Design and Validation of an In-Vehicle Data Recorder System for Testing Purposes

Julian Echeverry-Mejia, Felipe Arenas-Uribe, Diego Contreras and Virgilio Vásquez

Abstract— **This paper presents the design and statistical validation of an In-Vehicle Data Recorder System (IVDR) for testing purposes, implementing the use of CAN bus (Controller Area Network) communications protocol through the SAE J1962 port. The device works according to an interrupt-based algorithm that reads the vehicle's sensor data sent through the CAN bus network. This data is stored in an external memory card and sent to a server using a General Packet Radio Services (GPRS) module, it can also be connected to a computer to view the data in real time. The proposed IVDR was thoroughly validated by means of multiple experiments, including one in which its performance was compared to an industrial use device with similar functions, it has also already been used in the development of a Typical Driving Cycle (TDC) for the city of Chía in Colombia. Statistical analysis shows that the device performs similarly against commercial devices and complies with SAE standard guidelines, presenting the device as a low-cost and reliable option for testing purposes and massively distributed IoT applications.**

Index Terms— **Automotive communication protocols, CAN bus, In-Vehicle Data Recorder, OBDII, Typical Driving Cycles.**

I. Introduction

The transport industry finds itself in a transformation process to incorporate new technological trends such as, the Internet of Things (IoT) and Big Data. Multiple technological tools, apps, vehicle devices and external devices have been developed in the last decade with the objective of ensuring interconnection between older and new vehicle models. Following this trend, it is estimated that by 2030 more than 70% of vehicles will have internet connection [1]. With this advance, vehicles are able to transmit the information of their own physical variables, which are captured either by the vehicle's own sensors or by external devices.

By taking advantage of the data, developments used globally have been achieved. One example, that can be considered a Big Data application, is navigation and traffic detection applications, which capture the position and speed of the vehicle; with all this information, vehicular traffic maps are generated.

In response to these trends, data loggers, also known as In-Vehicle Data Recorders (IVDR), have been developed. These are devices that read and store data on the physical variables of the vehicle's dynamic behavior [2], [3]. IVDR devices have been used for several applications, for example in [4] the relationship between driving events, road characteristics and crashes is studied to predict crashes and identify high-risk places on the road network. In [5] an evaluation, based on IVDR and Survey Reviewed data on the driving behavior of inexperienced drivers with 3 to 4 years of driving experience is presented. The drivers improved their driving behavior while driving with IVDR data, compared to not having access to such information. The works developed in [6] and [7] present the measurement of vehicle performance following SAE (Society of Automotive Engineers) guidelines to identify driving habits and the usage of the information to design new vehicle parts, respectively. Collection of real data for the development of smart roads and the development of Typical Driving Cycles (TDC) to estimate fuel consumption and vehicle emissions are presented in [8], [9], [10].

J. Echeverry-Mejia, D. Contreras and V. Vásquez are part of the Escuela de Ingeniería y Ciencias at Tecnológico de Monterrey (e-mails: jecheverry@tec.mx, decontreras@tec.mx & vlopez@tec.mx).

F. Arenas-Uribe is part of the HCD Group at Universidad de La Sabana in Chía, Colombia (e-mail: felipearur@unisabana.edu.co).

This article proposes the design and validation of a highly scalable and reliable IVDR for testing purposes, which captures the behavior of the physical variables of the vehicle by reading the information that transits between its computers, using the SAE port J1962, better known as the OBDII (On Board Diagnostics version 2) diagnostic port [11]. The IVDR developed in this work aims to operate as a testing tool to identify the typical driving cycle of the city of Chia, Colombia. First, a review of the commercially available devices and the academic developments will be made. Then, the parameters of design for testability, massification, feasibility, reliability and total cost will be reviewed and defined. After that, both hardware and software will be designed and tested via a data loss test and a performance comparison with a commercially available device. Finally, an application message decoding for the sensor´s IDs is explained. It should be declared that this paper was developed as a result from a section of this thesis [6] with permission from the author.

II. Related Research

There are multiple works where the concept of dataloggers in passenger vehicles has been developed. The commercially available ones, in their majority, work with the OBDII protocol to read the data from the diagnostic computer in real time. IVDRs have been developed, commercially or as academic work, under different design parameters, such as those shown in Table I, which displays a collection of information from different devices and their characteristics.

Among the devices reviewed in Table I, the first eight are commercial devices, two of them were used for academic research [12], [13]. And the remaining ones are academic developments [14], [15], [16]. A weighting evaluation has been

TABLE I
IVDR CHARACTERISTIC COMPARISON FOR COMMERCIAL AND ACADEMIC DEVELOPMENTS

IVDR	CAN bus (0.2)	GPS (0.075)	Industry Validated (0.2)	Information Loss (0.2)	Programable (0.05)	Memory recording (0.15)	Cloud recording (0.075)	Portable (0.05)	Cost (USD)	weighting
ELM 327	0	0	1	1	0	0	0	5	20	0.65
Vector VN1600	5	0	5	5	0	0	0	2	3000	3.1
myRIO+X-CAN	5	5	2	3	5	3	4	5	950	3.625
NI9862 [12]	5	5	4	5	5	3	3	3	3000	4.25
Race Technology DL1 Mk3 [13]	5	5	4	5	0	5	0	3	960	4.075
OBD Mini Logger	5	5	2	4	0	5	4	5	800	3.875
DashDyno SPD	5	5	3	3	0	5	0	5	350	3.575
IOSiX OBDII Datalogger	5	0	0	4	0	5	0	5	700	2.8
OBDII Network [14]	0	0	2	5	5	5	3	3	50	2.775
Arduino OBDII Device [15]	0	5	0	1	5	1	3	4	160	1.4
Monitoring System [16]	0	5	0	3	5	5	5	5	45	2.6

applied to these devices, according to characteristics that are important for an IVDR with applications in IoT and Big Data. Each characteristic is evaluated from 0 to 5, this scale qualitatively evaluates the level of inclusion of a certain characteristic, with 5 being it is completely included and 0 that it is not included whatsoever. Regarding cost, it will be considered for a discussion of the best rated devices.

The characteristics shown in Table I allow the IVDRs to be evaluated according to parameters that make it possible to guarantee that the device is reliable and capable of being used in massive distribution IoT and Big Data applications. The table shows that the devices with the highest scores are high-cost commercial developments. Among these, the NI9862 [12] and Race Technology DL1 Mk3 [13], which have the highest scores, also present the highest costs, hindering their use for mass distribution applications. Additionally, they do not have the capability to record data in the cloud. Therefore, among the reviewed devices, there is none that proves to be viable for mass use in IoT and Big Data applications. Thus, sustaining the need for a development of an IVDR that can satisfy those needs.

III. DESIGN REQUIREMENTS

A. *IVDR Data Gathering Mechanism*

The IVDR information gathering process can be done in three ways: using external sensors, using the physical sensors of the vehicle's equipment, or extracting the information from the internal sensors of the vehicle that are available in the vehicle's communication protocol. The third method is selected for the present research because it is non-invasive, inexpensive, and repeatable on most vehicles. Two concepts relevant to the development of the device were studied: the CAN bus (Controller Area Network) communication protocol and the OBDII diagnostic system protocol.

1) *CAN bus Protocol*

This is the most widely used communication protocol in the automotive industry, it has the purpose to connect the different vehicle computers. The main characteristics of this protocol are listed in the ISO 15765-4 standard [17] and will be summarized below. The CAN bus is physically made up of two wires that carry a mirrored signal, CAN high and CAN low. Supporting speeds up to 1 Mbps with 8 bytes of data per message. For the version 2.0A, which is the one implemented in vehicles, 11 ID bits allow the bus access arbitration of messages. The lower the 11 ID bit field the higher the message priority.

In a CAN bus any of the controllers can function as master or slave, for automotive applications each controller can be a vehicle computer for a system, such as, the engine or brake system computers. Likewise, any controller can be disconnected or added to the bus at any time. Additionally, it has sophisticated methods of error detection and retransmission of erroneous messages: cyclic redundancy check, frame check, acknowledgment (ACK) error and Bit stuffing [18], [19].

The CAN bus protocol transmits messages of different nature, in this research two types of messages are used:

- CAN messages: These messages are periodically broadcasted to the bus (continuous), however, as there is no standard that defines them every manufacturer is free to define them as they see fit. This type of message is used by the vehicle's computers to communicate with each other and share, for example, sensor readings as fast as possible. Each sensor is identified by an ID, which helps the vehicle's computers to determine the origin of the message. Since this definition can change depending on the manufacturer and model of the vehicle, for its usage it is necessary to have the factory information or obtain it experimentally [18].
- OBDII messages: These are messages of the diagnostic system of the request type, that is, they only appear on the bus when they are requested. All OBDII parameters are known and standardized by the ISO 15765-4 [17] and ISO 15031-5 [20] standards. Although OBDII messages can contain information about different physical variables, they are all identified by the same ID in the CAN bus protocol.

2) *OBDII Diagnostic System*

The OBDII system is used in the automotive industry for vehicle diagnosis. In 2001 the SAE published the standards of this system, compiled mainly in the SAE J1962 [11], SAE J1978 [21] and ISO 15765-4 [17] standards. This diagnostic system can be implemented over any automotive communication protocol available, but it is implemented over CAN bus for most vehicles.

OBDII has ten standardized operation modes [17], each of them with different functionalities. The developed IVDR can operate in any of the ten modes; however, it mainly operates using mode one for real-time access to the standardized parameters known as PIDs (Parameter IDs). These identifiers are memory addresses that store dynamic variables of the vehicle, for example: engine speed, air mass flow, among others.

B. IVDR Design Requirements Definition

After reviewing the characteristics of the IVDRs available in Table 1, the following design parameters are proposed for this work:

- CAN/OBDII: must allow connection with the CAN communications protocol. And as a result of that, with the OBDII diagnostic protocol too.
- Standard: the measured variables with the IVDR must comply with the resolution and tolerance specified by the automotive standards.
- GPS (Global Positioning System): since not all vehicles have this equipment, the IVDR must integrate it.
- Industrial validation: at least the Microcontroller Unit must be a component validated in the automotive industry.
- Information loss: the processing speed must allow reading all the frames that pass through the CAN bus to ensure that there is no loss of information.
- Programmable: must be programmable, so that functionalities can be added.
- Real time PC connection: must allow real time connection to a PC to perform online analysis.
- High-capacity memory: a lot of information travels through the bus, making necessary to have a high-capacity external storage unit, in this case a microSD memory card.
- Cloud recording: must have internet connectivity to allow uploading information to the cloud in real time.
- Portable: it needs to be as small as possible, to not visually impede driving and allow easy installation.
- Power: must be powered from the SAEJ1962 connector [14].
- Cost: since the aim is to design a device that can be easily massified, it must be a low-cost device. This is one of the main design requirements.

IV. HARDWARE DESIGN AND DEVELOPMENT

The hardware used for the IVDR must be compliant with the design requirements mentioned previously. The most important device in the hardware is the Microcontroller Unit (MCU), as it is the device that could limit the operational scope of the IVDR. Consequently, the selection of the MCU must be careful and must follow an adequate process.

Table II is used for the MCU selection, in which a collection of the characteristics of different brands and models is qualified by means of a weighting matrix. Each of the characteristics is rated from 0 to 5, with 5 being that it fully complies with the parameter of that the evaluated characteristic is of excellent quality and 0 the opposite case. After this analysis, it was decided to use the NXP TRK-KEA128 MCU due to its industry validation and its compatibility with the CAN bus protocol.

Table III presents a list of the selected modules along with their costs. It is observed that the bill of materials is 110 dollars, which is within the range of the IVDR developed in academic works, [14], [15], [16]. The added value of this IVDR comes by implementing the use of a microcontroller validated in the industry; thus, complying with the design parameters previously established.

The IVDR prototype includes:

- Two LEDs working as operating indicators.

TABLE III
IVDR HARDWARE MODULES

Module	Reference	Cost
GPS Module	Ublox NEO-6M	*10*
SD Module	microSD	*2.5*
microSD	16GB	*7*
Power Module	DC-DC Buck LM2596	*5*
GPRS Module	SIM808	*20*
Microcontroller	NXP TRK-KEA128	50
SAEJ1962 Cable	OBDII-DB9	9
Miscellaneous	Cables, LEDs, fuses	*5*
Casing	Dexson DXN500DG	*1.5*
Total ---		*110*

- A pushbutton electrically connected to a digital input interruption of the microcontroller, to generate the definitive

TABLE II
IVDR MICROCONTROLLER COMPARISON

Development Board	Microcontroller (0.2)		Automotive Validation (0.2)		SD memory SPI Compatibility (0.05)		# of UARTs (0.05)		Direct CAN bus Compatibility (0.25)		MCU Cost (USD) (0.25)		weighting
Arduino Mega	ATMega1280	2	No	0	Yes	5	Yes	4	No	0	40	4	2.8
Raspberry PI 3 Model B	1.2GHz 64-bit quad-core ARM Cortex-A53 CPU	3	No	0	Yes	5	Yes	1	No	0	46	4	2.4
STM32F4DISCOVERY	STM32F407VGT6 32-bit ARM Cortex® -M4	3	No	0	Yes	5	Yes	3	No	0	20	5	2.95
NXP TRK-KEA64	32-bit MCU core from ARM's Cortex-M class	5	Yes	5	Yes	5	Yes	2	No	0	49	4	3.375
NXP TRK-KEA128	32-bit MCU core from ARM's Cortex-M class	5	Yes	5	Yes	5	Yes	3	Yes	5	49	4	3.825
Tiva C Series TM4C123G USB+CAN	Tiva TM4C123GH6PGE ARM® Cortex™-M4-based	5	Yes	5	Yes	5	Yes	4	Yes	5	150	2	3.725
Infineon KIT XMC43 RELAX_ECAT_V1	XMC4300-F100	5	Yes	5	Yes	5	Yes	2	Yes	5	62	3	3.475
Renesas RL78/F14	RL78/F14 16-Bit	5	Yes	5	Yes	5	Yes	2	Yes	5	275	1	3.175

stop of the IVDR operation and to be able to extract the data from the SD card in a safe way.
- DB9 connector for linking the IVDR with the SAE J1969 port. This type of connector is used since cables with this configuration are commercially available.
- A switch that allows the IVDR to be de-energized.
- It has a slot to access the microSD socket.
- It has a slot to access the micro-USB port so that the software can be updated if needed and for real time readings with a PC.
- The size should be as small as possible, complying the portability design requirement, considering that it does not visually impede driving.

In Fig. 1 the modular diagram of the electronic design is identified, in Fig. 2 the selected components are presented graphically. Additionally, Fig. 3 presents the final mechanical design.

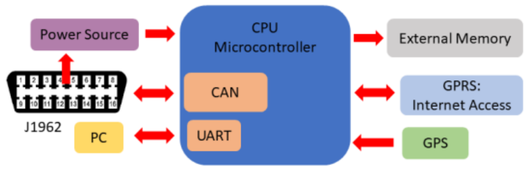

Fig.1. Modular electronic diagram for the IVDR.

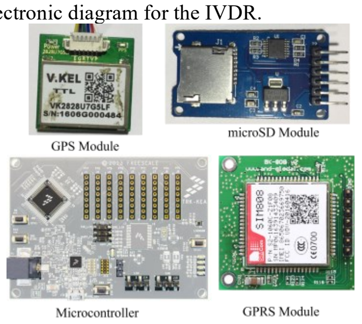

Fig. 2. Top view of the electronic modules and the microcontroller.

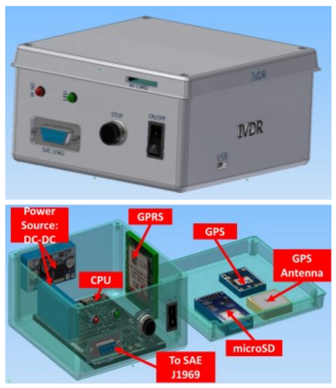

Fig. 3. Final IVDR design.

V. Software Development

As described above, the main function of the IVDR is to connect the datalogger to the vehicle's CAN bus. The storage means are: store the information in a removable SD memory, upload the data to the cloud through a GPRS (General Packet Radio Service) module and monitor the bus in real time through a PC. In addition to the CAN messages, it is necessary to collect the geographical position through a GPS module. Within the functionality of the IVDR, each of the peripherals (**SD, GPRS, PC, GPS**) can be deactivated at the convenience of the user without interrupting the operation of the IVDR.

On the other hand, the device has different operating modes, which have to do with how the device collects data from the CAN bus, the operating mode can be selected by the user each time the application is restarted. These modes of operation are:

- ***Sniffer* Mode**: reads and records every single message that passes through the CAN bus.
- **IDs Mode**: applies a mask over the ID section of the incoming messages to record the desired IDs exclusively.
- **OBDII Mode**: applies a mask over the ID of the incoming messages to read only OBDII messages, which are usually located in ID 0x7E8. Additionally, the desired PIDs must be selected.
- **IDs and OBDII Mode**: this operation mode is a combination of the two previous modes. Allowing to record specific CAN bus and OBDII messages.

Software development is supported on an interrupt-based state machine because both hardware and software are required to have control over the operation. This means that the operation of the algorithms does not depend on a sequential or synchronous program, but on themselves and on external events. The code developed is scalable and has an error verification system to have software monitoring control. The main program and interrupt routines are:

- <u>CAN reception interruption:</u> messages from the CAN bus are read and stored in a buffer. It also controls the request for data by OBDII.
- <u>Periodic Interrupt Timer (PIT) interruptions:</u> there are two interruptions of this type, one to synchronize the OBDII data requests and another to time some functions, such as, error handling and periodic functions.
- <u>Serial UART (Universal Asynchronous Receiver-Transmitter) interruptions:</u> there are two UART interruptions, one receives and stores to the buffer the incoming data from the GPS module and another to store the response of the GPRS module to the commands issued.
- <u>IRQ (external interrupt request) interruption:</u> it is in charge of completely stopping the program and moving to an infinite cycle. This to close the last open txt file on the SD memory card and cut off GPRS communication.
- <u>Main program routine:</u> it initializes all microcontroller resources and peripherals. Its function is to carry out the asynchronous transmission of information through buffers, either to the SD memory card, to the cloud (GPRS) or by UART to a PC.

Fig. 4 schematizes the state machine followed by the IVDR software; it is observed that the interruptions do not overlap. In

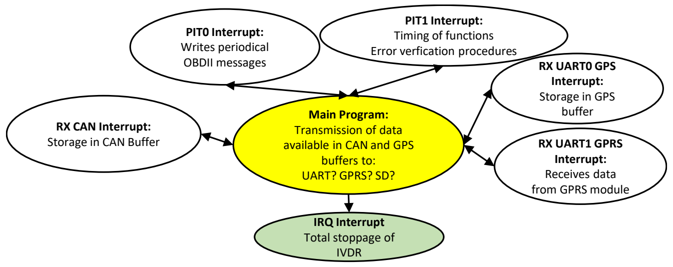

Fig. 4. IVDR State machine.

case of activating several interruptions at the same time, the order of priority is applied: IRQ, UART, PIT, CAN [22].

The microcontroller software is always in an infinite loop in the main program, waiting for an event, to transmit it asynchronously by any of the three peripherals (SD, GPRS, PC). In case of an external event, such as, a message on the CAN bus or a GPS message, the corresponding interrupt is activated, the interrupt function is executed and then the operation returns to the main program again. All the information flow between peripherals and the microcontroller is done through buffers.

The four most important algorithms (see Fig. 4) developed for the operation of the IVDR are RX CAN Interrup, RS UART0 GPS, RS UART0 GPRS and PIT0 serial interrupt. These algorithms are described in greater depth below.

A. *CAN Reception Interrupt*

Fig. 5 presents the general algorithm for the CAN reception interrupt, whose main function is to read the messages from the bus and store them in a buffer. The interrupt is activated when a CAN message available, which occurs with high periodicity since there can be messages every 200 microseconds on the bus [17]. Said buffer is cyclical with a fixed size, but with variable occupation that depends on the amount of data received from the CAN bus; this buffer is traversed with the help of a pointer. It is necessary that the size of the buffer allows significant storage, so that it will not overflow without transmitting the message.

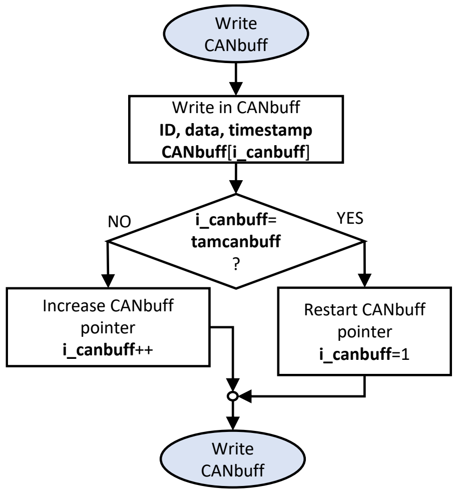

Fig. 5. Flowchart of the data storage algorithm for CAN messages.

The sequence of this routine is:

- When the interrupt is activated, the most relevant parameters of the CAN message are read, which are: the ID, 8 bytes of data and the timestamp. The latter has the functionality of locating each message in time and is extracted from the CAN bus hardware.
- Depending on the operation mode, the required messages are stored in the buffer, to later be transmitted to any of the peripherals in the main program routine; this is done using multiple pointers that together allow to identify the state of the buffer. It should be noted that this buffer stores CAN frames in binary format and not ASCII (American Standard Code for Information Interchange) to save capacity and computing time.

B. *GPS UART Serial Interrupt*

To communicate the GPS NEO-6M module with the microcontroller, only the reception channel is required. This module upon powering up and finding a geographical position starts transmitting frames with the NMEA (National Marines and Electronics Association) system coding [23], from which the GPGGA (Global positioning system fix data) protocol is extracted. This protocol contains time, latitude, longitude, and height above sea level. Fig. 6 presents the serial reception interrupt routine for the GPS. The main characteristics of this routine are described below:

- The UART hardware reads the frames received from the GPS byte by byte, these are stored in a static buffer, different from the CAN bus buffer, with the capacity for a single GPS message.
- When a frame finish transmitting, a carriage return line feed character is received; at this moment, the message identifier is checked and if it is equivalent to “GPGGA” a flag (msgGPS) is enabled. This flag indicates that there is a valid frame to store in the buffer and is used by the GPS routine of the main program to store the buffer in some peripheral.

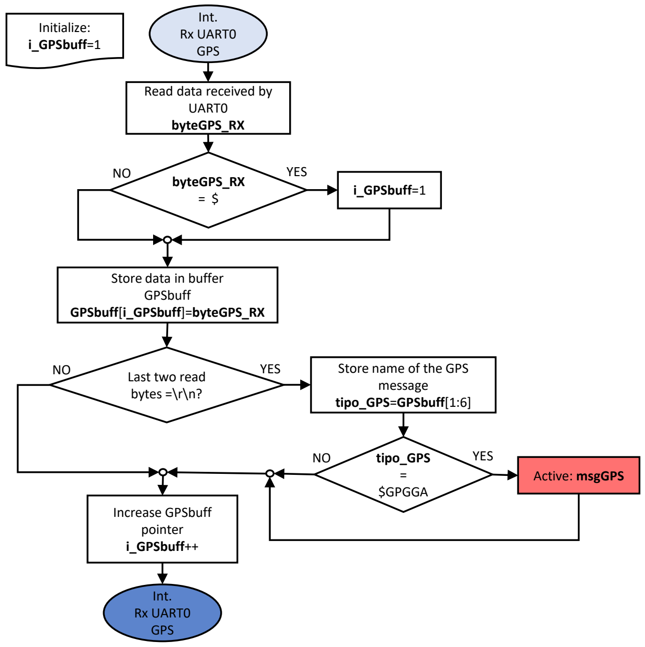

Fig. 6. Flowchart of the serial interrupt algorithm for GPS data.

C. *GPRS UART Serial Interrupt*

The selected GPRS module used for the present research can operate under different methods, the one used for the IVDR is the HTTP (Hypertext Transfer Protocol) GET method. The module is controlled using AT commands via a UART channel. Every time the module is interacted with, it generates a response that is processed through a serial interrupt. The responses used in this development are: "OK", "ERROR" or "200".

To use the GPRS module it is necessary to initialize the SIM808 chip. After being initialized, it goes to the infinite cycle of the main program and while the peripheral is operating, the required information is transmitted to the cloud. This process is outlined in Fig. 7 with the required AT commands and is carried out with the following considerations:

- To continuously upload parameters to the cloud with the

GET HTTP method and the GPRS module, it is necessary to carry out 2 steps: in step 1, write the command AT + HTTPPARA = "URL", "XXXX" where XXXX is the domain and in the step 2, after receiving the OK response from step 1, the command AT + HTTPACTION = 0 is written and the response 200 is expected, meaning that the message was received correctly.

- When a certain time is elapsed, 1 minute in this case, it means no response was received for the last AT command, therefore step 1 must be repeated. The time keeping function is described in the error handling algorithms below.

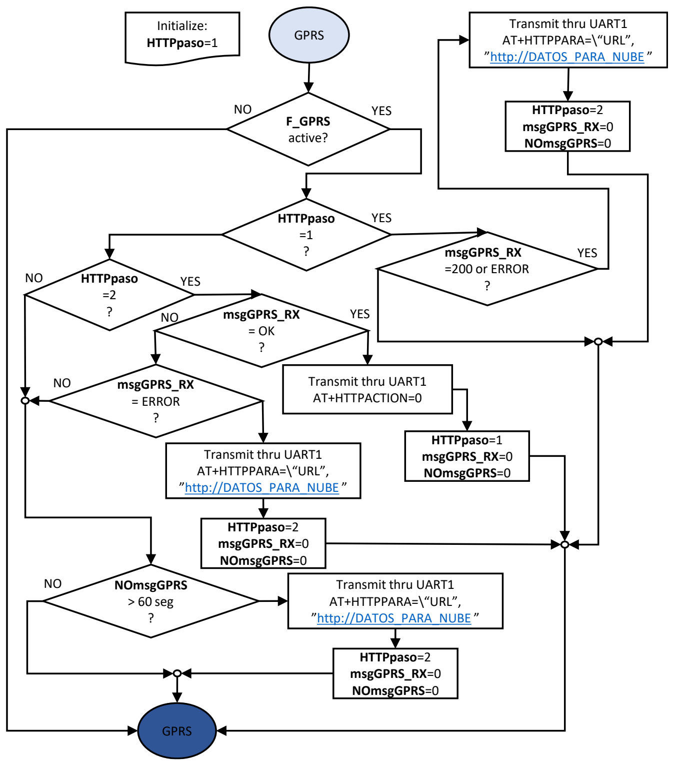
Initialize:
HTTPpaso=1
GPRS
F_GPRS active?
NO
YES
Transmit thru UART1 AT+HTTPPARA=\"URL", "http://DATOS_PARA_NUBE"
HTTPpaso=2 msgGPRS_RX=0 NOmsgGPRS=0
HTTPpaso =1 ?
msgGPRS_RX =200 or ERROR ?
HTTPpaso =2 ?
msgGPRS_RX = OK ?
msgGPRS_RX = ERROR ?
Transmit thru UART1 AT+HTTPACTION=0
HTTPpaso=1 msgGPRS_RX=0 NOmsgGPRS=0
NOmsgGPRS > 60 seg ?
GPRS

Fig. 7. GPRS module cloud data transmission flowchart algorithm.

D. Error Handling Algorithms

As previously mentioned, the error handling algorithms are synchronized and programmed in a Periodic Time Interruption (PIT). In this interrupt the states of different flags are checked to monitor the operation of the IVDR. These functions are presented in Fig. 8 and are described below:

- Error function to determine if the main program stopped its execution without it being requested by the user. In the main program, each time that a new infinite loop starts a flag is incremented. This function compares the current and the previous value of said flag, if this value has not changed in the last minute, it means that the microcontroller got stuck in some routine, different from the main one. If this error is generated, an alarm is activated and the execution of the IVDR stops definitively.
- CAN bus reading problem error function: a variable is used to save the elapsed time without reading CAN messages. Each time one minute is elapsed, a flag is set, and the elapsed time variable is reset. Both the flag and the variable are reset with the arrival of each CAN bus message, in this way non-consecutive times are not accumulated. The flag that is set in this function is used to write a message in the SD memory that indicates a CAN bus reading problem.
- GPRS response error function: in this function a variable is incremented every second, said variable is reset each time the GPRS routine gets a response from the module. This variable accumulates the time it takes for the module to respond. If the time is greater than 1 minute, the transmission of the GPRS module is restarted.
- SD txt file creation tracking function: to minimize the amount of information lost during prolonged experiments in case of an error happening, the information collected is split up into several files, each file containing five minutes of experimental data. The moment the IVDR continuously reads CAN messages, the elapsed time is stored in a variable. After the selected time (5 min) is reached, a flag is set that is used in the CAN bus routine to create a new txt file in the SD memory.

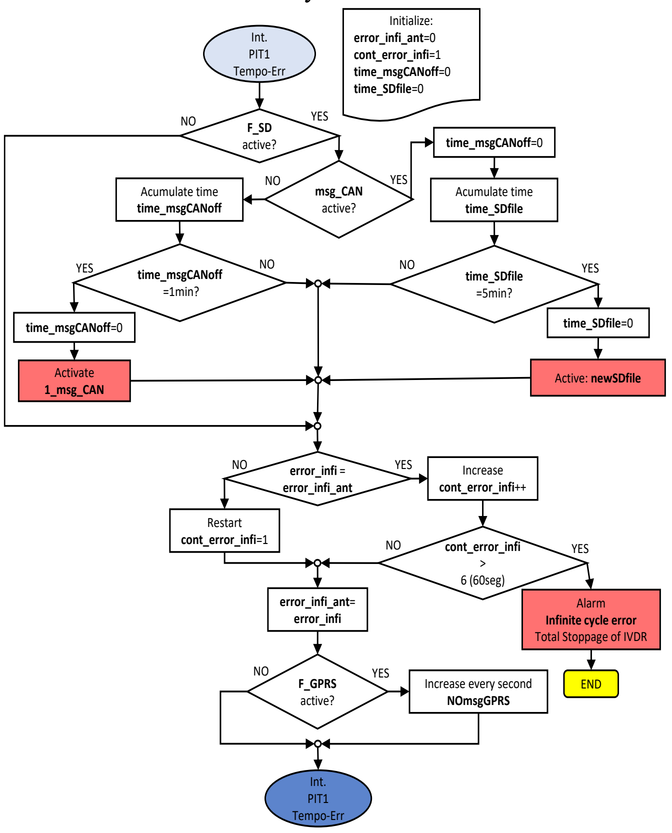

Fig. 8. Error handling algorithms flowchart.

E. Human-Machine Interface

In addition to software design in the IVDR, it is necessary to develop programs that allow data manipulation on a computer. For this, various human-machine interfaces are developed in MATLAB and LabVIEW depending on the IVDR peripherals activated:

- PC: this program allows to monitor the messages sent from the IVDR buffer to the PC in real time. Likewise, it creates a file where it stores the messages received through the serial port in the order they were received.
- SD: this program allows to integrate the data present in the separate files saved in the SD memory card during the IVDR operation.

- GPRS: for this peripheral it is necessary to develop a program that allows storing the IVDR information in the cloud. In the present research Google Cloud was used.

VI. Experiments and Validation

As part of the validation of the IVDR operation, two tests were performed:

1. Data acquisition comparison between the IVDR and an industrial datalogger.
2. IVDR data loss analysis.
3. CAN bus periodic message ID decoding methodology.

A. IVDR vs NI9862 Validation

For this validation test, the performance of the IVDR must be compared against an industry validated device, in this case the NI cRIO-9063 chassis plus NI9862 data acquisition card. For this purpose, an acceleration performance test following SAE J1491 standard [24] with five repetitions was done. In each test simultaneous data acquisition is carried out with both the IVDR and the NI9862.

During these experiments the IVDR is used in IDs and OBDII mode; on the other hand, the NI device is used in sniffer mode and afterwards with the help of a LabVIEW program the OBDII data is extracted. All experiments were performed with a *Jeep Patriot*, at sea level on the Mexican Pacific coast, on a street with an approximate slope of zero degrees and following all the requirements of the SAE standard for acceleration performance tests.

Fig. 9 shows the comparison of the linear speed reading with the two instruments, the NI and the IVDR, for the first experiment. The other four repetitions present a similar behavior. The similarity between the information acquired with both instruments can be observed. Table IV presents the comparison between the resolution of the instruments as required by the SAE standard for acceleration performance [24] and the resolution provided by the IVDR, clearly demonstrating that it can fulfill the requirements set by the standard.

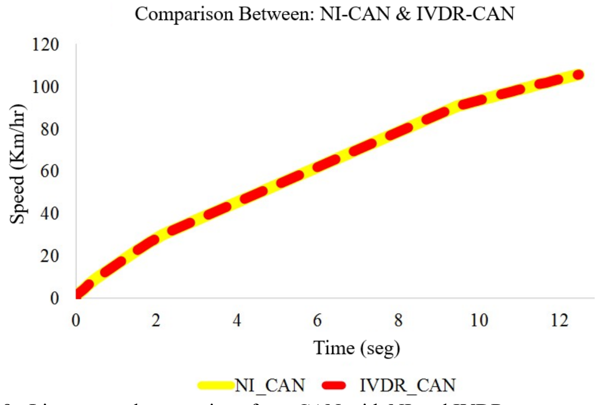

Fig. 9. Linear speed comparison from CAN with NI and IVDR.

TABLE IV
IVDR resolution vs SAE standard

Variable	Unit	SAE Standard	IVDR-CAN
Time	s	0.1	0.01
Linear Speed	km/hr	*0.4*	*0.0078*
Engine Speed	RPM	*25*	*0.25*

To prove the similarity of the results obtained, a statistical analysis consisting of three steps is carried out [25]. In the first step, a Kolmogorov-Smirnov normality test is performed on the data from each device, to determine whether to use parametric or non-parametric statistics. The numeric results of the normality test are presented in Table V. It is observed that the hypothesis (H) in both cases has a value of 1, indicating that they do not have normal behavior. In turn, the p-value is much less than 0.05 (5%), indicating that the null hypothesis, which proposes that the data present normal behavior is rejected. Therefore, the alternative hypothesis is accepted indicating the need to use non-parametric statistics for the data analysis.

TABLE V
Kolmogorov –Smirnov normality test for the NI and IVDR device data

Kolmogorov-Smirnov	NI-CAN	IVDR-CAN
Median	68.69	68.83
Standard Deviation	29.35	29.39
H	1	1
P	2.01E-06	1.85E-06

The second step is to perform the non-parametric analysis, using the Wilcoxon signed-rank test for two same sized dependent samples. Table VI shows the result for each repetition, it can be observed that the means are equal for each test since they present a p-value of 1, much higher than the alpha of 0.05. This indicates that for the 5 repetitions, the null hypothesis (h=0) is accepted, which proposes that the means of the two populations are equal, with no significant data difference. Also, the average for the different devices is exactly the same because they are reading the same data. If these averages had significantly different results, it would imply that there is information loss in one of the devices.

TABLE VI
Wilcoxon signed-rank test for NI vs IVDR

Wilcoxon test	P1	P2	P3	P4	P5
H	0	0	0	0	0
P	1	1	1	1	1
Mean NI-CAN	68.69	72.61	72.77	72.46	71.45
Mean IVDR-CAN	68.69	72.61	72.77	72.46	71.45

The last step is to perform a Pearson correlation test on the data obtained in each repetition. For all cases, a result of 1 was obtained both in speed and time. Thus, having a high positive correlation, which implies a high similarity in the form of the responses.

B. IVDR Data loss Analysis

For the data loss analysis, one hundred and fifty tests were carried out with an approximate duration of 18 minutes per test.

TABLE VII
IDs captured by the IVDR during data loss tests

ID	Sampling rate (seg)	ID Protocol
513	0.01	ID CANbus
517	0.01	ID CANbus
1056	0.1	ID CANbus
1060	0.1	ID CANbus
1200	0.01	ID CANbus
2024	0.03	ID OBDII

In each test a fixed route of five kilometers is traveled in the city of Chía, Colombia; driving a *Ford Fiesta*. With the help of the IVDR, in each test the acquisition of the IDs mentioned in Table VII are stored in an SD memory for later analysis.

To identify the amount of data lost, the average sampling time and the standard deviation of each observation are calculated for each ID. Subsequently, the sampling time of each observation is compared with the average sampling time for its ID plus an uncertainty given by the standard deviation. Any sampling outside the mentioned range is considered data loss. Table VIII shows the results of this test, indicating that according to the established definition of missing data, the IVDR presents no data loss.

TABLE VIII
DATA LOSS TESTS RESULTS

ID	Quantity of observations	Average sampling rate (seg)	Standard deviation (seg)	Lost data
513	13540796	0.010043825	0.00271150214	0
517	13540877	0.010043764	0.00270993422	0
1056	1353915	0.100440092	0.00915493517	0
1060	1353930	0.100438835	0.00914795529	0
1200	8095714	0.010035344	0.00253127891	0
2024	4063321	0.030107916	0.00455430137	0

C. CAN Bus Periodic Message ID Decoding

The procedure to find specific IDs for a sensor of the vehicle requires that the IVDR operate in sniffer mode. The identification of each ID requires different tests to be carried out on the vehicle, collecting the data with the IVDR, to be analyzed later using reverse engineering.

The procedure to find the odometer ID is presented below, whose procedure can be repeated to find other IDs.

1. Read the CAN bus messages with the engine running, without movement. In this case, all data fields and IDs that present changes must be discarded.
2. Read the bus with the engine off, the vehicle is pushed forward a few meters. All data fields and IDs that do not change are discarded.
3. A test with known distance is run and the remaining IDs from the previous step are analyzed. In this test approximately three hundred and fifty meters are traversed.
4. From experience it is known that the odometer ID is usually an incremental counter. Fig. 10 shows the ID that presents this behavior for the previous test.

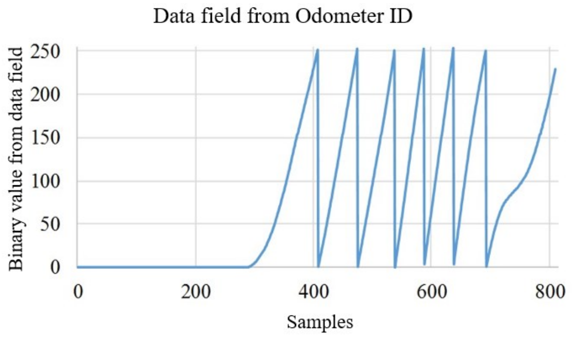

Fig. 10. Odometer ID data field analysis.

5. In Fig. 10 it can be seen that the byte is an incremental counter that goes from 0 to 253 and that this counter was reset 6.9. Meaning that if the test run is 350 meters, each counter reset is equivalent to 50.72. Rounding off, 0.2 meters are traveled for each counter increment, or 50.8 meters each reset. With this appreciation, the distance in meters of said test is plotted in Fig. 11.
6. Finally, the previous steps must be repeated several times with different routes, to ensure a proper decoding for the variable.

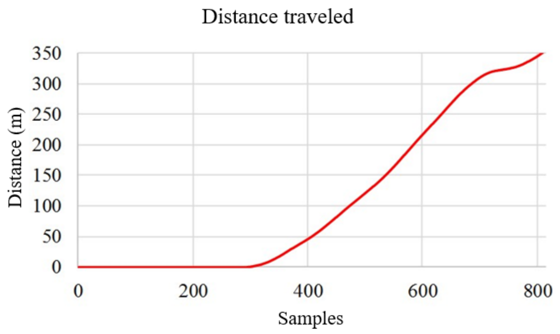

Fig. 11. Distance traveled during Odometer ID decoding test.

VII. CONCLUSIONS

In this work, an IVDR was developed complying with the initially proposed design requirements. This device, presents no data loss and can be easily used for testing purposes, allowing it to be massively distributed for IoT and Big Data applications.

A series of tests were applied over the developed IVDR to evaluate its performance, comparing it against an industry validated datalogger. A similar behavior is observed between both devices, which is statistically proved by having a high positive Pearson correlation. Also, with help of the Wilcoxon signed-rank test for mean equality in dependent samples, it is proven that the means of the two populations are not significantly different from a statistics point of view.

Likewise, the IVDR has been used in different research, highlighting the use to obtain the typical driving cycle for the city of Chía, Colombia, operating for more than 14 months and 10,000 kilometers. During the present work, 150 tests were run, demonstrating that the device did not present any significant data loss events and has the capability to adapt for multiple testing operations. An additional test was included to showcase a potential application for the IVDR. The methodology that permits the identification and decoding of the odometer ID can be extrapolated to any other variable present on the CAN bus.

Finally, the authors propose as future work to use this device as instrumentation equipment for the extraction of TDCs, methodologies to measure fuel consumption and greenhouse gas emissions in light commercial vehicles.

ACKNOWLEDGMENTS

We would like to thank Tecnológico de Monterrey Campus Estado de México and Universidad de La Sabana for their support in the development of this research.

REFERENCES

[1] H. Weber, J. Krings, J. Seyfferth, H. Güthner and J. Neuhausen, "Digital Auto Report 2019," 2019.

[2] A. Chidester, J. Hinch and T. Roston, "Real World Experience With Event Data Recorders," *National Highway Traffic Safety,* 2001.

[3] L. Moreira-Matias and H. Farah, "On Developing a Driver Identification Methodology Using In-Vehicle Data Recorders," *IEEE Transactions on Intelligent Transportation Systems,* vol. 18, no. 9, pp. 2387-2396, 2017.

[4] V. Gitelma, S. B. E. Doveh, F. Pesahov, R. Carmel and S. Morik, "Exploring relationships between driving events identified by in-vehicle data recorders, infrastructure characteristics and road crashes," *Transportation Research Part C: Emerging Technologies,* vol. 91, pp. 156-175, 2018.

[5] G. Albert, T. Lotan, T. Toledo, E. Grimberg and M. Lasebnik, "Are young drivers as careful as they deem? In vehicle data recorders and self reports evaluations," *Eur. Transp. Res. Rev.,* vol. 6, pp. 469-476, 2014.

[6] J. Echeverry, Metodología Para Reducir El Gasto De Combustible En Rutas Fijas Mediante El Uso De Hábitos De Conducción Eficiente, Empleando Un sistema IVDR Y Ciclos De Conducción, Estado de México: Instituto Tecnológico y de Estudios Superiores de Monterrey, 2018.

[7] J. Huertas, J. Díaz, D. Cordero and K. Cedillo, "A new methodology to determine typical driving cycles for the design of vehicles power trains," *Int J Interact Des Manuf,* 2017.

[8] S. Kamble, T. Mathew and G. Sharma, "Development of real-world driving cycle: Case study of Pune, India," *Transportation Research Part D,* pp. 132-140, 2009.

[9] D. Cordero and J. Huertas, "Metodología para minimizar el consumo de combustible en autobuses, que sirven rutas fijas, mediante la reconfiguración del tren motriz," Instituto Tecnológico y de Estudios Superiores de Monterrey, Toluca, 2015.

[10] G. Toledo and Y. Shiftan, "Can feedback from in-vehicle data recorders improve driver behavior and reduce fuel consumption?," *Transportation Research Part A: Policy and Practice,* vol. 94, pp. 194-204, 2016.

[11] Society of Automotive Engineers (SAE), *On-Board Diagnostic Connector, J1962,* 2016.

[12] J. Echeverry, V. Vasquez, J. Aguirre and D. Contreras, "Low Cost Obtainment of Vehicle Performance Curves and Values Experimentally by Means of the OBD2 Port," *SAE Technical Paper,* 2015.

[13] S. Birrell and M. Fowkes, "Glance behaviours when using an in-vehicle smart driving aid: A real-world, on-road driving study," *Transportation Research Part F,* 2014.

[14] B. Sung-hyun and J. Jong-Wook, "Implementation of integrated OBD-II connector with external network," *Information Systems,* 2013.

[15] K. Smith and J. Miller, "OBDII Data Logger Design for Large-Scale Deployments," in *Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems*, 2013.

[16] S. Garcia, "System for the realization of advanced mobility studies based on driver, cabin and vehicle monitoring," *IEEE Latin America Transactions,* vol. 100, no. 1, 2020.

[17] ISO, *ISO 15765-4:2016. Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) — Part 4: Requirements for emissions-related systems,* 2005.

[18] M. D. Natale, H. Zeng, P. Giusto and A. Ghosal, Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice, Springer, 2012.

[19] B. Kai, "1.3.1 CAN Bus," in *EMC and Functional Safety of Automotive Electronics*, Institution of Engineering and Technology., 2019.

[20] ISO, *ISO 15031–5:2015. Road vehicles — Communication between vehicle and external equipment for emissions-related diagnostics — Part 5: Emissions-related diagnostic services.,* 2005.

[21] Society of Automotive Engineers (SAE), *OBD II Scan Tool -- Equivalent to ISO/DIS 15031–4, J1978,* 2011.

[22] SIMCom, *SIM800 Series AT Command Manual,* 2015.

[23] SiRF Technology Inc., *NMEA Reference Manual,* San Jose, California, USA, 2007.

[24] Society of Automotive Engineers (SAE), *Vehicle Acceleration Measurment, J1491,* 2006.

[25] V. V. Lopez, J. E. Mejia and D. C. Dominguez, "Design and Statistical Validation of Spark Ignition Engine Electronic Control Unit for Hardware-in-the-Loop Testing," *IEEE Latin America Transactions,* vol. 15, no. 8, pp. 1376-1383, 2017.

Julian Mauricio Echeverry Mejía. Dean at Tecnológico de Monterrey - Escuela de Ingeniería y Ciencias, Aguascalientes Campus, Mexico. Prior of that, he was a professor and chair of mechanical engineering at the Universidad de La Sabana in Colombia. He received his Ph.D. degree in engineering sciences (2018) and his M.Sc. degree in automotive engineering (2010) from Tecnológico de Monterrey in Mexico. Julian is electronic engineer from Universidad Autonoma de Manizales (2006 Colombia). Among his lines of research are the following: vehicle dynamics, automotive communication protocols, testing engineering, and biomedical equipment design.

Felipe Arenas Uribe. Mechanical Engineering student from Universidad de La Sabana. Member of the Human Centered Design center, where he has focused on working in intelligent transportation systems. Captain of the Unisabana Herons EV since 2020, team with which he has participated in 4 Shell Eco-Marathon Americas events. His research interests include vehicle dynamics, instrumentation, and control.

Diego Ernesto Contreras Domínguez. Mechatronics Engineer (2011) and Ph.D. of Engineering Sciences (2018) from the Tecnologico de Monterrey, Mexico. He is currently a professor in the mechanics department at the Tecnológico de Monterrey Campus Estado de México, lecturing on automotive topics. His research interests include automotive electronics and control, vehicle dynamics, and automotive instrumentation.

Virgilio Vásquez López. Electrical Mechanical Engineer from the Universidad Veracruzana, Veracruz Mexico and completed his master's and doctoral studies at the Center for Advanced Studies of the Instituto Politecnico Nacional (CINVESTAV-IPN), Mexico City in 1997 and 2005, respectively. He is currently a full-time professor in the Mechatronics department of the Tecnológico de Monterrey Campus Estado de México. His research interest includes mechatronics instrumentation, automation, and control theory.