Intelligent Acquisition and Analysis System for ECUs (IASE) is a joint effort of LISHA and Renault to investigate the possibility of applying **Artificial Intelligence (AI)** techniques within the paradigm of the **Internet of Things (IoT)** to optimize the operation of Internal Combustion Engines, particularly in respect to the calibration of controller's parameters and anomaly detection.
This project is funded by Fundep within the Rota 2030 Line V axis 2.

Table of Contents

[Show/Hide]

1. **Development**

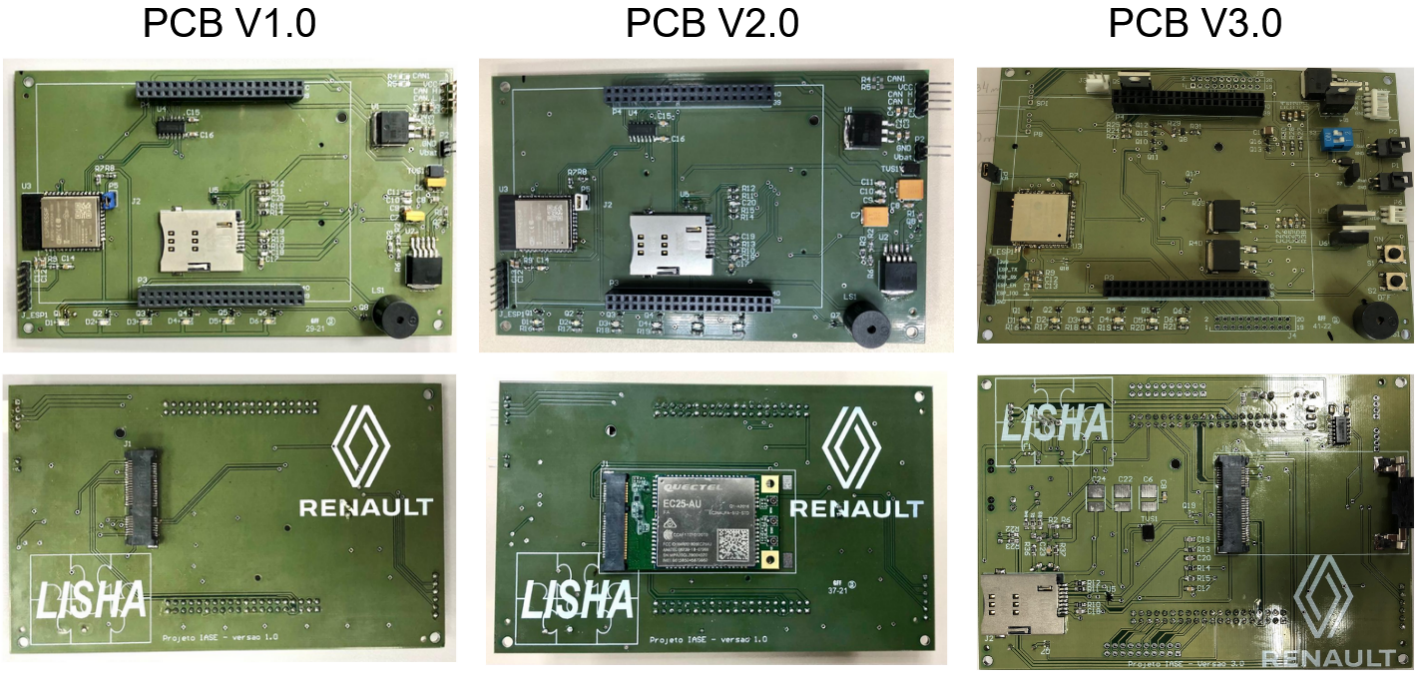

Figure 1. Developed Versions of the IASE hardware

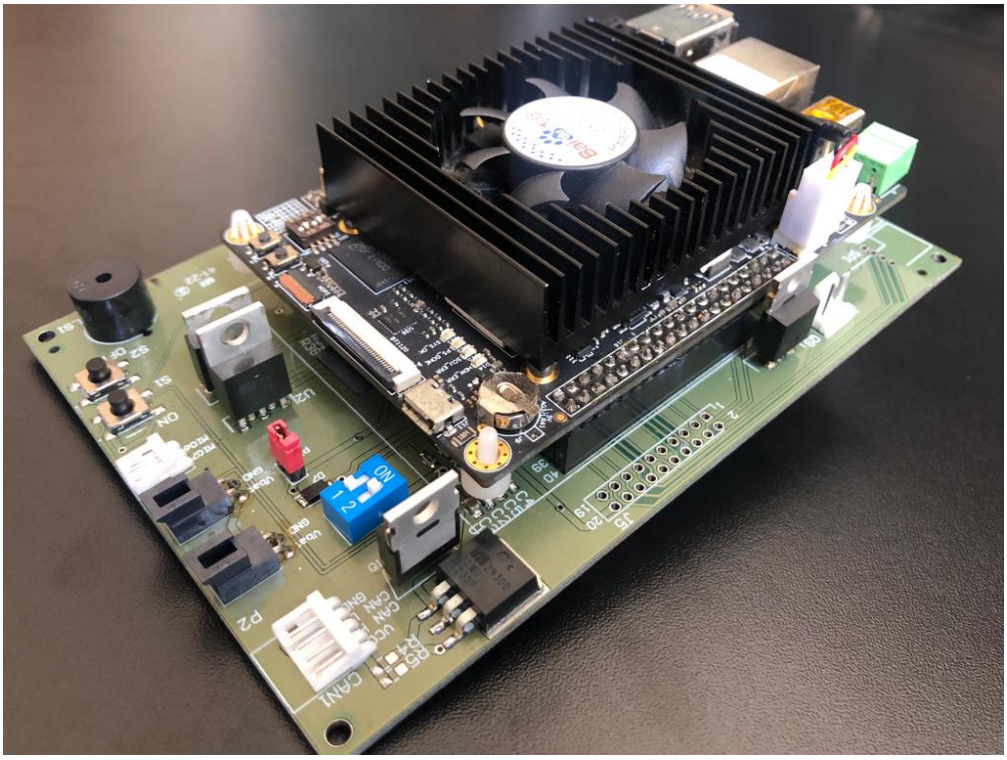

Figure 2. IASE hardware (PCB + FZ3 board)

Hardware

- CAN interface for data acquisition
- Memory capacity of 16GB
- Data upload via 4G cellular network
- LED status indicators
- Bluetooth Low Energy connection
- Ethernet and USB connection
- Power conditioning circuit
- 4GB RAM memory
- Zynq Ultrascale processor
- Quadcore ARM Cortex A53
- Dualcore ARM Cortex R5

Software

- CAN Protocol selection (XCP / CCP)
- Customizable list of variables
- 4G communication
- Offline operation possible
- More than 1 day data backup
- Android UI via Bluetooth
- Live status in LEDs and UI
- Safe communication with servers (SSL certificates)

Data Analysis

- Knock noise
  - **Description**
    - Engine knock is what happens when a portion of the fuel inside the cylinder detonates before the rest of the fuel. When your engine is running properly, the spark plug begins the process of combustion, burning up all the fuel inside the cylinder in one, controlled

explosion. If the octane rating of the fuel you have used is too low, or there is another damage elsewhere in the engine, some of this fuel may detonate prematurely.
    - Under specific input conditions, the autoignition of end-gas during the flame propagation leads to high temperature, acoustic resonances and high-intensity pressure waves whose reflection inside the chamber causes the knocking noise. These detonations create the knocking or pinging sound.
  - **Causes**
    - Too Low Octane: High octane fuels burn more uniformly and resist knock, contrastingly, low octane fuels are not as good for the vehicle and may produce noise when the engine works.
    - Bad Timing: The spark is not firing exactly when it should, this can cause multiple detonations in the cylinder, leading to engine knock.
    - Lean Air/Fuel Mixture: Oxygen sensor, fuel injectors, fuel pump or mass airflow sensor can create a lean air/fuel mixture in the engine which will not burn fast enough, allowing for multiple detonations.
    - Bad Knock Sensor: The knock sensor is designed to detect engine knock and inform the ECU to correct the problem automatically.
    - High Temperature: Within a high-temperature environment, the vehicle will be more likely to this phenomenon occurs.
  - **Symptoms**
    - Knocking or pinging noise coming from the engine, or even in worst cases, faults on the engine. This phenomenon can significantly increase carbon dioxide emissions, a better understanding and handling of this event reduces CO2 emissions. As expected, having those problems in detonations will degrade fuel economy.
    - Knocking/Pinging noise
    - Higher CO2 emissions
    - Lower fuel economy
  - **Detection**
    - This Section presents more details on the achieved results for each algorithm developed.

SELECTED VARIABLES

Sets							
A	B	C	D	E	Description	Target	Signals
x	x	x	x		Knock Signal	Input	4
x					Intake Air Temperature	Input	1
x		x			Fuel Consumption	Input	1
x			x	x	Intake Manifold Temperature	Input	1
x		x	x	x	Intake Manifold Pressure	Input	1
x					Engine Coolant Temperature	Input	1
x		x			Torque index	Input	1
x		x			Duration of 30° Crank	Input	6
x		x	x	x	Engine Speed	Input	1
x		x	x	x	Mean Knock Noise	Input	1
x		x	x	x	Engine Air Load	Input	1
x	x	x	x	x	Knock Detection	Output	1

Figure 3.SELECTED VARIABLES

- Isolated Forest
  - The development of this software component was based on IsolationForest method from sklearn ensemble library. In each experiment was defined parameters values, for instance, the row identified with 31 was defined with the contamination (0.0233), max features (18), and max samples (0.25). The row identified with 32 defined the contamination (0.021), max features (18), and max samples (0.25), and the 33 investigation considered the contamination (0.0138), max features (4), and max samples (0.25). Firstly, all variables presented in set A were used, but even adjusting parameter contamination, it did not reach better results than a F1-score of 26%.

So, we used only the knock signals (set B) with this method, and achieved a f1-score of 31% as presented in Figure 4.

RESULTS OF THE EXPERIMENTS

Id.	Algorithm	Split [Train/Test]	DS.	Exp.	F1-Score	Recall	Precision
1	Classifier	80%, 20%	1-4	D	0.49	0.40	0.61
2	Classifier	80%, 20%	1-6	D	0.81	0.83	0.80
3	Classifier	80%, 20%	2-4	D	0.63	0.52	0.79
4	Classifier	50%, 50%	1-4	D	0.20	0.79	0.11
5	Classifier	50%, 50%	1-6	D	0.26	0.79	0.16
6	Classifier	50%, 50%	2-4	D	0.21	0.79	0.12
7	Classifier	80%, 20%	1-6	E	0.12	0.13	0.12
8	Classifier	50%, 50%	1-6	E	0.05	0.04	0.09
9	Classifier	80%, 20%	1-6	D	0.45	0.46	0.44
10	Classifier	80%, 20%	1-6	E	0.00	0.00	0.00
11	Classifier	80%, 20%	1-6	D	0.12	0.13	0.12
12	Classifier	80%, 20%	6	D	0.28	0.50	0.19
13	Classifier	50%, 50%	6	D	0.14	0.26	0.10
14	Classifier	80%, 20%	4	D	0.50	0.36	0.81
15	Classifier	50%, 50%	4	D	0.01	0.01	0.03
16	Classifier	80%, 20%	1-5	A	0.00	0.00	0.00
17	Classifier	80%, 20%	1-6	C	0.78	0.80	0.77
18	Classifier	80%, 20%	1-6	D	0.12	0.12	0.11
19	Classifier	80%, 20%	1-5	C	0.00	0.00	0.00
20	Classifier	80%, 20%	1-6	B	0.72	0.78	0.66
21	Isolated Forest	75%, 25%	2	A	0.22	0.37	0.15
22	Isolated Forest	75%, 25%	2	A	0.26	0.42	0.19
23	Isolated Forest	75%, 25%	2	B	0.31	0.67	0.20
24	SVM	75%, 25%	2	B	0.00	0.00	0.00
25	SVM	75%, 25%	2	B	0.07	0.04	0.50
26	SVM	75%, 25%	2	B	0.21	0.15	0.38
27	SVM	75%, 25%	2	B	0.28	0.19	0.56
28	SVM	75%, 25%	2	B	0.36	0.27	0.56
29	SVM	75%, 25%	2	B	0.39	0.47	0.33
30	SVM	75%, 25%	2	B	0.48	0.50	0.46
31	Conv. Autoencoder	Healthy, Faulty	2	A	0.53	0.79	0.39
32	Conv. Autoencoder	Healthy, Faulty	2	A	0.55	0.72	0.44
33	Dense Autoencoder	Healthy, Faulty	2	A	0.31	0.67	0.20
34	Dense Autoencoder	Healthy, Faulty	2	A	0.32	0.82	0.20

Figure 4. RESULTS OF THE EXPERIMENTS

- Support Vector Machine - SVM
  - The implementation of this method used the sklearn SVM module, which has the C-Support Vector Classification (SVC) function allowing to apply different kernels as parameter. The dataset was split using the sklearn.model_selection.train_test_split function, setting the test part to be 25% to the dataset. This method used variables presented in set B, and we iteratively changed the kernel of the models to try to improve the f1-score. The use of linear and sigmoid kernels in this model was not able to predict anything from the present dataset, as presented on Figure 4 on id 34.

    - At row 25, it was defined the parameter using the Radial Basis Function (RBF) kernel, but it achieved a f1-score of less than 10%. The polynomial kernel achieved better results with degree 3 to 7 on rows identified by 26 to 30, but increasing the degree level also increased the training time. After the degree level of 7, it could not converge, so best performance for this model was a f1-score of 48%, as presented in Figure 4 identified by the rows 24 to 30.
- Dense Autoencoder
    - The dataset used in this method had 7877 data points to each of the 19 signals analyzed and was divided as mentioned before, in healthy and faulty periods. The implementation of this method used the Keras library to build the dense autoencoder architecture, starting with the total number of inputs (signals) decreasing until a point and then returning to the number of inputs, resulting in the characteristic bottleneck.
    - With this model the loss threshold was tweaked to understand how it would affect the f1-score, using the same architecture. The best result was an f1-score of 32%, where it got 246 false positives, and 14 false negatives. The comparison between different thresholds can be seen in the Figure 4 from row 33-34.
- Convolutional Autoencoder
    - According to the literature review, a CNN can be a great choice for detecting faults using the vibrational signals. So, a model was created, making use of the convolutional layers. The development of this model was similar to a dense autoencoder, but it can analyze windows of datapoints at a time, instead of single datapoints in time. It predicted 70 false positives which was better than the dense model, and 21 false negatives. The comparison between different window sizes can be seen in the Figure 4 from row 31-32.
- Feature Extraction Classifier
    - This model is a classifier, but instead of directly classifying values into temporal sequences, it first extracts a more meaningful representation using a convolution layer. Thus, the classification phase can better identify the differences between the samples. Many parameters have been applied to find the best accuracy, so each experiment was conducted with a different setup. It was considered a split process, dataset (DS), experiment (exp.) that defines the input/output variables, and a different parameter such as internal architecture. Experiments from row 1-8,12-20 considering batch size 128 and sequence length of 64, rows 9-10 uses the same architecture but batch size 64 and sequence of 32. The experiment on row 11 uses another architecture, starting with an Input Layer of sequence length by feature numbers, followed by two layers of convolution 1D and max polling 1D, it is followed by the flattening process. Afterward, four layers of dense(sequence length // w) and dropout(rate=0.01), thus, each layer has a different w starting on 2, 4, 8 until 16. Finally, a dense(1) layer is set to define the classification process. All the results are presented in Figure 4 from row 1-20.
- Comparison between algorithms
    - Results are presented and the best result was the selected variables and feature extraction variables on the Feature Extraction Classifier with f1-score of 81% and 78% of engine knock detection in comparison to using just the signals 72%, at Figure 4 respectively 2, 17, 20. The convolutional autoencoder with 55% of accuracy of f1-score, followed by SVM 48%, dense autoencoder 32%, and isolated forest with 31% as presented in the Figure 5 of f1-score. It is also presented the recall results that vary between 50% to 83%, and the precision results were between 20% to 80%.
    - Experiments were conducted with the same architecture, dataset, and input variables, but with different split process on train/test achieved lower accuracy on experiments with the same amount of the dataset for train/test. It is possible to see at Figure 4 that 1, 2, 3 had 80/20, and 4, 5, 6 had 50/50 with the same characteristics but lower f1-score. The experiments that were conducted not using the knock signals (set E) achieved a maximum accuracy of 12%, so it leads to understanding that it is difficult to detect the engine knock fault without the vibrational signal just analyzing other aspects.
    - Some of the reasons known that could be interfering in the results of the algorithms are that the

piezoelectric is acquired in high frequency, but while entering the ECU, due the sampling of the XCP causes the downsampling of the signal. Another hypothesis is that the vehicle has already implemented some features that try to minimize the knocking noise occurrence, so ECU may be detecting its occurrence and trying to readjust so that the occurrence does not interfere much in the process. Even though this readjustment could affect other variables acquired, that leads to low accuracy in our models.

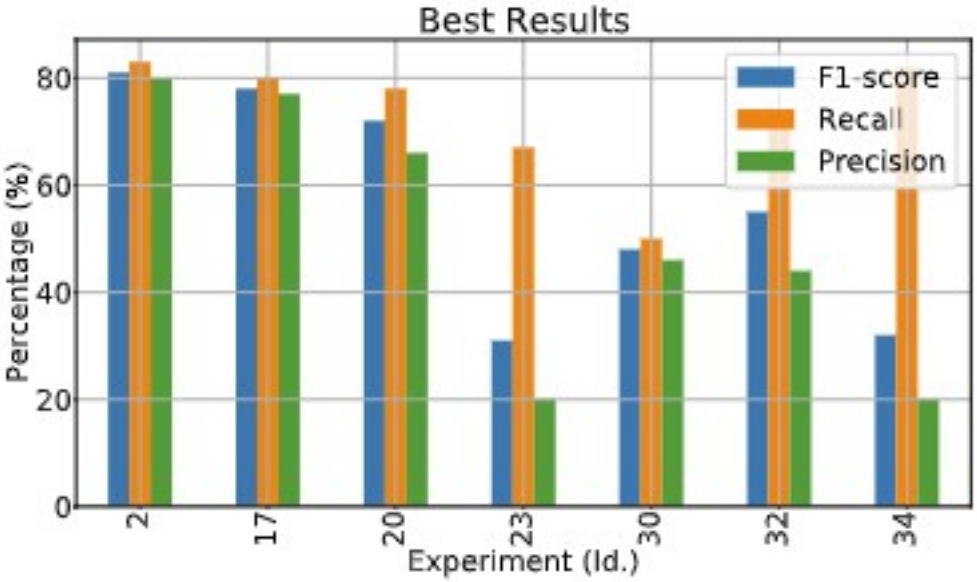

Figure 5. BEST RESULTS

- Misfire
  - **Description**
    - Internal combustion engines use combustion gases themselves generated as a form of work. In other words, the gases generated in combustion perform the compression stages and the temperature increases, that cause the gases to burn, expand and the exhaust. In general, there is an increase in internal pressure due to the high temperature and burning of gases, the misfire failure is when this burn is not carried out correctly in the combustion chamber, leaving the engine without power.
    - Misfiring creates a unique pattern attributed to a particular cylinder. When a misfire occurs, the balance of the engine is destroyed, and the generalized force at the centre of gravity of the engine is changed.
  - **Causes**
    - Worn spark plug: the wear of the electrodes can causes the inefficiency of the spark, in addition, this can overload the spark plug, the ignition coil cables and the catalyst;
    - Spark plug wire: if the wire seal is damaged, the spark does not reach its destination correctly, impairing the combustion;
    - Ignition coil: this coil transforms the voltage received by the battery into voltage to create an electrical spark in the spark plugs to start combustion of the fuel. If the voltage produced is not correct, it may cause the misfire. This is considered the most common cause when it comes to misfires;
    - Valves: the chamber where the valve is located may contain the presence of charring, which prevents proper sealing of the same.
    - Pistons, rings and cylinder liners: engine wear, especially excessive clearance between its components, can cause oil to burn and foul the spark plug, preventing a perfect burn.
    - Fuel: using bad or adulterated fuels, in addition to damaging the engine and its components, produces a higher rate of pollutant gases and causes this type of failure.
    - Inlet manifold: leakage from the inlet manifold gasket near the cylinder heads;

- Fuel Pressure: Low fuel pressure can be caused by a faulty pressure regulator, faulty fuel pump, or a clogged fuel filter. Low pressure will cause a lean mixture in the engine, consequently causing misfires in all cylinders;
- Low Compression: A faulty timing belt adjustment can cause low compression and therefore misfire;
- Wrong air-fuel mixture: Sensors can cause a faulty air-fuel mixture, such as MAF sensor, O2 sensor, coolant temperature sensor, etc.
- Some other situations can be the cause such as defect in the injector, water in the washing coils, incorrect adjustment of the ignition system, incorrect adjustment of the valves, etc.

◦ **Symptoms**
- It is possible to notice symptoms of misfire during everyday driving of the vehicle.
- Brute acceleration: car moves in jerks when stepping on the acceleration pedal, you can feel it as a strong jolt coming from the engine. The most common situation to detect misfire is in high gear, low RPM and the throttle on the ground. Raw throttle is a typical sign that the engine is failing;
- Rough Idle: very irregular idling, which may cause the engine to shut down. Thus engine sensors will get faulty values and the fuel mixture will be confused;
- Vibrations: when one or more engine cylinders are not firing correctly, the engine is out of balance, causing strong vibrations inside the cab when accelerating or idling;
- Engine light: If a sensor has failed or has detected that something is not right with the engine, it will send the information to the engine control unit. The unit, upon receiving the data, will decide if the problem is serious or not. If the problem repeatedly occurs, the engine control unit will illuminate the check light. When the ECU detects misfires, it is very common to turn on the engine light and store a fault code in the cylinder where the engine fired;
- Slow Acceleration: As stated earlier, O2 sensors can receive incorrect information and generate very lean or rich mixtures. Poor or too rich mixtures can cause a decrease in acceleration, and may even turn off the turbocharger pressure;
- Strong smell of unburnt fuel gas in the exhaust;
- Change of engine sound.

◦ **Detection**
- In the following topics, details related to the development of the project and the experimental evaluations are discussed, explaining how the data acquisition was carried out, the standards used, with an overview of the platform and, finally, the results found.
- A 2019 Renault Sandero, model 1.0, with a four-stage four-cylinder spark-ignition gasoline engine was allocated for testing and experimentation. Thus, with the car available, it was possible to collect a significant amount of real data, with or without misfires. The project is integrated by other students, each one is responsible for a part of the work and it is essential that everyone collaborate and advance together so that the general objective is achieved. In the course of the text some activities of other members are mentioned because they complemented the development of this dissertation.
- The hardware responsible for collecting the desired data directly from the ECU was embedded in the vehicle, via the CAN Calibration Protocol. This device communicates via 4G with the LISHA cloud server, so that the data is processed in the cloud, with greater computing power available, and stored in a database for later use, including in new studies and projects. Figure 6 illustrates the communication, from reading the ECU variables to their respective storage on the server, more details are explained below.

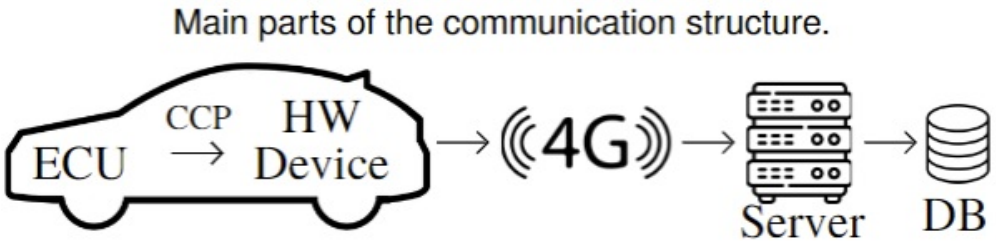

Figure 6 MAIN PARTS OF THE COMMUNICATION STRUCTURE

- Data acquisition and Experiments
  - One of the points that differs this study from most related works found in the literature is that all data used in the experiments were exclusively taken from the ECU, with the sampling rate limited by its capacity, which is lower than that of the sensors. Thus, a communication architecture was structured and the initial part of the project was intended to build a device capable of communicating with the ECU through protocols such as the CAN Calibration Protocol and the Universal Measurement and Calibration Protocol. For the construction of the data sets, more than thirty experiments of fifteen minutes each were carried out, distributed over a semester. The occurrence of misfire in the experiments ranged from none to more than one hundred, because of its unpredictability, this control was difficult.
  - To start the data acquisition process, the hardware embedded in the vehicle must be loaded with the experiment file. Which indicates the location of the signal in the ECU, including the definition of Smartdata (Unit, Dev, X, Y, Z, Signature, Type, Period, Workflow) and the range of minimum and maximum values of the signals (pre-set to check if the measurement is as expected). The data from the experiments are sent to the server. The communication between the device and the ECU has a limitation on the sampling rate, separated into four rates: 4ms, 5ms, 10ms and 100ms. The data acquisition used in this work was time-triggered, during this phase, data is buffered and prepared for serial upload via 4G to the IoT platform using the SmartData format.
  - With the device embedded in the vehicle, numerous experiments can be loaded onto it. This allows a wide range of tests with different objectives to be performed, each of these experiments can take into account different approaches to misfire detection, which is excellent for comparing and evaluating the developed algorithms.
  - At the moment, the focus of this qualification was to develop the algorithms to test the first detection techniques to be applied and needed more controlled data. Therefore, most of the ECU data acquisition was done using the Integrated Calibration and Application Tool (INCA) from ETAS, which is used by Renault. This software allows misfires to be forced into the engine by decalibrating system variables related to misfire. Although it was used to communicate the ECU with the server and even though it can read the variables, the hardware developed in the project does not do this decalibration, with it the car needs to be forced by the driver, it is not possible to control when the failure occurs and in what amount.
  - Unlike most of the related works, which carry out simulations of failures on test benches, the measurements in the project occurred at the car in operation, driven by project members, on the limited test track within the Joinville campus of the Federal University of Santa Catarina. In order to obtain different data sets with and without failure for training the algorithms, the experiments in the vehicle took place on different days and more than once a day, with 1 to 3 experiments of 8 to 15 minutes being generated. The practices were limited to that time due to overload in the engine, the test car showed signs of degradation when being forced for longer periods, precisely because the failure can harm the engine and its functioning.
  - Several experiments were carried out to understand the misfire in different scenarios (engine in fault and normal conditions, car in stationary and moving state, high and low gear and different speeds). The experiments were performed with variables that describe basic car components and systems, such as engine status, and others related to misfire, initially taken from the bibliography of causes and related works. Numerous machine learning algorithms were applied to analyze each set of variables. The algorithms training was done with the data collected in the

experiments, for the supervised training the misfire counter variable of the ECU was used, for the other cases it was disregarded. Varying according to the algorithm, the training parameters were defined during its execution, observing metrics such as cross entropy and squared error loss, among others, and observing the convergence of the loss. Some of these experiments are detailed in the results section.

- Subsequently, to confirm the understanding of the causes of the failure and to improve the performance of the misfire detection algorithms, feature selection algorithms were developed by another member of the project, to be applied in the construction of new data sets. Having more than twenty thousand variables available in the ECU, it is necessary to analyze their importance and choose the essential ones for the misfire detection. It is noteworthy that the fault counter generated by the ECU was used as a reference parameter to confirm whether the data read was a fault or not, and also used to label the data so that the feature selection algorithms could be trained with the variables that symbolize failure over time and also for supervised machine learning training. Thus, after the process of selecting the variables and forcing the car to fail, it was possible to obtain large data sets for analysis of the failure identification algorithms.

- The feature selection methods applied were:
- SelectPercentile: selects resources with a higher scoring percentage specified as parameters, 10\%, 15\% and 20\% were used;
- SelectKBest: selects the features with the highest score according to the function chosen as a parameter, the perason correlation coefficient was used to find the best attributes of a dataset;
- Sequential Feature Selector: is a greedy procedure that finds the best feature to add to the subset of features that starts empty. First finds the feature that maximizes a cross-validation score when an estimator is trained with a single feature. Once the first feature is selected, the procedure is repeated adding a new feature to the subset and stopping when the desired number of selected features is reached;
- Recursive Feature Elimination: Select features recursively considering smaller and smaller sets. Initially it is trained on the complete set where the importance of each feature is obtained in relation to a specific attribute. Subsequently, the less important features are removed from the set, a procedure that is repeated until the desired number of features to be selected is reached. A version of this method in a cross-validation loop to find the optimal number of features was also used, called \textbf{Recursive Feature Elimination with Cross-Validation}.
- SelectFromModel: is a metatransformer used with any estimator that assigns importance to each resource, in relation to a specific attribute, the random forest regressor was used as an estimator with square error criterion. Resources are removed if their importance is below the given threshold parameter. Also, the process ends when a desired number of features to be selected is reached.
- Among the most selected features, which were used in the fault detection, the following systems and their respective variables stand out (Figure 7):

Feature selection

Engine systems	Variables
General	Engine coolant temperature, engine speed, vehicle speed, current gear engaged, dead center counter;
Air Control	Engine air load, atmospheric pressure, intake manifold pressure, intake air temperature, throttle valve position, intake manifold temperature;
Fuel And Richness Control	Final alcohol adaptive, injection time - cylinder 1 - 1st injection, injection time - cylinder 2 - 1st injection, ijection time - cylinder 3 - 1st injection, injection time - cylinder 4 - 1st injection, air/Fuel Ratio, richness regulation status, richness adaptation factor, catalyst Diagnosis Criteria (OSC Level), catalyst exhaust gas upstream oxygen sensor voltage, catalyst exhaust gas downstream oxygen sensor voltage
Ignition Advance	Intake camshaft phaser position, ignition advance
Engine Torque	Engine torque without gearbox request, estimated effective engine torque, effective engine torque target requested by real (pedal) and virtual (ACC/CC/SL) drivers, engine torque losses, final indicated torque raw, final indicated torque target.
Alternator and Battery Management	Alternator load, battery voltage, alternator power, battery state of charge, filtered alternator rotor current.

Figure 7 FEATURE SELECTION

- IoT Platform and Workflows
  - In an overview, the IoT Platform is organized as microservices that provide secure storage and data processing for SmartData series. The Microservice Manager is an IoT front-end and the Domain Manager handles microservices requests, being responsible for authentication and mapping SmartData sets. The SpaceTime Mapper, on the other hand, maps regions of space and time to the SmartData that are stored or to be stored on the platform. The execution of data science algorithms on SmartData entering the platform are the responsibility of the Insertion Managers and those leaving the platform are of the Retrieval Manager.
  - SmartData is characterized by a version, a unit and source coordinates that show where and when SmartData was produced, created, captured, sampled, etc. Stored on the Platform, it is a data point in a SmartData time series. The SmartData stored and processed by the platform have the following Json representation:

□□□□□□□□

{ "version" : unsigned char "unit" : unsigned long "value" : double "uncertainty" : unsigned long "x" : long "y" : long "z" : long "t" : unsigned long long "dev" : unsigned long "signature": string }

  - SmartData series are classified based on the mode of operation: time-triggered or event-triggered, the former must define a period, while the others are considered event-triggered. The start of a series can be specified by time (by giving t0), by event, or manually (by not giving t0, which is then assumed to be the current time). Thus, the start of a time-driven series can be an event, and similarly, event-driven series can start at a certain time. The end of a series can be specified by time (by giving tf), by event, manually (with the finish method, which makes tf equal to the current time), or in terms of event count (by giving count). Events are internal (stored on the platform) or external SmartData, arithmetic and logical operators. The SmartData Series stored and processed by the platform have the following Json representation:

□□□□□□□□

"Series" : Object { "version" : unsigned char "unit" : unsigned long "x" : long "y" : long "z" : long "r" : unsigned long "t0" : unsigned long long "tf" : unsigned long long "type" : char[3] "period" : unsigned long "count" : unsigned long "event" : string "accuracy" : unsigned long "workflow" : unsigned long }

  - The SmartData can be directed to a specific Workflow for data processing before its definitive insertion into the platform or its manipulation afterwards. Workflows are used by the IoT platform to execute server-side algorithms, related to the received series, which can be defined as input or output. The SmartData concept is used to add metadata to the read data in order to make it semantically complete, with a spatial location and reliability. In this way, it is possible to identify different devices in different locations, in this case vehicles, without adjusting the settings of each one. Also, being a common data format, there is no need to adapt for each ECU, so the same algorithms can be used in all cars.
  - An input workflow can be specified during the creation of the series and its execution occurs during insertions (PUT method) of SmartData in this Series, being applied to each SmartData individually and persisted through daemons. It can be used to pre-process data, execute algorithms, fix data points, generate notifications and interact with other series. Daemons are sub-processes that receive data from the workflow, do the processing and return it to the workflow, or insert the processed data into a new series, preserving the original data.
  - An output workflow can be specified during a query request (GET method) and its execution is applied at the end of a query process to consider all returned SmartData records, It can be used to post process the data, to perform aggregations or transformations.
  - Workflows are used because they can be executed in real time, applying corrections to measurements and running machine learning algorithms for misfire detection, for example,

notifying at runtime if faults are detected, in addition to providing real-time data visualization.

- Results
  - The accuracy of all models were high due to the nature of the failure, more data under normal conditions are measured and for this situation the models are great to evaluate. As the failure occurrence is periodic, with few measurements, the datasets contain less cases for training and testing this scenario, so even though it is not good at detecting the occurrence of failure, this metric has a high performance in the general model, so it was not considered. Instead, F1 was defined as the metric to be considered, it takes into account how much the model can differentiate the classes and the percentage of hits made by the model, that is, how much of the failures were correctly identified. Figure 8 summarizes the average results found for the developed and tested algorithms, in very specific scenarios the rating reached 60% F1, but not constantly and therefore are not being considered.

Results in failed scenarios

Algorithm	Accuracy	F1-Score
KNN	95.00%	12.00%
Isolation Forest	86.00%	02.00%
CNN	91.49%	17.38%
1D CNN	98.20%	28.72%
Autoencoder	88.00%	18.00%
LSTM Autoencoder	95.30%	23.08%
MLP	97.26%	20.12%

Figure 8. RESULTS IN FAILED SCENARIOS

- After understanding how each algorithm evaluates the data, a number of tests with different data sets and different parameterizations were made. As can be seen, the maximum F1 found was using an 1D CNN algorithm, although it did not vary much in performance compared to the other algorithms based on neural networks, the distance algorithms performed worse. All algorithms performed with 100% F1 and accuracy in faultless scenarios.
- Despite the large amount of work allocated to the development of machine learning algorithms and the creation of training and test sets, the results found were not sufficient to replace the detection methodology used by Renault at the time. It was concluded that because it arises from numerous engine systems and external factors such as environment and driver, a misfire model is quite complex and difficult to build, even with the application of machine learning algorithms. The results found are not applicable in the industrial context, it was not possible to surpass the methodology currently used in terms of performance in the classification. A failure detection delay was observed, impairing the evaluation metrics. It is noteworthy that the expected detection may be lower than that found in related works due to the difference in the sampling rate provided by the ECU, as the algorithms only have access to ECU data, it is not possible to read directly from the sensors and this implies sampling rate and data quality, different from what was seen in most of the related work articles.
- It is noteworthy that although the results of the model were not better than the strategy used by Renault, the extensive work developed proved to be adequate for the operationalization of ML for use in production. As it is a machine learning system designed to be applied at the production level, a number of problems encountered in its implementation and certain concerns are not found in the literature. Therefore, testing and monitoring are important considerations to ensure the system's operation and applicability, avoiding its failure, these issues were constantly worked on in the project.
- As a methodology for data acquisition and processing was developed, new alternatives analysis were made. So, even though the misfire detection results are not applicable at the industrial level, the developed framework is. In this way, a change of objective was made for the continuity of the master's degree, starting to be focused on the analysis of vehicle consumption, with the proposal of creating a detailed feedback for the engineers, in bench tests, and also for the drivers, with the car in use. This new proposal is detailed in the next topics of the document.

- Driver profile analysis (Classification)
  - This research aims to perform an analysis on two aspects of fuel consumption: a first one related to driver behavior, a second on engine and vehicle calibration parameters.
  - In this section, our goal is to implement models to classify the fuel consumption as pro or non-economic, low or high consume, through the instantaneous data gathered from the vehicle.
  - Classifiers are learning models applied to data labeled with classes or categories, which objective is to predict what class each data record belongs to. In this research context, the classes consist in levels of consumption, categorizing how much the driving behavior at the moment stands for a higher or lower consumption.
  - For this research, we chose to partition the data in two classes: a “low consumption” and a “high consumption” class. The division criterion is based on an analysis of the distribution of recorded instantaneous fuel consumption, gathered on experiments. Observing the distribution patterns, a threshold of 5.5 l/h was set, considering that above this is a high consumption record, and below this a moderate to low consumption.

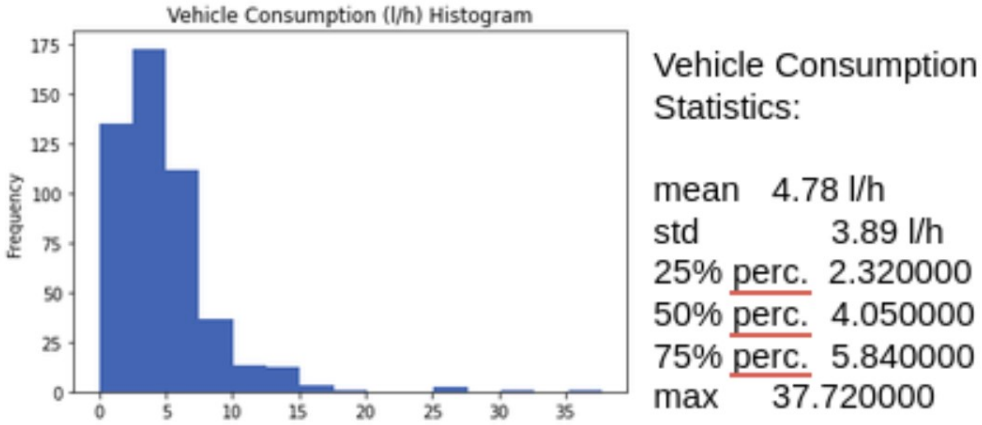

Figure 9. HISTOGRAM AND DISTRIBUTION STATISTCS OF FUEL CONSUMPTIOM (l/h)

- There are different models of classifiers, being the mainly used in this research the XGBoost, optimized implementation of the Gradient Boost method. The Gradient Boost is an ensemble, a method of combining weaker learners to perform as a strong one, in this case of decision trees, where different learners are trained in sequence with the errors of the previous.
  - The model was trained and tested using data collected from experiments with the vehicle. For an analysis of the results, we use the following metrics:
    - Precision: tp/(tp+fp)
    - Accuracy: (tp+tn)/N
    - Recall: tp/(tp+fn)
  - Where:
    - tp: number of true positives
    - tn: number of true negatives
    - fp: number of false negatives
    - fn: number of false negatives
    - N: number of total samples
  - Interpreting “positive” as classifying the data as “high consumption”, and “negative” as “low consumption”.
  - The three used metrics are ratios, ranging from 0 to 1, or equivalently 0% to 100%, where higher measures means the better performance. Precision measures how much the model predicts correctly the positives; recall measures the proportion of predicted and existing positives; and accuracy measures the proportion of total correct predictions in relation to all tested samples.

- Below, the training and testing results, where one experiment is used as training set for the model, and another one as test set.

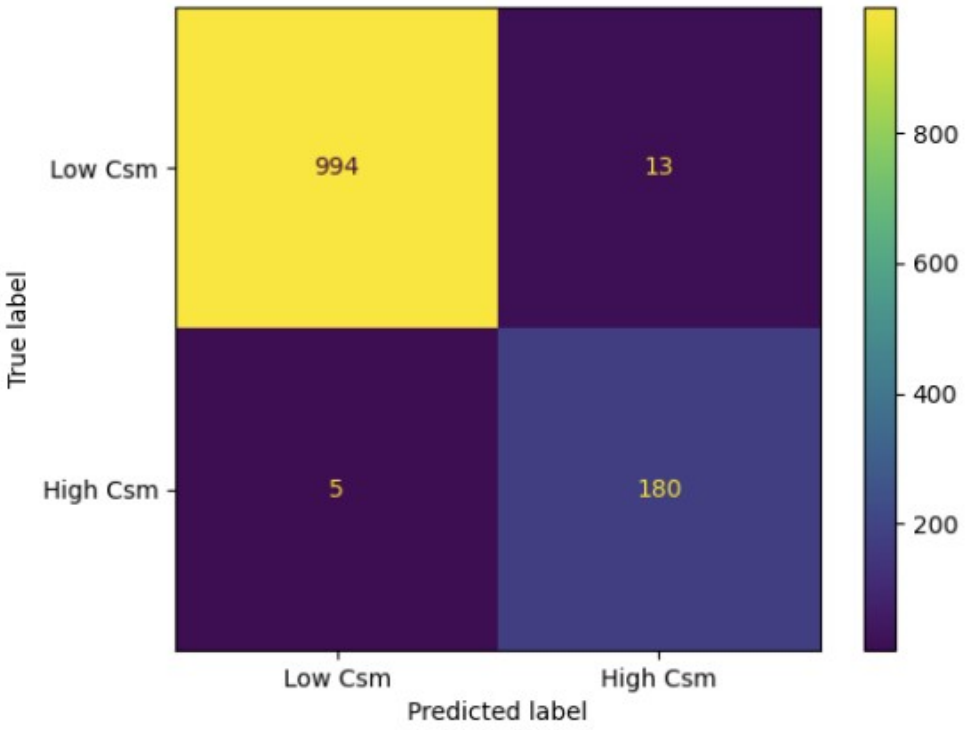

Figure 10. PREDICTED LABEL

- Accuracy: 0.9848993288590604
- Recall: 0.972972972972973
- Precision: 0.9326424870466321
- It may be seen that the model have a good performance, showing good results for all three analyzed metrics, in special a 98% of accuracy. By the confusion matrix, it can be observed as well that very few data points are incorrectly classified, counted in the numbers shown in the lower left and upper right entries, while the majority of the points classes are correctly predicted, that can be seen in the remaining entries.

- Fuel Consumpition Regression
  - Regression models are statistical and machine learning methods that aim to predict continuous numerical values. This section objective consists in applying regression methods to predict the fuel consumption rate, based on another available measures gathered from the vehicle.
  - Among the different existent regression models, we focus on two of them that shown good results at the executed tests: the Ridge and the XGBoost regressors.
    - Ridge is a type of Linear regression applied with a regularization technique, the Tikhonov Regularization. Essentially, this model determines the parameters $\theta 1, \ldots, \theta n$ minimizing the loss function $L(\theta) = (\hat{y}1-y1)^2 + \ldots + (\hat{y}n - yn)^2 / n + \alpha(\theta 1^2 + \ldots \theta n^2) / 2$, in order to construct the predictor in the form: $\hat{y}(x) = \theta 1x1 + \ldots + \theta nxn$. In this notation, $x = (x1,\ldots,xn)$ is the feature vector, $\hat{y}$ the prediction vector, y the real value vector and α an regularization parameter. In this work, the Sci-kit Learn implementation of the Ridge Regression is applied, with the default parameters provided by the library.
    - The XGBoost Regressor is an implementation of the Gradient Boosting applied for regression learning. It uses the same working principle as the Gradient Boosting Classifier, interpreting value ranges and some intermediate value as classes results. Regularization techniques are applied to control the number and range of output values.
  - In order to analyze both methods performances, the following metrics are applied:

- Root mean square error (RMSE);
- Absolute mean error (AME);
- Max error;
- Coefficient of determination(R^2).

- Both RMSE and AME are mean errors of the predicted set, where the first uses the quadratic error that show greater penalty for higher errors. The Max Error, as the name suggests, is the maximum error in absolute value obtained by the model predictions. Finally, the R^2 Score is a ratio between the explained variance and the observed variance, going from 0 to 1, being better as greater the value.

- Below, some of the found results using the experimental data gathered in the vehicle, where one of the experiments is used for test and evaluation, and the remaining for training the models.
- Ridge:

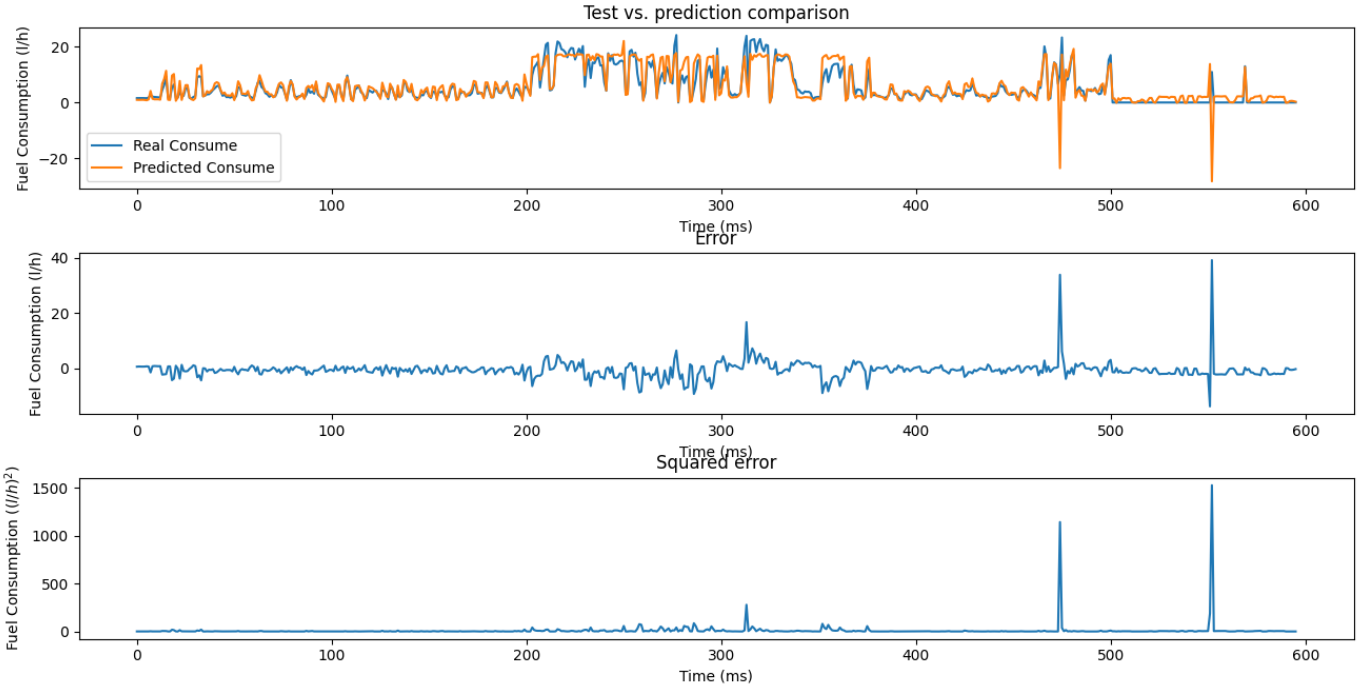

Figure 11.TEST VS. PREDICTION - COMPARISON

- Root Mean Squared Error: 10.075410892285648
- Abs Mean Squared Error: 1.6895357220015024
- Max (abs) error: 39.107547185749105
- R^2 determination coefficient: 0.7182110380492899

- XGBoost:

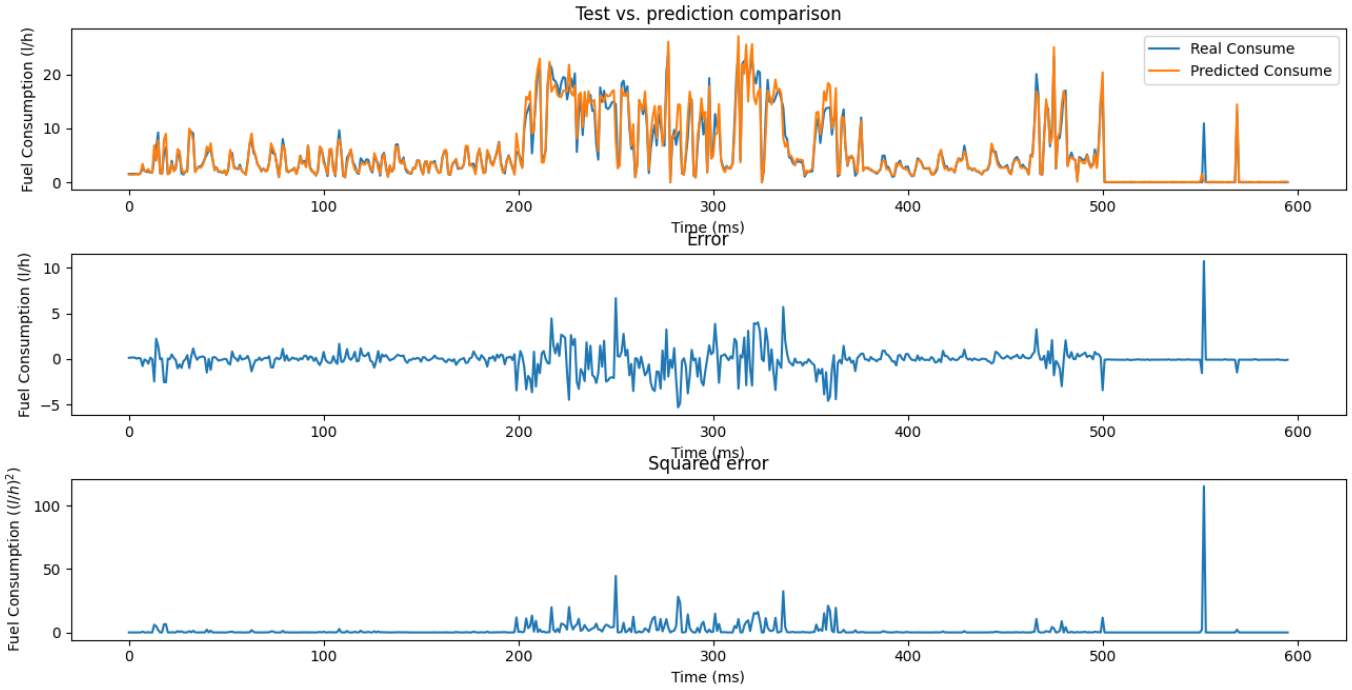

Figure 12.TEST VS. PREDICTION - COMPARISON

- Root Mean Squared Error: 1.6858840509531254
- Abs Mean Squared Error: 0.7039533121878284
- Max (abs) error: 10.733944220542908
- R^2 determination coefficient: 0.9490652829412747

- Both models show an overall good performance, having an absolute mean error around 1 l/h and 2 l/h. Also, the two models presents R^2 scores above 70%, meaning that the models can predict quite correctly the variance in the measured data.
- As may be seen, the Ridge mode shows a lower performance, having greater errors and a lower R^2 coefficient, what may be caused by the nature of the linear model, that may not consider the complexity of non-linear phenomenon. Whereas, the XGBoost regressor shows a great performance, that may be observed by the metrics as well by the plotted graphics, showing an error oscillating close to 0 in general, with few peaks.

- Automatic Report
  - ECU's Failures
    - During an experiment, the electronic part of the vehicle may present some failures, which the ECU reports to the IASE Board. Then the board sends the value of this failure to the server, where it is treated and reported directly by a Workflow to the tester engineer by email, with the failure information, such as the component and type of failure, the time when the failure happened, the vehicle chassis and the repair hints.

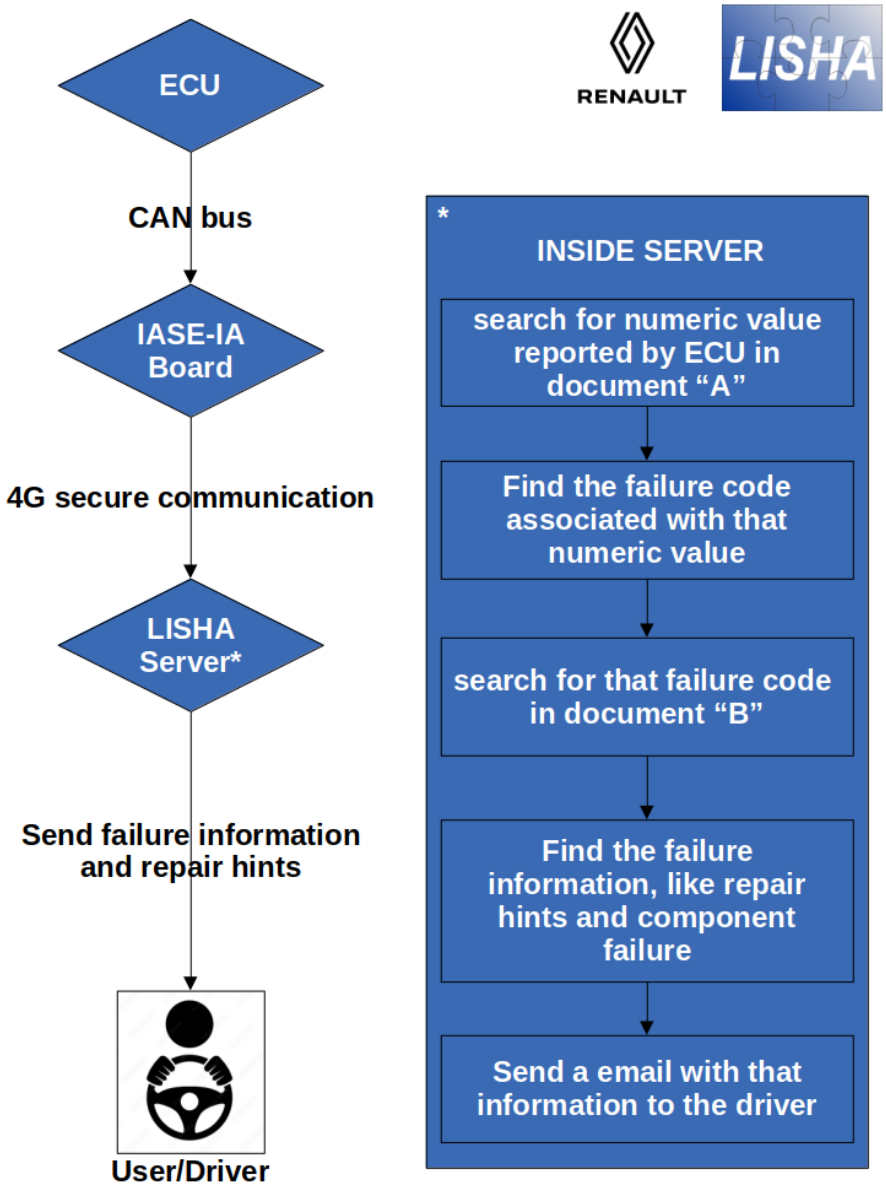

Figure 13. ECU's failures automatic treatment diagram

- An email report example:

GPS	**x:1500 ; y:567 ; z:1298 ;**
Timestamp	**2023-02-07 19:45:30;**
Faulty Component	**Temperature Sensor;**
Repair Hints	**Check the “sensor ABC”.** **If is ok:** **…** **If is not ok:** **…**

Figure 14. ECU's failures report example

  - Rangecheck Report
    - Upon completion of an experiment, it is necessary to conduct an assessment of its overall situation to check whether variables are within minimal and maximal ranges (defined by the engineering team). Therefore, a workflow processes the signal indicating the conclusion of the experiment then extracts both the starting and ending times and analyzes all previously configured variables. The output of the workflow is a PDF report file summarizing the

experiment's general status, including graphics for all variables that have at least one value outside the defined ranges.

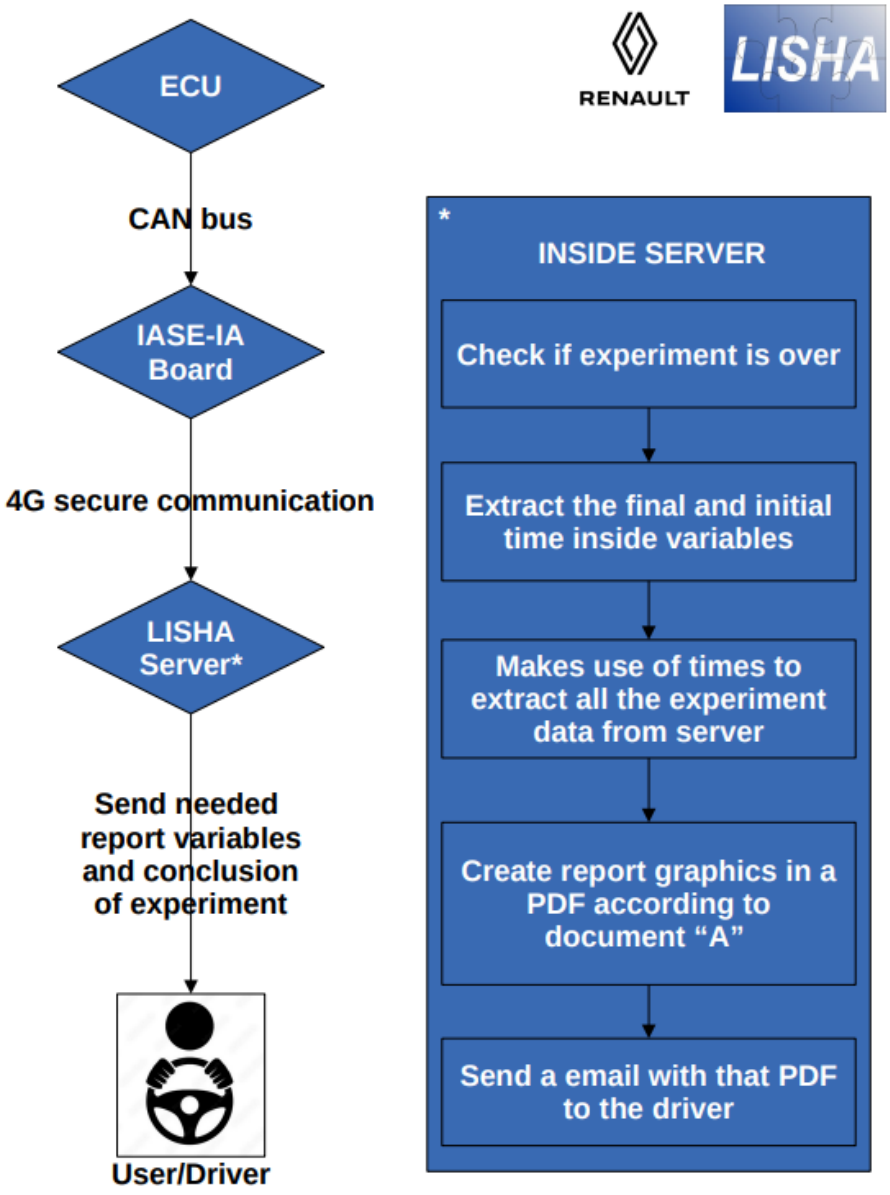

Figure 15. ECU's rangecheck automatic report diagram

2. **Demonstrations**

Demonstration video - IASE

3. **Team**

Researchers

- Prof. Giovani Gracioli - General Coordinator - Software
- Prof. Antônio Augusto Fröhlich - Associated Coordinador - IoT
- Prof. Anderson Wedderhoff Spengler - Hardware
- Prof. Gustavo Medeiros de Araujo - AI

Graduate Students

- João Paulo Bedretchuk (M.Sc.)

- Rafael Canal (M.Sc.)
- Sergio Arribas Garcia (M.Sc.)

Undergrad Students

- Felipe Kaminsky Riffel
- João Paulo Araujo Bonomo
- João Pedro Dandolini de Souza
- Tiago Porsch Dopke

Former Members

- Arthur Haickel Nina (B.Sc.)
- Leonardo Tomasi Francis (M.Sc.)
- Lucas Matheus dos Santos (B.Sc.)
- Maria Eduarda Rosa da Silva
- Miguel Angelo Romanichen Suchodolak (B.Sc.)
- Thiago Nogiri Igarashi (B.Sc.)
- Victor Elizio Pierozan (B.Sc.)
- Vitor Sorpile Geraldo (B.Sc.)

4. **Publications**

Master's Theses

1. Anomaly Detection of Engine Knock in Automotive ECU using Machine Learning Algorithms. Leonardo Tomasi Francis. Dissertação de Mestrado. PPGCC/UFSC.
2. Proposta e Desenvolvimento de um Hardware Embarcado com Suporte à Computação de Borda para Aquisição, Análise e Processamento Avançado de Dados Veiculares. João Paulo Bedretchuk. Dissertação de Mestrado. PPGESE/UFSC.
3. Machine Learning Applied in Automotive ECUs for Real-Time Engine Behavior Analysis. Rafael Canal. Dissertação de Mestrado. PPGESE/UFSC.

Undergraduate Theses

1. Desenvolvimento de sistema de aquisição e calibração para ECUs. Thiago Nogiri Igarashi. TCC em Engenharia Mecatrônica. UFSC. Disponível aqui.
2. Inteligência artificial aplicada para detecção de Knock em motores automotivos. Victor Elízio Pierozan. TCC em Engenharia Mecatrônica. UFSC. Disponível aqui.
3. Seleção de Atributos em aprendizado de máquina para identificação de falha em motores de combustão interna. Maria Eduarda Rosa da Silva. TCC em Engenharia Mecatrônica. UFSC. Disponível aqui.

Research Papers

1. Misfire Detection in Combustion Engines Using Machine Learning Techniques. R. Canal, F. K. Riffel, J. P. Bonomo, R. Carvalho, and G. Gracioli. 2023 XIII Brazilian Symposium on Computing Systems Engineering (SBESC), 1-8.
2. Low-Cost Data Acquisition System for Automotive Electronic Control Units. João Paulo Bedretchuk, Sergio Arribas García, Thiago Nogiri Igarashi, Rafael Canal, Anderson Wedderhoff Spengler, Giovani Gracioli. Sensors. 2023.
3. Driving Profile Analysis Using Machine Learning Techniques and ECU Data. R. Canal, F. K. Riffel, and G. Gracioli. 32nd International Symposium on Industrial Electronics (ISIE 2023).
4. Improving the Execution Time of Industrial Applications through Planned Cache Eviction Policy Selection. S. A. García, G. Gracioli, D. Hoornaert, T. Kloda and M. Caccamo. 32nd International

Symposium on Industrial Electronics (ISIE 2023).

5. Minimizing Cache Usage for Real-time Systems. B. Sun, T. Kloda, S. A. García, G. Gracioli, and M. Caccamo. 31th International Conference on Real-Time Networks and Systems (RTNS 2023).
6. Feature Selection in Machine Learning for Knocking Noise Detection. M. E. R. da Silva, G. Gracioli, and G. M. de Araujo. 2022 XII Brazilian Symposium on Computing Systems Engineering (SBESC), 1-8.
7. Supporting Single and Multi-Core Resource Access Protocols on Object-Oriented RTOSes. L. M. dos Santos, G. Gracioli, T. Kloda, and M. Caccamo. Springer Journal of Design Automation for Embedded Systems. 2023.
8. Data-driven Anomaly Detection of Engine Knock based on Automotive ECU. L. T. Francis, V. E. Pierozan, G. Gracioli, and G. M. de Araujo. 2022 XII Brazilian Symposium on Computing Systems Engineering (SBESC), 1-8.
9. Integrating Autonomous Vehicle Simulation Tools using SmartData. J. L. C. Hoffmann, L. P. Horstmann, and A. A. Fröhlich. 2022 XII Brazilian Symposium on Computing Systems Engineering (SBESC), 1-8.
10. Development of Embedded Software for Data Acquisition and Calibration of Automotive ECUs. S. A. Garcia, T. N. Igarashi, J. P. Bedretchuk, M. A. Suchodolak, A. W. Spengler and G. Gracioli. 29th International Symposium of Automotive Engineering (SIMEA 2022), 2022, São Paulo.