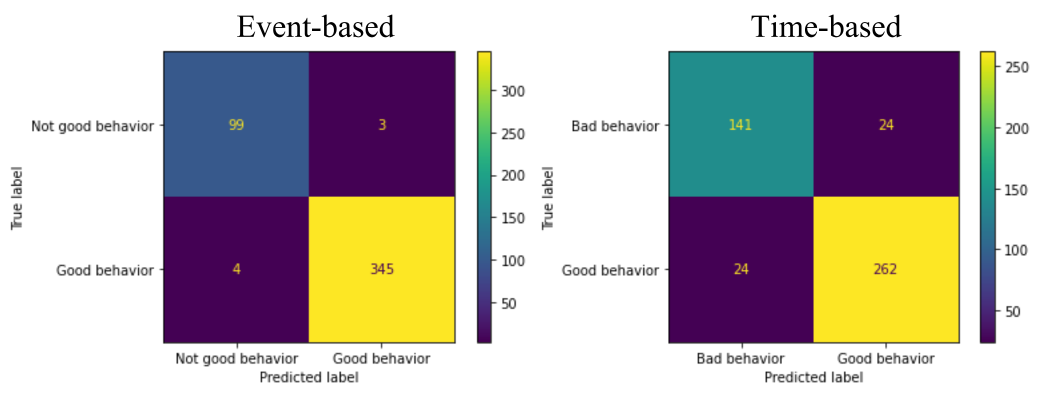


Figure 30. Event-based confusion matrix and Time-based confusion matrix in urban track model

Table 7. Experiment 2 urban track model metrics

	Event-based precision	**Event-based recall**	**Event-based f1-score**	**Time-based precision**	**Time-based recall**	**Time-based f1-score**
Bad behavior	0.97	0.97	0.97	0.85	0.85	0.85
Good behavior	0.99	0.99	0.99	0.92	0.92	0.92
Accuracy	0.99			0.89		

Second, the rural track SVM classification model was created and this time only with event-based classification. From the Figure 31 confusion matrix, we can see that the model made only 1 wrong classification total. From Table 8 the overall accuracy can be seen as very high at 99%.

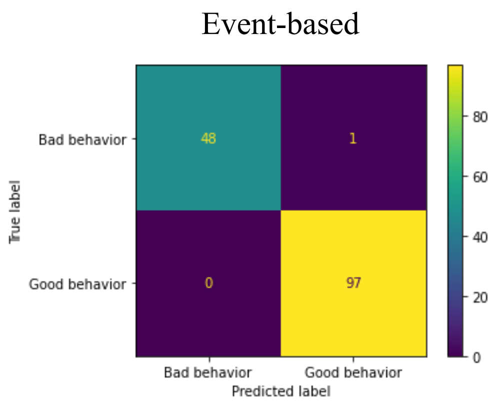


Figure 31. Event-based confusion matrix in rural model

Table 8. Experiment 2 rural track model metrics

	Event-based precision	**Event-based recall**	**Event-based f1-score**
Bad behavior	1.00	0.98	0.99
Good behavior	0.99	1.00	0.99
Accuracy	0.99		

Third, a combined SVM classification model was made with event-based classification. This dataset was made by combining the previous urban track and rural track datasets. This meant that in this model both are being used to train and test which makes it harder for the model. From the confusion matrix in Figure 32, it is seen that there were only 8 misclassifications. The total accuracy of the model shown in Table 9 was 99%.

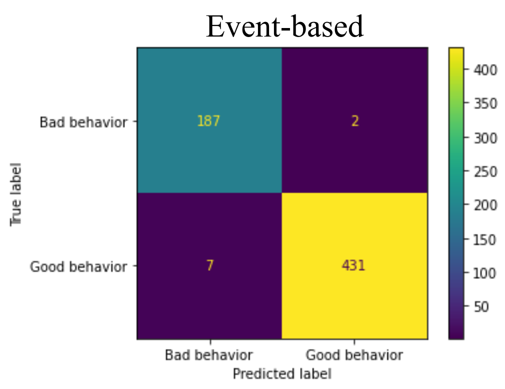


Figure 32. Event-based confusion matrix in combined model

Table 9. Experiment 2 combined model metrics

	Precision	**Recall**	**F1-score**
Bad behavior	0.96	0.99	0.98
Good behavior	1.00	0.98	0.99
Accuracy	0.99		

Finally, a test run track data was used on all previous 3 models. This meant that this track data was not used to train the model and the models see data from this track for the first time. This lets us see how these different models can handle classifying data taken from a different locations on different roads but with similar behavior.

First, the combined model was used. This model has seen both urban driving and rural driving in both bad and good behaviors. From Figure 33 we can see that the model made a total of 24 classifications wrong out of the total 218 test data points. Overall accuracy is 89% as seen in Table 10.

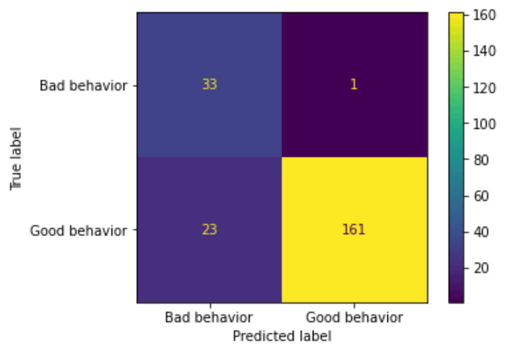


Figure 33. Test drive data confusion matrix in combined model

Table 10. Experiment 2 test drive combined model metrics

	Event-based precision	**Event-based recall**	**Event-based f1-score**
Bad behavior	0.59	0.97	0.73
Good behavior	0.99	0.88	0.93
Accuracy	0.89		

For additional comparison, the same test drive data was used in the urban model and rural model (Figure 34). As expected the models independently without knowing both road types can not classify bad driving behavior well. The urban model had an accuracy of 80% and the rural model had an accuracy of 82% as seen in Table 11.

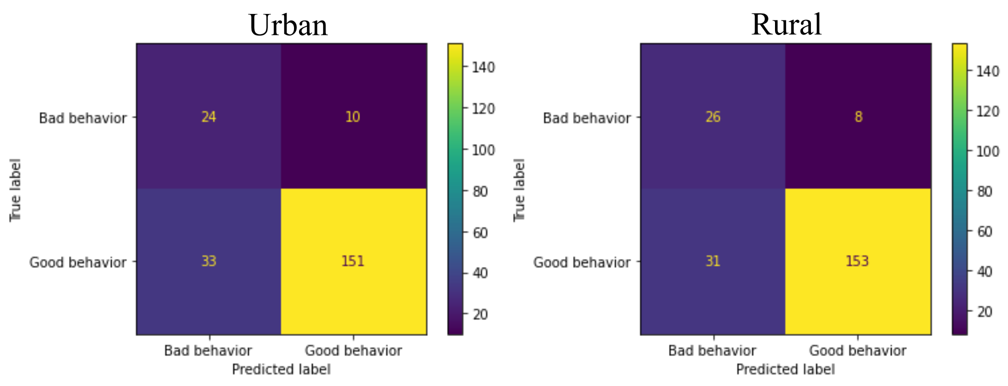


Figure 34. Test drive data confusion matrix in urban model and rural model

Table 11. Experiment 2 test drive urban and rural metrics

	Urban model precision	**Urban model recall**	**Urban model score**	**Rural model precision**	**Rural model recall**	**Rural model f1-score**
Bad behavior	0.42	0.71	0.53	0.46	0.76	0.57
Good behavior	0.94	0.82	0.88	0.95	0.83	0.89
Accuracy	0.80			0.82		

4.2.4 Conclusion of Experiment 2

In the Experiment 2 the driver behavior was classified differently than in experiment 1. We moved from time-based classification to event-based classification as the driver was not breaking thresholds 100% of the time of the bad driving lap. In an urban environment, two models were created just to compare this situation. From that comparison, it was shown that event-based classification is 10 percentage points more accurate than time-based.

Then in the second part of Experiment 2, different models were tried on test data. From this, we can see that the model accuracy drops significantly if the model is used on totally new data from a different location. When the model is trained on data taken from the same location, then the accuracy tends to be near 99% but in the case of a new location, it was around 89%.

4.3 Packet loss

Additionally, to test the quality of Cat-M1 communication and the prototype device the Packet Loss Rate can be calculated. The calculation is done with the following equation [44]:

$$Packet\ Loss\ Rate = \frac{sent\ packets - received\ packets}{sent\ packets} * 100\%$$

In Table 12 the packet loss of all tracks from experiment 2 can be seen. The packet loss is highest in the urban environment, which can be explained due to the high buildings and multiple handovers of cells. The rural drive had significantly better results and the test drive which was conducted last had only 1 lost packet. On that route, there were not any high buildings or obstructions. The sections where packets were lost can also be seen from the track plots shown in Figure 35. From this figure, we can see how there are sections where the navigation line is jumping from one place to another.

Table 12. Packet loss metrics

	Urban	**Rural**	**Test drive**
Sent packets	1572	590	871
Received packets	1503	584	870
Lost packets	69	6	1
Packet loss	4.38%	1.02%	0.1%

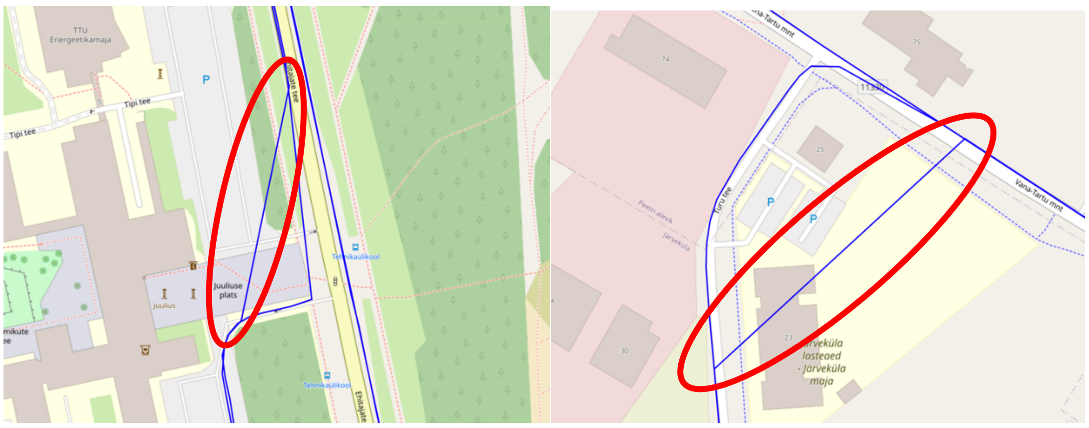


Figure 35. Plot sections where packets were lost

5 Summary

The main purpose of this thesis was to develop a driver analysis system that would classify driver behavior and work with cellular IoT communication technology. First in chapter 1 the problem and background were described. In chapter 2 the state of the art of driver behavior classification with machine learning along with machine learning metrics and related works on this topic were given.

From the state of the art, it was found that driver behavior analysis and classification can be done in multiple ways. Furthermore, the ways for testing and developing in the related works were different. The many machine learning algorithms and thresholds were not common between the related works. There were various devices used and sometimes a lot of information about used machine learning models or devices were hidden. The number of classes and rules of them was also different among the works.

Based on state of the art it was decided to build a simple device that could read data from a vehicle OBD-II reader and add additional sensor data to it. For machine learning, the most commonly seen algorithm SVM was used. ML model Output was divided to two classes, either bad driving or good driving. Thresholds were decided by investigating differences in values of good and bad driving. Finally, the models were trained and tested to see how well machine learning can classify driving behavior.

A prototype device was proposed with 6 modules that fulfill the two requirements of gathering all needed data and sending it to the cloud server for further analysis. Modules consisted of Raspberry Pi Zero W computer, SIM7070G modem, G-mouse GNSS module, ELM327 OBD-II reader, GY-521 accelerometer/gyroscope, and Hama USB hub. These devices were connected with USB cables and jumper cables. The device was installed inside the vehicle so that the sensor module was firmly attached facing the correct axis.

In the experiments, machine learning and behavior classification were tested. At this point, the classifications were investigated from two perspectives. First was time-based behavior classification where the driver was labeled as a bad driver for the whole duration of the bad driving section. Since a driver cannot be driving badly 100% of the time, another way had to be tested. Therefore, in the second experiment, the driver was labeled as a bad driver only when certain thresholds were exceeded, which made it an event-based classification. From the results of these experiments, the event-based model was 10 percentage points more accurate at 99% compared with the time-based which was at around 89%.

The only issues on the hardware side were connected with the SIM7070G modem. During the tests, the LTE NB-IoT was used, but because the modem did not support mobility by cell HandOver, it was changed to Cat-M1. LTE Cat-M1 is a cellular IoT communication technology that suits this type of device very well.

Beside this issue, the modem also does not support GNSS usage at the same time as the modem is using communication. GNSS issue was resolved by adding an additional GNSS USB module. Other modules and sensors did not have any issues.

The packet loss rate of all the experiments was also investigated. LTE Cat-M1 suffered the most packet loss in an urban environment with results of around 4.4%. When driving on the rural roads, the loss rate was only about 1% whereas, during a final test drive which consisted of both rural driving and urban driving, the packet loss rate was only 0.1%.

Although the machine learning models were tested on only one car type and one driver, based on these results the developed device and model could be used by car insurance companies or rental companies to classify the proportion of bad driving behaviors during a user's whole driving session.

The software developed in this thesis for the device is located on the following Google Drive URL https://drive.google.com/drive/folders/1FMfWqhpAoYKP11DHd8W1R54XGZr2HALp?usp=sharing together with all the datasets to allow future work.

During the experiments and writing of the thesis, additional ideas and questions were raised, which will remain for future work. These can be divided into short-term and long-term categories.

Short-term:

- A self-working data collection device – At this moment the Software on the device needs manual starting with the help of a laptop, but this could be automated.

- Scoring of the driver based on behavior – Driver behavior could be graded to give feedback to the model users or the driver themself on how good or bad they are driving.
- Data analysis automation – At the moment the data was analyzed manually by downloading .csv files from the server and then preparing the data and training the model, but this could be also be a fully automatic process and therefore less time-consuming.
- Other machine learning algorithms – During this thesis, only one machine learning algorithm was used, there could be alternate algorithms that could prove to be better for this dataset.
- Testing with more drivers and different vehicles – In this thesis only one driver with one car was used, but with more drivers and vehicles, the results could be more reliably generalized for larger populations.

Long-term:

- Facilities for real testing – At the moment the bad driving was not done completely as it can be in real scenarios. For example, the speed limit exceeding had to be done by altering the dataset.
- Testing other methods of classifying – Classification could be done with more classes and with other ways to divide them.

References

[1] N. Peppes, T. Alexakis, E. Adamopoulou and K. Demestichas, "Driving Behaviour Analysis Using Machine and Deep Learning Methods for Continuous Streams of Vehicular Data," *Sensors,* vol. 21, no. 14, p. 4704, 2021.

[2] Gurtam, "Understanding Cellular IoT Connectivity in GPS Tracking," [Online]. Available: https://gurtam.com/en/gps-hardware/news/understanding-cellular-iot-connectivity-in-gps-tracking. [Accessed 11 10 2021].

[3] S. Chalapati, "Comparison Of LPWA Technologies And Realizable Use Cases," [Online]. Available: https://www.nctatechnicalpapers.com/Paper/2018/2018-comparison-of-lpwa-technologies-and-realizable-use-cases/download. [Accessed 13 10 2021].

[4] Ublox, "What 3GPP Release 14 means for NB-IoT and LTE-M," [Online]. Available: https://www.u-blox.com/en/blogs/insights/what-3gpp-release-14-means-nb-iot-and-lte-m. [Accessed 19 03 2022].

[5] Kvaser, "Introduction to the OBD II Standard," [Online]. Available: https://www.kvaser.com/about-can/can-standards/introduction-to-obd-ii/. [Accessed 01 05 2022].

[6] P. Els, "How electric vehicles are rewriting the rules of diagnostics," 19 September 2019. [Online]. Available: https://www.automotive-iq.com/electrics-electronics/articles/how-electric-vehicles-are-rewriting-the-rules-of-diagnostics. [Accessed 04 05 2022].

[7] N. Guenther and M. Schonlau, "Support vector machines," *The Stata Journal,* vol. 16, no. 4, pp. 917-937, 2016.

[8] A. M. Kumar, "C and Gamma in SVM," 17 December 2018. [Online]. Available: https://medium.com/@myselfaman12345/ c-and-gamma-in-svm-e6cee48626be. [Accessed 07 05 2022].

[9] "javaTpoint," [Online]. Available: https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning. [Accessed 2 05 2022].

[10] IBM, "IBM," 7 December 2020. [Online]. Available: https://www.ibm.com/cloud/learn/random-forest. [Accessed 1 05 2022].

[11] A. Mishra, "Towards Data Science," 24 February 2018. [Online]. Available: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234. [Accessed 04 05 2022].

[12] A. Ragan, "Towards Data Science," 10 October 2018. [Online]. Available: https://towardsdatascience.com/taking-the-confusion-out-of-confusion-matrices-c1ce054b3d3e. [Accessed 03 05 2022].

[13] D. Matter, "About Digital Matter," [Online]. Available: https://www.digitalmatter.com/about-us/. [Accessed 24 03 2022].

[14] MyOrien, "MyOrien Features," [Online]. Available: https://myorien.com/features. [Accessed 24 03 2022].

[15] Z. Marafie, K.-J. Lin, D. Wang, H. Lyu, Y. Liu, Y. Meng and J. Ma, "AutoCoach: An Intelligent Driver Behavior Feedback Agent with Personality-Based Driver Models," *Electronics,* vol. 10, no. 11, p. 1361, 2021.

[16] L. Zhang, B. Tan, T. Liu and J. Li, "Research on recognition of dangerous driving behavior based on support vector machine," *Twelfth International Conference on Graphics and Image Processing (ICGIP 2020),* 2021.

[17] S. Navneeth, K. P. Prithvil, N. R. Sri Hari, R. Thushar and M. Rajeswari, "On-Board Diagnostics and Driver Profiling," *2020 5th International Conference on Computing, Communication and Security (ICCCS),* pp. 1-6, 2020.

[18] S.-H. Chen, J.-S. Pan and K. Lu, "Driving Behavior Analysis Based on Vehicle OBD Information and AdaBoost Algorithms," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Hong Kong, 2015.

[19] M. Karrouchi, I. Nasri, H. Snoussi, I. Atmane, A. Messaoudi and K. Kassmi, "Black box system for car/driver monitoring to decrease the reasons of road crashes," *2021 4th International Symposium on Advanced Electrical and Communication Technologies (ISAECT),* pp. 01-06, 2021.

[20] C.-C. Wu, "The development and implementation of driving behavior analysis system," *2021 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS),* 2021.

[21] T. Dadhich and S. Gupta, "Detecting Aggressive Driving Behavior Using Spectral Kurtosis and MEMS Accelerometers," in *In Proceedings of the Second International Conference on Information Management and Machine Intelligence*, Springer, Singapore, 2021.

[22] A. Kashevnik, I. Lashkov and A. Gurtov, "Methodology and Mobile Application for Driver Behavior Analysis and Accident Prevention," *IEEE Transactions on Intelligent Transportation Systems,* vol. 21, no. 6, pp. 2427-2436, 2020.

[23] A. BinMasoud and Q. Cheng, "Design of an IoT-based Vehicle State Monitoring System Using Raspberry Pi," *2019 International Conference on Electrical Engineering Research & Practice (ICEERP),* pp. 1-6, 2019.

[24] D. Sivaraj, S. H. Kumar, D. V. Sai Jogarao, S. Dutta, K. Ezhumalai and E. Rabjor, "Analysis of Driving Behaviour for Improved Safety in Commercial Fleet Management using Onboard Diagnostics (OBD-II)," *2021 Smart Technologies, Communication and Robotics (STCR),* pp. 1-5, 2021.

[25] A. Aksjonov, P. Nedoma, V. Vodovozov, E. Petlenkov and M. Herrmann, "A Novel Driver Performance Model Based on Machine Learning," *IFAC-PapersOnLine,* vol. 51, no. 9, pp. 267-272, 2018.

[26] Z. Li, H. Wang, Y. Zhang and X. Zhao, "Random forest–based feature selection and detection method for drunk driving recognition," *International Journal of Distributed Sensor Networks,* vol. 16, no. 2, 2020.

[27] U. Fugiglando, E. Massaro, P. Santi, S. Milardo, K. Abida, R. Stahlmann, F. Netter and C. Ratti, "Driving Behavior Analysis through CAN Bus Data in an Uncontrolled Environment," *IEEE Transactions on Intelligent Transportation Systems,* vol. 20, no. 2, pp. 737-748, 2019.

[28] A. S. Yuksel and S. Atmaca, "Driver's black box: a system for driver risk assessment using machine learning and fuzzy logic," *Journal of Intelligent Transportation Systems,* vol. 25, no. 5, pp. 482-500, 2021.

[29] A. EI Basiouni EI Masri, H. Artail and H. Akkary, "Toward self-policing: Detecting drunk driving behaviors through sampling CAN bus data," *2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA),* pp. 1-5, 2017.

[30] L. Pounder, "Raspberry Pi Zero Guide: Projects, Specs, GPIO, Getting Started," [Online]. Available: https://www.tomshardware.com/features/raspberry-pi-zero. [Accessed 20 03 2022].

[31] peppe8o, "Raspberry PI Zero Pinout," [Online]. Available: https://peppe8o.com/raspberry-pi-zero-pinout/. [Accessed 17 03 2022].

[32] Waveshare, "SIM7070G NB-IoT/Cat-M/GPRS/GNSS HAT for Raspberry Pi, global band support," [Online]. Available: https://www.waveshare.com/sim7070g-cat-m-nb-iot-gprs-hat.htm. [Accessed 20 03 2022].

[33] SIMCom, "SIM7080 Series_AT Command Manual_V1.02," 26 February 2020. [Online]. Available: https://www.waveshare.com/w/upload/3/39/SIM7080_Series_AT_Command_Manual_V1.02.pdf. [Accessed 04 05 2022].

[34] u-blox, "u-blox 7," [Online]. Available: https://www.u-blox.com/sites/default/files/products/documents/u-blox7-V14_ReceiverDescriptionProtocolSpec_%28GPS.G7-SW-12001%29_Public.pdf. [Accessed 16 03 2022].

[35] "GPS - NMEA sentence information," [Online]. Available: http://aprs.gids.nl/nmea/. [Accessed 4 05 2022].

[36] "python-OBD," [Online]. Available: https://python-obd.readthedocs.io/en/latest. [Accessed 14 03 2022].

[37] COMPONENTS101, "MPU6050 Accelerometer and Gyroscope Module," [Online]. Available: https://components101.com/sensors/mpu6050-module. [Accessed 20 03 2022].

[38] ElectronicWings, "MPU6050 (Accelerometer+Gyroscope) Interfacing with Raspberry Pi," [Online]. Available: https://www.electronicwings.com/raspberry-pi/mpu6050-accelerometergyroscope-interfacing-with-raspberry-pi. [Accessed 20 03 2022].

[39] H. electronics, "GY-521 MPU6050 3-Axis Acceleration Gyroscope 6DOF Module," [Online]. Available: https://www.hotmcu.com/gy521-mpu6050-3axis-acceleration-gyroscope-6dof-module-p-83.html. [Accessed 20 03 2022].

[40] Hama, "Hama "Slim" 1:4 USB 2.0 Hub, Bus-Powered, black," [Online]. Available: https://www.hama.com/00012324/hama-slim-14-usb-2-0-hub-bus-powered-black. [Accessed 17 03 2022].

[41] P. Kullerkupp, "5G NB-IoT Implementation for Predictive Car Maintenance," Taltech, Tallinn, 2020.

[42] J. R. Koelsch, "Why the MQTT Protocol is So Popular," 25 February 2022. [Online]. Available: https://www.automationworld.com/factory/iiot/article/22080946/why-the-mqtt-protocol-is-so-popular. [Accessed 07 05 2022].

[43] jacohend, "overpass_speed.py," [Online]. Available: https://gist.github.com/jacohend/1e943771d89ab3de067d. [Accessed 24 04 2022].

[44] F. Bu, "An Exploration Of Calculating The Packet Loss Rate By Using The Block Rate," *Advances in Computer Science Research,* vol. 65, 2018.

[45] G. Hermawan and E. Husni, "Acquisition, Modeling, and Evaluating Method of Driving Behavior Based on OBD-II: A Literature Survey," in *IOP Conference Series: Materials Science and Engineering*, 2020.

[46] H. Harkous and H. Artail, "A Two-Stage Machine Learning Method for Highly-Accurate Drunk Driving Detection," *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob),* pp. 1-6, 2019.

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.