Scalable IoT Accelerometer

A Degree Thesis
Submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
and to the Faculty of Electrical Engineering in CTU Prague
by
Pau Tomàs Culubret

In partial fulfilment
of the requirements for the degree in
TELECOMUNICATIONS TECHNOLOGIES AND SERVICES ENGINEERING

Advisor: Lukáš Vojtěch and Israel Martin Escalona

Barcelona, June 2020

Abstract

Driving style behavior in public transport can be divided into aggressive, normal and efficient. Usually, different kinds of hardware sensors are used to recognize these styles.

The project presented in this thesis shows how using only an accelerometer sensor is possible to recognize the different styles of driving and it is also possible to achieve fuel-consumption optimization and improvement in safety without need of a big inversion. A score method scale that helps classifying the different styles of driving has been proposed when driving in the same route. Also, statistics have been used in order to complete this driving style scale.

Resum

El tipus de conducció en el transport públic pot dividir-se en agressiu, normal i eficient. Normalment, s'utilitzen diferents tipus de sensors de hardware per reconèixer aquests estils.

El projecte presentat en aquesta tesi mostra com utilitzant només un sensor de acceleròmetre és possible reconèixer els diferents estils de conducció i també és possible aconseguir l'optimització del consum de combustible i la millora de la seguretat sense necessitat d'una gran inversió. S'ha proposat una escala de mètode de puntuació que ajuda a classificar els diferents estils de conducció quan es condueix per la mateixa ruta. A més, s'han utilitzat mètodes estadístics per completar aquesta escala d'estils de conducció.

Resumen

El tipo de conducción en el transporte público puede dividirse en agresivo, normal y eficiente. Normalmente, se utilizan diferentes tipos de sensores de hardware para reconocer estos estilos.

El proyecto presentado en esta tesis muestra cómo usando sólo un sensor de acelerómetro es posible reconocer los diferentes estilos de conducción y también es posible lograr la optimización del consumo de combustible y la mejora de la seguridad sin necesidad de una gran inversión. En el proyecto se ha propuesto una escala de método de puntuación que ayuda a clasificar los diferentes estilos de conducción cuando se conduce por la misma ruta. Además, se han utilizado estadísticas para completar esta escala de estilos de conducción.

Acknowledgements

First of all, I would like to thank Ing. Lukáš Vojtěch, Ph. D, and Ing. Marek Neruda, Ph. D, for giving me the opportunity of developing my bachelor thesis abroad. Moreover, I would like to express my gratitude towards them for being always willing to help me in every matter and for his implication on this project.
Also, to my advisor at UPC, Israel Martin Escalona for his willingness to help.

In addition, I would like to thank to all the people that I met in Prague, who have made this experience unforgettable.

Finally, I would like to thank to my parents who have been always showing their support to me.

Revision history and approval record

Revision	Date	Purpose
0	03/07/2020	Document creation
1	24/07/2020	Document revision
2	26/07/2020	Document revision
3	29/07/2020	Final Document

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Pau Tomàs Culubret	paautc@gmail.com
Lukáš Vojtěch	lukas.vojtech@fel.cvut.cz
Israel Martin Escalona	israel.martin@upc.edu

Written by:		Reviewed and approved by:	
Date	24/07/2020	Date	29/07/2020
Name	Pau Tomàs Culubret	Name	Lukáš Vojtěch
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables:

1. Introduction

One of the main existing problems of this modern era is related to the environmental situation. Nowadays there is a booming economy that increments and increases urbanization in cities, and in fact, transport is the main global pollutant of environment in cities and what is an important step to reduce this pollution is to encourage people to use more public transport and reduce the use of cars. For this, an important feature from these mains of transport is to be as more efficient as possible, in both fuel-efficiency and safety.
Most of the means of transport are powered by electric motors, but there is still one that still have fuel engine vehicles and it is the bus. It is important to find a way to improve the fuel efficiency from the bus without a big amount of inversion such as changing mechanical pieces or introducing very expensive hardware in the vehicle.

1.1. Statement of purpose

The aim of this project is to develop a system to monitor the bus movements and to distinguish the style of driving to motivate the bus driver to be more efficient in both fuel-efficiency and safety.
Usually, GPS sensors are good to extract information about driving because of the inertial measurement signal, but, as the system is only focused on the public transport, it is not necessary. In fact, only an accelerometer sensor will be used to extract information that will help classify the driver between aggressive, normal, and efficient driving. Moreover, it is important to distinguish between these three different driving styles due to the fact that one of the principal ways to reduce aggressive driving comes from motivate the driver to be fuel-efficient and to perform a safe driving, to do so, an efficient driving should come along a salary increase.
In addition, and related to the project, many research studies have shown that an accelerometer captures driving style, skill and driving behaviour from the bus driver. Only longitudinal and lateral accelerations are used to measure safety, economy and comfort.

1.2. Requirements and specifications

Requirements and specifications for this project are detailed below.

1.2.1. Requirements

- Monitor bus movements and driving: With monitoring bus movements and driving, it is possible to know the efficiency from the bus driver and to know when an aggressive behaviour is taking place.
- Predictable monitoring tool capturing and classifying driving styles and fuel consumption optimization: Low-cost intelligent monitoring system that is capable to classify aggressive, normal and efficient drivers and moreover reduce the fuel consumption from aggressive drivers.

1.2.2. Specifications

The main parts of this projects are an accelerometer GY-801 and a Raspberry Pi.

Specifications	
Measuring accelerometer range	±16g
Pressure range	300 to 1100hPa
Communication Mode	I2C
Power Supply	3-5V
Measuring gyroscope range	±2000º/sec
Measuring range magnetic field	±8 Gauss

Table 1: Project Specifications

1.3. Resources

Throughout this project both hardware and software resources have been used.
Hardware overview
Two main components: Raspberry Pi, GY-801 sensor.
Its information and references sources are presented in section 2.4.1.
Software overview
Two main different software: Node-RED, Grafana.
Moreover, to storage the data from the accelerometer, two different databases are used to communicate, transmit and receive information with the two main software: SQLite database and InfluxDB.
Its information and references sources are presented in section 2.4.2.

1.4. Planning

Work plan with tasks, milestones and Gantt diagram are included in Appendix I.

2. State of the art of the technology used or applied in this thesis

On this part, the definition of styles of driving will be exposed along with the benefits and the importance of efficient driving focusing it on the public transport, in addition to related recent research about this topic in many different ambits and aspects, more specifically, smartphone systems and fixed systems will be discussed as well as the system, analytics and technology used in this thesis.

2.1. Styles of driving

The style of driving can be referred as the way a person choose to drive taking into account the human factor. It includes features such as speed, acceleration, headway and habitual levels of attentiveness. Also, the driving style is one of the important problems in the field of traffic safety and fuel consumption.
These styles can be divided into three main categories: (1) *aggressive driving*, [1] defined as the behavior of an individual who "commits a combination of moving traffic offences so as to endanger other persons or property", moreover it can be described as an anxious, irregular and careless style, considering it to be both fuel-consuming and unsafe. It is important to detect, prevent and warn the driver, in order to reduce the occurrence of accidents and to be fuel-efficient; (2) *efficient driving*, [2] defined as adaptive driving with attention to the road, patience, calmness, fuel-efficient, low carbon dioxide emissions and considered as safety driving; (3) *normal driving*, between aggressive and efficient driving, considered as non-danger driving but non fuel-efficient, it is important to divide the style of driving into three different categories because this can help to potentiate and motivate the public transport driver to be efficient.
Moreover, the style of driving is expected to be influenced by attitudes and beliefs regarding driving, with the meaning that this style can be corrected within these human behaviours.

2.2. Benefits of efficient driving in public transport

The transport sector is one of the main causes of CO_2 emissions, [3] the European Commission have set a limit to decrease these CO_2 emissions, in fact, there is an agree to reduce the 40% of the carbon dioxide from the vehicles by the end of 2030, and 60% by the end of 2050.
Nowadays, there are many countable actions with the aim of improving driving techniques to reduce fuel consumption that nowadays are gaining more relevance.
Methods to reduce the fuel consumption can be found in many directions, improving the technology in vehicles, reducing energy consumption and in consequence the pollutant emissions, converting the conventional combustion engine buses to hybrid or full battery buses are being investigated as solutions to obtain environmental benefits. Another method aimed to reduce the fuel-consumption trying to be more efficient with the existing technology optimizing the routes in the public transport by applying maintenance or by using efficient driving programs.
Focusing in this last method and comparing it with the others, it is the most economically and undemanding to work with. This method takes advantage of the existing vehicle with no need to invest in new technology. The proposal is a system based on training and feedback to try to modify the driver's behavior and then maintain the modifications with a bounty.

Efficient driving takes into account the human factor, allowing a reduction in fuel consumption of around 10%, in fact, studies by Toyota automobile company prove that this reduction could be reduced due to the driver behavior, and also an increase in the safety and security, defined this as the actual degree of safety from accidents and the feeling of security from the passengers.

2.3. Survey of related works

2.3.1. Smartphone systems

Nowadays there are many solutions to analyse and monitor the acceleration of a vehicle. In particular, there are 3 that are related to this thesis:

1. Accelerometer Meter (Google Play) [4]:
   App based on the accelerometer sensor from the Smartphone. You can interact and view the output from the accelerometer sensor. The app has six screens to choose from: Meter (maximum and minimum values), graphic (accelerometer output over time), spectrum, light, music and info.
2. Android Smarthphone Applikation für Fahrstilerkennung (Android Smartphone Application for Driving Style Recognition) by *Radoslav Stoichkov* [5]:
   The application presented in this work shows how the driving style recognitions can be achieved using the smartphone sensors.
3. BMW M Power Meter (App Store) [6]:
   App that uses the smartphone accelerometer sensor and GPS, and gives information about acceleration (lateral and longitudinal, speed (maximum, average, real time), distance…

2.3.2. Fixed systems

As the system proposed in this thesis is a fixed system, these next related works and researches, are on-board built-in systems that doesn't need a smartphone for its development.

4. Adaptive Learning for Efficient Driving in Urban Public Transport [7]:
   Personalized adaptive Learning System for efficient driving, which allows the evaluation of professional drivers of urban public transport in their work environment.
5. IoT On-Board System for Driving Style Assessment [8]:
   The proposed system for driving style is based on eight indicators, which are associated with the vehicle's speed, acceleration, jerk, engine rotational speed and driving time. The useful data are acquired from the car diagnostic port and from an additional accelerometer sensor and GPS module.
6. Driver Classification and Driving Style Recognition using Inertial Sensors [9]:
   The proposed system uses the vehicle's inertial sensors from the CAN bus to build a profile of the driver to ultimately provide a feedback of the way of driving.
7. Simulating autonomous driving styles: Accelerations for three road profiles [10]:
   Experimental approach to simulate projected autonomous driving style based on the accelerations of the driver. Experimental investigations have been carried out at three different road profiles (junction, speed hump and corner).

8. Driving style classification using long-term accelerometer information [11]:
   Pattern recognition approach to classify driving style when driving the same route using accelerometer data.
9. *Scania Fahrer Eco-module* [12]:
   On-board system module that provides a real-time feedback on how to optimize the style of driving. A score is displayed on the on-board computer, this score is an average value of four elements (hill driving, gear choice, anticipation and brake use).
10. Volvo – *Driver Alert Control* (DAC) [13]:
    On-board system that monitor the movements of the steering wheel and the data from a camera that controls the edges of the road. When a way of driving is aggressive or decontrolled the driver is alerted by an alarm.

2.4. System overview

This subsection introduces the technology and analytic methods used in this project.

2.4.1. Hardware description

The device hardware is built as the integration of two hardware modules:

11. Raspberry Pi 3 Model B V1.2.
12. GY-801 sensor.

Raspberry Pi 3 Model B V1.2 [14] – Third generation low-cost embedded board. With a Quad Core 1.2GHz Broadcom BCM2837 64bit CPU and 1 GB of RAM. It has wireless LAN and Bluetooth connectivity that allows to communicate via SSH in a private network. Also, the General Purpose Input/Output (GPIO) port consists in 40 pins allowing the Raspberry Pi to communicate with external elements, in this case, it allows the communication with the GY-801 module. Also the Ethernet port is used for the Internet and SSH connection with a PC. Moreover, the device support I2C communication, which will be used in this project.

Figure 1: Raspberry Pi 3 Model B V1.2

GY-801 Module [15] – It is a 10-axis module, updated version of the predecessor GY-80 and includes 4 I2c sensors:

13. Gyroscope sensor: L3G4200D 3-Axis Angular Rate Sensor
14. Accelerometer sensor: ADXL345 3-Axis Digital Accelerometer, it communicates with the Raspberry Pi via the GPIO ports.
15. Magnetometer sensor: HMC5883L 3-Axis Digital Compass
16. Pressure sensor: BMP180

17. I2C interfaces, manages the communication with Raspberry Pi.

The module is designed for use with a large variety of microcontrollers using communication through the I2C bus, it is therefore taken to be handled by in this case Raspberry Pi or other many development kit.

Figure 2: GY-801 sensor hw

Protocol communication

In this project, two different protocols have been used to establish a communication between the module, Raspberry Pi and PC.

I2C communication:

[16] As both devices (GY-801 and the Raspberry Pi) supports I2C connectivity, the communication will be with this protocol. This Inter-Integrated Circuits (I2C) Protocol [14] is a protocol intended to allow multiple "slave" digital chips to communicate with one or more "master" chips. It is only intended for short distance communications within a single device and it only requires two signal wires to exchange information, SCL (Serial Clock) and SDA (Serial Data). The clock signal is always generated by the current bus master and is this one who initiates the communication, the slave receives the clock and responds when addressed by the master. Raspberry Pi is master and GY-801 is slave.

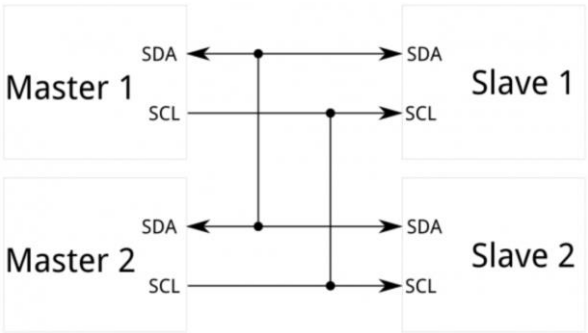


Figure 3.1: I2C Protocol Communication

SHH communication:

SSH™ [26] (or Secure SHell) is a protocol which facilitates secure communications between two systems using a client/server architecture and allows users to log into server host systems remotely. Unlike other remote communication protocols, such as FTP or Telnet, SSH encrypts the login session, making it impossible for intruders to collect unencrypted passwords.

SSH is designed to replace older, less secure terminal applications used to log into remote hosts, such as telnet or rsh. A related program called scp replaces older programs designed to copy files between hosts, such as rcp. Because these older applications do not encrypt passwords transmitted between the client and the server, avoid them whenever possible. Using secure methods to log into remote systems decreases the risks for both the client system and the remote host.

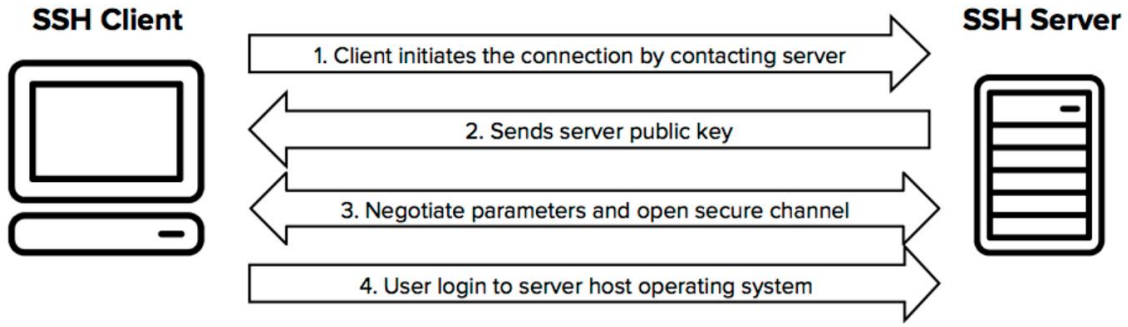


Figure 3.2: SSH protocol communication.

2.4.2. Software description

Node-RED

Node-RED [17] is an open-source mashup tool developed by IBM and released under Apache 2 license. Based on the server side JavaScript framework Node.js. It is uses an event-driven, non-blocking I/O model suited to data-intensive, real-time applications that run across distributed devices.

It provides a GUI where users drag-and-drop blocks that represents components of a large system which can either be devices, software platforms or web services that are to be connected, these blocks are called nodes. A node is a visual representation of a block of JavaScript code designed to carry out a specific task. Additional nodes can be placed in between these components to represent software functions that manipulate and transform the data during its passage. After connecting many nodes, the final diagram is called a flow.

Node-RED flows are represented in JSON and can be serialized in order to be imported and exported easily.

Moreover, since nodes are blocks of JavaScript code it is possible to wrap any kind of functionality and encapsulate that as a node in the platform.

To make a device or a service compatible with Node-RED, a native Node.js library capable of talking to the particular device or service is required, in fact, the project has support of different libraries in order to be capable to communicate with the sensor and data bases.

JavaScript

As Node-RED is based on the server side JavaScript framework Node.js, all functions are coded using JavaScript code.

JavaScript (JS) [18] is a lightweight, interpreted, object-oriented programming language with first-class functions. Although it is best known as the scripting language for web pages, many non-browser environments also use it, such as in the project case, node.js, and also Apache CouchDB, and Adobe Acrobat. It is a multi-paradigm, prototype-based, dynamic scripting language that supports object-oriented, imperative and declarative styles. Moreover, the standard for JavaScript is ECMAScript. As of 2012, all modern browsers fully support ECMAScript 5.1.

Grafana

Grafana is a software that allows to query, visualize, alert on and understand the metrics no matter where they are stored. It is possible to visualize and alert the data with plenty of options, from heatmaps to histograms, moreover as it is opensource, it gives plenty of options to extend this graphics to many dashboards and plugins.

Databases

Two main databases have been used to storage all the data from the project

18. SQLite
    SQLite is a C-language library that implements a small, fast, self-contained, high-reliability and full-featured SQL database engine. The SQLite file format is stable, cross-platform, and backwards compatible and the developers pledge to keep it that way through at least the year 2050. SQLite is open source and in the public domain.
19. InfluxDB
    To interact with Grafana, influxDB has been used. It is an open-source time series database (TSDB) developed by InfluxData. It is written in Go and optimized for fast, high-availability storage and retrieval of time series data in fields such as operations monitoring, application metrics, Internet of Things sensor data, and real-time analytics.

2.5. Statistics analysis

Apart from the central characteristics statistics (mean and median) and also the principal analytic features (maximum and minimum values, histogram, three moment skewness, variance, covariance and standard deviation) a statistical method introduced by Horn has been used.

2.5.1. Horn Method

[19] With a very small sample sizes 4 < n < 20 available for evaluation process, a procedure based on order statistics introduced by Horn30 was used. This is based on the depths that correspond to the sample quartiles. The pivot depth is expressed by:

$$H_L = int[(n+1)/2]/2 \text{ or } H_L = int[(n+1)/2+1]/2 \quad (1)$$

According to which H_L is an integer. The lower pivot is $x_L = x_{(H)}$ and the upper one is $x_U = x_{(n+1\ H)}$. Note that the $x_{(i)}$ are ordered statistics, that is, $x_{(i)} <= x_{(I+1)}$. The estimate of the parameter of location is then expressed by *the pivot half sum*:

$$P_L = 0.5(x_L + x_U) \quad (2)$$

and the estimate of the parameter of spread is expressed by the pivot range:

$$P_L = 0.5(x_U - x_L) \quad (3)$$

The 95% confidence interval of the mean is expressed by pivot statistics as:

$$P_L - R_L t_{L,0.95}(n) \leq \mu \leq P_L + R_L t_{L,0.95}(n) \quad (4)$$

and analogously hypothesis testing may also be carried out. For small samples (4 < n < 20), the pivot statistics lead to more reliable results than the application of Student's F-test or robust t-tests.

3. Methodology / project development

In this chapter is included all the project development, starting for the system architecture, all the methodology, scripts, algorithms and analytics used for the development of the project.
Moreover, as the code from each part are a big amount of data, in this chapter there will only be the explication and flows figures, leaving the big amount of code and tables in the annex part.

3.1. System architecture

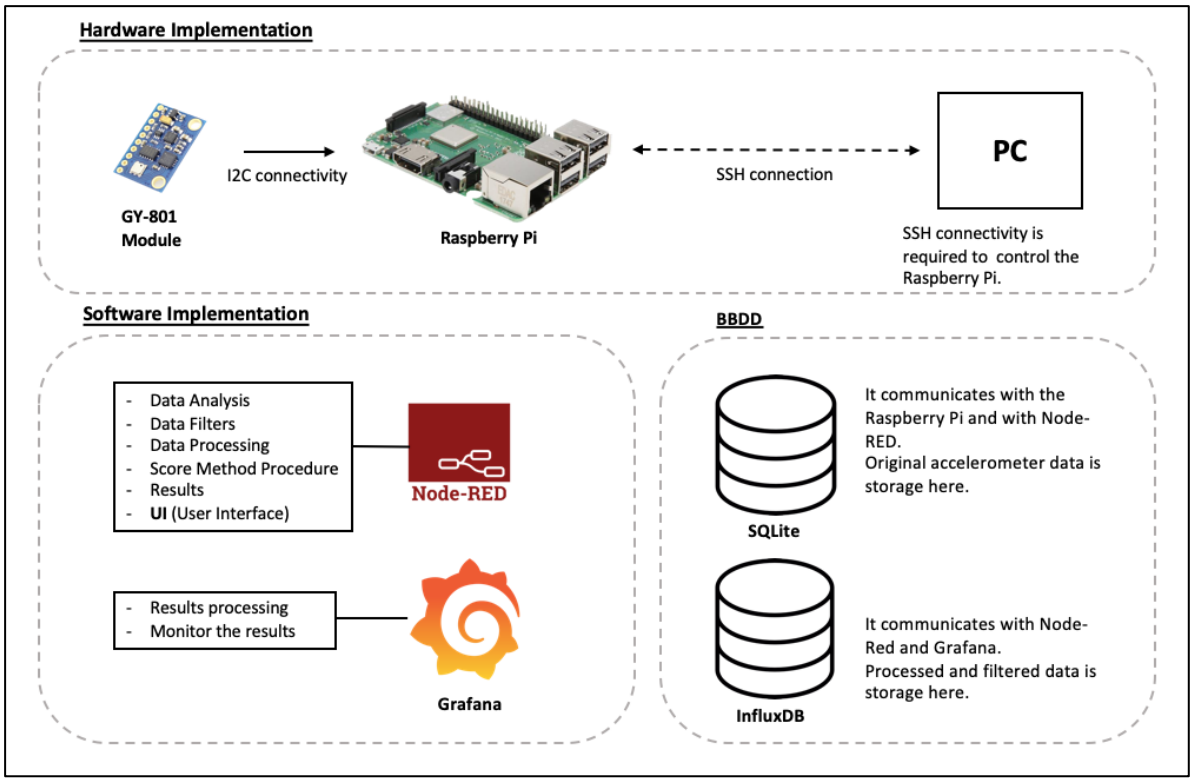


Figure 4: System architecture

As can be seen in the figure above, the project can be divided into hardware and software implementation. The hardware part includes I2C communication between the module and the Raspberry Pi and the SSH (secure shell protocol) connection in a private network between the Raspberry Pi and a PC. Moreover, the software part includes the data extraction, data processing, data analysis, monitoring results as well as the traffic of data between the data bases and the different type of software used.

3.2. Hardware implementation

Hardware implementation consists of an integration between the GY-801 module and the Raspberry Pi with a SSH connection with a PC on a private network to manage and control the system remotely.
Due to the I2C connectivity, only 4 pins are required to be connected between the GY-802 and the Raspberry Pi (Vcc, Ground, SCL: Serial Clock and SDA: Serial Data).

Connection	GY-801 PIN	Raspberry Pi GPIO PIN
Vcc	*VCC_IN*	*Pin 1: 3.3V*
Ground	*GND*	*Pin 6: GND*
SCL: Serial Clock	*SCL*	*Pin 3: GPIO 3*
SDA: Serial Data	*SDA*	*Pin 2: GPIO 2*

Table 2: Pin connection between Raspberry Pi and GY-801

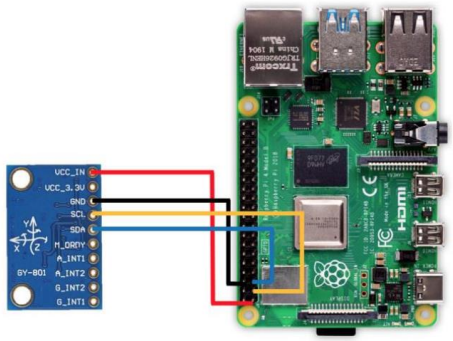


Figure 5: Connectivity between GY-801 and Raspberry Pi

The communication between the devices can be in two directions:

- Raspberry Pi tx → GY-801 rx:
  To initialize and enable measurement modes from the ADLX345 sensor.
- GY-801 tx → Raspberry Pi rx:
  3-axis data transmission from the accelerometer sensor.

3.3. Software implementation

The project has been developed under Raspbian operating system, as said, with a SSH connection to a PC. This SSH connection is stablished in a private network (192.168.0.X). Moreover, the main software used in this project is Node-RED (v1.0.6 (npm)), as introduced in the section above, it is a flow-based development tool for visual programming with a web browser-based flow editor. Apart from the node-red internal packages, 4 different packages have been installed in node-red in order to work with the project, the explication of each of package is explained in the following subsections: *node-red-contrib-I2C*, *node-red-node-sqlite*, *influxdb-package*, *node-red-dashboard.*
In this project, all the scripts, data analysis and data processing has been coded with JavaScript, the functions or libraries used are Math library and Covariance function from local JavaScript.
In addition, Grafana software has been also used to help monitor the accelerometer data.
The next subsections are explained the process since the data extraction to final results.

3.3.1. Data Extraction

To transmit and extract the data from the accelerometer sensor, the package *node-red-contrib-i2c* [18] has been used. It allows the communication with Raspberry Pi I2C driver and it provides three nodes:

- **i2c scan:** This will scan the I2C bus for connected devices.
- **i2c in** (input i2c): This node will request data from a given device.
- **i2c out** (output i2c):This node will send a given String/array/buffer to a given device.

Also, all the data extracted from the sensor, is stored in a data base, specifically into a SQLite DB. A SQLite package (*node-red-node-sqlite*) [19] has been used to store the extracted data.

For the data extraction, as can be seen in the next figure, a node-red flow has been developed to extract the data from the accelerometer sensor and to store it into a SQLite database.

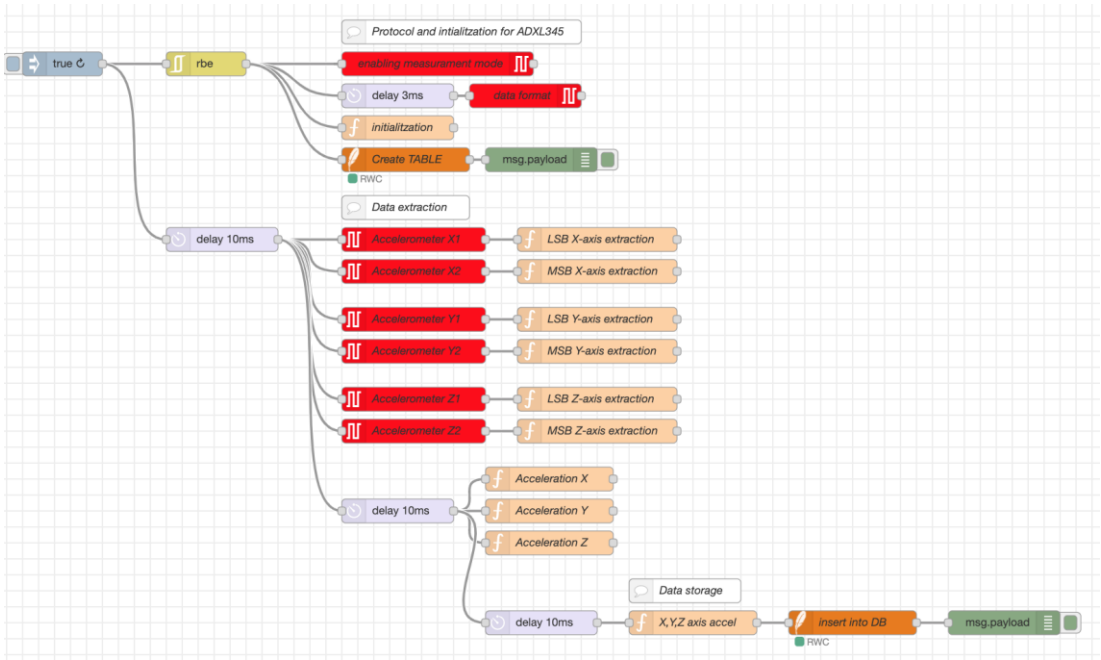


Figure 6: Accelerometer extraction and storage flow

The following scheme compacts the scripts and procedures that are in the figure above.

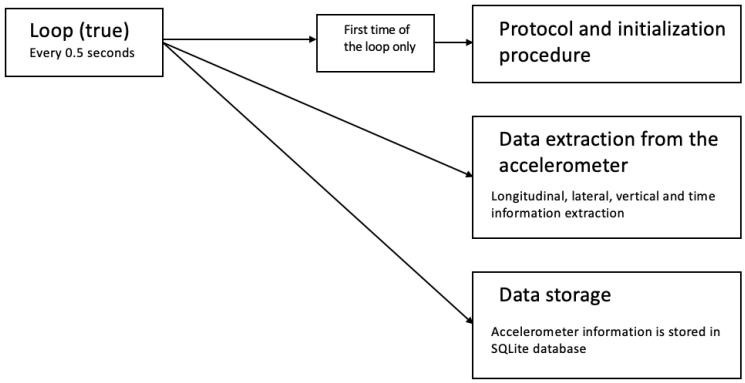


Figure 7: Scheme for the data extraction flow

The process of extraction and storage the data is divided into mainly three parts with a true node that will act as a timestamp every 0.5 seconds to repeat the process of the flow every 0.5 seconds.

- Protocol and initialization for ADXL345:
  This part covers the initialization for the ADXL345, all variables and SQLite table creation. As seen it comes first with a *rbe node*, the functionality of this node is to let this part run only the first repetition of the process. The other times will run only the *data extraction* and *data storage* parts.
  The initialization consists on:

- Enable measurement mode: With an *I2C out node* the first thing to do is to enable measurement mode with register POWER_CTL from ADXL345 which corresponds to the command 0x2D (45d) and we have to enable the D3 bit (1000 -> 0x8 -> 8d). Also, as we are referring to the ADXL345 sensor, we have to set a bus address to 83 (0x53). All this registers are reported in the ADXL345 sensor datasheet [20].

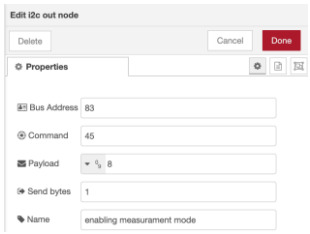


Figure 8: Capture of the i2c out node (enable measurement mode)

- Data format: Also with a *I2C out node* it is needed to format the data, which means, set the resolution of the measurements.
  In this case, only 3 bits are required (D3, D1 and D0):
  - D3 is the FULL_RES bit, which enable the full resolution mode.
  - And D1 and D0 are the operating range:

Table 21. *g* Range Setting

Setting		
D1	D0	*g* Range
0	0	±2 *g*
0	1	±4 *g*
1	0	±8 *g*
1	1	±16 *g*

Figure 9: Range Setting in ADXL345 datasheet.

Where higher *g Range* values will have a wider measurement range and lower values will have more sensitivity.
As for driving monitoring both are important, 4G Range is selected.

- Initialization of flow variables.
- SQLite table creation: A SQLite table needs to be created in order to storage all the data from the accelerometer. This table will be located in */home/pi/sqlite* in the ROM memory of the Raspberry Pi and it will storage four different data: X-axis (lateral information), Y-axis (longitudinal), Z-axis (vertical) and time information.

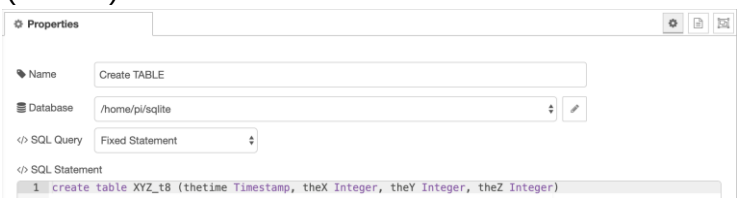


Figure 10: Capture of the creation of the table

- Data extraction:
  In this section, the data is extracted from the sensor using an *I2C in* node, where the data from the sensor will be transmitted to the Raspberry Pi.
  As reported in the ADXL345 datasheet each axis has the output information divided in 2 bytes, one for LSB (Less Significant Bit) and one for MSB (Most Significant Bit). and so, each byte information needs to be transmitted in different command lines:
  - X-axis: LSB [0x32 or 50 (in decimal)] and MSB [0x33 or 51 (in decimal)]
  - Y-axis: LSB [0x34 or 52 (in decimal)] and MSB [0x35 or 53 (in decimal)]
  - Z-axis: LSB [0x36 or 54 (in decimal)] and MSB [0x37 or 55 (in decimal)].

Figure 11: Information from the Accelerometer X1 (i2c in node).

This node will return as the data value of the LSB X-axis.

In the three last nodes of this section (Acceleration X, Y, Z) the LSB and MSB of each acceleration (X, Y, Z axis) are integrated together. As said before, the code will be displayed in the annexed part.

- Data storage:
  The last section is the storage of the data in the SQLite DB previously created. To do so, we introduce the values X, Y, Z and time data to the DB.

Moreover, this process will be repeated each 0.5 seconds, so, there will be 2 data extraction every 1 second until the monitoring is finished. What is more, the flow is totally autonomous, node-red software has been programed to be *autostarted* at every boot to use it in the bus and recording the data without need of a screen neither any kind of initialization.

3.3.2. Data Processing and Data Analysing

Once the data is stored in the BD, needs to be processed and analysed with analytics methods to take a final score for the driver, also the data need to be presented in both UI and monitoring software.

In the next figure, the flow responsible for the processing and analysing is presented.

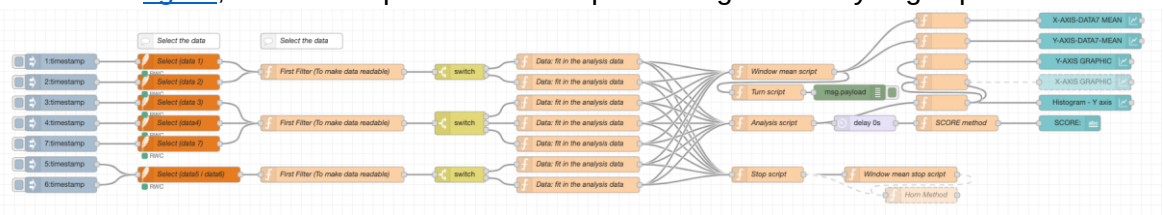


Figure 11: Data processing and analysing flow

The following scheme compacts the scripts and procedures that are in the figure above.

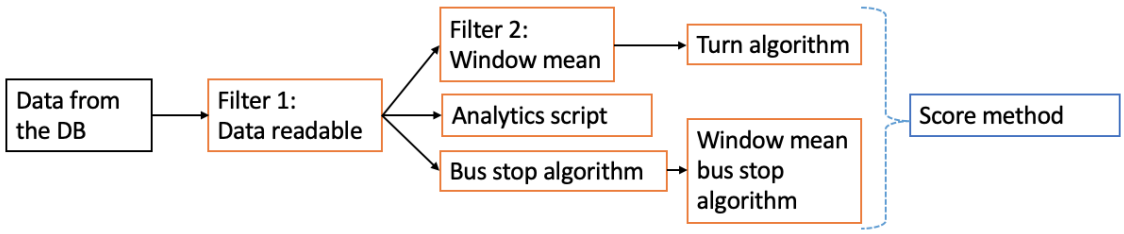


Figure 12: Scheme for the processing and analysing scripts and procedures

3.3.2.1. Filters

As seen in the scheme, there are two different filters:

- Filter 1: Data readable
  The first scheme makes data readable. This filter is needed because as the wires from the accelerometer are not welded and as the bus makes vibrate the

accelerometer, the received data comes with some errors such as the ones in the figure 13, which Y-axis is exposed in g's and the X-axis in time.

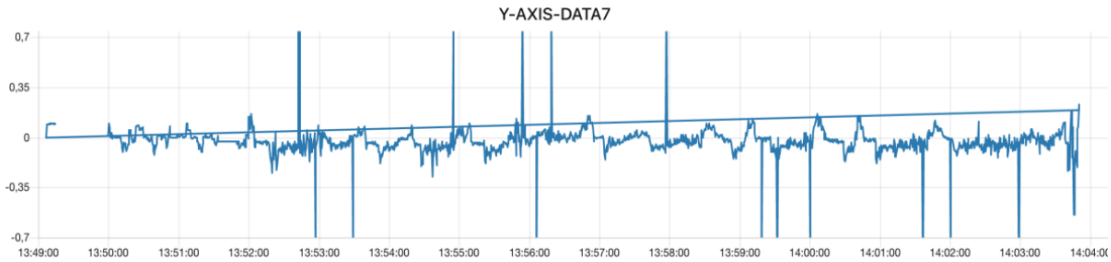


Figure 13: DB data Y-AXIS acceleration from Test 7 in g's

This filter eliminates and replaces the high accelerations (not correct) and eliminates repeated and useless data. The replaced data is possible due to the fact that the initial data is extracted every 0.5 seconds, so there are 2 data points every 1 seconds, but the filtered data will use half of the data, the other half will be replace the useless and high accelerations data, consequently the data points are every 1 second. The reason for using only half of the data is that if one data point is not correct (high acceleration or not time correct) it will be replaced for the next data point.
The code in JavaScript and explication of the filter is in the annexed part.

Figure 14: Result of the first filter script applied on the Data 7 Y-AXIS data, Y-axis is exposed in g's and the X-axis in time.

- Filter 2: Window mean

The second filter script works in a 5 second window frame, which means that the script averages 5 followed data points and repeats it for all the data. The result of this script helps not only to determine the bus turns but also to determine where are the higher 5 seconds accelerations from the breaks and accelerations. Moreover, the filter helps reducing the vibration produced by the bus. This part will be discussed in the next chapter.

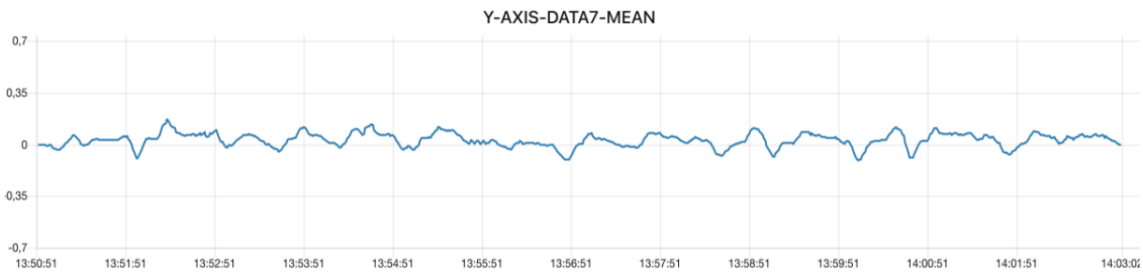


Figure 15: Results of the second filter script applied to the Data 7 Y-AXIS data, Y-axis is exposed in g's and the X-axis in time.

3.3.2.2. Bus stop algorithm

The bus stop algorithm detects the stops that the driver does. As the data is not repeatable, because of traffic conditions, this algorithm is obligatory if a dynamic analysis of all the stops wants to be performed to compare bus stops. In fact, this algorithm tries to avoid the quickly traffic lights stops, because, only bus stops wants to be compared (in fact is the only thing that all buses from the route repeat, is better to do only a comparative with what every bus does).
The algorithm is based on the acquisition of five thresholds limits acquired from the analysing of the tests which is explained in the results part:

- Bus stop thresholds:
  When the bus stops, the perfect measured acceleration should indicate 0 m/s^2, but due to the fact that the public-bus motor makes the accelerometer vibrate and also the sharp lane changes make the accelerometer not indicate this 0 m/s^2, consequently this acceleration cannot be considered, a higher and lower threshold must be used to indicate the stop.
- Maximum and minimum acceleration thresholds:
  Same as before, a higher and lower threshold must be considered to detect when a public transport bus is stopping (breaking) and is accelerating.
- Time threshold:
  As the main factor is to have data comparable, it is important to have the same actions in all buses, that is why a time threshold is needed to detect the bus stops and try to avoid all the traffic light pit-stops. This time threshold is measured as the lower time a bus stays in a bus-stop (7 seconds).

The algorithm works with these 5 thresholds. When a break acceleration is detected due to the minimum acceleration threshold, the bus stop thresholds are waiting to receive at least 7 followed data points, after this, when a high acceleration is introduced, there are two threshold waiting, the maximum acceleration threshold and the time threshold, if the result is true it will indicate that this is a stop. Also, the algorithm returns a vector with 4 variables (*minimum_value, maximum_value, distance_value, start_stop, finish_stop*), the amount of vector points are the stops that the bus did.

- Minimum_value:
  A minimum value is set when the break acceleration is detected, if there is a lower value, the variable will change the value for the lower one. Moreover, as we need to consider the sharp lane changes because the road is not always at 0 degrees, this value could be not correct at all, because of this, the value needs to be adjusted with the stop acceleration value, this value is calculated using the mean of the 5 middle stop values of that bus top.
  This variable will contain the minimum acceleration value of each stop.
- Maximum_value:
  A maximum value is set when a high acceleration is detected, if there is a bigger value, the variable will change the value for the bigger one. Same as the minimum value, it has to be adjusted with the stop acceleration value using the mean of the 5 middle stop values of that bus stop.
  This variable will contain the maximum acceleration value of each stop.

- Distance_value:
  Apart from the maximum and minimum value, it is also important to analyse the distance between maximum and minimum values, because, as before, the sharp lane changes may give a non-correct value.
- *Start_stop*:
  This variable contains the time value of the first data from the stop.
- *Finish_stop*:
  This variable contains the time value of the last data from the stop.

The algorithm code and explication is included in the annexed part.

3.3.2.3. Turn algorithm

As the bus stops, the turns needs also to be taken into consideration. In this case the data will be analysed from the window mean script, because it is more easy to detect the turns due to the fact that normal data has a lot of vibrations. This turn algorithm uses only one threshold, a maximum turn threshold. When a value is detected for more than 6 seconds, means that there is a bus turn. The algorithm will return a vector with three variables (*maximum_value, start_turn, stop_turn*):

- Maximum value:
  The variable will contain the maximum turn value.
- Start_turn:
  It will contain the start time of the turn.
- Stop_turn:
  It will contain the stop time of the turn.

The algorithm code and explication is included in the annexed part.

3.3.2.4. Analytics script

This script contains the principal statistic features from the longitudinal and lateral accelerations in the time domain. These statistics are extracted from the accelerometer signal and take into account all the route data for the analysis. Moreover, if the driving samples were separable (same traffic conditions, same stops, same amount of data), the 98% accuracy for the style of driving [11] can be detected from the central characteristics (mean and median), but as it cannot be easily repeatable, statistics need to be taken into account:

- Central characteristics:
  - Mean
  - Median
- Maximum sample values:
  - Maximum value
  - Minimum value
- Histogram characteristics:
  - Histogram
  - Third moment (skewness)
- Variance
- Covariance

- Standard deviation

The script code and explication is included in the annexed part. The analytics are exposed in the results section.

3.3.2.5. SCORE method

Using a numerical score to detect a style of driving has been performed in many previous projects (for example Scania on-board system [12]) without taking into account the analytics features, which are also important to this classification. The score method is based on setting thresholds for the statistics, these thresholds are settled using Horn Method. Its implementation and explication is described in the next section, as well as the statistical analysis for the data.

SCORE Algorithm:

The score algorithm is based in all statistical features and repeatable and comparable data from every test. The score is extracted using 13 driving style features.

The first 6 features are related to the bus driver stops, that are achieved using the *bus stop algorithm*. This algorithm as said and explained before, will return all the peak maximum, minimum and distance values from each stop and will return it as a vector. All stops will be compared to extract the final score value. Features:

- Break features:
  - Peak value.
  - Window mean peak value.
- Acceleration features:
  - Peak value.
  - Window mean peak value.
- Distance features:
  - Peak value,
  - Window mean peak value.

Another feature is related the bus driver turns. Using the turn algorithm and as explained before, we can get the peak maximum turn acceleration of each turn of the samples. As the bus stops, all the turns will be also compared:

- Bus turn:
  - Window mean peak value.

And finally, using the analytic script we can get an overall information of the samples, which give an important value to the final score. The principal analytics features can be obtained:

- Mean value.
- Median value.
- Maximum value.
- Minimum value.
- Standard deviation
- Histogram information:
  - 3th moment skewness.

The score algorithm is based in the next formula:

$$SCORE = Stop\_SCORE + Turn\_SCORE + Central\_Characteristic\_SCORE + Maximum\_SCORE$$

The score formula is divided into four main parts:
Stop_SCORE: All the bus stops feature (break peak values, acceleration peak value and distance value).

$$Stop\_SCORE = 100 - \left(\frac{Peak\_value\_mean}{Peak\_value\_threshold} - 1\right) * 50 + \left(\frac{Counter\_bad\_accel}{Counter\_peak\_values}\right) * 50 \quad (5)$$

Peak_value: Corresponds to the mean of all the stop values of the sample.
Peak_value_threshold: Threshold that delimitates if an acceleration is aggressive or not (explained in the results section).
Counter_bad_accleration: Number of bus stop peak values that exceed the threshold limit.
Counter_peak_values: Total number of bus stop peak values.
This first part of the script is repeated for the 6 peak features (break features: peak value and window mean peak value; acceleration features: peak value and window mean peak value; distance features: peak value and window mean peak value).

Turn_SCORE: All the bus turns:

$$Turn\_SCORE = 100 - \left(\frac{Peak\_value\_mean}{Peak\_value\_threshold} - 1\right) * 50 + \left(\frac{Counter\_bad\_accel}{Counter\_peak\_values}\right) * 50 \quad (6)$$

Peak_value_mean: Corresponds to the mean of all the turn values of the sample.
Peak_value_threshold: Threshold that delimitates if a turn is aggressive or not (explained in the next subsection).
Counter_bad_accleration: Number of bus turns peak values that exceed the threshold limit.
Counter_peak_values: Total number of bus turn peak values.

Central_Characteristic_Score: Both mean and median characteristics, as the data is not repeatable, mean and median are the only central characteristics measured.

$$Central\_Characteristics\_Score = 100 - \frac{\left(\frac{Characteristic\_value}{Characteristic\_threshold} - 1\right)}{Distance\_average\_value} \quad (7)$$

Characteristic_value: Mean or median sample value.
Characteristic_threshold: Threshold that delimitates if the mean corresponds to aggressive driving or not (explained in the next section).
Distance_average_value: Distance value between the characteristic threshold and the characteristic interval, this interval (explained in the next subsection) is extracted using the Horn Method.
This part is repeated for mean and median statistics.

Maximum_score: The last 3 features are maximum value, minimum value and standard deviation.

$$Maximum\_score = 100 - \left(\frac{Peak\_maximum\_value}{Peak\_maximum\_threshold} - 1\right) * 100 \quad (8)$$

Peak_maximum_value: Maximum peak sample value.
Peak_maximum_threshold: Threshold that delimitates if the maximum peak value corresponds to aggressive driving or not (explained in the next subsection).

As seen each part of the score is calculated from basic 100 points. As there are 12 features, the result should be around 1200 points. In the next subsection, the intervals and limits for the score method to determine aggressive, normal and efficient driving are exposed as well as the thresholds.

3.3.2.6. Monitoring results

To present the data, two types of software have been used:

- User Interface (UI) in Node-RED:
  A UI has been programmed to interact with the different data storage in the data base. To do so, *dashboard* [18] module has been used, this module provides a set of nodes in node-RED to create a live data dashboard.
  All graphics nodes need the input to be in a json format, in fact, all the function before the graphic dashboard nodes take care of the conversion of the data to json. The UI consists of three tabs, one with the general monitoring information of the 7 tests used for deciding the style of driving, another with specific information about each test and the last one to start a new test.
  - General monitoring information:
    Data storage in 7 different SQLite data bases is projected all the time in the tab. To do so, the analysing and processing flow needed to be modified in order to introduce buttons, texts and graphics in the user interface. Moreover, the flow for this tab can be seen in the annexed part.
  - Test information:
    Data storage in 7 different SQLite data bases is projected individually in the tab. To do so, a function has been introduced only to filter the data of only one test at a time. The analysing and processing flow has been modified in order to introduce buttons, texts and graphics in the user interface. Moreover, the flow for this tab can be seen in the annexed part.
  - New test information
    The third tab projects a real-time graphics and information of the sensor. This information is stored in a SQLite data base as well as to an InfluxDB also in real time. Moreover, there is a button that can delete the data, this button will delete the data from both databases. The flow for this tab can be seen in the annexed part.

- Grafana monitoring:
  Another proposed solution for monitoring the bus data is Grafana. As said, it is a powerful software to treat, analyse and monitor data.
  Grafana software is programmed to run in the Raspberry Pi where it is installed. Furthermore, SQLite is used to storage the data, but Grafana does not support this DB. Instead of SQLite, Influxdb data base has been used to present the data to Grafana.
  To incorporate the data in Influxdb, the *influxdb-package* has been used [19]. Moreover, the flow for the translation of the data from one DB to another is the followed one:

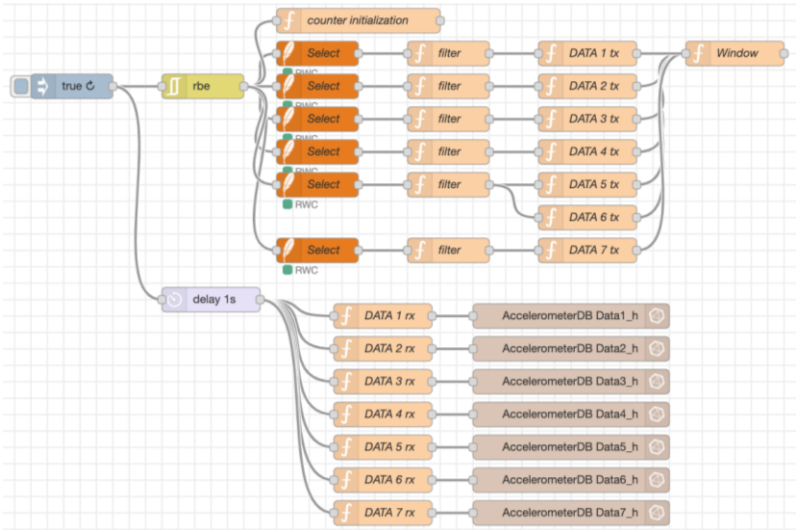


Figure 16: Data transferred to a InfluxDB database.

As can be seen, the data is transferred from SQLite to influxDB data base with the filter 1 already applied, so, the data received in the second data base, is already processed. All the code and explication from the flow is exposed in the annexed part.
A scheme easy to link with the flow above is presented next.

Figure 17: Procedure to monitor data in Grafana

Furthermore, once the data is in the influxDB database, and as Grafana ships with a feature-rich data source plugin for backends such as InfluxDB, the data can be monitored in dashboards.
Grafana software has different formats to present the data, in this project only two different formats have been used, the data directly from the influxdb database and its histograms.
The dashboards are presented in the results section.

3.4. Data bases

As said, two different types of data bases have been used. Both are installed in Raspberry Pi have been installed in Raspberry Pi system.

- SQLite (v 3.32.3)
  In SQLite database all tests raw data are stored, moreover all data tests are storage in a same database called *AccelerometerDB* and in different tables which is how a database is structured in this type of database.
  As seen in the flow above, node-red software is responsible for the communication (tx and rx) with the data base with the *node-red-node-sqlite* library.
  In the next table is presented all the tables that are stored in the database *AccelerometerDB* in Raspberry Pi:

Table	Information
Data_1	Raw accelerometer data from test 1.
Data_2	Raw accelerometer data from test 2.
Data_3	Raw accelerometer data from test 3.
Data_4	Raw accelerometer data from test 4.
Data_5	Raw accelerometer data from test 5.
Data_6	Raw accelerometer data from test 6.
Data_7	Raw accelerometer data from test 7.
New_test	Raw accelerometer data for a new test.

Table 3: SQLite table information.

- InfluxDB (v1.8):
  In influxDB all the filtered data from SQLite is stored. As in the other data base, all data tests are storage in a same database called *AccelerometerDB* and in different series which is how a database is structured in this type of database.
  This database communicates with both node-red and Grafana software. The communication is only in one direction: Node-RED → InfluxDB → Grafana
  In the next table is presented all the tables that are stored in the database *AccelerometerDB* in Raspberry Pi:

Serie	Information
Data1	Filtered accelerometer data from test 1 in g's
Data1_h	Filtered accelerometer data from test 1 in bits
Data2	Filtered accelerometer data from test 2 in g's
Data2_h	Filtered accelerometer data from test 2 in bits
Data3	Filtered accelerometer data from test 3 in g's
Data3_h	Filtered accelerometer data from test 3 in bits
Data4	Filtered accelerometer data from test 4 in g's

Data4_h	Filtered accelerometer data from test 4 in bits
Data5	Filtered accelerometer data from test 5 in g's
Data5_h	Filtered accelerometer data from test 5 in bits
Data6	Filtered accelerometer data from test 6 in g's
Data6_h	Filtered accelerometer data from test 6 in bits
Data7	Filtered accelerometer data from test 7 in g's
Data7_h	Filtered accelerometer data from test 7 in bits
New_test	Filtered accelerometer data for a new test both in g and bits.

Table 4: InfluxDB table information.

InfluxDB has both the data in g's and bits because as bits are integers is easier to extract the histogram.

4. Results

In this chapter all the project results are included, starting with the statistical analysis to find the thresholds for the score method and the score scale election analysis, and finishing with the UI and Grafana application to monitor the data.

4.1. Statistics analysis: Horn Method and threshold election

The statistical analysis consists on 7 different test samples for a public bus route. These different test samples have been storage in different SQLite data bases. Each database has been processed and analysed to find all features from the flow shown in the Methodology section.
After performing the 7 different tests and due to the small sample size ($4 < n < 20$) available for the evaluation process, a procedure based on order statistics introduced by Horn has been used in order to find the thresholds for the score method. These thresholds will be delimited using the 95% confidence interval formula (4) from Horn.
This study is divided into two parts, first a statistical analysis with the different statistical features and then the bus stop and turns analysis.

Statistical analysis:

As introduced in the previous section, this analysis consists into 18 features, but for the score method, only 5 have been used:

- Central characteristics:
  - Longitudinal mean: Y-axis
  - Longitudinal median: Y-axis
- Maximum sample values:
  - Longitudinal maximum value
  - Longitudinal minimum value
- Standard deviation.

As seen, all statistical features are from the longitudinal axis, because as the lateral acceleration correspond to the X-AXIS, it measures the turns, so the mean, median standard deviation should be different depending on how many turns does the bus and in which direction.
With this said, in the next table are summarized the 5 features from each of the 7 tests. The entire statistical analysis from the 18 features can be found in the annexed part.
Moreover, the features below values are expressed in both g's and bits acceleration due to the fact that as the acceleration in g's is very low (order of E-3) it is better to treat the values in bits, the table with real values can be found in the annexed part. To do so, all values must be divided by 0.00390625. This factor corresponds to the bit value instead of g. About its origin, the sensor's default resolution is 10-bits, so a 1 would be the full span of the current range, which is 4g, divided by the resolution of 2^10 bits. Therefore, 4g/2^10bits = 0.00390625 g/bits.

Test	Mean	Median	Maximum value	Minimum value	Standard deviation
Test 1	-3,63 E-3	-19,53 E-3	0,265	-0,320	20,11
Test 2	12,83 E-3	0	0,203	-0,179	15,91
Test 3	38,64 E-3	23,43 E-3	0,293	-0,363	19,89
Test 4	7,00 E-3	3,90 E-3	0,242	-0,238	17,3
Test 5	-2,25 E-3	-15,625 E-3	0,277	-0,242	18,18
Test 6	-19,17 E-3	-31,25 E-3	0,183	-0,273	18,49
Test 7	27,75 E-3	27,34 E-3	0,277	-0,164	14,67

Table 5.1: Mean, median, maximum and minimum values and standard deviation analysis with G's

Test	Mean	Median	Maximum value	Minimum value	Standard deviation
Test 1	-0,9289	-5	68	-82	20,11
Test 2	3,2848	0	52	-46	15,91
Test 3	9,8922	6	75	-93	19,89
Test 4	1,7928	1	62	-61	17,30
Test 5	-0,5756	-4	71	-62	18,18
Test 6	-4,9082	-8	47	-70	18,49
Test 7	7,1049	7	71	-42	14,67

Table 5.2: Mean, median, maximum and minimum values and standard deviation analysis with bits

- Mean: The ideal situation should be a mean very close to 0g, which means that there are same negative and positive acceleration, and consequently that the bus accelerates and deaccelerate alike. A far value from 0g means that the acceleration is higher in accelerating or deaccelerating. It is necessary to emphasize that the values are presented in bits instead of g (in g, a 10 value (in bits) means 0.039g)
- Median: As the median is the middle of a sorted list of numbers, it will give us the middle number, which for the same reason as the mean, should be close to 0g.
- Maximum and minimum values: The maximum and minimum values allow us to know which are the maximum and minimum accelerations of the route in every test. It is important to distinguish this value to the peaks of the stops to know if the maximum value of the route coincides with a stop or it is from another situation.
- Standard deviation: As bigger the standard deviation, the further away the values are from the mean.

As said, Horn Method has been used to decide the thresholds for the *score method*. These thresholds have been found applying the 95% confidence interval formula from Horn (formulas 1,2,3,4) to the mean, median, maximum value, minimum value and standard deviation and are resumed in the next table.
Like all the previous features, Horn Method values are presented in bits instead of g.

Statistic	Pivot Half Sum	Pivot Range	95% confidence interval
Mean	4.658	10.467865	**-1.099** < u < **10.415**
Median	1.500	11	**-4.550** < u < **7.550**
Maximum value	68.500	13	61.350 < u < **75.650**
Minimum value	-56	28	**-71.400** < u < -40.600
Standard deviation	18.705	2.809	17.159 < u < 20.250

Table 6: Horn Method 95% confidence interval

The threshold used in the score method depend on the analytic feature:

- Central Characteristics (mean and median):
  The central characteristics score formula (7) is formed by three thresholds:
  - *Characteristic_threshold*: Corresponds to the 95% confidence interval.
  - *Distance_average_value*: Corresponds to the difference between the average value of the characteristic feature of all tests and the 95% confidence interval.
- Maximum and minimum value:
  The threshold for the maximum score formula (8) corresponds to the higher confidence interval value from the Horn Method.
- Standard deviation:
  Analysing the confidence interval, the higher value is too high to consider as a threshold, in fact all test are below the limit by far so, instead of this one, the average value of all standard deviations has been used (**17.7928**).

Bus stops and turns analysis:

To compare the bus stops of each test, two different methods have been used to obtain the thresholds for the 6 bus stop features obtained in the algorithms (break peak values, break window peak values, acceleration peak values, acceleration window peak values, distance values, distance window peak values).

- Method 1:
  Within the same sample test, all the stops of the test obtained from the bus stop algorithms have been compared. In this comparison, the 95% confidence interval from Horn has been obtained for the maximum peak values, minimum peak values and distance peak values individually.
  Moreover, this has been repeated for all test to extract all the confidence intervals.
  After this, and as the interval has a lower and a higher value (interval: X < u < Y, where X is the lower and Y the higher), the higher interval value of each test of each feature has been compared again to the other tests feature to obtain another time the 95% confidence interval of the feature.
  As is kind of difficult to explain, the feature maximum peak value has been used in order to explain this method with an example:
  As said, the bus stop algorithm returns all the maximum, minimum and distance values of the test. All these values have been compared separately with Horn to obtain the 95% confidence. Next is the table with the confidence interval for the maximum value of each test features and its confidence intervals:

Test	Pivot half sum	Pivot range	95% confidence interval
Test 1	0.2124	0.0849	0.1725 < u < 0.2522
Test 2	0.1577	0.0927	0.1066 < u < 0.2087
Test 3	0.2221	0.0966	0.1689 < u < 0.2753
Test 4	0.1811	0.0400	0.1536 < u < 0.2087
Test 5	0.1699	0.0468	0.1441 < u < 0.1957
Test 6	0.1538	0.0615	0.1071 < u < 0.2005
Test 7	0.1918	0.0849	0.1451 < u < 0.2386

Table 7: 95% confidence interval table for the maximum

After this, the higher interval value has been analysed again (underlined in the table) with the value in all the test, to do a second time Horn Method analysis to obtain another interval to act as a threshold:

Pivot Half Sum: 0.2420205078125
Pivot Range: 0.06664257812499999
Upper interval: **0.27867392578125**
Lower interval: 0.20536708984375002

With an upper interval of **0.2786**.

This upper interval value has set as the threshold for the first method maximum value feature, after the explanation of the second method is a table with all the thresholds.

- Method 2:

  Same as in method 1, within the same sample test, all the stops of the test obtained from the bus stop algorithms have been compared. In this comparison, the 95% confidence interval from Horn has been obtained for the maximum peak values, minimum peak values and distance peak values individually.

  After obtaining this interval again, the higher interval value of each test of each feature has been compared again to the other tests feature to obtain in this case an average value that will act as a threshold for that feature.

  To explain it in the example above, after getting the intervals in the table, all the values underlined have been compared to extract an average value (0.2256) that will act as the threshold for the maximum value

All the analytic development for method 1 and method 2 can be found in the annexed part.
In the next table is the comparison of both methods values.

	Method 1	**Method 2**	
Maximum value	0.2786	0.2256	**No window frame**
Minimum value	-0.2091	-0.1895	
Distance value	0.4805	0.4104	
Maximum value	0.1873	0.1467	**Window frame**
Minimum value	-0.1558	-0.1391	
Distance value	0.3376	0.2835	

Table 8: Method 1 and Method 2 threshold results.

Comparing and analyzing both methods with the stop features data, it can be concluded that using method 1 as thresholds, the values are too high to consider aggressive driving, letting all stop features values out of the aggressive driving, and contrary to this, method 2 establishes accurately the thresholds for the driving style.
And regarding to the analysed turn data, as said, it will only be studied the window mean lateral data.
Appling the Horn Method analysis to the data:

- Pivot Half Sum: 0.1836
- Pivot Range: 0.0529
- 95% confidence interval: 0.1545 < u < **0.2050**

The upper interval value has been set a threshold for the turn score formula, in fact, this value will act as a threshold for right and left turns (its absolute value).

4.2. Score scale election

As the score method returns a numerical value, a scale has to be set to distinguish between efficient, normal and aggressive driving. To do this, a simulation to represent the value of the features to decide when the score is efficient and when is aggressive have been made. As before, the two methods above have been used to do the simulation adding one characteristic:
Method 1:
In method 1, the higher 95% confidence interval value that act as the threshold for the method score, will act as the simulating value for the aggressive driving limit, moreover, another value has been used in order to act as the simulating value for the efficient driving. As before, is difficult to explain, that is why, the same example as before of the maximum peak value will be useful to explain the method for the efficient driving:
Now comparing the table of the example with the confidence interval for the maximum value of each test features:

Test	Pivot half sum	Pivot range	95% confidence interval
Test 1	0.2124	0.0849	**0.1725** < u < 0.2522
Test 2	0.1577	0.0927	**0.1066** < u < 0.2087
Test 3	0.2221	0.0966	**0.1689** < u < 0.2753
Test 4	0.1811	0.0400	**0.1536** < u < 0.2087
Test 5	0.1699	0.0468	**0.1441** < u < 0.1957
Test 6	0.1538	0.0615	**0.1071** < u < 0.2005
Test 7	0.1918	0.0849	**0.1451** < u < 0.2386

Table 9: 95% confidence interval table for the maximum

To extract the efficient driving simulation value, a second time Horn Method has been done with all the lower interval value (in **bold**) to extract this value. After finding this interval, the higher interval value has been used as the simulating value.
After repeating this process for all the features, and after setting the features values for the simulated ones above for both cases (aggressive and efficient) we get two values that as said, as a limits.
The scale proposed applying this method is:

AGRESIVE DRIVER:
1217.2062 < X
NORMAL DRIVER
1217.2062 < X < 1363.3508
EFFICIENT DRIVER
X > 1363.3508

Second method:
This method applies the same as the second method from the threshold election. But in this case and as before, we take the average value from the higher and lower values from the 95% confidence interval. The higher average value will simulate the aggressive value, and the lower average value will simulate the efficient driving.
All the explication is summarized in the annexed part.
The scale proposed applying this method is:

AGGRESSIVE DRIVER:
1085.0686 < X
NORMAL DRIVER
1085.0686 < X < 1287.9977
EFFICIENT DRIVER
X > 1287.9977

Comparison of method 1 and method 2 score scale election
Test 1 to 7 in addition of two new tests are being compared.

Test	**Method 1**	**Method 2**
Test 1	1108.6	1108.6
Test 2	1367.39	1367.39
Test 3	1162.21	1162.21
Test 4	1238.31	1238.31
Test 5	1180.95	1180.95
Test 6	1182.25	1182.25
Test 7	1336.6	1336.6
Test 8 (new)	1304.46	1304.46
Test 9 (new)	1015.54	1015.54

Table 10: Comparison of tests 1 to 9 for the score scale election
Colours are related to the style of driving: aggressive, normal, efficient.
Method 1:
The values are extracted applying the Horn Method twice, unluckily, the first threshold is too high to consider as aggressive driving. Comparing these two thresholds with the confidence interval of the tests from the annexed, the values cannot be considered as aggressive, due to the fact that in test 1, 3 and 7 the value is closer to the lower value than the upper one.
Method 2:

The values are extracted applying the average of the Horn Method. Again, comparing this threshold with the confidence interval of the tests from the annexed, the values can be considered as aggressive due to the fact that in all tests the value is closer to the higher interval value than to the lower. This are the intervals for the driver classification:

AGGRESSIVE DRIVER: 1085.0686 < X
NORMAL DRIVER: 1085.0686 < X < 1287.9977
EFFICIENT DRIVER: X > 1287.9977

4.3. User interface and monitoring tools

As said in the methodology part, we can distinguish two monitoring tools to visualize the data.

- User Interface in Node-RED
  It is a monitoring tool that is very easy to interact. Apart from monitoring all the data from the different tests storage in the different SQLite data bases it allows to create a new test and visualize the features directly from it.
- Monitoring tool in Grafana
  It is a monitoring tool that helps visualize all the data storage in influx data base.

4.3.1. User Interface (UI) in Node-RED

This User Interface is made using JavaScript and node-red dashboard nodes which provide a comprehensive set of UI components for building dashboards for the Internet of Things (IoT).
This UI offers different type of graphics, feature results, driving style and overall monitoring view for all the data. Moreover, this UI is easily accessible with a smartphone or with a computer due to the fact that it can be open with a web browser. Moreover, as this project has been developed in a private network, the UI can be open within this private network (http://@IP:1880/ui/) but it can easily be in a public network for a global utilization. The access of the User Interface is free without log-in if a web browser is used in the private network.
The UI provides three different tabs:

- Test information tab:
  In this first tab, the user can explore the data from tests 1 to 7 which are the ones that have been used for the analysis. In it 10 different features are exposed to make it easier to interact and understand the style of driving in the platform. Moreover, as seen above in the figure, you can introduce a test, if this test is not between 1 and 7 it appears a message to introduce a value between 1 and 7.
    - Lateral monitoring: A monitoring view for the lateral acceleration data from the window mean filter. As the vibrations of the bus do not provide a useful information, the window mean must be used.
    - Longitudinal monitoring: A monitoring view for the longitudinal acceleration data.
    - Longitudinal window mean monitoring: A monitoring view for the longitudinal acceleration data from the window mean filter.
    - Longitudinal Histogram: A histogram for the longitudinal acceleration data.

- Score value obtained using the score method explained in the methodology part. This score is used for style of driving election with the scale proposed above.
- Driving style: The driving style can be aggressive, normal and efficient.
- Comments: In this part, depending on the driving, there can be seen the different features that are above the threshold limit.
- Mean, maximum peak and minimum peak values from the data test.

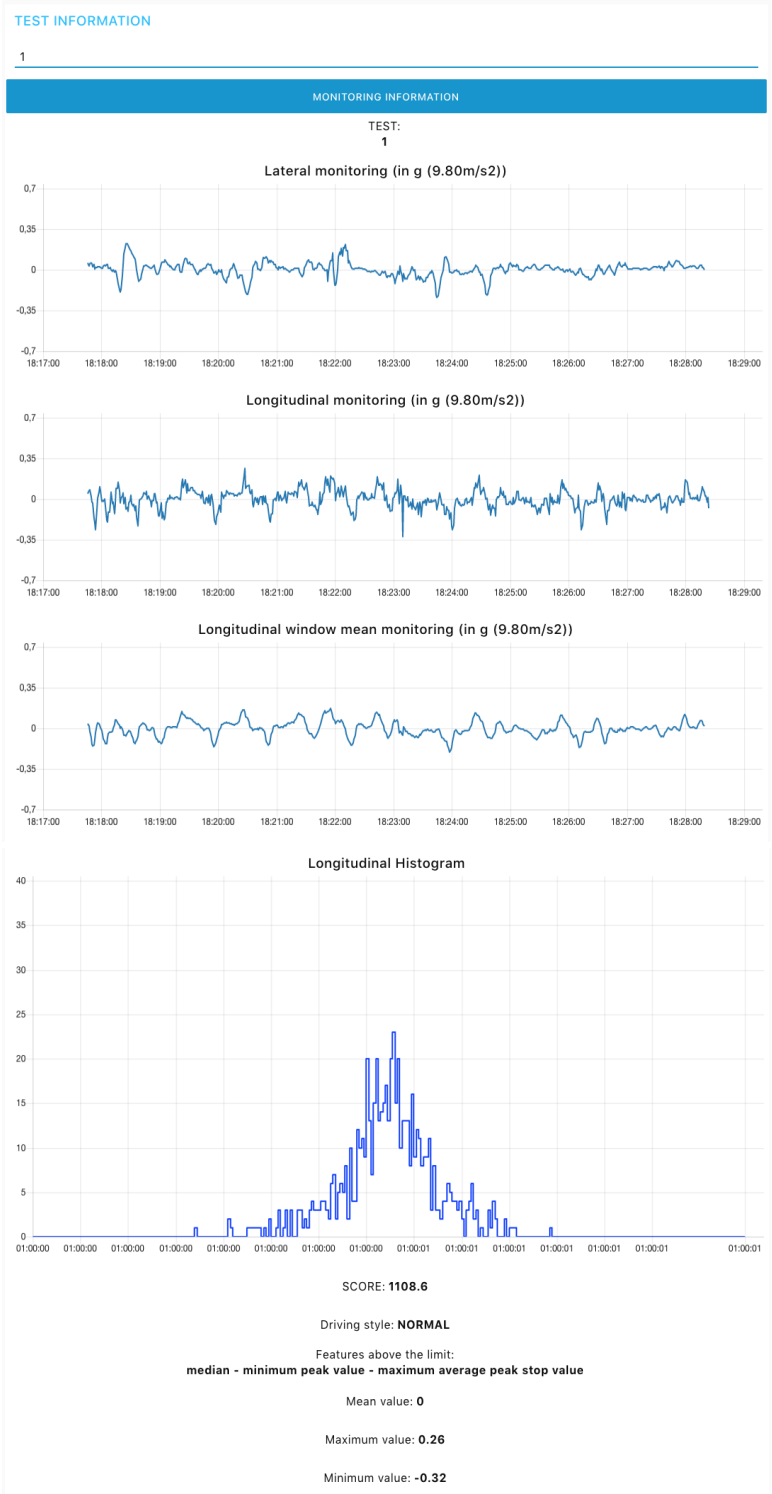
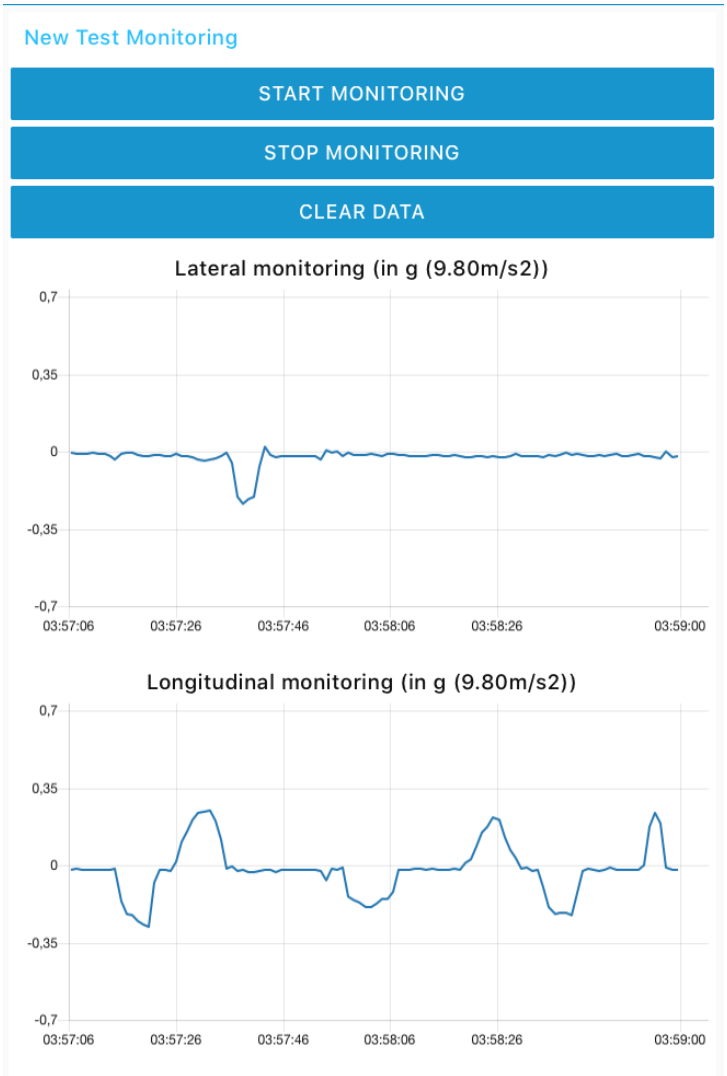


Figure 18: UI, first tab screenshot

- New test tab:
  In the second tab of the UI, the user can monitor a new test. This test is storage in a new SQLite database.
  In this tab, and as the one before, the user can see nine features:
  - Start monitoring button: To start recording data.
  - Stop monitoring button: To stop recording data.
  - Clear data: To clear the data that was before in the accelerometer.
  - Lateral monitoring: A monitoring view for the lateral acceleration data.
  - Longitudinal monitoring: A monitoring view for the longitudinal acceleration data.
  - Longitudinal Histogram: A histogram for the longitudinal acceleration data.
  - Score value obtained using the score method.
  - Driving style: The driving style can be aggressive, normal and efficient.
  - Comments: In this part, depending on the driving, there can be seen the different features that are above the threshold limit.
  - Mean, maximum peak and minimum peak values from the data test.



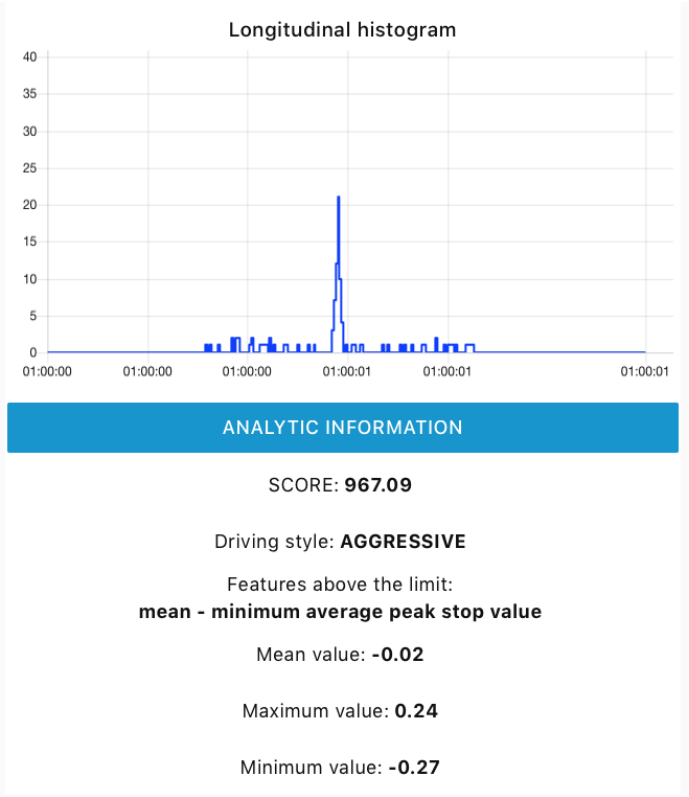


Figure 19: UI, second tab screenshot

- General information tab:
  In this tab, all the tests used for the study are shown. In it, the data is divided into longitudinal and lateral information.
  As the tab is very large, only a small screenshot is shown in the next figure.

Figure 20: UI, third tab screenshot

4.3.2. Monitoring tool in Grafana:

Another way to visualize the data is with Grafana software. With the help of influx data base and node-red it is possible to insert the data in Grafana. This data is divided in 5 different dashboards:

- General: In it can be found the data with and without the window mean from every test overlapped (with the help of the filter it is possible to compare tests).

Figure 21: Grafana: general dashboard

- Tests:
  In tests can be found the same information as in the general dashboard but one by one. With this we can compare more easily the data from each test.

Figure 22: Grafana test dashboard

- Histogram:
  The histogram of each test overlapped, as in the general dashboard we can filter where we want.

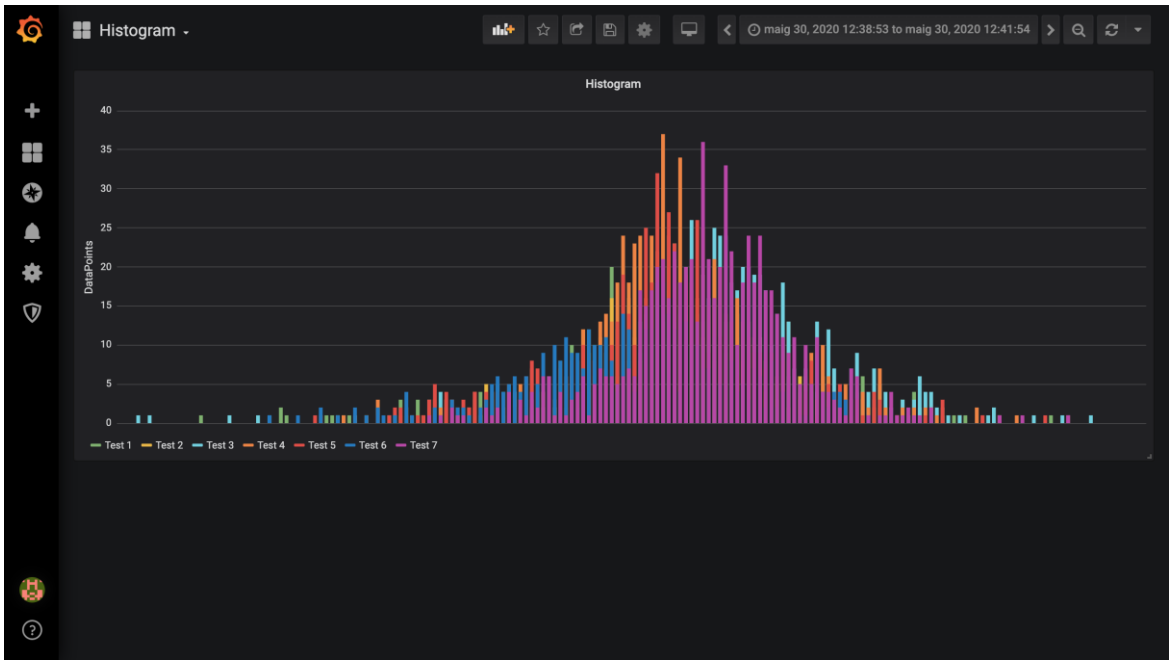


Figure 23: Grafana histogram dashboard

- Test histogram:
  In test histogram can be found the same information as in the histogram dashboard but one by one. With this we can compare more easily the data from each test

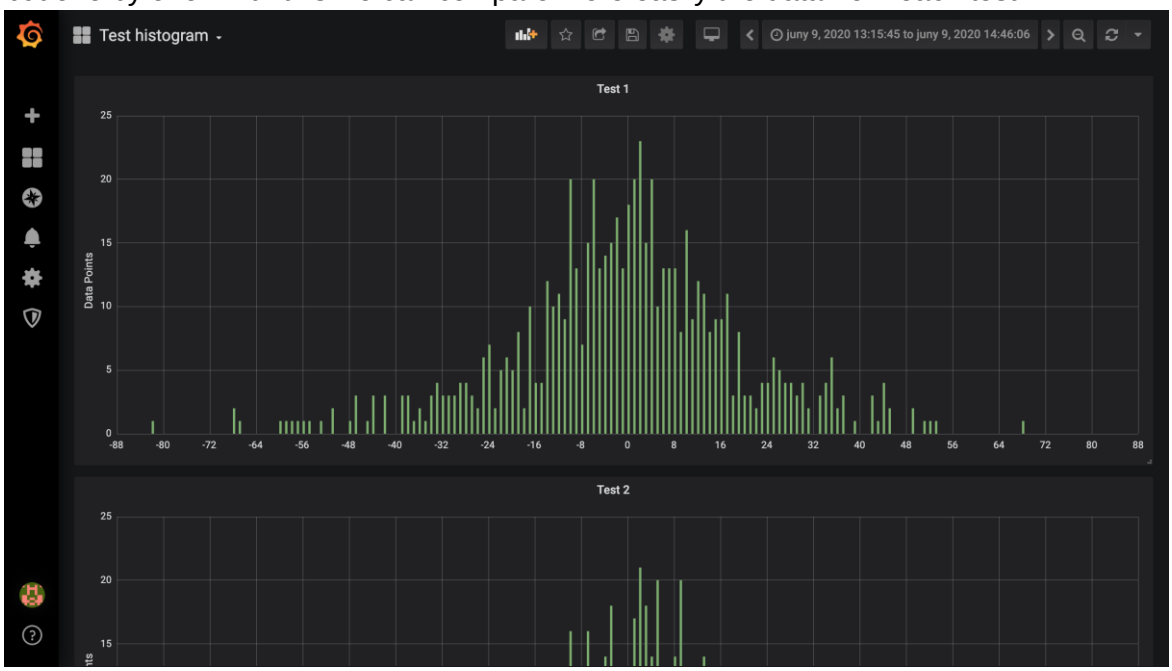


Figure 24: Grafana: test histogram dashboard

- New test data: Real time monitoring for the new data started in the UI. This monitoring can be in real time and it is storage in the influx database as well as in SQLite database. As seen, the dashboard offers the lateral and longitudinal information and in addition the histogram in a real time.

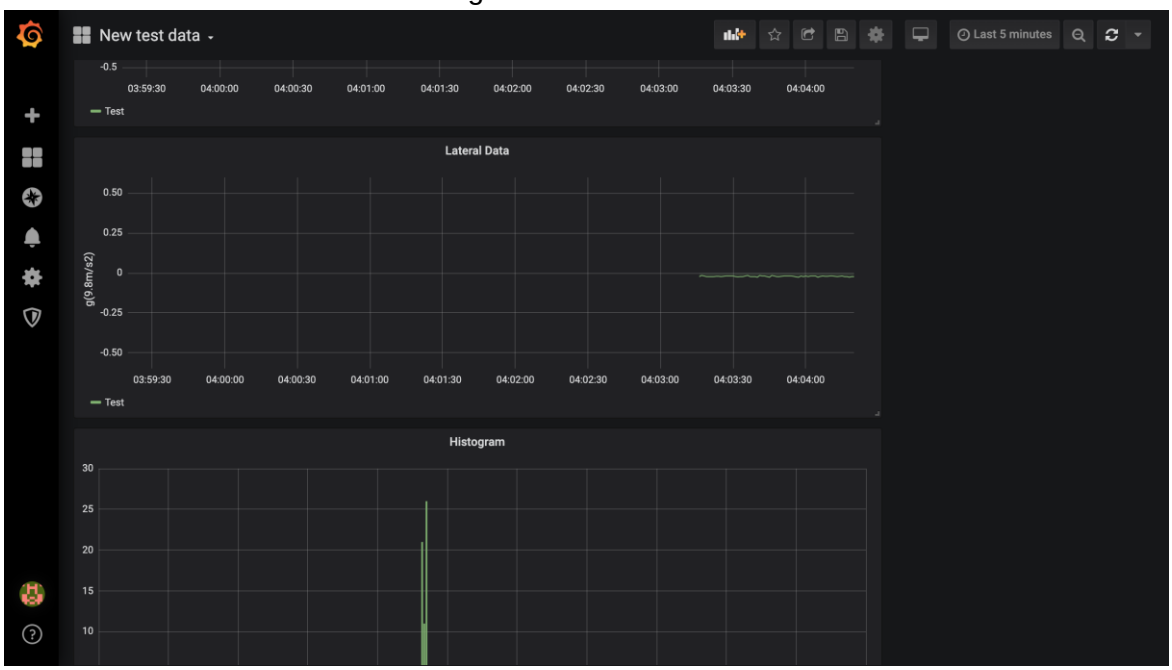


Figure 25: Grafana: new test data dashboard

Also the Grafana dashboards can be visualized within the private network (http://@IP:3000/) but it can easily be in a public network for a global utilization. For the access to the Grafana dashboards it is need a user and a password in order to visualize the data, in this case only *usr: admin* and *pw: admin* have been settled.

4.4. Results comparison

A comparison with a style of driving project (Android Smartphone Application for Driving Style Recognition by Radoslav Stoichkov [5]) has been done in order to find out if the system proposed in this project is more reliable than others.

As said in the state of art, the application presented in that work shows how the driving style recognition can be achieved using the smartphone sensors, specifically using an accelerometer (x and y axis) and sensor fusion (pitch and roll).

The system provided in the compared project only takes into account accelerations, breaks and turn analysis generally, without taking into account if the bus is stopping or is a hard acceleration, the main difference between this and the system proposed in this thesis is that the data compared for the recognition takes into account if the bus is stopping in a bus stop or not, in order to reduce the bad comparison between for example a bus stop and a transit stop. Moreover, the compared project does not take into account central characteristics neither standard deviation, that according to researched projects, these features provides the 98% accuracy for the style of driving if the data is totally repeatable, still useful if the data is not repeatable due to the fact that provides a full data analysis.

Finally, and talking about statistical analysis, the compared project does not provide a statistical analysis to find the thresholds, on the contrary in this project Horn Method

statistics have been used in order to find the different thresholds for the score method and for the scale, which is the correct way to detect the accelerations.

In conclusion, as the system proposed in this project takes into account not only accelerations, breaks and turns but also an entire analysis for all test data (with central characteristics, standard deviation and maximum and minimum values) the result is more accurate and reliable than previous projects.

5. Budget

In this chapter all project costs including price of components list (prototype) used as well as indicative salaries of a junior engineer are provided.

Regarding to the prototype, only hardware components and a 10% of contingency have taken into account without the software license because all software used is open source and in the public domain.

Total price for the components:

MATERIAL			
Product	Price/u [€]	Total nº of product	Total price [€]
Raspberry Pi	50	1	50
SD card (64GB)	9,79	1	9,79
GY-801 sensor	12,69	1	12,69
Wires	0,5	4	2
10 % contingency			7,45
TOTAL PRICE			**81,93**

Table 11: Components total price

Salary in function of developed work:

SALARY			
Work	Price/hour [€]	Total hours	Total price [€]
Salary of a junior engineer	9	540	4860
Social security	33% of net salary		1603.8
TOTAL PRICE			**6463.8**

Table 12: Salary total price

6. Environment Impact

A booming economy and increasing urbanization are leading to increasing utilization of transport, and worldwide traffic density contributes largely to the emission of greenhouse gases and air pollutants, releasing carbon dioxide, methane, nitrous oxide, sulphur dioxide, nitrous gases, carbon monoxide, volatile organic compounds and particulates. Over millions of years, the earth has developed a natural greenhouse effect which is caused by emission of water vapor and the corresponding gases, which prevents the Earth from cooling down. Most of this greenhouse effect is the result of carbon dioxide released through the combustion of fossil fuels (petroleum, coal and gas) as well as deforestation, soil erosion and wood burning.

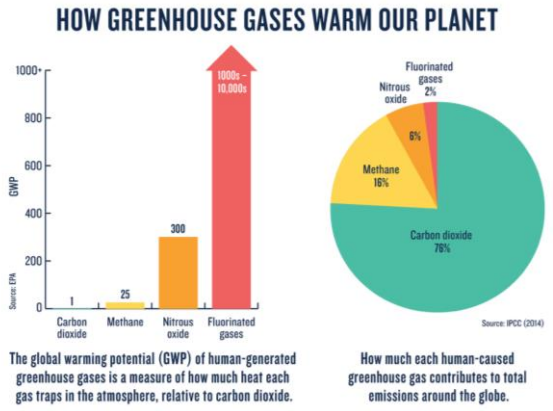


Figure 26: GreenHouse effect [25]

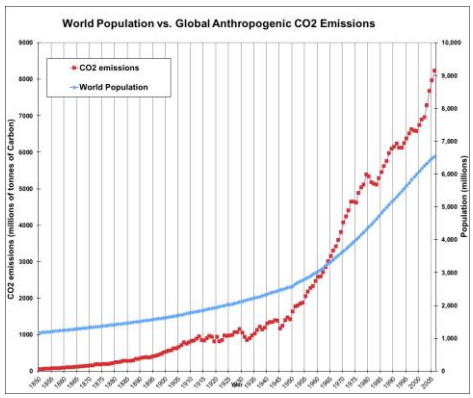


Figure 27: Population increasing and CO2.

Due to this increasing urbanization, traffic in the cities is growing heavily, more than ever, and one portion of the citizens use their own car to move but in fact, another portion of the people are using public transport, and nowadays it is becoming more popular than ever.

In these days, public transport is becoming more and more sustainable due to the fact that most of the transport is powered by electric engine and most of the cities have a big infrastructure to cooper with the climate change, but, something that needs to be taken into consideration is that this sustainability comes with a big amount of money as inversion to reduce emissions.

Therefore, the task that takes this thesis in the actual world situation is to reduce the emissions from fuel-engine public buses without need of changing to an electric motor. This is an important feature due to the fact that there is no need of inversion, using only a small system which cost does not exceed the 100€.

7. Conclusions and future development:

As stated in the introduction and as it has been observed throughout the thesis, this project shows how the driving style can be estimated using only an accelerometer sensor and statistical features. With the help of Horn Method, a classification of styles of driving based on the driver behavior has been achieved.

Moreover, this classification concludes that public transport fuel consumption optimization and efficiency in driving exists without need of a big inversion, and it can be achieve using only an accelerometer sensor that is capable to capture driving style and driving behavior information.

In addition, previous works obtain less reliable results due to the fact that only bus stops features were considered to recognize the style of driving. With the system proposed in this thesis apart from these features, central characteristics as well as different type of measurements have been taken into account for the recognition.

Also, the research behind this project shows how public transport plays an important role in the actual environmental situation. Pollution in cities comes mostly from fuel engine vehicles, mostly from cars. If there is an increasing use of public transport, is possible to reduce this pollution in cities and consequently, this must come with an efficient driving style to reduce fuel consumption and increase safety in the public transport.

Relating to difficulties in the project it must be taking into account that even though the tests were performed in the same season, same hours and on the same route, it was not easy to have a separable data, that is why algorithms and scripts have been programmed and used in order to have comparable data. Moreover, the pandemic (covid-19) has taken a big role in the project due to the fact that the first idea was to monitor car movements. However, changing the topic has been a great idea due to the fact that most of the works related to styles of driving using accelerometer are aimed at cars and not bus, so it has been a great opportunity to create a system useful for future implementations.

These future implementations can come with several improvements based on new sensors implementations such as a GPS to detect speed and distance in various areas and to make data repeatable without need of scripts. Moreover, for the safety and to detect distance between the bus and a vehicle, a camera could be used. Therefore, using a proper dynamic product formed by many sensors would provide more accurate and reliable results.

Bibliography:

[1] National Highway Traffic Safety Administration (NHTSA), "*Aggressive Driving",* https://one.nhtsa.gov/people/injury/research/aggdrivingenf/pages/introduction.html
[2] Government of Canada, "Fuel-efficient driving techniques", https://www.nrcan.gc.ca/energy-efficiency/energy-efficiency-transportation/personal-vehicles/fuel-efficient-driving-techniques/21038 [Date modified: 2020-04-23]
[3] Roberto Garcia, Gabriel Diaz, Xabiel G. Pañeda, Alejandro G. Tuero, Laura Pozueco, David Melendi, Jose A. Sanchez, victor Corcoba and Alejandro G. Pañeda, "Impact of Efficient Driving in Professional Bus Fleets" 2017, Spain, pp. 1-50
[4] Accelerometer Meter App for Android Smartphones, in Google Play: https://play.google.com/store/apps/details?id=com.keuwl.accelerometer&hl=ca
[5] *Radoslav Stoichkov*, "Android Smartphone Application for Driving Style Recognision", 2013 Department of Electrical Engineering and Information Technology, Germany
[6] BMW M Power Meter for iPhone, iPod touch, and iPad on the iTunes App Store: https://downloads.tomsguide.com/BMW-M-Power-Meter,0301-53905.html
[7] L. Pozueco et al., "Adaptive learning for efficient driving in urban public transport," 2015 International Conference on Computer, Information and Telecommunication Systems (CITS), Gijon, 2015, pp. 1-5, 0.1109/CITS.2015.7297724
[8] Bartosz Jachimczyk, Damian Dziak, Jacek Czapla, Pawel Damps, and Wlodek J. Kulesza, "IoT On-Board Sytem for Driving Style Assessment", 2018, 10.3390/s18041233
[9] Minh Van Ly, Sujitha Martin and Mohan M. Trivedi et al.,"Driver Classification and Driving Style Recognition using Inertial Sensors", 2013 IEEE Intelligent Vehicles Symposium (IV) June 23-26, 2013, Gold Coast, Australia, pp 1040 – 1045
[10] Juffrizal Karjanto, Nidzamuddin Md. Yusof, Jacques Terken, Frank Delbressine, Muhammad Zahir Hassan, and Matthias Rauterberg, "Simulating autonomous driving styles: Accelerations for three road profiles" AiGEV 2016, Department of Industrial Design, Eindhoven University of Technology, Eindhoven, The Netherlands, pp1-16
[11] V. Vaitkus, P. Lengvenis and G. Žylius, "Driving style classification using long-term accelerometer information," 2014 19th International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, 2014, pp. 641-644, doi: 10.1109/MMAR.2014.6957429JJ
[12] *Per-Erik Nordström. Scania Fahrer Eco-Modul -– Wirtschaftlichkeit mit eigenem TraineranBord.* http://www.scania.de/images/P09903DE2009
[13] Volvo Technik-Lexikon - Technik von A-Z. d.aspx. https://www.volvocars.com/py/support/manuals/xc60/2019-early/apoyo-del-conductor/driver-alert-control/driver-alert-control
[14] Raspberry Pi 3 Model B Specifications in .pdf: https://www.terraelectronica.ru/pdf/show?pdf_file=%252Fds%252Fpdf%252FT%252FTechicRP3.pdf
[15] GY-801 module datasheet in .html: https://www.plexishop.it/en/10-axis-gy-801-module-gyroscope-accelerometer-magnetometer-pressure-sensor.html
[16] I2C tutorial and explication: https://learn.sparkfun.com/tutorials/i2c/all
[17] Michael Sheng, Yongrui Qin, Lina Yao, Boualem Benatallah "Managing the Web of Things: Linking the Real World to the Web", 2017, Australia, pp. 77:
[18] JavaScript document in Mozilla developer organization: https://developer.mozilla.org/es/docs/Web/JavaScript

[19] Electromagnetic shielding properties of woven fabrics made from high-performance fibers by *Veronika Safarova and Jiri Militky*
[20] I2C node-red library https://flows.nodered.org/node/node-red-contrib-i2c
[21] SQLite node-red library https://flows.nodered.org/node/node-red-node-sqlite
[22] ADXL345 sensor datasheet in PDF from sparkfun: https://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf
[23] Dashboard node-red library https://flows.nodered.org/node/node-red-dashboard
[24] InfluxDB node-red library https://flows.nodered.org/node/node-red-contrib-influxdb
[25] https://www.nrdc.org/stories/greenhouse-effect-101
[26] SSH Protocol Procedure http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-en-4/ch-ssh.html

Appendices

Appendix 1: Work plan, milestones and Gantt diagram

Work plan

WP: Node-RED introduction	WP ref: 1	
Major constituent: SW	Sheet 1 of 11	
Short description: Get used with the SW tool (Node-RED) for control both Raspberry Pi and accelerometer module.	Planned start date: 25/02/2020 Planned end date: 03/03/2020	
	Start event: 18/02/2020 End event: 08/03/2020	
Internal task T1: Node-RED communication with Raspberry Pi Internal task T2: Communication between Raspberry Pi and Accelerometer module.	Deliverables: None	Dates:

WP: PC and Raspberry Pi communication	WP ref: 2	
Major constituent: HW and SW	Sheet 2 of 11	
Short description: With a private network stablish a SSH connection to control Raspberry Pi from PC.	Planned start date: 09/03/2020 Planned end date: 11/03/2020	
	Start event: 08/03/2020 End event:	
Internal task T1: PC – Raspberry Pi communication.	Deliverables: None	Dates: 20/03/2020

WP: Laboratory adaptation at home	WP ref: 3	
Major constituent: HW and SW	Sheet 3 of 11	
Short description: Recreate the scenario from laboratory at home (due to covid-19 restrictions).	Planned start date: 11/03/2020 Planned end date: 20/03/2020	
	Start event: 08/03/2020 End event:	
Internal task T1: Laboratory adaptation.	Deliverables: None	Dates: 20/03/2020

WP: Raspberry Pi and Accelerometer Communication	WP ref: 4
Major constituent: HW and SW	Sheet 4 of 11
Short description: Establish the communication between Raspberry and Accelerometer.	Planned start date: 03/03/2020 Planned end date: 25/03/2020
	Start event: 08/03/2020 End event:

Internal task T1: HW communication. Internal task T2: SW communication.	Deliverables: Report	Dates: 25/03/2020

WP: Graphic representation of the measurements.	WP ref: 5	
Major constituent: SW	Sheet 5 of 11	
Short description: Create graphics to represent and plot the three-axis accelerometer information.	Planned start date: 25/03/2020 Planned end date: 15/04/2020	
	Start event: End event:	
Internal task T1: Create graphics to represent the 3-axis accelerometer information. Internal task T2: Test the final solution.	Deliverables: Results	Dates: 15/04/2020

WP: Data Analysis	WP ref: 6	
Major constituent: SW	Sheet 6 of 11	
Short description: 7 tests analysis for the accelerometer data and its statistical features.	Planned start date: 15/04/2020 Planned end date: 30/04/2020	
	Start event: End event:	
Internal task T1: Perform 7 different tests. Internal task T2: Present its statistical features	Deliverables: Results Report	Dates: 20/04/2020 30/04/2020

WP: Propose a method and scale for the style of driving classification	WP ref: 7	
Major constituent: SW	Sheet 7 of 11	
Short description: Using statistical analysis and using Horn Method analytics propose a method based on a score for the style of driving classification. Propose the scale for this score.	Planned start date: 01/05/2020 Planned end date: 10/05/2020	
	Start event: End event:	
Internal task T1: Discussion for a method classification. Internal task T2: Method proposal. Internal task T3: Propose the scale for the driving style.	Deliverables: Report	Dates: 01/05/2020

WP: User Interface	WP ref: 8	
Major constituent: SW	Sheet 8 of 11	
Short description: Development of a User Interface based in the node-red dashboard.	Planned start date: 10/05/2020 Planned end date: 20/05/2020	
	Start event: End event:	
Internal task T1: Development of a User Interface	Deliverables: Results	Dates: 10/05/2020

WP: Grafana software interface for monitoring system	WP ref: 9	
Major constituent: SW	Sheet 9 of 11	
Short description: Use and program a Grafana software interface to monitor the bus driving.	Planned start date: 20/05/2020 Planned end date: 30/05/2020	
	Start event: End event:	
Internal task T1: Program a Grafana software interface	Deliverables: Results	Dates: 10/05/2020

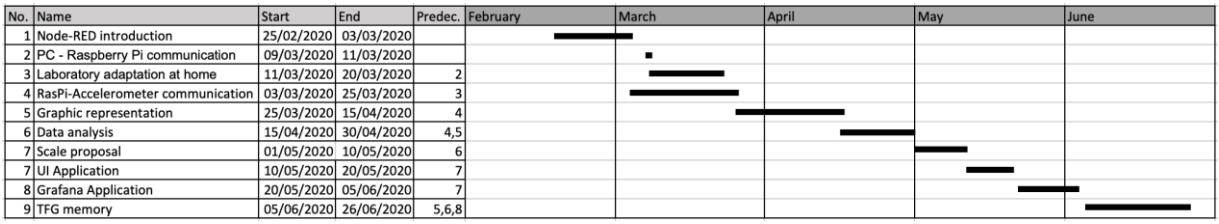
WP: Test a final solution	WP ref: 10	
Major constituent: HW and SW	Sheet 10 of 11	
Short description: Test the final solution.	Planned start date: 30/05/2020 Planned end date: 05/06/2020	
	Start event: End event:	
	Deliverables: Results Report	Dates: 27/05/2020 05/05/2020

WP: TFG Memory	WP ref: 11	
Major constituent: SW and study	Sheet 11 of 11	
Short description: Realization of the thesis final report	Planned start date: 05/06/2020 Planned end date: 23/06/2020	
	Start event: End event:	
Internal task T1: Memory of final project	Deliverables: Results	Dates: 15/06/2020

Milestones

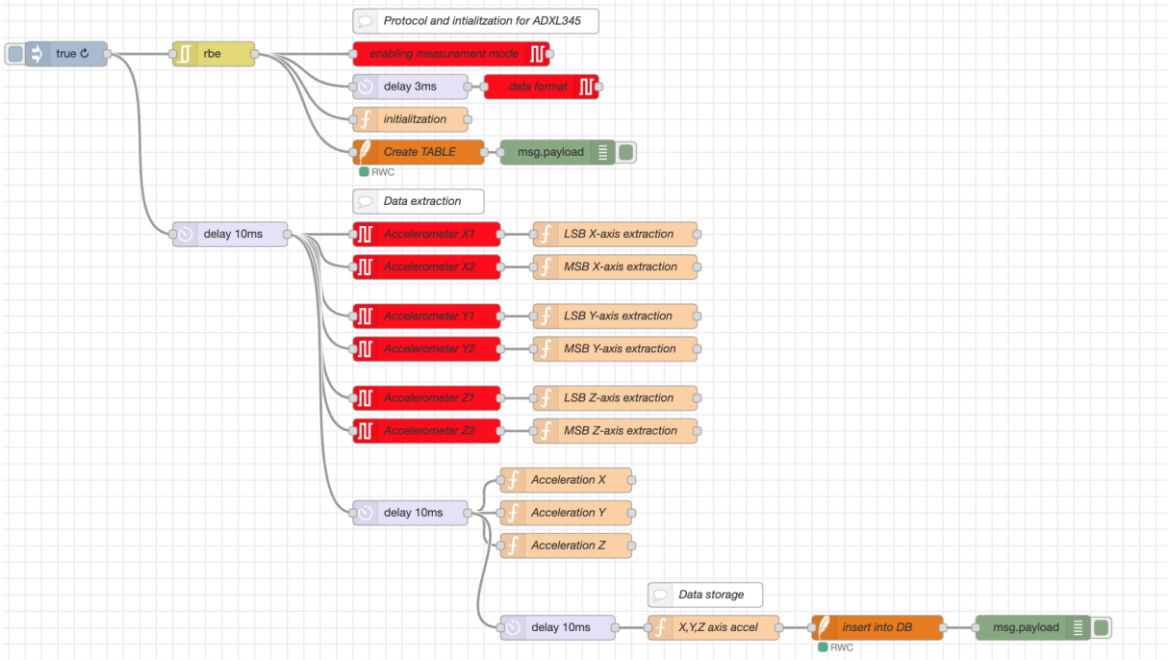
WP#	Task#	Short title	Milestone / deliverable	Date (week)
WP4	1	Raspberry Pi and Accelerometer HW communication	None	25/03/2020
WP4	1	Raspberry Pi and Accelerometer SW communication	Results	25/03/2020
WP5	1	Graphic Representation	Results	15/04/2020
WP6	1	Accelerometer scale proposal	Report	10/04/2020
WP9	2	UI Application	Results	20/05/2020
WP10	1	Grafana Application	Results	05/06/2020
WP11	1	TFG memory	Report	26/06/2020

Gantt



Appendix 2: Node-RED Sources

Data extraction flow



- Initialization code from protocol and initialization part:

```
//X-axis initialitzations (lateral accelerometer)
var accel_X = global.get("globalX");
var v_X = 0;
global.set("v_X", 0);
var v0_X = 0;
global.set("v0_X", 0);
//Y-axis initialitzations (longitudinal accelerometer)
var accel_Y = global.get("globalY");
var v_Y = 0;
global.set("v_Y", 0);
var v0_Y = 0;
global.set("v0_Y", 0);
//Z-axis initialitzations (vertical accelerometer)
var accel_Z = global.get("globalZ");
var v_Z = 0;
global.set("v_Z", 0);
var v0_Z = 0;
global.set("v0_Z", 0);
var i = 0;
global.set("i", 1);
var vmax;
global.set("vmax", 0);
//counter initialization
var cont;
global.set("cont", 0);

return msg;
```

- Data extraction: Accelerometer X1 LSB X-axis extraction act like a model for the rest LSB and MSB extractions, all the extractions are the same.

- Bus Address: 83 (0x53) //Accelerometer sensor ADXL345
- Command: 50 (0x32) // LBS X-axis
- Bytes: 1 //We only expect one-byte information for each command
  This simple flow returns the position of the LBS X-axis.

As reported in the ADXL345 datasheet we found that each axis has 2 bytes, one for LSB (Less Significant Bit) and one for MSB (Most Significant Bit), and so, each byte is represented in a different command:

- X-axis:
  - LSB: 0x32 or 50 (in decimal)
  - MSB: 0x33 or 51 (in decimal)
- Y-axis:
  - LSB: 0x34 or 52 (in decimal)
  - MSB: 0x35 or 53 (in decimal)
- Z-axis:
  - LSB: 0x36 or 54 (in decimal)
  - MSB: 0x37 or 55 (in decimal)

***I express it in decimal because the I2C library works in decimal notation instead of hexadecimal.

For this application we will be using i2c in node, in which we request the information to the ADXL345.

Where accelerometer X1 is the i2c in node with the LSB of the X-axis:

In the function next to the i2c node, we put the LSB or MSB byte as a variables global:

```
var X0; //Declare the variable.
global.set("X0",msg.payload); //Info from the sensor --> X0
return msg;
```

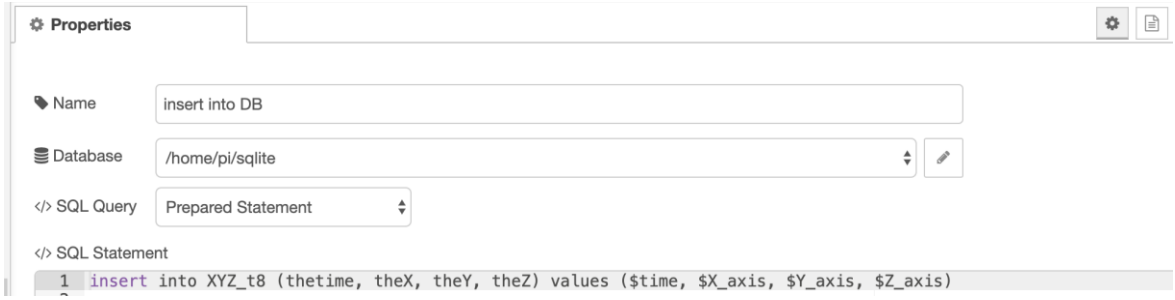
Now that we have our global variables, in the function nodes of Acceleration X,Y,Z the MSB and LSB are putted together in one:

```
var data0 = global.get("X0"); //information LSB
var data1 = global.get("X1"); //information MSB
var globalX;
var resultat = ((data1) * 256) + data0; //LSB and MSB together (256d = 100000000 we move MSB to another bit to not collapse with the LSB)
//the first bit is used as negative (1) and posivite (0) indicator
if(resultat > 32767){ //If the value exceed 32767 (2^15), means that the first bit is 1 and in fact is a negative value
    resultat -= 65536 //Recalculated as negative
}
var accel_X = resultat*0.00390625; //0.00390625 g/bit
global.set("accel_X",accel_X); //we set the acceleration as a global variable
msg.payload= global.get("accel_X");
return msg;
```

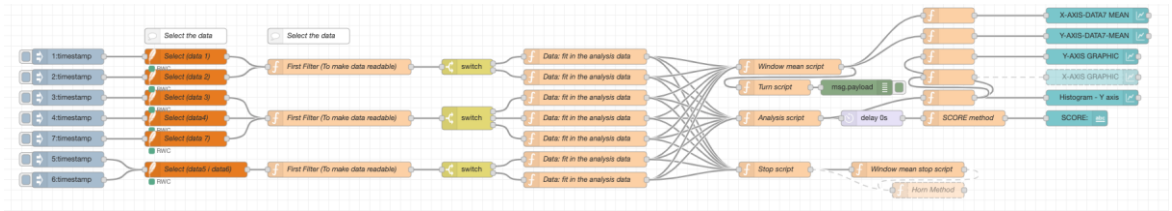
And finally for the data storage, we need to put these values in the *msg.params* which are equivalent to the parameters field from the message:

```
msg.params = {$time:Date.now(), $X_axis:global.get("accel_X"), $Y_axis:global.get("accel_Y"), $Z_axis:global.get("accel_Z") }
return msg;

```

After this the data is introduced in the SQLite data base using the SQLite node:



- Data processing and analysing flow:



- First filter code:

```
var bool = true; //bool variable is used to know if the bus is stopped
var dataX, dataY, dataZ, time;
var sum = 0;
var corrector = 0; //variable to correct the unuseful data
var canvi = true; //if the data is formed by two tests, this can separate this tests
var data1 = [{}]; //new variable for the data 1
var data2 = [{}]; //new variable for the data 2
//first part of the data and last part are unuseful due to the fact that the accelerometer is positioning in a seat or being removed.
for (var i=3; i < msg.payload.length-85; i++) {
    if(i%2 === 0){ //as said, only half of the data is used, the other is to correct the unuseful data
        if((msg.payload[i+4].thetime < msg.payload[7].thetime) && bool){ //to correct and detect repeated data we use the time variable
            bool = false;
        }
        if(bool){ //once we know that the data is not repeated we proceed to save this data correctly
            dataX = msg.payload[i].theX;
            dataY = -msg.payload[i].theY;
            dataZ = msg.payload[i].theZ;
            time = msg.payload[i].thetime;
            corrector = i+1;
            //if both lateral and logitudinal exceeds at the same time 0.7g means that the data took a bad value
            while((dataX>0.7 || dataX<-0.7)&&(dataY>0.7 || dataY<-0.7)){
                dataX = msg.payload[corrector].theX;
                dataY = -msg.payload[corrector].theY;
                corrector++;
            }
            //if lateral or longitudinal exceeds 0.8g means that the data took a bad value
            while(dataX > 0.8 || dataX<-0.8){
                dataX = msg.payload[corrector].theX;
                corrector++;
            }
            while(dataY > 0.8 || dataY < -0.8){
                dataY = -msg.payload[corrector].theY;
                corrector++;
```

```
        }

        //we put all the data correctly
        data1[sum] = {"thetime":time, "theX":dataX, "theY":dataY, "theZ":dataZ};
        sum++;

    }
    if(!bool){ //second data, same proceadure
        if(canvi){
            if(i > 800){
                canvi = false;
                sum = 0;
            }
        }
        else{
            dataX = msg.payload[i].theX;
            dataY = -msg.payload[i].theY;
            dataZ = msg.payload[i].theZ;
            time = msg.payload[i].thetime;
            corrector = i+1;
            while((dataX>0.7 || dataX<-0.7)&&(dataY>0.7 || dataY<-0.7)){
                dataX = msg.payload[corrector].theX;
                dataY = -msg.payload[corrector].theY;
                corrector++;
            }
            while(dataX > 0.7 || dataX<-0.7){
                dataX = msg.payload[corrector].theX;
                corrector++;
            }
            while(dataY > 0.7 || dataY < -0.7){
                dataY = -msg.payload[corrector].theY;
                corrector++;
            }

            data2[sum] = {"thetime":time, "theX":dataX, "theY":dataY, "theZ":dataZ};
            sum++;
        }

    }
  }
}
global.set("data1", data1);
global.set("data2", data2);
msg.payload = data1;
return msg;
```

- Fit in the analysis data:
  Each data needs to be readjusted to match the other tests for the analysis to take into account the same stops, which means that all the data must start in the same route moment and finish in the route moment. And as the initialization of the raspberry may diverse for some seconds between tests, these needs to be readjusted to catch the same data in all tests.

```
var data = [{}];
for(var i=20; i < msg.payload.length-100; i++){
    data[i-20] = msg.payload[i];
}
msg.payload = data;
flow.set("filtered_data", data);
return msg;
```

- Second filter: Window mean filter code

```
var j = 0;
var m_X = 0;
var m_Y = 0;
var m_Z = 0;
var tempo;
var wf_data = [{}];
var window_n = 5;
for(var i = 0; i < msg.payload.length-window_n; i++){
    j = i;
    while((j-i) < window_n){ //every 5 data points, we do the mean
        m_X += msg.payload[j].theX;
        m_Y += msg.payload[j].theY;
        m_Z += msg.payload[j].theZ;
        j++;
    }
    m_X = m_X/window_n;
    m_Y = m_Y/window_n;
    m_Z = m_Z/window_n;
    tempo = msg.payload[i].thetime;
    wf_data[i] = {"thetime":tempo, "theX":m_X, "theY":m_Y, "theZ":m_Z}; //we save this data into a new msg
    m_X = 0;
    m_Y = 0;
    m_Z = 0;
}
global.set("wf_data", wf_data);
msg.payload = wf_data;
return msg;
```

- Analysis script code:

```
//variable initialization
var data_Y = Array.apply(null, new Array(2000)).map(Number.prototype.valueOf,0);
var datap_Y = Array.apply(null, new Array(1000)).map(Number.prototype.valueOf,0);
var datan_Y = Array.apply(null, new Array(1000)).map(Number.prototype.valueOf,0);
var dataY;
var data_X = Array.apply(null, new Array(2000)).map(Number.prototype.valueOf,0);
var datap_X = Array.apply(null, new Array(1000)).map(Number.prototype.valueOf,0);
var datan_X = Array.apply(null, new Array(1000)).map(Number.prototype.valueOf,0);
var dataX;
//third moment
var third_moment = 0;
//covariance function
var COV = context.global.get('cov'); //a function called cov has been used to calculate var, covar and st. dev
//mean
var mean_X=0;
var mean_Y=0;
//max
var max_X = 0;
var max_Y = 0;
//min
var min_X = 0;
var min_Y = 0;
//median
var medianX = [];
var medianY = [];
var median_X = 0;
var median_Y = 0;
var dif_zero_X = 0;
var dif_zero_Y = 0;
//moda
var moda_X = 0;
var moda_Y = 0;
var moda_int_X = 0;
var moda_int_Y = 0;
//standar deviation
var st_dev_X = 0;
var st_dev_Y = 0;
//mean deviation
var mean_dev_X = 0;
var mean_dev_Y = 0;
//Histogram data points
for (var i=0; i < msg.payload.length; i++) {
    dataY = msg.payload[i].theY;
    dataX = msg.payload[i].theX;
    dataX = dataX/0.00390625; //its easier to work with bits than with g because in bits the values are integers
    dataY = dataY/0.00390625;
    mean_Y += dataY; //mean
    mean_X += dataX;
    if(dataX !== 0){
            datan_X[-dataX]++;
    }
    if(dataY !== 0){
            datan_Y[-dataY]++;
    }
    datap_X[dataX]++;
    datap_Y[dataY]++;
    //Max and Min value
    if(max_X < dataX){
        max_X = dataX;
    }
    if(max_Y < dataY){
        max_Y = dataY;
    }
    if(min_X > dataX){
        min_X = dataX;
    }
    if(min_Y > dataY){
        min_Y = dataY;
    }
}
//readjustment on the 1000 for the histogram values
for(i = 1; i < 999; i++){
    data_X[1000-i] = datan_X[i];
    data_X[i+999] = datap_X[i];
    data_Y[1000-i] = datan_Y[i];
    data_Y[i+999] = datap_Y[i];
}

//median
for(i = 0; i < 2000; i++){
    if(data_X[i] !== 0){
        medianX[dif_zero_Y]= i;
        dif_zero_X++;
    }
    if(data_Y[i] !== 0){
        medianY[dif_zero_Y]= i;
        dif_zero_Y++;
    }
    if(data_X[i] > moda_X){
```

```
        moda_X = data_X[i];
        moda_int_X = i;
    }
    if(data_Y[i] > moda_Y){
        moda_Y = data_Y[i];
        moda_int_Y = i;
    }
    //standard deviation
    st_dev_X += (Math.abs(mean_X-data_X[i]))^2;
    st_dev_Y += (Math.abs(mean_Y-data_Y[i]))^2;
    //mean deviation
    mean_dev_X += Math.abs(mean_X-data_X[i]);
    mean_dev_Y += Math.abs(mean_Y-data_Y[i]);
}

st_dev_X = ((st_dev_X/i)^(1/2));
st_dev_Y = ((st_dev_Y/i)^(1/2));
mean_dev_X = (mean_dev_X/i);
mean_dev_Y = (mean_dev_Y/i);

if(medianX.length%2 === 0){
    median_X = (medianX[(medianX.length/2)]+ medianX[medianX.length/2+1])/2;
}
if(medianX.length%2 !== 0){
    median_X = medianX[(medianX.length/2-0.5)];
}
if(medianY.length%2 === 0){
    median_Y = (medianY[(medianY.length/2)+1]+ medianY[medianY.length/2+1])/2;
}
if(medianY.length%2 !== 0){
    median_Y = medianY[(medianY.length/2-0.5)];
}

moda_X = (moda_int_X-1000); //lateral moda (not used for the final analysis)
median_X = (median_X-1000); //lateral median
mean_X = mean_X/(msg.payload.length); //lateral mean
moda_Y = (moda_int_Y-1000); //longitudinal moda (not used for the final analysis)
median_Y = (median_Y-1000); //longitudinal median
mean_Y = mean_Y/(msg.payload.length); //longitudinal mean

//we set the variables as globals
global.set("mean_X", mean_X);
global.set("mean_Y", mean_Y);
global.set("max_X", max_X);
global.set("max_Y", max_Y);
global.set("min_X", min_X);
global.set("min_Y", min_Y);
global.set("median_X", median_X);
global.set("median_Y", median_Y);
global.set("moda_X", moda_X);
global.set("moda_Y", moda_Y);
global.set("st_dev_X", st_dev_X);
global.set("st_dev_Y", st_dev_Y);
global.set("mean_dev_X", mean_dev_X);
global.set("mean_dev_Y", mean_dev_Y);

var cov_hist = new COV(data_X, data_Y);
global.set("cov_hist", cov_hist);
```

```
var d_X = [];
var d_Y = [];
for(i = 0; i < msg.payload.length; i++){
    d_X[i] = msg.payload[i].theX/0.00390625;
    d_Y[i] = msg.payload[i].theY/0.00390625;

}
//setting the st.deviation, covariance and the third moment
var cov_n = new COV(d_X, d_Y);
st_dev_X = Math.pow(cov_n[0][0],1/2);
st_dev_Y = Math.pow(cov_n[1][1],1/2);
global.set("st_dev_X", st_dev_X);
global.set("st_dev_Y", st_dev_Y);
global.set("cov_n", cov_n);
for(i = 0; i < msg.payload.length; i++){
    dataY = msg.payload[i].theY;
    third_moment += Math.pow((dataY-mean_Y)/st_dev_Y, 3);
}
third_moment = third_moment/msg.payload.length;
global.set("third_moment", third_moment);
msg.payload = data_Y;
return msg;
```

As seen this script calculate and set all statistics as global variables. Starting with the histogram, as seen, we use the bit value instead of the g value because as it is an integer is easier to separate the points.
Moreover, as it cannot be negative values in a vector, a vector of 2000 columns has been used with the 1000 representing the 0.
Also, standard deviation and covariance have been calculated using the function COV, imported in the settings.js document.
At the end all the statistical features have been settled as global variables to use in all the flows.

- Stop script code:

```
var tempo = 0;
var mean = [[]];
var vec_max = [];
var dif = [];
var position = [[]];
var vec_max2 = [];
var stop_v = [];
var sum = 0;
const MAX = 0.07; //max for the stopped value
const MIN = -0.07; //min for the stopped value
const MAX_S = 0.08; //value for the acceleration detection
const MIN_S = -0.094 //value for the break detection
var b_stop = false; //bus stop boolean, when is true, the bus is with acceleration 0.
var v_max = 0;
var i_min = 0;
var v_min = 0;
var int_max = 0;
var int_min = 0;
var stop_mean = 0;
var stop_mean_v = [];
for(var i = 0; i < msg.payload.length-20; i++){
    if(msg.payload[i].theY < MIN_S){ //we detect the bus break
        b_stop = true;
        i_min = i;
    }
    if(msg.payload[i].theY > MAX_S){ //we detect the bus accelerating
        b_stop = false;
    }
    if(b_stop){ //once a break has been detected
        if(MIN < msg.payload[i].theY > MAX){ //if the value of the acceleration is between max and min means that the bus is with accel 0.
            tempo++; //every second that goes on
            stop_mean += msg.payload[i].theY; //to set the mean of the stop we will do the average of the entire stop.
        }
        else{   //sometimes can occur that the accelerometer is higher or lower, due to vibrations or bus semiaccelerations.
                //if this occurs 2 times, we will conclude that it is not a stop and we will set the tempo to 0.
            var falsa_alarma = 0;
            for(var k = i; k < i+4; k++){
                if(MIN < msg.payload[i].theY > MAX){
                    falsa_alarma++;
                }
            }
            if(falsa_alarma < 3){
                tempo = 0;
            }
        }
    }
    if((tempo > 7)&&!b_stop){   //if the tempo is higher or equal than 7 and a acceleration value has been detected, we will conclude
                                //that is a stop.
        stop_mean_v[sum] = stop_mean/tempo;
        tempo = 0;
        for(var j = i; j <= i+6; j++){
            if(msg.payload[j].theY > v_max){
                v_max = msg.payload[j].theY;
                int_max = j;
            }
        }
        for(j = i_min; j >= i_min-6; j--){
            if(msg.payload[j].theY < v_min){
                v_min = msg.payload[j].theY;
                int_min = j;
            }
        }
        mean[sum] = [msg.payload[int_min].thetime,msg.payload[int_max].thetime];
        position[sum] = [int_min, int_max];
        //to take the mean in the middle of the stop, we peak the 5 values between min and max
        var middle_stop = Math.trunc((int_max+int_min)/2);
        for(i = middle_stop-2; i < middle_stop+2; i++){
            stop_mean = msg.payload[i].theY;
        }
        //we save all the useful information
        stop_v[sum] = stop_mean/4; //mean
        dif[sum] = v_max - v_min;   //difference between max and min
        vec_max[sum] = [v_min-stop_v[sum],v_max-stop_v[sum]]; //max and min values taking into account the mean value when the bus is stopp
        vec_max2[sum] = [v_min,v_max];
        sum++;
        v_max = 0;
        v_min = 0;
        stop_mean = 0;
    }
    if(!b_stop){
        tempo = 0;
        v_max = 0;
        v_min = 0;
        stop_mean = 0;
    }
}
//we stablish as global all the variables
global.set("stop_mean_v", stop_mean_v);
global.set("maxs_stop", vec_max);
global.set("dif", dif);
msg.payload = [mean, vec_max, dif, position, stop_mean_v, stop_v, vec_max2];
return msg;
```

- Window mean script code:
  Using the stop script information, we can detect the same information from the window mean data. Next is the script that returns this information. We will introduce two nulls in the return message because we use the stop script information for the mean and the position. The return will contain the max and min value, the distance value and the stop mean value.

```
var wf_data = global.get("wf_data");
var min_wf_Y = 0;
var max_wf_Y = 0;
var vec_max = [[]];
var stop_mean_v = [];
var dif = [];
var pos = 0;
var stop_mean = 0;
var posmin = 0;
var posmax = 0;
var count_stop = 0;
for(var j = 0; j < msg.payload[3].length; j++){
    pos = msg.payload[3][j][0] - 15;
    while(msg.payload[3][j][1] + 5 > pos){
        if(msg.payload[3][j][0]+5 > pos){
            if(wf_data[pos].theY < min_wf_Y){
                min_wf_Y = wf_data[pos].theY;
            }
        }
        if(msg.payload[3][j][1]-5 < pos){
            if(wf_data[pos].theY > max_wf_Y){
                max_wf_Y = wf_data[pos].theY;
            }
        }
        if(msg.payload[3][j][0] < pos < msg.payload[3][j][1]){
            if(-0.05 < wf_data[pos].theY < 0.05){
                stop_mean += wf_data[pos].theY;
                count_stop++;
            }
        }
        pos++;
    }
    var middle_stop = Math.trunc(Math.trunc((msg.payload[3][j][0]+msg.payload[3][j][1])/2));
    for(i = middle_stop-1; i < middle_stop+2; i++){
            stop_mean = wf_data[i].theY;
        }
    stop_mean_v[j] = stop_mean/3;
    vec_max[j] = [min_wf_Y-stop_mean_v[j], max_wf_Y-stop_mean_v[j]];
    dif[j] = max_wf_Y - min_wf_Y;
    min_wf_Y = 0;
    max_wf_Y = 0;
    stop_mean = 0;
    count_stop = 0;
}
global.set("maxs_stop_w", vec_max);
global.set("dif_w", dif);
msg.payload = [null, vec_max, dif, null, stop_mean_v];
return msg;
```

- Turn script code:

```
var data = global.get("wf_data");
var count = 0;
const TURN_TH = 0.12; //threshold for the turn
var turn = [[]];
var min = 0;
var max = 0;
var k = 0;
var start, stop;
for(var i = 0; i < data.length; i++){
/*we will diferenciate the left turn and the right turn_
- the right turn will be negative acceleration
- the left turn will be positive acceleration*/
//right turn
    if(data[i].theX < -TURN_TH){
        if(i < data.length){
            start = data[i].thetime; //set the time when the turn is started
            while((data[i].theX < 0)&(i < data.length)){
                if(data[i].theX < 0){
                    count++;
                    stop = data[i].thetime; //set the time when the turn is finished
                    if(min > data[i].theX){ //find the maximum acceleration value of the turn
                        min = data[i].theX;
                    }
                }
                i++;
            }
            if(count > 8){
                turn[k] = [start, stop, min]; //start time, stop time and minimum accelerometer values are storage in the turn vector
                k++;
                min = 0;
            }
            count = 0;
        }
```

```
    }
    if(data[i].theX > TURN_TH){
        if(i < data.length){
            start = data[i].thetime; //set the time when the turn is started
            while((data[i].theX > 0)&(i < data.length)){
                if(data[i].theX > 0){
                    count++;
                    stop = data[i].thetime; //set the time when the turn is finished
                    if(max < data[i].theX){ //find the maximum acceleration value of the turn
                        max = data[i].theX;
                    }
                }
                i++;
            }
            if(count > 8){
                turn[k] = [start, stop, max];
                k++;
                max = 0;
            }
            count = 0;
        }
    }
}

msg.payload = turn; //the returned msg is a vector with all the turns
return msg;
```

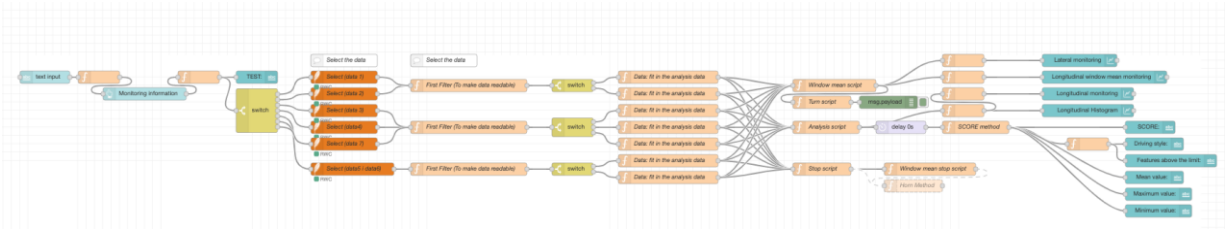
As seen, the turn script acts for both right turn and left turn. Same as in the stop script, there is a turn threshold that will act in both turns (in the practice in the positive and negative values).

- SCORE method code:
  As explained before, all thresholds have been calculated using Horn Method statistics and using 7 performances in the route.

```
//thresholds
//1- peak stop value
const MAX_STOP_VALUE_TH = 0.2256944754385714;
const MIN_STOP_VALUE_TH = -0.189495535692857;
const MAX_STOP_W_VALUE_TH = 0.1467040;
const MIN_STOP_W_VALUE_TH = -0.139140625;
const DIS_STOP_VALUE_TH = 0.4104045758928571;
const DIS_STOP_W_VALUE_TH = 0.28347993949983336;
const TURN_TH = -0.20504464285714288;
var peak_stop_th = [MAX_STOP_VALUE_TH,MIN_STOP_VALUE_TH,MAX_STOP_W_VALUE_TH,MIN_STOP_W_VALUE_TH,DIS_STOP_VALUE_TH,DIS_STOP_W_VALUE_TH];

//2- AVERAGE VALUES
const MEAN_MAX_VALUE_TH = 10.41563425;
const MEAN_MIN_VALUE_TH = -1.0990172500000002;
const MEDIAN_MAX_VALUE_TH = 7.550;
const MEDIAN_MIN_VALUE_TH = -4.550;
const MEAN_TH = 2.239772;
const MEDIAN_TH = 0.428;
var average_th = [[MEAN_MAX_VALUE_TH, MEAN_MIN_VALUE_TH, MEAN_TH],[MEDIAN_MAX_VALUE_TH,MEDIAN_MIN_VALUE_TH, MEDIAN_TH]];
//THIRD MOMENT IMPOSIBLE TO USE!!!!
//3- PEAK MAXIMUM VALUES
const MAX_VALUE_TH = 0.2955078125;
const MIN_VALUE_TH = -0.27890625;
const ST_DEV_VALUE_TH = 17.79285714285714;
var peak_maximum_th = [MAX_VALUE_TH, MIN_VALUE_TH, ST_DEV_VALUE_TH];

//We get all the feature values from the test previosly set as global variables.
var mean_v = global.get("mean_Y");
var median_v = global.get("median_Y");
var max_v = global.get("max_Y")*0.00390625;
var min_v = global.get("min_Y")*0.00390625;
var maxs_stop_v = global.get("maxs_stop");
var dif_stop_v = global.get("dif");
var maxs_stopw_v = global.get("maxs_stop_w");
var dif_stopw_v = global.get("dif_w");
var st_dev_v = global.get("st_dev_Y");
var right_turn_v = global.get("turn_v");

//Initialization to detect all bus stops and turns that exceed the threshold needed for the score calculation
var cont_max = 0;
var cont_min = 0;
var cont_dif = 0;
var cont_maxw = 0;
var cont_minw = 0;
var cont_difw = 0;
var mean_max = 0;
var mean_min = 0;
var mean_dif = 0;
var mean_maxw = 0;
var mean_minw = 0;
var mean_difw = 0;
var mean_right_turn = 0;
var cont = dif_stop_v.length;

//Bool is a boolean vector for mean, median, maximum and minimum value, av max, av min, av maxw, av minw values.
//If the feature exceeds the threshold the bool[i] will be true.
//[0]- mean [1]- median [2]- max. value [3]- min. value [4]- average max [5]- average min
//[6]- average max window [7]- average min window [8]- average turn
var bool = Array.apply(null, new Array(9)).map(Boolean.prototype.valueOf,false);
```

```
//Bus stop counter for all features that are higher than the threshold
for(var i = 0; i < cont; i++){
    if(MAX_STOP_VALUE_TH < maxs_stop_v[i][1]){
        cont_max++;
    }
    if(MIN_STOP_VALUE_TH > maxs_stop_v[i][0]){
        cont_min++;
    }
    if(DIS_STOP_VALUE_TH < dif_stop_v[i]){
        cont_dif++;
    }
    if(MAX_STOP_W_VALUE_TH < maxs_stopw_v[i][1]){
        cont_maxw++;
    }
    if(MIN_STOP_W_VALUE_TH > maxs_stopw_v[i][0]){
        cont_minw++;
    }
    if(DIS_STOP_W_VALUE_TH < dif_stopw_v[i]){
        cont_difw++;
    }
    //mean value
    mean_max += maxs_stop_v[i][1];
    mean_min += maxs_stop_v[i][0];
    mean_dif += dif_stop_v[i];
    mean_maxw += maxs_stopw_v[i][1];
    mean_minw += maxs_stopw_v[i][0];
    mean_difw += dif_stopw_v[i];
}

//all features divided by its score formula, as seen, it is divedid in 3.
var peak_stop = [[mean_max/cont, cont_max],[mean_min/cont, cont_min],[mean_maxw/cont, cont_maxw],[mean_minw/cont, cont_minw],[mean_dif/cor
var average = [mean_v, median_v];
var peak_maximum = [max_v, min_v, st_dev_v];

var score = 0; //score initializated to 0.

//bus stop script: formula
for(i = 0; i < peak_stop_th.length; i++){
    score +=100  - (peak_stop[i][1]/cont)*50 - (peak_stop[i][0]/peak_stop_th[i]-1)*50;
    if(Math.abs(peak_stop[i][0]) < Math.abs(peak_stop_th[i])){
        bool[i+4] = true;
    }
}

//bus turn script: counter for all features that are higher than the threshold and formula
var cont_turn = 0;
if(right_turn_v.length !== undefined){
    for(i = 0; i < right_turn_v.length; i++){
        if(right_turn_v[i][2] < TURN_TH){
            cont_turn;
        }
        mean_right_turn += right_turn_v[i][2];
    }
    if(mean_right_turn < TURN_TH){
        bool[10] = true;
    }
    score += 100 - (cont_turn/right_turn_v.length)*50-((mean_right_turn/right_turn_v.length)/TURN_TH-1)*50;
}
if(right_turn_v.length === undefined){
    score += 100 - ((right_turn_v/TURN_TH)-1)*100;
    if(right_turn_v < TURN_TH){
        bool[10] = true;
    }
}

//average value (central characteristics): formula
for(i = 0; i < average_th.length; i++){
    var mean = (average_th[i][0] - average_th[i][1])/2;
    if(average[i] < average_th[i][2]){
        score += 100 - Math.abs((average[i]/average_th[i][2])-1)/(-average_th[i][1]+average_th[i][2])
    }
    if(average[i] > average_th[i][2]){
        score += 100 - Math.abs((average[i]/average_th[i][2])-1)/(-average_th[i][0]-average_th[i][2])
    }
    if((average[i] > average_th[i][1])&&(average[i] < average_th[i][0])){
        bool[i] = true;
    }
}
//maximum peak values: formula
for(i = 0; i < peak_maximum_th.length; i++){
    score += 100 - ((peak_maximum[i]/peak_maximum_th[i])-1)*100;
    if(Math.abs(peak_maximum[i]) < Math.abs(peak_maximum_th[i])){
        bool[i+2] = true;
    }
}
//returned message: [final score,bool,mean value, max value, min value] (this values will be presented in the UI)
//all the returned messages are trunc to show only 2 decimals
msg.payload = [Math.trunc(score*100)/100,bool,Math.trunc(mean_v*0.00390625*100)/100,
Math.trunc(max_v*100)/100,Math.trunc(min_v*100)/100];
return msg;
```

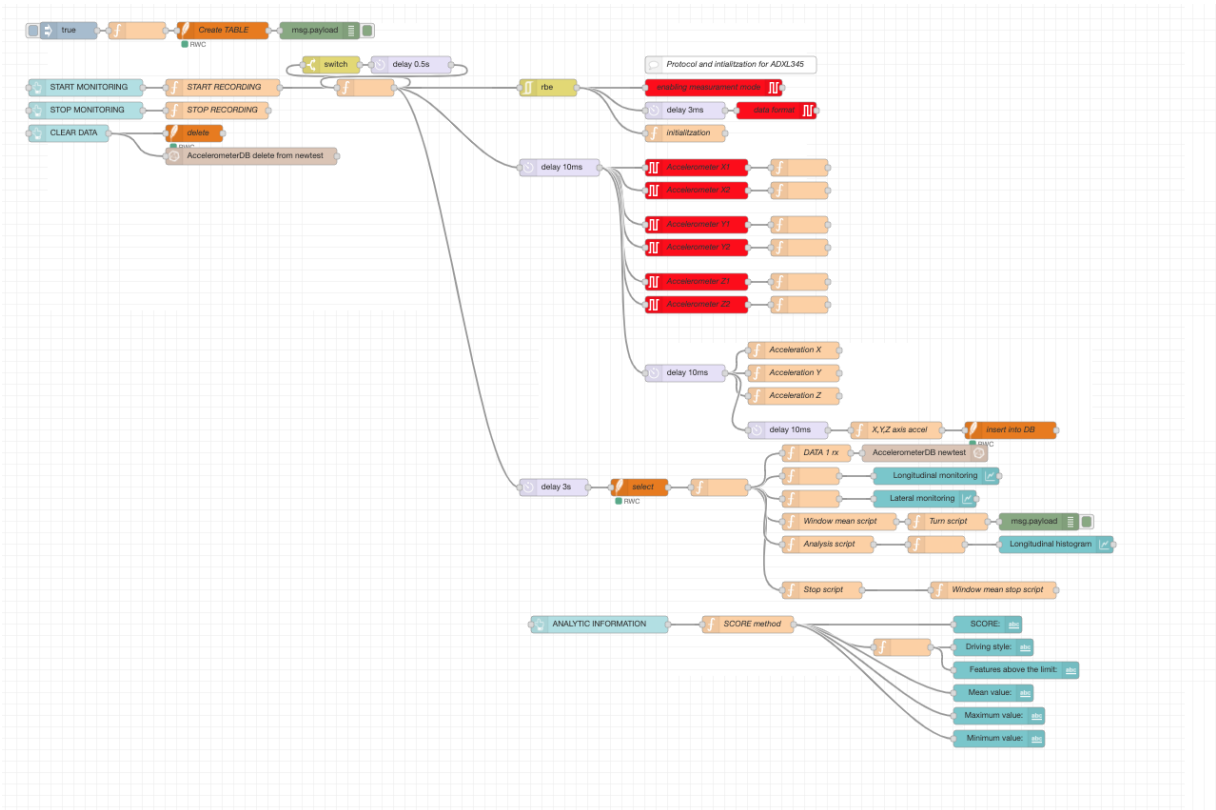
- UI dashboard flows:
  The UI consists of 3 tabs. In the next figures are represented the flows of each tab. The function nodes are the same that the ones above.
  - Test information tab flow:



In blue are all the buttons, texts and graphics nodes (dashboard nodes) are represented in blue.
The graphics represents the information with a json input. The function that tx the json to the graphic need to convert the information into json. The code for the longitudinal monitoring is exposed in next (lateral, window mean and histogram are the same):

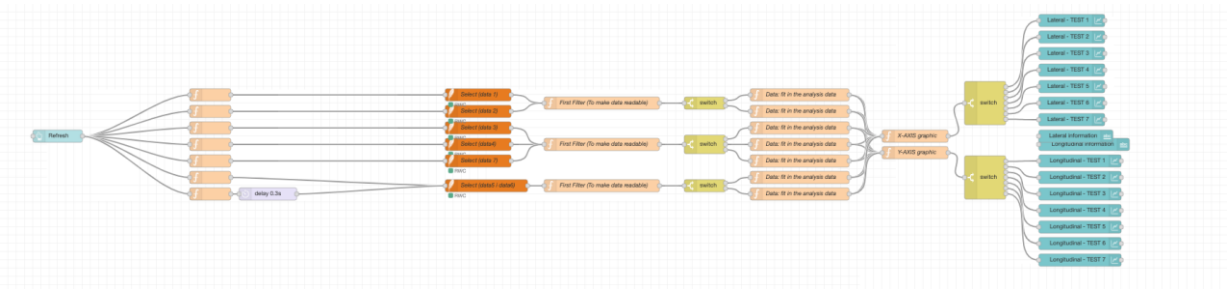
```
var series = ["temp DegC"];
var labels = ["Data Values"];
var data = "[[";
var dataY;
for (var i=0; i < msg.payload.length; i++) {
    dataY = msg.payload[i].theY;
    data += '{ "x":' + msg.payload[i].thetime + ', "y":' + dataY + '}';
    if (i < (msg.payload.length-1)) {
     data += ","
    } else {
     data += "]]"
    }
}
var jsondata = JSON.parse(data);
msg.payload = [{"series": series, "data": jsondata, "labels": labels}];
return msg;

```

- New test tab flow:



For the new test, the extraction, the processing and the analysis are exposed in the same flow. In blue are exposed the dashboard nodes for the UI.

- General information tab flow:



Horn Method script code:

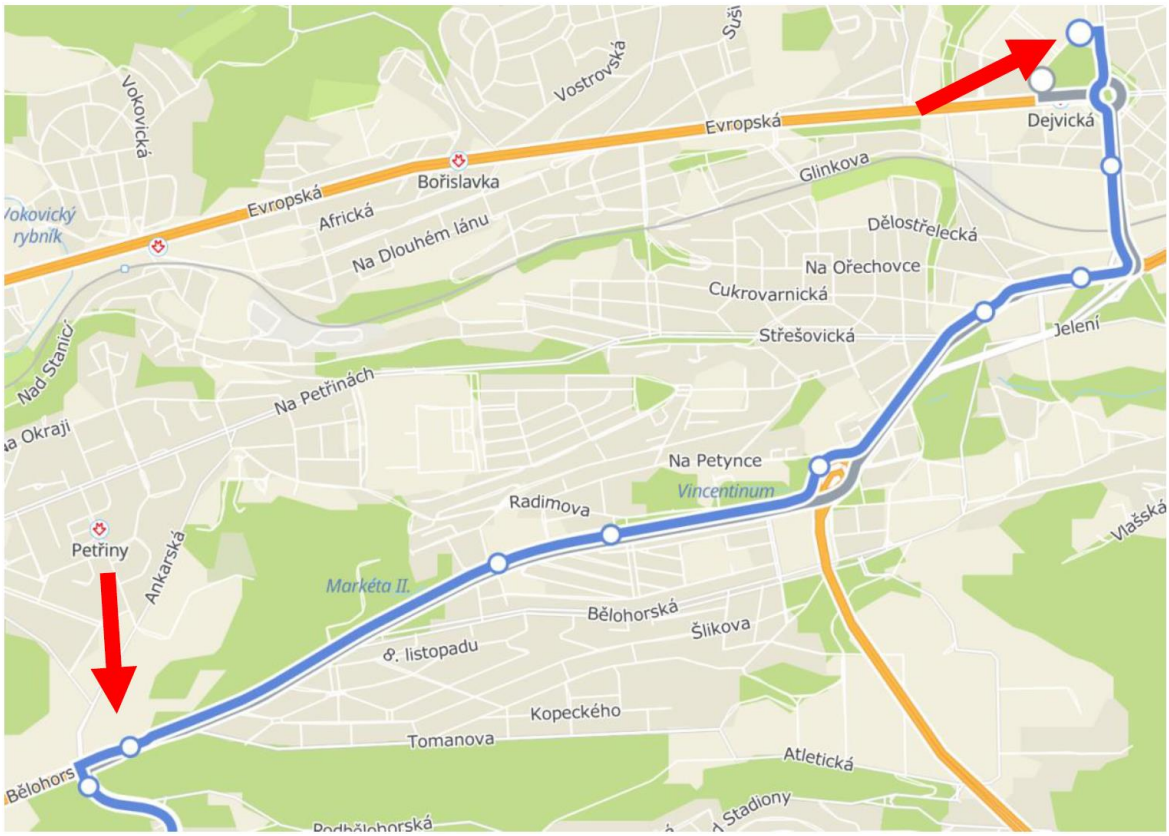
To extract all the Horn Method analysis data, a script has been programmed and used in order to get the 95% confidence interval. The script returns directly the Horn method for the maximum value, the minimum value and the distance value.

```
var max_v = [];
var min_v = [];
var dif_v = msg.payload[2]; //distance vector
var mean_max;
var mean_min;
var mean_dif;
var aux;
var COV = context.global.get('cov');
for(var i = 0; i < msg.payload[2].length; i++){
    max_v[i] = msg.payload[1][i][1]; //input for the maximum vector
    min_v[i] = msg.payload[1][i][0]; //input for the minimum vector
    mean_max += max_v[i]; //first we do the mean of the vector
    mean_min += min_v[i];
    mean_dif += dif_v[i];
}
mean_max = mean_max/max_v.length;
mean_min = mean_min/min_v.length;
mean_dif = mean_dif/dif_v.length;
/*1. As Horn method says, first, the data need to be ordered from min to max
To do so, BUBBLE METHOD has been used */
for (var k = 1; k < msg.payload[1].length; k++) {
    for (i = 0; i < (msg.payload[1].length - k); i++) {
        if (max_v[i] >= max_v[i + 1]) {
            aux = max_v[i];
            max_v[i] = max_v[i + 1];
            max_v[i + 1] = aux;
        }
        if (min_v[i] >= min_v[i + 1]) {
            aux = min_v[i];
            min_v[i] = min_v[i + 1];
            min_v[i + 1] = aux;
        }
        if (dif_v[i] >= dif_v[i + 1]) {
            aux = dif_v[i];
            dif_v[i] = dif_v[i + 1];
            dif_v[i + 1] = aux;
        }
    }
}
/*2. Find pivots: Following the formulas from Horn Method, pivots must be found for maximum value, minimum value and distance value */
//Max variables
//First to find is HL, for it, it depends if the number of stops is par or even:
var HL_max = parseInt((max_v.length+1)/2)/2;
if(HL_max%2 !== 0){
    HL_max = parseInt((max_v.length+1)/2+1)/2;
}
//Pivot Half Sum and Pivot Range calculation
var max_max = max_v[max_v.length+1-HL_max];
var min_max = max_v[HL_max];
var pivh_max = (max_max+min_max)*0.5;
var pivr_max = max_max - min_max;
var SE_max = 0;
//Min variables
//First to find is HL, for it, it depends if the number of stops is par or even:
var HL_min = parseInt((max_v.length+1)/2)/2;
if(HL_min%2 !== 0){
    HL_min = parseInt((max_v.length+1)/2+1)/2;
}
//Pivot Half Sum and Pivot Range calculation
var max_min = min_v[min_v.length+1-HL_min];
var min_min = min_v[HL_min];
var pivh_min = (max_min+min_min)*0.5;
var pivr_min = max_min - min_min;
var SE_min = 0;
//Distance variables
//First to find is HL, for it, it depends if the number of stops is par or even:
var HL_dif = parseInt((dif_v.length+1)/2)/2;
if(HL_dif%2 !== 0){
    HL_dif = parseInt((dif_v.length+1)/2+1)/2;
}
//Pivot Half Sum and Pivot Range calculation
var max_dif = dif_v[dif_v.length+1-HL_dif];
var min_dif = dif_v[HL_dif];
var pivh_dif = (max_dif+min_dif)*0.5;
var pivr_dif = max_dif - min_dif;
var SE_dif = 0;

/* in order to get the 95% confidence we need to find the tL0.95 value in the t distribution table
Depending on the samples we will use one t or another*/
var t;
if(min_v.length == 4){
    t = 0.555;
}
if(min_v.length == 5){
    t = 1.730;
}
if(min_v.length == 6){
    t = 0.759;
}
if(min_v.length == 7){
    t = 0.550;
}
if(min_v.length == 8){
    t = 0.469;
}
if(min_v.length == 9){
    t = 0.688;
}
if(min_v.length == 10){
    t = 0.523;
}

```

```
//calculation of the 95% confidence interval
var interv_max_max = pivh_max+t*pivr_max;
var interv_min_max = pivh_max-t*pivr_max;
var interv_max_min = pivh_min+t*pivr_min;
var interv_min_min = pivh_min-t*pivr_min;
var interv_max_dif = pivh_dif+t*pivr_dif;
var interv_min_dif = pivh_dif-t*pivr_dif;

//return of all the information
msg.payload = [{"Pivot Half Sum": pivh_min, "Pivot Range": pivr_min, "Upper interval": interv_max_min, "Lower interval": interv_min_min, m
{"Pivot Half Sum": pivh_max, "Pivot Range": pivr_max, "Upper interval": interv_max_max, "Lower interval": interv_min_max, max_v},
{"Pivot Half Sum": pivh_dif, "Pivot Range": pivr_dif, "Upper interval": interv_max_dif, "Lower interval": interv_min_dif, dif_v}];

return msg;
```

STATISTIC ANALYSIS explication:

The statistical analysis consists on 7 different test samples for a public bus route. This bus route has been taken from *Dejvická 160 00 Prague 6* to *Vypich 169 00 Prague 6*:

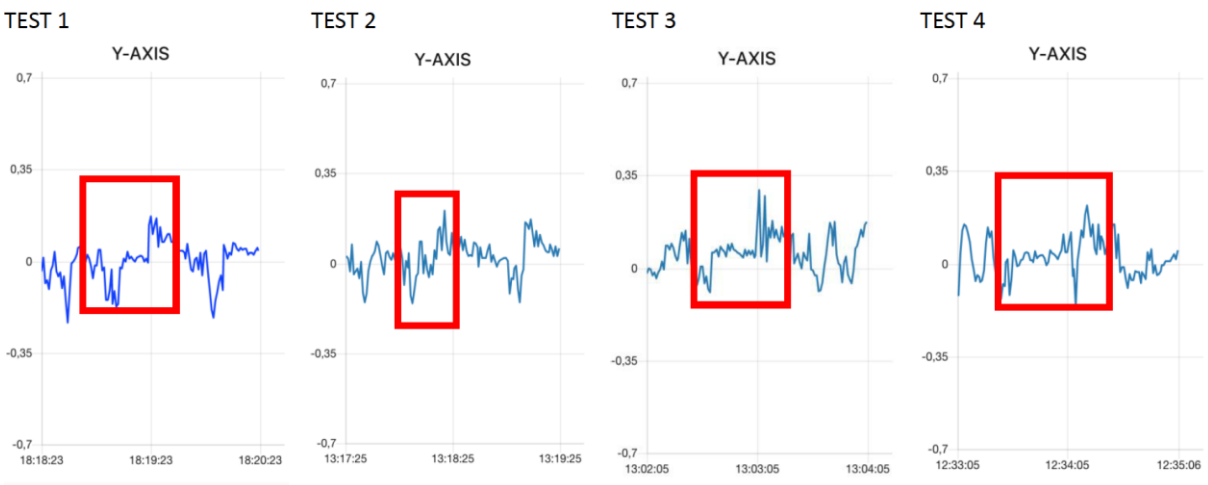


The comparison and bus stop script threshold has been done analysing all the stops from the route, first from the first 4 tests and then for the last 3 tests:

Stops:

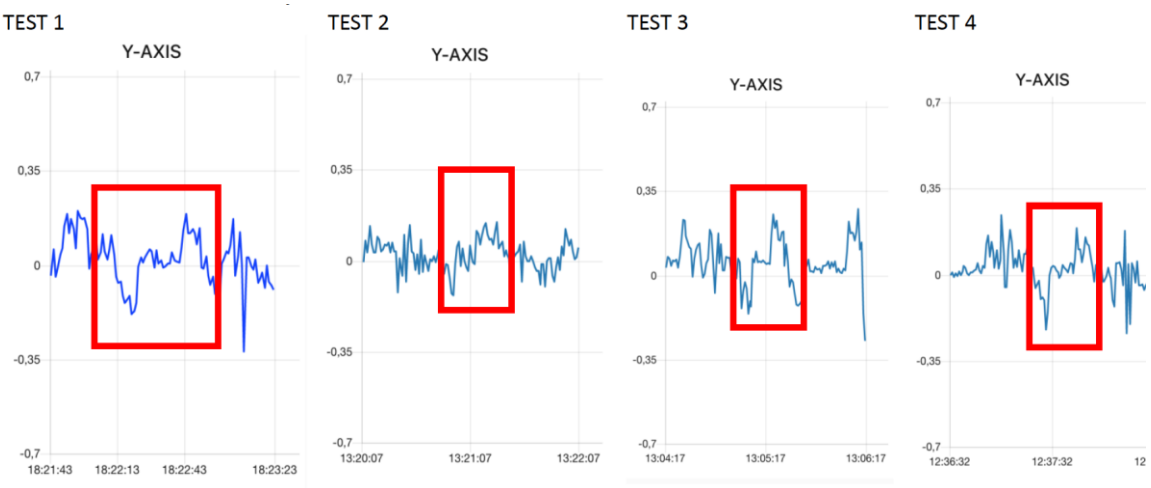
All stops have been compared with the first 4 tests next.

First stop

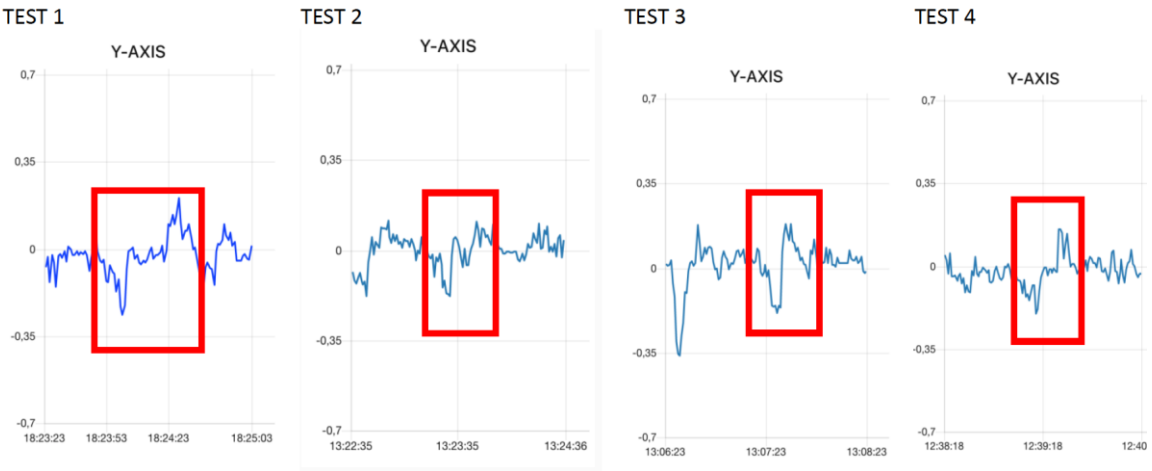


Second stop

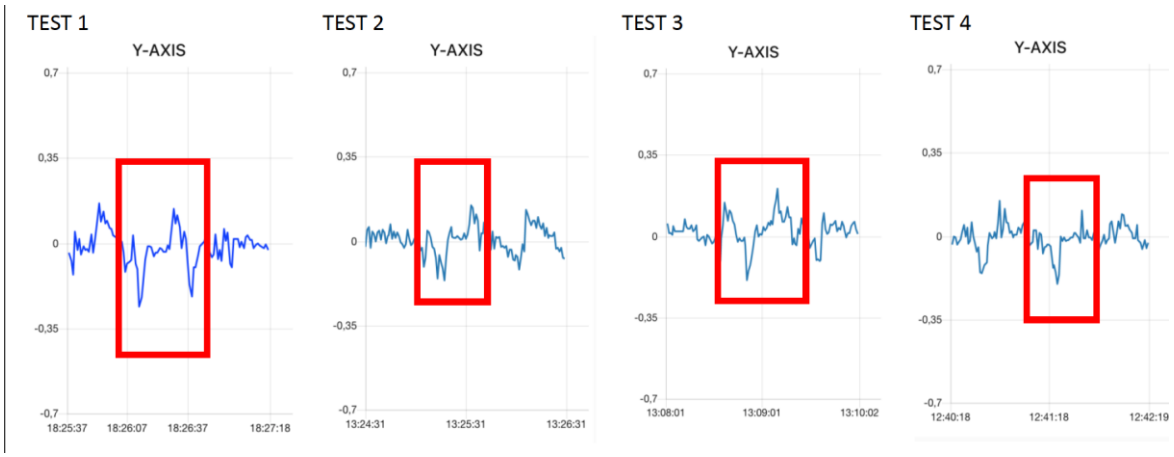
Third stop



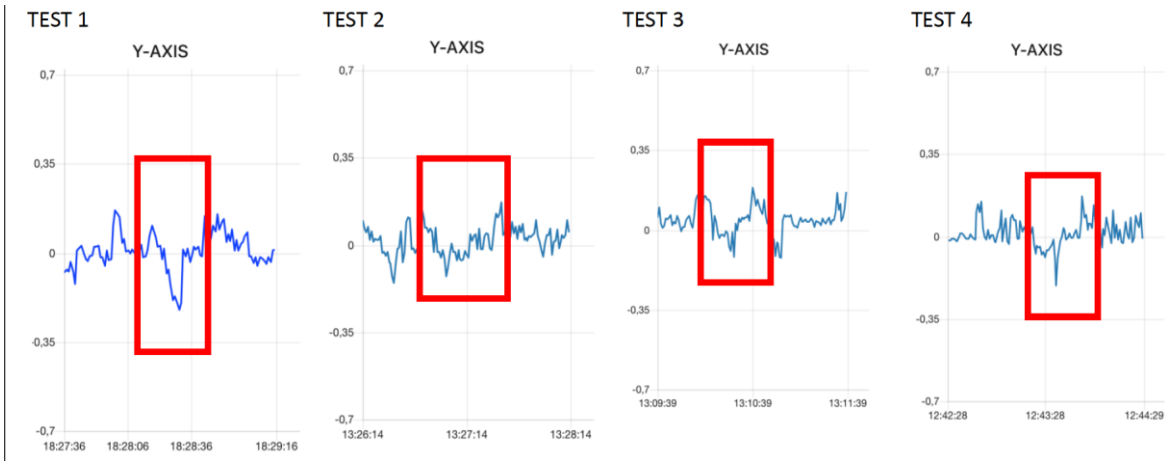
Fourth stop



Fifth stop

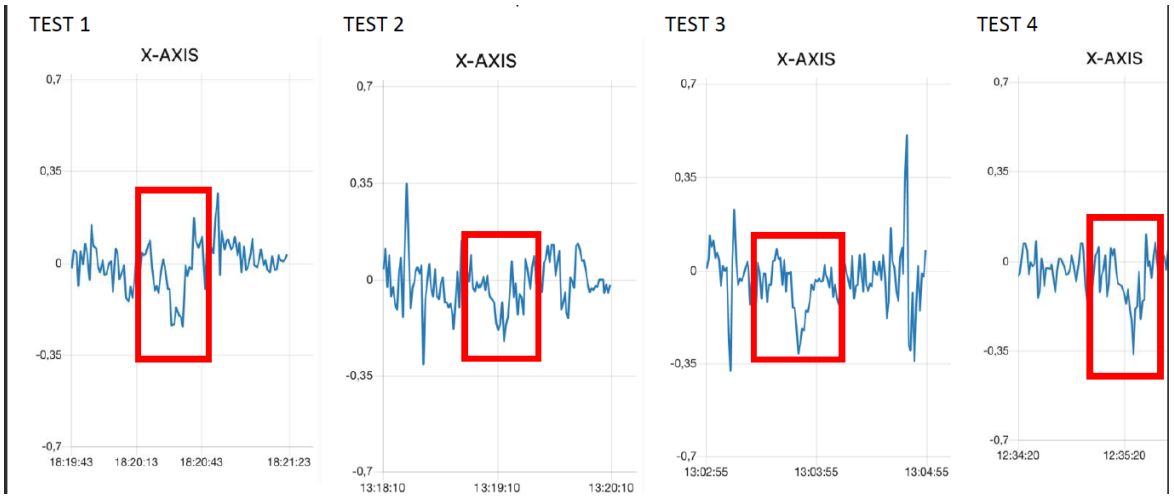


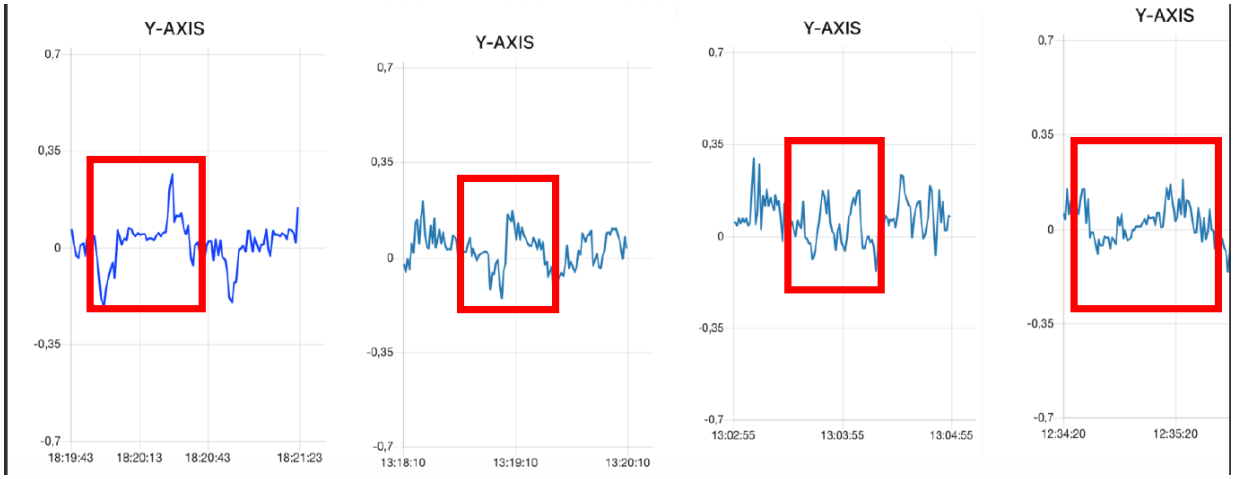
Sixth stop



Turn:

In the turn section, both X axis and Y axis graphics are important due to the fact that there is a break and acceleration in a turn.





Statistical analysis:

Next are exposed the hole statistical analysis for the accelerometer data. It is presented in 3 parts:

- Histogram analysis
- Statistic features: Mean, Median, Maximum and Minimum values, Variance, Covariance and Standard Deviation.
- Stops and turns analysis.

Histograms:

One important feature to take into consideration is the histogram information.

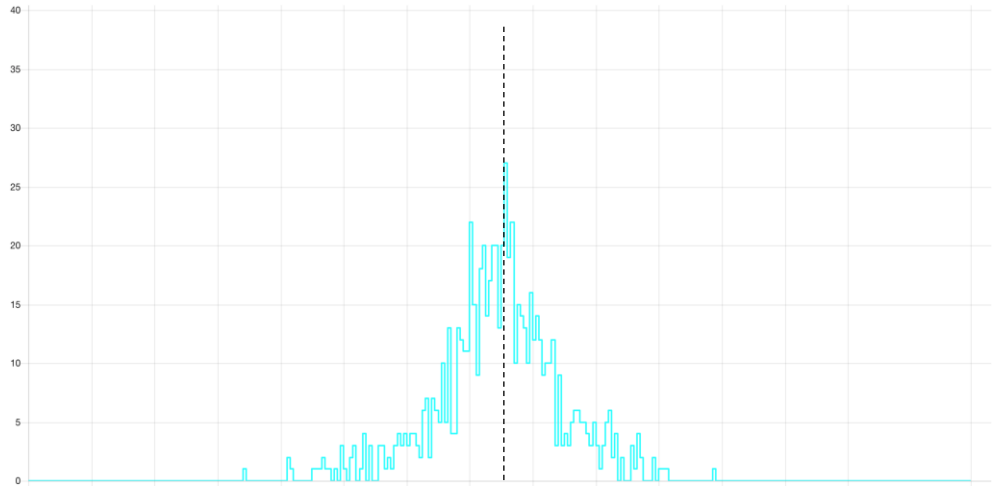
Talking about the Y-AXIS histogram, the ideal situation would be an histogram with the majority of the points near the centre, which is the 0 acceleration, and a symmetric function in relation to the centre (0). Also, as it has many stops, it would be a histogram with a similar distribution as the normal one.

Also, it is important not to have a histogram with dispersed values far from the centre, because, what it indicates is that the accelerometer had abrupt acceleration in some point of the route, and moreover is not an efficient driving.

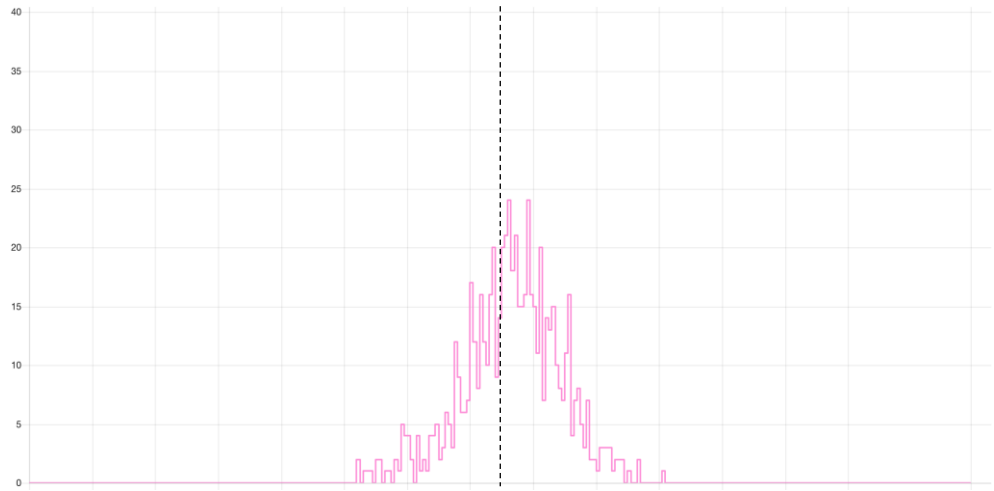
Furthermore, one more thing to take into consideration is that, higher histogram (in the 0) can also help to see which test stayed in a bus stop for more time.

With this said, next is exposed the comparison for the 7 tests Y-AXIS histograms:

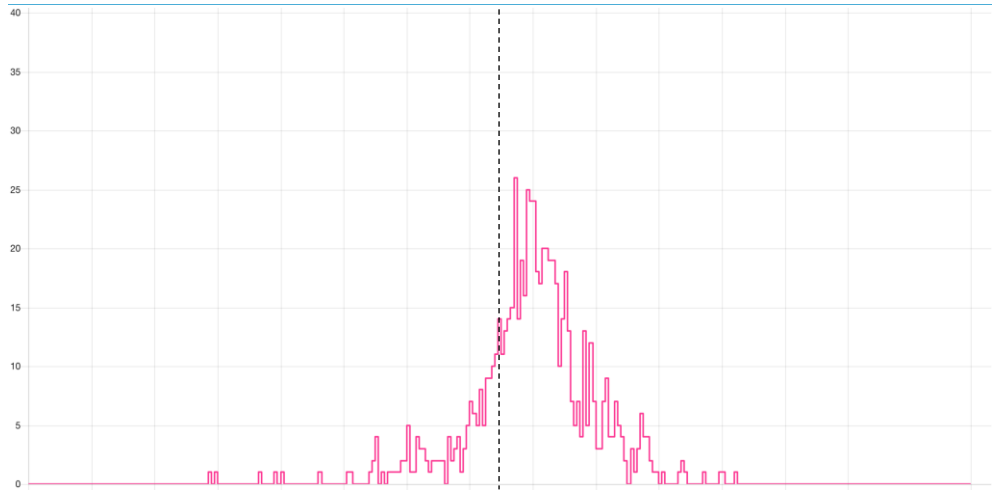
Test 1



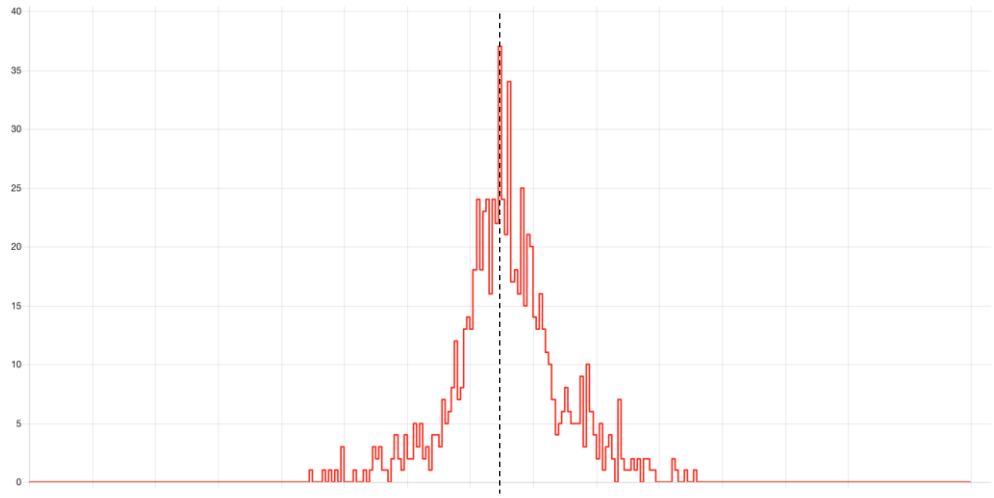
Test 2



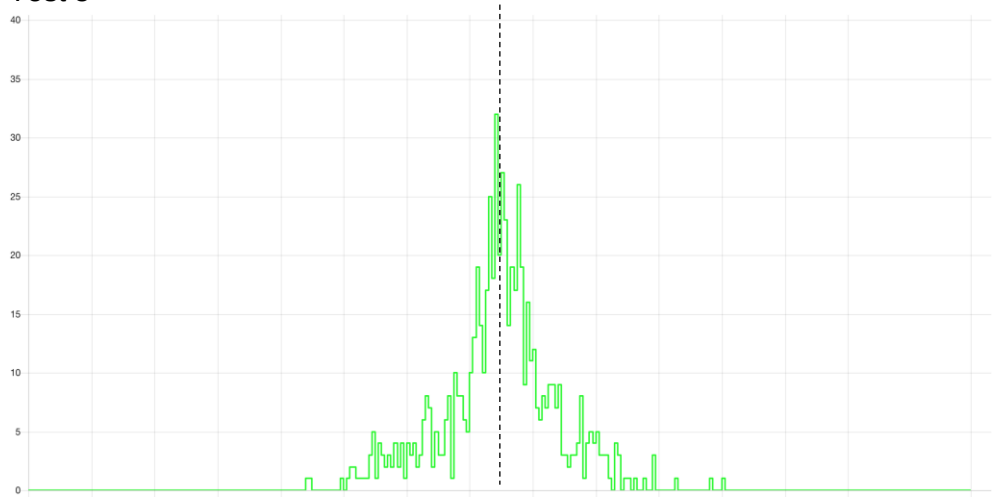
Test 3



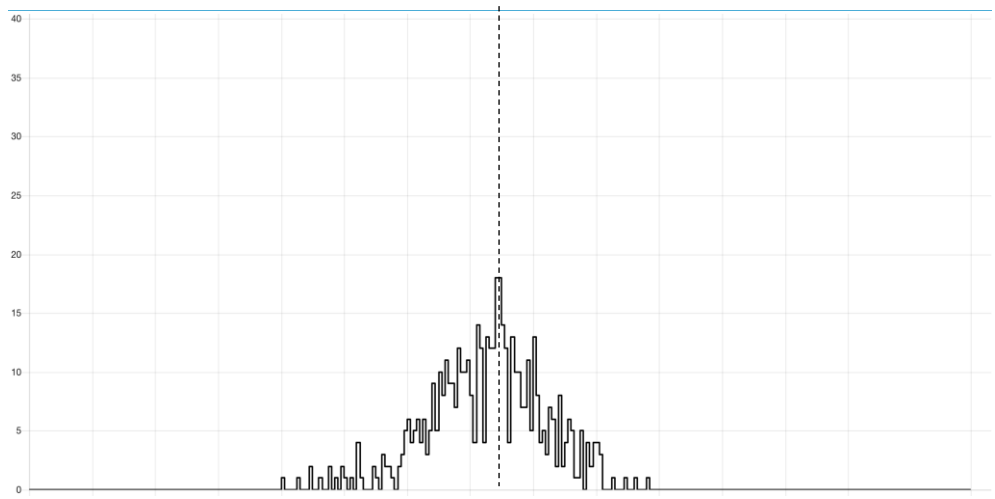
Test 4



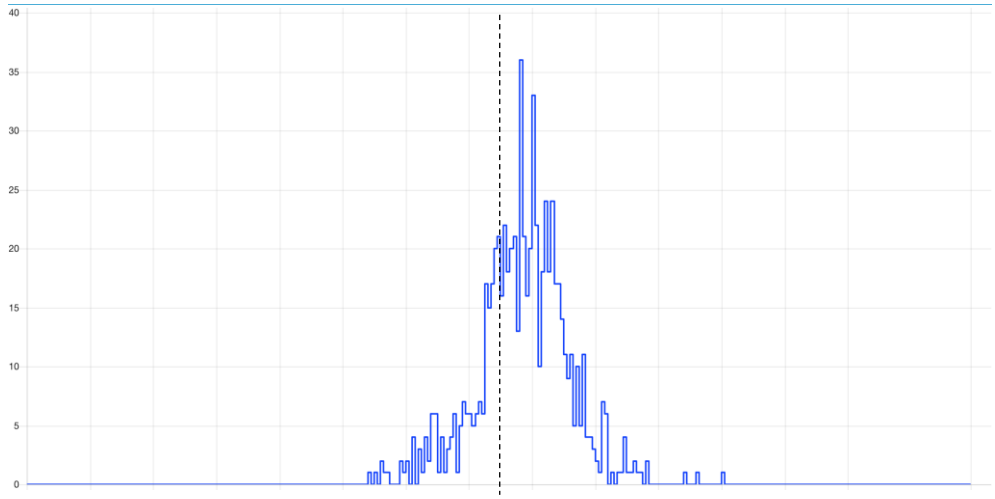
Test 5



Test 6



Test 7



Skewness analysis:

Before going into statistics features, it is important to do a skewness analysis inside the histogram part. Because it gives information very related to the histogram.

This feature is extracted as well as the mean and variance (deviation), with moments. In fact, the shape of any distribution can be described by its various 'moments'. The first three are:

1. The mean, which indicates the central tendency of a distribution.
2. The second moment is the variance, which indicates the width or deviation.
3. The **third moment** is the **skewness**, which indicates any asymmetric 'leaning' to either left or right.

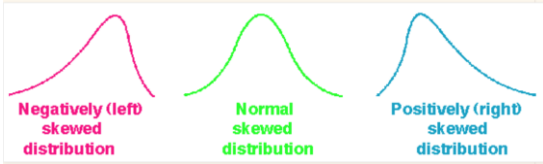
Skewness formula:

$$Skew = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{(X_i - \bar{X})}{\sigma}\right]^3$$

TEST	SKEWNESS
TEST 1	0.000094597
TEST 2	-0.00869966
TEST 3	-0.121538514
TEST 4	-0.001103560
TEST 5	0.000032796
TEST 6	0.0184970995
TEST 7	-0.112123982

Horn Method:
Pivot Half Sum: 0.00489871975
Pivot Range: 0.0271967595
Upper interval: 0.019856937475000003
Lower interval: -0.010059497975000001

- Negative skew commonly indicates that the *tail* is on the left side of the distribution.
- Positive skew indicates that the tail is on the right.



In fact, the values that are very close to 0 (order of 0.00X) are the ones which have a more normal distribution.

Statistic features: Mean, Median, Maximum and Minimum Value, Variance, Covariance and Standard Deviation

As there are lots of statistics features, a previous research has been done in order to compare the most relevant statistics:

- Y - MEAN
- Y - Median
- X and Y Maximum and minimum values
- Covariance (between lateral and longitudinal acceleration)
- Standard deviation (to see how dispersed the data is)

It doesn't make sense to calculate the X-AXIS mean and median because maybe in the route there are more left turns than right or vice versa.

Also, Horn method was used in order to get the 95% confidence interval with the mean, median and maximum and minimum values.

Result:

The values that are presented next are **not** multiplied by the factor of 0.00390625 because it is easy to treat the data without so many decimals. So, if we want to have it with G's, we have to multiply by that factor.

Mean and Median

TEST	X - Mean	**Y - Mean**	X - Median	**Y - Median**
Test 1	1.1796657	**-0.92896**	-3.5	**-5**
Test 2	-1.928024	**3.284839**	1.5	**0**
Test 3	-2.044540	**9.892241**	5	**6**
Test 4	-2.220715	**1.792848**	2.5	**1**
Test 5	-2.879588	**-0.575624**	-2.5	**-4**
Test 6	-3.847656	**-4.908203**	-6.5	**-8**
Test 7	-0.538147	**7.104904**	6	**7**

Mean:

Y-AXIS

The same that happens with the histogram, the ideal situation is to have the mean very close to 0. Because it means that there are (more or less) the same negative accelerations and positive acceleration, and it means that the bus accelerates and deaccelerate alike. With a value far from 0 means that the acceleration is higher accelerating or deaccelerating.

And talking about Horn method to determine the 95% confidence interval:

Pivot Half Sum: 4.6583085

Pivot Range: 10.467865

Upper interval: **10.41563425**

Lower interval: **-1.0990172500000002**

Median:

Y-AXIS

As the median is the middle of a sorted list of numbers, it gives the middle number, which for the same reason as the mean, should be close to 0.

With the Horn method the results are:

Pivot Half Sum: 1.5

Pivot Range: 11

Upper interval: **7.550000000000001**

Lower interval: **-4.550000000000001**

Maximum and minimum values:

TEST	X – Max	X – Min	**Y – Max**	**Y – Min**
Test 1	153	-118	**68**	**-82**
Test 2	147	-139	**52**	**-46**
Test 3	156	-179	**75**	**-93**
Test 4	80	-170	**62**	**-61**
Test 5	116	-141	**71**	**-62**
Test 6	124	-174	**47**	**-70**
Test 7	145	-64	**71**	**-42**

The maximum and minimum values allow to know which are the maximum and minimum accelerations of the route in every test. It is possible to compare these values to the peaks of the stops (in the last section of this report are analysed) to know if the maximum value of the route coincide with a stop or it is from another situation.

Y-AXIS:

- Maximum value:
  With Horn Method we get:

  Pivot Half Sum: 68.5
  Pivot Range: 13
  Upper interval: **75.65**
  Lower interval: 61.35
- Minimum value:
  With Horn Method we get:

  Pivot Half Sum: -56
  Pivot Range: 28
  Upper interval: -40.599999999999994
  Lower interval: **-71.4**

X-AXIS:

- Maximum value:
  With Horn Method we get:

  Pivot Half Sum: 140
  Pivot Range: 32
  Upper interval: 157.6
  Lower interval: 122.4
- Minimum value:
  With Horn Method we get:

  Pivot Half Sum: -117
  Pivot Range: 106
  Upper interval: -58.699999999999996
  Lower interval: -175.3

Covariance and Standard Deviation

TEST	VAR X	**COVAR XY**	**COVAR YX**	VAR Y
Test 1	545.453	**-21.177**	**-21.177**	404.665
Test 2	533.971	**-44.745**	**-44.745**	253.283
Test 3	660.186	**-17.667**	**-17.667**	395.762

Test 4	475.488	**-29.149**	**-29.149**	299.500
Test 5	676.347	**-2.870**	**-2.870**	330.588
Test 6	647.808	**-34.773**	**-34.773**	341.899
Test 7	311.907	**-9.610**	**-9.610**	215.430

TEST	X – St Deviation	**Y – St Deviation**
Test 1	23.35	**20.11**
Test 2	23.10	**15.91**
Test 3	25.69	**19.89**
Test 4	21.80	**17.30**
Test 5	26.01	**18.18**
Test 6	25.45	**18.49**
Test 7	17.66	**14.67**

Horn Method for st. deviation
Pivot Half Sum: 18.705
Pivot Range: 2.8099999999999987
Upper interval: 20.2505
Lower interval: 17.159499999999998
The covariance gives us a very important information, because as the values are negative, the greater values of one variable mainly correspond to the lesser values of the other.
Also, both covariances are the same because of one of the properties:
Cov(XY) = Cov(YX)

Bus stops and turns analysis:

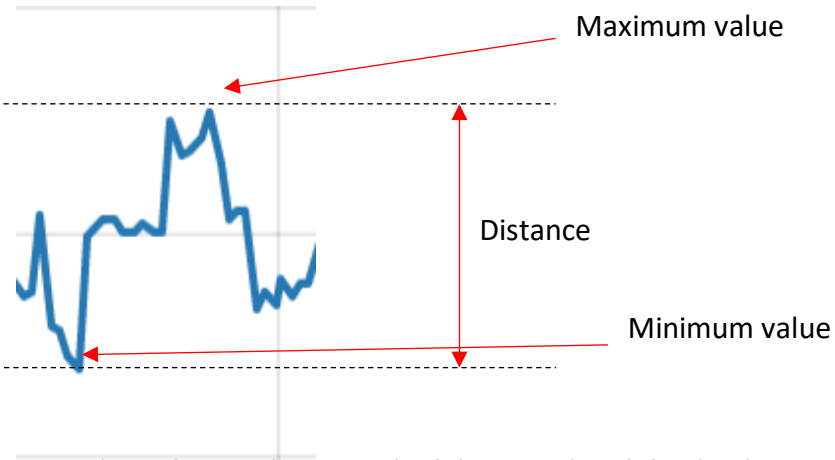
Regarding to the analysed bus stops data, two different methods have been used to obtain the thresholds for the 6 bus stop features:

- Method 1:
  Within the same sample test, the maximum, minimum and distance peaks values extracted from the bus stop algorithm, have been compared and analysed using the Horn Method to obtain the 95% confidence interval from each test. After this a second Horn Method analysis has been executed with the higher confidence interval value of all the tests.
- Method 2:
  Within the same sample test, the maximum, minimum and distance peaks values extracted from the bus stop algorithm, have been compared and analysed using the Horn Method to obtain the 95% confidence interval from each test. After this, an average value of the higher confidence interval value of all tests has been used as a threshold.

Compared data:

To compare the data, 3 different values:

- Maximum value peak.
- Minimum value peak.
- Distance between peaks.



Apart from the maximum and minimum value, it is also important to analyse the distance between these values, because, as the road is not always in 0 degrees, sometimes with the sharp lane changes, the accelerometer can give us wrong accelerations (higher or lower). So, if the distance is taken, the value (in an efficient driving style) between these values is always the same.

SCRIPT

Regarding to how to compare the data, not all the buses stop the same number of times and many times the data is not repeatable. So, in order to analyse this data, a script was made to extract the information about the stops of the route. This script detects when a bus stops for more than 8 seconds it is the minimum of a bus stop, removing all the “pit stop” from a traffic light (that are lower than 8 seconds)

Until now, the script is working perfectly fine.

Returns of the script:

- Hour of the minimum and maximum peak.
- Minimum and maximum peak values.
- Distance between this two peaks.

Example:


```
▾ array[3]
  ▾ 0: array[8]
    ▾ 0: array[2]
        0: 13/5/2020, 12:48:51 [UTC+2]
        1: 13/5/2020, 12:49:30 [UTC+2]
    ▸ 1: array[2]
    ▸ 2: array[2]
    ▸ 3: array[2]
    ▸ 4: array[2]
    ▸ 5: array[2]
    ▸ 6: array[2]
    ▸ 7: array[2]
  ▾ 1: array[8]
    ▾ 0: array[2]
        0: -0.1875
        1: 0.1796875
    ▸ 1: array[2]
    ▸ 2: array[2]
    ▸ 3: array[2]
    ▸ 4: array[2]
    ▸ 5: array[2]
    ▸ 6: array[2]
    ▸ 7: array[2]
  ▾ 2: array[8]
      0: 0.3671875
```


The first two values are the exact hour of the peaks (before stop and after stop), the next two values are the minimum and maximum values and for last there is the difference between these two values.

TURNs:

For the turns the window frame data has been used directly, because, as see in the next graph, the data is not very clear, and compared with the window frame one, it is very clear:

STOPS ANALYSIS:

METHOD 1:

In the next tables are expressed the values of the pivot half sum, pivot range and 95% confidences interval of the Horn Method for maximum value, minimum value and distance between peaks value.

Data without a window frame will be analysed first, and then with the window frame.

As explained before the formula to extract the 95% confidence interval is:

$$P_L - R_L t_{L,0.95}(n) \leq \mu \leq P_L + R_L t_{L,0,95}(n)$$

P_L: Pivot Half Sum

R_L: Pivot Range

And the $t_{L,0.95}(n)$, which depends on n, and not all the tests have the same stops (n), we will use different $t_{L,0.95}(n)$.

Test 1: n = 8 → $t_{L,0.95}(9)$= 0.688

Test 2: n = 7 → $t_{L,0.95}(8)$= 0.469

Test 3: n = 8 → $t_{L,0.95}(8)$= 0.469

Test 4: n = 10 → $t_{L,0.95}(10)$= 0.523

Test 5: n = 8 → $t_{L,0.95}(8)$= 0.469

Test 6: n = 6 → $t_{L,0.95}(6)$= 0.759

Test 7: n = 8 → $t_{L,0.95}(8)$= 0.469

Without a window frame:

Maximum value:

Test	Pivot half sum	Pivot range	Interval
Test 1	0.2124	0.0849	0.1725 < u < 0.2522
Test 2	0.1577	0.0927	0.1066 < u < 0.2087
Test 3	0.2221	0.0966	0.1689 < u < 0.2753
Test 4	0.1811	0.0400	0.1536 < u < 0.2087
Test 5	0.1699	0.0468	0.1441 < u < 0.1957
Test 6	0.1538	0.0615	0.1071 < u < 0.2005
Test 7	0.1918	0.0849	0.1451 < u < 0.2386

The only threshold that is important for the maximum is the higher one, because, as lower the acceleration, more efficient the driving style.

Applying again the Horn method with this maximum we get:

Pivot Half Sum: 0.2420205078125

Pivot Range: 0.06664257812499999

Upper interval: 0.27867392578125

Lower interval: 0.20536708984375002

With an upper interval of **0.2786**.

Minimum value:

Test	Pivot half sum	Pivot range	Interval
Test 1	-0.1743164	0.10253906	-0.2224072265 < u < -0.1262255
Test 2	-0.1508789	0.032226562	-0.1686035156 < u < -0.13315429687
Test 3	-0.13232421	0.032226562	-0.150048828 < u < -0.1145996093
Test 4	-0.176757812	0.04296875	-0.2063203125 < u < -0.1471953125

Test 5	-0.1850585	0.02246093	-0.1974121094 < u < -0.172705078
Test 6	-0.19287109	0.0419921875	-0.2247431641 < u < -0.1609990234
Test 7	-0.1279296875	0.052734375	-0.15693359375 < u < -0.09892578125

Same as with the maximum, only the inferior threshold will be analysed.
Applying again the Horn method with this minimum we get:
Pivot Half Sum: -0.17818457025
Pivot Range: 0.056271484499999996
Upper interval: -0.14723525377500002
Lower interval: -0.209133886725
With a lower interval of **-0.2091**.

Distance value:

Test	Pivot half sum	Pivot range	Interval
Test 1	0.4140625	0.1328125	0.3517734375 <u< 0.4763515625
Test 2	0.32421875	0.0703125	0.285546875 <u< 0.362890625
Test 3	0.359375	0.109375	0.29921875 < u < 0.41953125
Test 4	0.318359375	0.13671875	0.24685546875 < u < 0.38986328125
Test 5	0.37109375	0.0390625	0.349609375 < u < 0.392578125
Test 6	0.3671875	0.0625	0.31975 < u < 0.414625
Test 7	0.341796875	0.13671875	0.2666015625 < u < 0.4169921875

Applying again the Horn method:
Pivot Half Sum: 0.43446484375
Pivot Range: 0.08377343749999999
Upper interval: 0.480540234375
Lower interval: 0.388389453125
With a upper interval of **0.4805.**

With a window frame:
Maximum value:

Test	Pivot half sum	Pivot range	Interval
Test 1	0.13372395	0.022656250	0.1230981770 < u < 0.14434973958
Test 2	0.10390625	0.042708333	0.0804166666 < u < 0.131395833
Test 3	0.15325520	0.057552083	0.1216015625 < u < 0.1849088541
Test 4	0.119010416	0.033333333	0.0960770833 < u < 0.14194375
Test 5	0.121484375	0.027343749	0.1064453125 < u < 0.13652343749
Test 6	0.101302083	0.036979166	0.0732348958 < u < 0.12936927083
Test 7	0.12578125	0.059374999	0.0931250000 < u < 0.1584375

The values of the Horn method with a window frame are lower than the real value because it does the mean with 5 values.
Applying again the Horn method:
Pivot Half Sum: 0.160716145795
Pivot Range: 0.04838541660999998
Upper interval: 0.18732812493049997
Lower interval: 0.1341041666595

The upper interval is **0.1873.**

Minimum value:

Test	Pivot half sum	Pivot range	Interval
Test 1	-0.114322916	0.08437499	-0.153894791 < u < -0.0747510416
Test 2	-0.09635416	0.04843750	-0.122994791 < u < -0.0697135416
Test 3	-0.0938802083	0.03880208	-0.115221354 < u < -0.0725390625
Test 4	-0.110026041	0.03671875	-0.135288547 < u < -0.0847635416
Test 5	-0.1380208333	0.03697916	-0.158359375 < u < -0.1176822916
Test 6	-0.1360677083	0.05078125	-0.174610677 < u < -0.09752473958
Test 7	-0.0963541666	0.03958333	-0.11812500 < u < -0.0745833

Applying again the Horn method:
Pivot Half Sum: -0.13455807250000001
Pivot Range: 0.038673437000000005
Upper interval: -0.11328768215000001
Lower interval: -0.15582846285000002
The lower interval is **-0.1558**.

Distance value:

Test	Pivot half sum	Pivot range	Interval
Test 1	0.295703125	0.08359375	0.2564976562 < u < 0.3349085937
Test 2	0.21015625	0.04062500	0.1878124999 < u < 0.2325
Test 3	0.2640625	0.075	0.22888749999 < u < 0.2992375
Test 4	0.218359375	0.06640625	0.18362890625 < u < 0.2530898437
Test 5	0.278125	0.0328125	0.260078125 < u < 0.296171875
Test 6	0.266015625	0.05234375	0.226286718 < u < 0.305744531
Test 7	0.234765624	0.082031249	0.1896484375 < u < 0.2798828125

Applying again the Horn method:
Pivot Half Sum: 0.3073957031
Pivot Range: 0.05502578119999996
Upper interval: 0.33765988276
Lower interval: 0.27713152343999997
The upper interval is **0.33766**.

ANALYSING METHOD 2:
And now with the second analysis and as before, data without a window frame will be analysed first, and then with the window frame.
Without window frame:
Maximum value:

Stop	Pivot half sum	Pivot range	Interval
Stop 1	0.232421875	0.12109375	0.1658203125 < u < 0.2990234375
Stop 2	0.224609375	0.08203125	0.1794921875 < u < 0.2697265625
Stop 3	0.205078125	0.09765625	0.1513671875 < u < 0.2587890625
Stop 4	0.1640625	0.078125	0.12109375 < u < 0.20703125
Stop 5	0.162109375	0.08203125	0.1169921 < u < 0.2072265625
Stop 6	0.177734375	0.01953125	0.16699218< u < 0.1884765625

Mean value of the higher interval value: 0.2256

Minimum value:

Stop	Pivot half sum	Pivot range	Interval
Stop 1	-0.134765625	0.08203125	-0.1798828125 < u < -0.0896484375
Stop 2	-0.12695275	0.1054695	-0.184960975 < u < -0.068944525
Stop 3	-0.15625	0.046875	-0.18203125 < u < -0.13046875
Stop 4	-0.1738278	0.050781749	-0.2017578375 < u < -0.1458979125
Stop 5	-0.1640625	0.0703125	-0.202734375 < u < -0.125390625
Stop 6	-0.150390625	0.06640625	-0.1869140625 < u < -0.1138671875

Mean value of the higher interval value: -0.1895

Distance value:

Stop	Pivot half sum	Pivot range	Interval
Stop 1	0.37890625	0.0625	0.34453125 < u < 0.41328125
Stop 2	0.421875	0.1171875	0.357421875 < u < 0.486328125
Stop 3	0.3671875	0.09375	0.315625 < u < 0.41875
Stop 4	0.396484	0.144532	0.3169914 < u < 0.475976599
Stop 5	0.353515625	0.08203125	0.3083984375 < u < 0.3986328125
Stop 6	0.341796875	0.07421875	0.3009765625 < u < 0.3826171875

Mean value of the higher interval value: 0.4104

With window frame:

Maximum value:

Stop	Pivot half sum	Pivot range	Interval
Stop 1	0.14140625	0.065625	0.1053125 < u < 0.1775
Stop 2	0.149609375	0.06484375	0.11394531249 < u < 0.1852734375
Stop 3	0.155859375	0.08828125	0.1073046875 < u < 0.2044140625
Stop 4	0.1053906249	0.05796875	0.083515625 < u < 0.136796874999
Stop 5	0.112890625	0.04453125	0.088398437 < u < 0.1373828125
Stop 6	0.1109375	0.0453125	0.086015625 < u < 0.135859375

Mean value of the higher interval value: 0.1467

Minimum value:

Stop	Pivot half sum	Pivot range	Interval
Stop 1	-0.085546875	0.06640625	-0.1220703125 < u < -0.04902343749
Stop 2	-0.0921875	0.075	-0.1334375000 < u < -0.0509375
Stop 3	-0.101953125	0.06328125	-0.1367578125 < u < -0.06714843749
Stop 4	-0.14140625	0.05468750	-0.1714843750 < u < -0.111328125
Stop 5	-0.1	0.1078124999	-0.159296875 < u < -0.040703125
Stop 6	-0.08515625	0.0484375	-0.111796875 < u < -0.058515625

Mean value of the higher interval value: -0.1391

Distance value:

Stop	Pivot half sum	Pivot range	Interval
Stop 1	0.250191375	0.04569524999	0.2250589875 < u < 0.275323762499
Stop 2	0.25703125	0.04375	0.2329687499 < u < 0.28109375
Stop 3	0.2703125	0.04999999999	0.2428125000 < u < 0.2978125
Stop 4	0.290535	0.09393	0.2388734999 < u < 0.3221965
Stop 5	0.2417968	0.0289062	0.2258984375 < u < 0.257695312
Stop 6	0.228515625	0.06953125	0.2002734374 < u < 0.2667578125

Mean value of the higher interval value: 0.2835

Next can be found the comparison of both methods values.

	Method 1	**Method 2**	
Maximum value	0.2786	0.2256	**No window frame**
Minimum value	-0.2091	-0.1895	
Distance value	0.4805	0.4104	
Maximum value	0.1873	0.1467	**Window frame**
Minimum value	-0.1558	-0.1391	
Distance value	0.3376	0.2835	

Comparing and analysing both methods with the stop features data, it can be concluded that using method 1 as thresholds, the values are too high to consider aggressive driving, letting all stop features values out of the aggressive driving, and contrary to this, method 2 stablish accurate thresholds for the driving style.

Turn analysis:
As the only turn that I have a proper data is the right turn, I will be analysing this right turn. For this analysis the turn has been taking “manually”.

Test	**Acceleration**
Test 1	-0.21015625
Test 2	-0.1571875
Test 3	-0.23828125
Test 4	-0.23125
Test 5	-0.19453125
Test 6	-0.19453125
Test 7	-0.209375

Applying the Horn method:
Pivot Half Sum: -0.18367187499999998
Pivot Range: 0.05296874999999998
Upper interval: -0.1545390625
Lower interval: -0.2050468749999998
Threshold: **-0.2050** (also the positive value)

Glossary

I2C: Inter-Integrated Circuit (Protocol)

LSB: Less Significant Bit

MSB: Most Significant Bit

SCL: Serial Clock

SDA: Serial Data

SSH: Secure Shell (Protocol)

UI: User Interface

DB: Data Base

RBE: Report by Exception

Tx: Data Transmit

Rx: Data Receive