Transport, Mobility & Society. 2025; 4:202
doi: 10.56294/tms2025202

ORIGINAL

Computer-to-vehicle linking system, Alexa auto

Sistema de enlace para el funcionamiento entre computador y vehículo, Alexa auto

Alexis Javier Villacis Ninasunta[1], Mariana Pinargote Basurto[1]

[1]Universidad Técnica Estatal de Quevedo, Facultad de Ingeniería, Departamento de Ingeniería Mecánica. Ecuador.

Cite as: Villacis Ninasunta AJ, Pinargote Basurto M. Computer-to-vehicle linking system, Alexa auto. Transport, Mobility & Society. 2025; 4:202. https://doi.org/10.56294/tms2025202

Submitted: 31-07-2024 **Revised:** 26-10-2024 **Accepted:** 15-07-2025 **Published:** 16-07-2025

Editor: Prof. Emanuel Maldonado

ABSTRACT

The automotive industry has seen the need to improve the comfort of the user by integrating intelligent voice-controlled systems. This aims to prevent potential distractions while driving and enhance the user experience on their journeys. Nowadays, voice assistants that interact with specific vehicle information are exclusive to high-end models or specific brands. However, there are other voice assistant alternatives available on the market such as the Echo Auto, which is designed for use in vehicles allowing access to the functionalities of the Alexa voice assistant. Conversely, its functions are limited to access to entertainment and information applications. Considering this context, this work presents the development of a system that allows extracting information from sensors of the Toyota Hilux CS 2020 vehicle, such as the status of the fuel system, ignition timing, coolant temperature and the air-fuel ratio. It also includes control of the activation of the door locks of a Mazda CX-3 vehicle. To do this, an MCP2515 CAN bus controller has been used to extract or write information from the bus and an Arduino Nano microcontroller to process it and send it to the cloud using a WIFI module. The Blynk IoT platform is responsible for receiving this data or sending it to the microcontroller and presenting it in an interface so that the user can monitor it. Through an Alexa skill programmed in Voiceflow, it is possible to access this data or modify it using voice commands given by the user. The main limitations encountered stem from the characteristics of the CAN bus in the vehicle, as not all the modules for function control are always connected to the main bus, for this reason it was necessary to use different vehicles.

Keywords: Echo Auto; Alexa; CAN bus; Vehicle Information; Function Control.

RESUMEN

La industria automotriz ha visto la necesidad de mejorar el confort del usuario integrando sistemas inteligentes controlados por voz. Esto busca prevenir distracciones mientras se conduce y mejorar la experiencia del usuario en sus travesías. Actualmente, los asistentes de voz que interactúan con información específica del vehículo son exclusivos de modelos de alta gama o marcas específicas. Existen otras alternativas de asistentes de voz disponibles en el mercado como el Echo Auto, que está diseñado para uso en vehículos permitiendo acceder a las funcionalidades del asistente de voz Alexa. Sin embargo, sus funciones se limitan al acceso a aplicaciones de entretenimiento e información. En este contexto, este trabajo presenta el desarrollo de un sistema que permite extraer información de sensores del vehículo Toyota Hilux CS 2020, como el estatus del sistema de combustible, el tiempo de avance, la temperatura del refrigerante y la relación aire-combustible. También incluye el control de la activación de los seguros de las puertas de un vehículo Mazda CX-3. Para ello, se ha utilizado un controlador de bus CAN MCP2515 para extraer o escribir información del bus y un microcontrolador Arduino Nano para procesarla y enviarla a la nube mediante un módulo WIFI. La plataforma IoT Blynk se encarga de recibir estos datos o enviarlos al microcontrolador y presentarlos en una interfaz para que el usuario pueda monitorearlos. A través de una skill para Alexa programada en Voiceflow, es posible acceder a estos datos o modificarlos mediante comandos de voz dados

por el usuario. Las principales limitaciones encontradas se deben a las características del bus CAN en el vehículo, ya que no siempre los módulos para el control de funciones están conectados al bus principal, por esta razón fue necesario utilizar vehículos diferentes.

Palabras clave: Echo Auto; Alexa; Bus CAN; Información del Vehículo; Control de Funciones.

INTRODUCTION

Background

With the advent of Industry 4.0 and the introduction of new digital technologies into everyday life, there has been a change in the way users interact with machines. As explained by Ortiz et al.[1], the need to facilitate information retrieval and the advancement of the Internet of Things (IoT) have favored the development of virtual voice assistants, which in turn have changed the way users search for or request information. In this case, the use of voice commands is prioritized to achieve faster interaction on the Internet, so that users do not need to use their hands or eyes.

Among the best-known options on the market are Amazon's Alexa, Google Assistant, Microsoft's Cortana, Apple's Siri, Samsung's Bixby, and others. Their use is particularly notable for home automation applications due to their compatibility with a wide range of sensors and actuators. Options for use inside the vehicle include assistants derived from the above, such as Echo Auto, Android Auto, Carplay, JBL Link Driv, Anker Roav Bolt, and other alternatives.

The functionalities available to these assistants will depend on their specific characteristics, as well as the vehicle model. Several brands, such as BMW, Audi, and Mercedes-Benz, have their own virtual assistants, which allow users to access vehicle information and even specific control functions, such as regulating the temperature inside the cabin, operating the windshield wipers, locking the doors, and more. Among the most current assistants are Hey Mercedes! (Mercedes), Laura (Skoda), Ok Honda (Honda), etc.[2] The rest of the assistants are limited to more common functions that are not directly related to the status of the vehicle or its control, much less have free software that allows for expanding the commands for accessing or managing information within the vehicle.[3]

Thus, major companies such as Google, Apple, Microsoft, Amazon, and even well-known vehicle brands are currently investing significant capital to integrate voice technology into their products, naturally including the automotive sector to achieve greater user comfort.[4] More and more vehicle models are now coming with this technology built in, while for those that do not have it or older models, devices are being developed to work in these environments, such as Amazon's Echo Auto or Android Auto. These have improved their voice recognition capabilities in noisy environments.

Amazon's official website features a list of vehicles in which Alexa has already been integrated. Among the most recognized brands and models are Audi, BMW, Chevrolet, Ford, Toyota, and Nissan, among others. However, the lists only include models from recent years, mostly from 2019 onwards.[5] And their control functions over the vehicle are limited and will depend on the hardware and software features of that vehicle. In addition, in 2021 Amazon announced Alexa Custom Assistant, which allows device manufacturers and service providers to create their own assistants based on Alexa technology and thus integrate it from the manufacturing stage.[6]

Most devices that connect to the vehicle via its OBDII (On Board Diagnostics) port are limited to scanners for fault detection. These may have their own screens for displaying information or require a mobile application that connects mainly via Bluetooth. The best known and lowest cost device on the market is the ELM327, a diagnostic device used to read and analyze vehicle codes that report problems.[7]

Mobile applications compatible with the ELM327, such as Infocar, Car Scanner, OBD Auto Doctor, among others, available in the Play Store, allow you to view various data such as speed, temperature, fuel level, etc.[8] However, these devices do not have control options that can be processed by voice commands, as their main function is to diagnose the condition of the vehicle. On the other hand, this offers an important opportunity, as there is an interface where the information obtained can be quickly verified by building your own interface.

As for the connection of the microcontroller to the voice assistant, the main function is communication and linkage between the reading of variables from the vehicle's OBDII port and the voice assistant and its ability to audibly provide this information to the user.[8] Currently, there are several platforms that allow this connection to be made, such as Blynk, Particle, Home Assistant, among others, so that the user can give voice commands to access data from sensors or control actuators that are connected to the microcontroller. This makes it possible to connect from the vehicle to the microcontroller and from there to the virtual assistant to achieve the proposed goals.

How can we develop a hardware and software solution that allows voice assistants, specifically Alexa Auto,

to be integrated with a vehicle's OBDII port to control functions and access variables in real time, overcoming the current compatibility and control limitations present in commercial assistants and devices?

Objective

To develop hardware and software that allows interaction with Alexa Auto to control functions and read vehicle variables.

METHOD

Study design

An applied technological development study was carried out with experimental and descriptive evaluation. The engineering phase (hardware-software prototyping) was complemented by validation tests in a single group with repeated measurements for: (a) accuracy of sensor variable readings via OBDII/CAN and (b) effectiveness of actuator activation via voice commands.

Population, setting, and materials

- Vehicles tested: Toyota Hilux CS 2020 (standard SAE J1979 sensor reading) and Mazda CX-3 2019 (actuator activation on secondary CAN bus).
- Vehicle interface: OBDII port (CAN_H 6 pins, CAN_L 14 pins; 500 kbps and 125 kbps buses depending on model).
- Hardware: Arduino Nano (ATmega328), MCP2515/TJA1050 CAN controller/transceiver, ESP01 WiFi module (ESP8266), DC-DC regulation (MP1584 → 5 V; LM1117 → 3,3 V), OBDII cable, and duplex extension.
- Assistant and platforms: Echo Auto (Alexa), skill designed in Voiceflow; Blynk IoT backend for reading/writing variables; "CANSniffer" PC app for frame inspection in Mazda.
- Test environment:
- Vehicle at rest (engine running, no gear engaged).
- Vehicle in motion (≈30–40 km/h, windows closed).
- Reference operating conditions were recorded (approximate ambient noise, rpm).

Variables

- Independent/controlled: vehicle condition (stationary vs. moving), command type (query vs. action), sampling interval (500 ms for sending to Blynk), CAN filters, link quality (WiFi/Bluetooth).
- Dependent (sensors in Toyota):
- Fuel system status (PID 03).
- Coolant temperature (PID 05, °C).
- Ignition advance time (PID 0E, ° before TDC).
- Air-fuel ratio (AFR) derived from λ (PID 24). Accuracy quantified as % error relative to a parallel-connected ELM327 reference scanner.
- Dependent (actuators in Mazda): Command compliance (success/failure) for door lock/unlock; effectiveness (%) reported.

Procedure

1. HW/SW implementation: Circuit assembly (dedicated PCB board) and OBDII–MCP2515–Arduino Nano (SPI)–ESP01 (UART) connection. CAN bus initialization (500 kbps or 125 kbps as appropriate) and ID filters.
2. Sensor reading (Toyota):
    - Sending OBDII requests (ID 0x7DF, Service 0x01, PID according to variable).
    - Receiving/parsing responses (ID 0x7E8), decoding and normalizing to physical units.
    - Transmission to Blynk every 500 ms for consumption by the Alexa skill.
    - Validation: ELM327 in parallel using a duplex cable; 5 measurements were taken for each variable and condition (rest/movement); point and average errors were calculated.
3. Actuator activation (Mazda):
    - Empirical/reverse engineering identification of relevant CAN frames (support with CANSniffer and community documentation).
    - Design of intents in the skill; update of virtual variables in Blynk; propagation to ESP01 and publication of the CAN frame from Arduino Nano.
    - Validation: 10 attempts per action (lock/unlock) with the vehicle in parking mode; compliance record.
4. Voice recognition robustness: execution of commands in both scenarios (moderate noise at rest and in motion) to observe practical sensitivity of the voice channel.

Data analysis

- Sensors: descriptive statistics; comparison with ELM327; calculation of error (%) per attempt and average error per variable and condition.
- Actuators: compliance rate and total effectiveness (%) by command type.
- Acceptance criteria: errors ≤5 % for continuous variables and effectiveness ≥80 % for voice actions were considered adequate for the proof of concept.

Ethical and safety considerations

- Operational safety: low-speed driving tests and controlled routes; activation of actuators only in parking mode to avoid risks; no critical safety systems were altered.
- Vehicle integrity: use of standard SAE J1979 messages for reading; for writing in Mazda, limited to documented/non-critical frames; no permanent or invasive modifications.
- Privacy and data: no personally identifiable information was collected; Blynk tokens and credentials were managed in a restricted manner; no sensitive user data was stored.
- Regulatory compliance: compliance with current traffic regulations and good laboratory practices; the system was evaluated as an experimental prototype, not intended for commercial use without additional certifications.

RESULTS AND DISCUSSION

Design of the Vehicle-Alexa Auto Communication System Introduction

This stage details the design of both the hardware and software. As first part, a general proposal is made for the hardware and components to be used, and to achieve communication between the vehicle's ECU and a microcontroller that will be responsible for processing the data to be sent via Wi-Fi to the cloud.

This is based on the diagram in figure 1, which shows the connection between the different components from the vehicle to the voice assistant. Regarding the software required, further information is provided on the structure and content of the CAN messages that must be used to achieve communication with the data bus. Once the data has been obtained, the Blynk and Voiceflow IoT platforms need to be configured, which are used to store the variables read from the vehicle and to develop the Alexa skill, respectively.

Design of the hardware used

Figure 1 shows the block diagram of the hardware components used for the link system between the vehicle computer and Alexa Auto, starting from the connection with the vehicle to the Wi-Fi communication with the voice assistant.

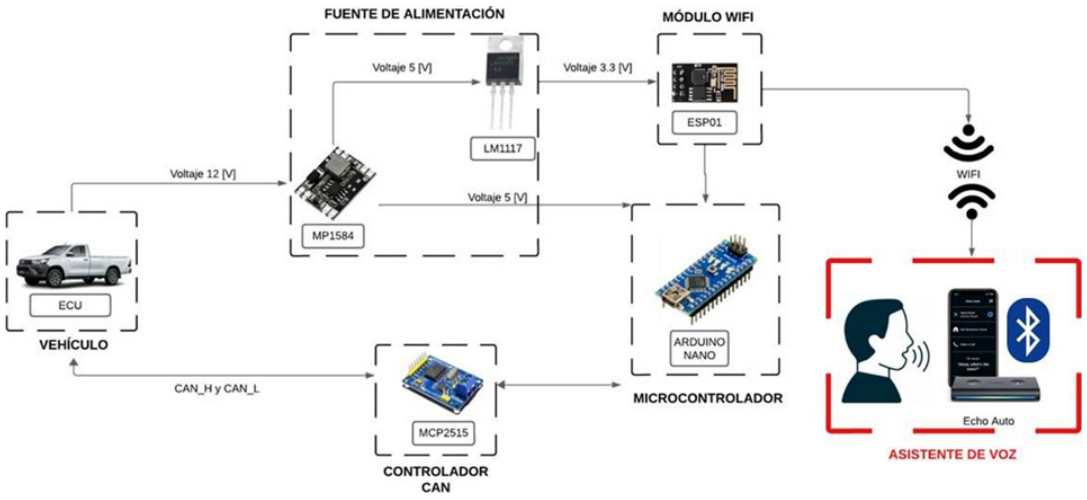

Figure 1. Block diagram of the hardware for interconnecting the vehicle computer and Alexa Auto

Subsequently, the specifications of each element that makes up each block of the aforementioned diagram are described.

Technical specifications of the components in the vehicle

It is important to detail the functionality of the components and devices essential for implementing the link system between the vehicle computer and the Echo Auto voice assistant. These components include the OBDII cable and the vehicles with the sensors or actuators used in this work.

Vehicles used

To establish the connection with the vehicle computer, it is essential to know the protocol governing the OBDII port, as this may vary depending on the vehicle manufacturer, and this information will provide the basic criteria for selecting the electronic components necessary for the overall system.

Sometimes, manufacturers incorporate one or two CAN buses in the same vehicle to distinguish the modules that interact with the vehicle's computer, as shown in figure 2, where pins 3 and 11 are also designated as CAN Bus (Object Detection), enabling communication with the vehicle's instrumental modules, i.e., with the actuators. Likewise, pins 6 (CAN_H) and 14 (CAN_L), defined by the ISO 15765-4 standard, are responsible for connecting the main modules of the vehicle.

In the case of the 2020 Toyota Hilux CS pickup truck, it was determined that it only has one CAN bus, which means that the modules for controlling windows, lights, door locks, and others are independent modules that are not directly connected to the ECU. Therefore, another alternative was chosen, namely a Mazda CX3, which does have two CAN buses: the main bus used for modules that are important for the vehicle's operation and a secondary bus for modules such as climate control, audio, door locks, among others.

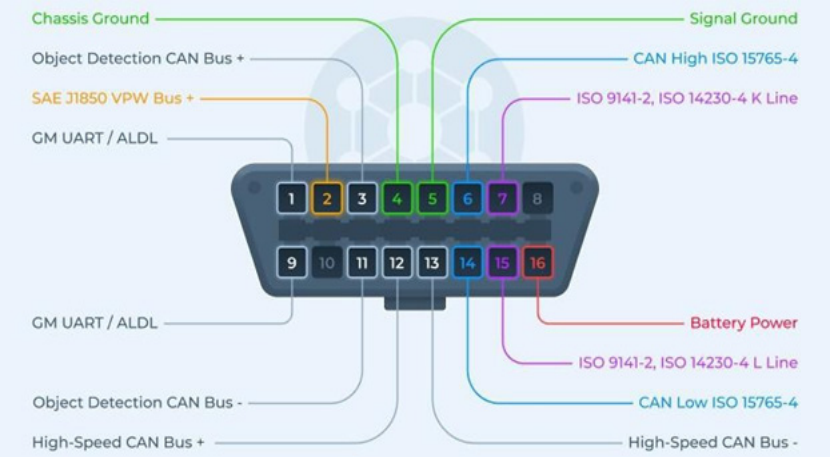

Figure 2. OBDII port pin designation with one or two CAN buses[9]

2020 Toyota Hilux CS

This is a Toyota pickup truck, known for its quality, durability, and off-road capabilities. It has a very robust design, incorporating a front bumper, a revamped grille, and LED lights to improve visibility of the surroundings. Shows an image of the vehicle described, as well as its technical characteristics. In addition, the vehicle's OBDII port can be seen, with the enabled pins identified. These are highlighted with a metallic coating and correspond to pins 4, 5, 6, 7, 13, 14, and 16. According to the information provided in figure 3, this vehicle has a high-speed CAN bus (500 kbps).

Figure 3 shows a representation of the CAN bus with several of the modules that connect to it. This diagram is based on a diagram of the CAN communication system of another Toyota vehicle model,[10] since such information is limited with respect to the specific model being used. However, there are certain similarities between models that are very useful.

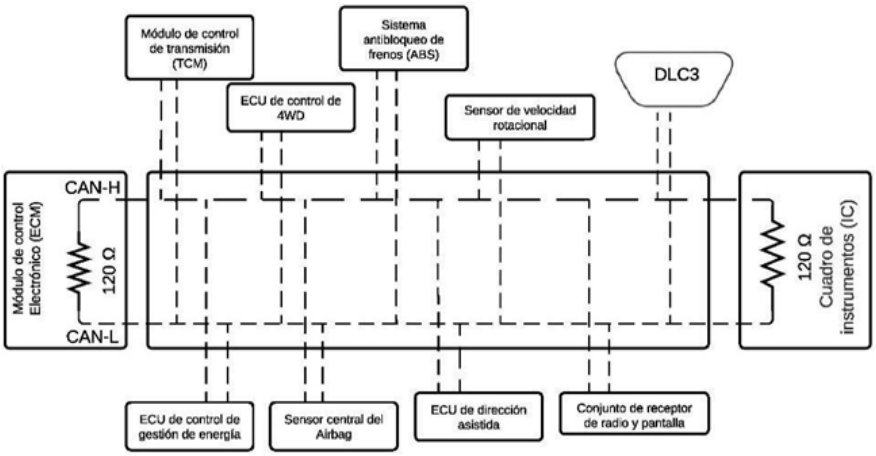

Figure 3. CAN bus of the 2020 Toyota Hilux CS vehicle[10]

As mentioned above, sensor readings will be taken in this vehicle, which are described below:

Oxygen sensor

Converts the amount of oxygen in the gases produced by the car into an electrical signal, which is then read by the Engine Control Unit (ECU) to compare and see if the air-fuel mixture is ideal or adequate. If there is too much oxygen in the exhaust gases, the injection time is increased to obtain optimal engine power, resulting in a stoichiometric ratio of 14,7 parts air to one part fuel.[11] Shows this sensor and several of its technical characteristics.

Coolant temperature sensor

This sensor reports the coolant temperature to the central computer. The physical model can be seen, along with its technical characteristics. This allows the computer to check whether the engine is cold or hot and what temperature it is at, in order to adjust the injection and start-up of the vehicle. Its basic operation is based on the variation of an internal resistance depending on the measured temperature.[12]

Position sensors

To measure the advance time or ignition advance, information from two position sensors is required. The first is the crankshaft position sensor, which detects the position of the crankshaft and measures the number of revolutions of the engine.[13] This sensor and its technical characteristics can be seen.

The second is the camshaft position sensor, which is used to monitor the position and rotational speed of the camshaft.[14] Details the respective characteristics of the camshaft sensor. The combination of data from both sensors allows the vehicle's computer to calculate and adjust the vehicle's ignition timing.

Fuel system

The fuel system, as shown in figure 4, consists of the fuel tank, pump, filter, injectors, and pressure regulator, which are responsible for supplying fuel to the engine. Once activated, this system indicates to the vehicle's computer that it is on, off, has a fault, among other options. Within the aforementioned context, the parameter provided to the ECU by this system will be used to determine the general status of the car once it has been started.

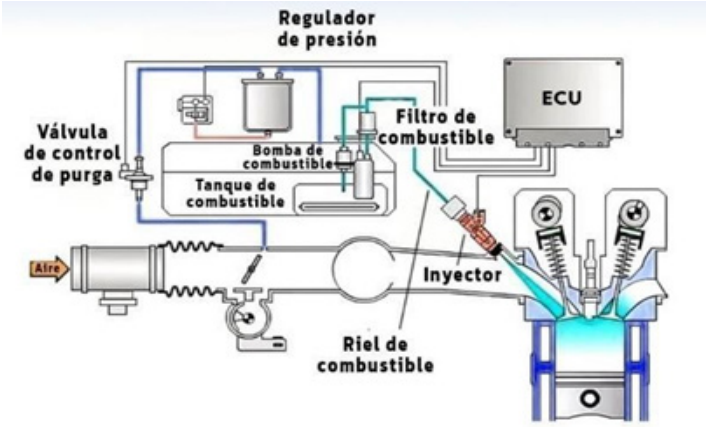

Figure 4. General representation of a fuel system

2019 Mazda CX-3

The 2019 Mazda CX-3 is a compact vehicle that offers a variety of features, including its design, interior quality, and sporty performance, with good acceleration and handling characteristics. It has a touchscreen entertainment system, intrusion sensor alarm, electronic brake assist, among other features. An image of the vehicle described and several of its technical features. The vehicle's OBDII port can also be seen, and the enabled pins have been identified. These correspond to pins 3, 4, 5, 6, 8, 11, 14, 15, and 16. According to the information provided in figure 4, this vehicle has two CAN buses, which allow access to the vehicle's sensors and certain actuators connected to the ECU.

In this case, the Mazda CX-3 vehicle has two CAN buses, one high-speed (500 kbps) and one lower speed (125 kbps). Figures 5 and 6 show a representation of these buses. It should be noted that this diagram is based on information obtained from the ForSCAN diagnostic software, which is capable of reading all vehicle modules connected to the CAN network. Similarly, additional information is provided by the CAN network diagram of a MAZDA CX-5[14], which is a model that shares certain characteristics with the Mazda CX-3.

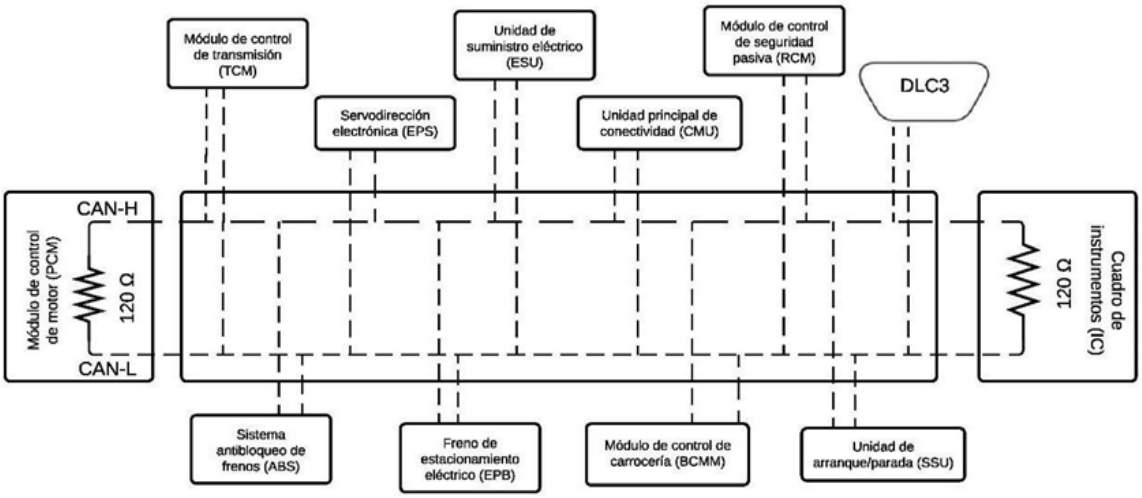

Figure 5. CAN bus (500 kbps) of the 2019 Mazda CX-3 vehicle

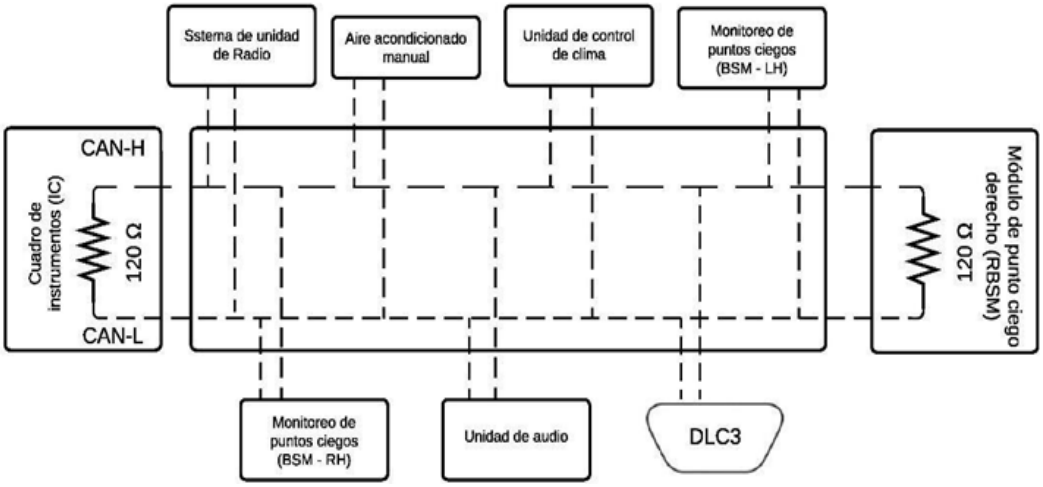

Figure 6. CAN bus (125 kbps) of the 2019 Mazda CX-3 vehicle

Due to these characteristics in this vehicle, an actuator will be activated, in this case the door lock, which is described below:

Electric actuator for door locks

This system allows the vehicle door lock to be moved in order to access or deny access to the vehicle, as shown. The control signal comes from an electrical system that can be connected to a button, a key, or the vehicle's computer.[15]

Extension cable for OBDII connector

To avoid any inconvenience or obstruction of the system that must be connected to the OBDII port, an extension cable connected to this port is used. Shows the cable used and its characteristics. It is designed according to SAE J1962 specifications. With a length of approximately 60 cm, it allows the system to be connected to the side of the driver, since the OBDII port is normally located under or near the steering wheel, so it is necessary that it does not obstruct any area that the driver uses while driving.

Technical specifications of the power supply

The power supply circuit consists of two parts. First, the voltage source from the vehicle is regulated, reducing it from 12[V] DC to 5 [V] DC, which is the voltage level required to power the microcontroller. Second, the resulting 5 [V] is regulated to 3,3 [V], which is the voltage required to power the ESP01 WiFi module.

Fixed DC/DC voltage regulator (MP1584)

The MP1584 DC-DC converter is a step-down or buck switching regulator with high conversion efficiency, excellent line regulation, and low ripple voltage.[16] The physical component can be seen with its technical

characteristics. This voltage regulator was chosen based on the power requirements of the Arduino Nano microcontroller. A fixed output voltage of 5[V] was established to prevent possible variations that could damage the electronic components.

LM1117 Transistor

This is a three-terminal positive voltage regulator, whose fixed regulated output voltage is very useful in standard electronics applications. Although it is primarily designed as a fixed voltage regulator, it can be used with external components to obtain adjustable voltages and currents.[17] It was specifically configured to provide a 3,3 [V] output for the purpose of powering the ESP01 WIFI module. The technical characteristics of the transistor can be found.

Technical specifications of the CAN bus controller: MCP2515 CAN module

The MCP2515 CAN module allows different devices to communicate using the CAN protocol. This module has the ability to receive and send data packets in standard and extended format, although it has masks and access filters, which reduce the load on the main microcontroller.[17] It should be noted that it includes the MCP2515 CAN controller chip with SPI interface and the TJA1050 CAN transceiver chip to facilitate communication with microcontrollers and development boards, in this case Arduino. Shows the physical module mentioned and its technical characteristics.

Technical specifications of the Microcontroller: Arduino Nano

Arduino Nano is a microcontroller board based on the ATmega328, notable for its small size compared to other Arduino boards, as can be seen. It is compatible with a wide variety of electronic components and is specially designed to work on protoboards and facilitate the prototyping of circuits or projects with limited space.[18]

Technical specifications of the ESP01 WiFi Module

The ESP01 WIFI module integrates the ESP8266 System On a Chip (SOC) and is compatible with Arduino. It integrates a powerful 32-bit architecture processor and WiFi connectivity, allowing it to work as an application host or reduce the WiFi networking load of another processor, making it suitable for IoT applications.[19] It can be used in a variety of applications, such as remote device control, data collection, monitoring, and control of IoT devices, among others.[20]

Technical specifications of the voice assistant: Echo Auto

Amazon Alexa's Echo Auto, as shown, is a voice assistant that connects via Bluetooth to the Amazon Alexa mobile application. It is used to receive user requests and process the desired information through the programming of the specific skill for this project. A notable feature of this device is its integration of eight microphones, which makes it suitable for noisy environments, an important factor in voice command recognition.

General hardware diagram for connecting the vehicle to a microcontroller

According to the block diagram shown in figure 7, the schematic in figure 7 was implemented, showing each of the main components of the circuit.

The first section of the circuit deals with the regulation of the voltage source from the vehicle, which supplies 12 [V] to power the entire system. Using the MP1584 regulator, an output voltage of 5 [V] is generated, which is used by the Arduino Nano and MCP2515 microcontrollers. Additionally, this voltage is used to power the LM1117 transistor, which is configured to adjust its output voltage to 3,3 [V], in order to supply power to the ESP01 WIFI module.

To connect the Arduino Nano to the MCP2515 module, SPI communication is used, connecting the respective pins (CS, MISO, MOSI, SCK, and INT). Similarly, in the MCP2515 module, the connection to the vehicle's CAN bus is made via the CAN_H and CAN_L pins, respectively. Finally, the connection between the Arduino Nano microcontroller and the ESP01 WiFi module is made using serial communication between the two components via the RX and TX pins.

After verifying the correct operation and the respective connections of all the electronic components, we proceed with the design of a printed circuit board (PCB). This has dimensions of 73,66 mm wide by 73,53 mm long. Shows the PCB design for the interconnection of all components. Likewise, shows the same design in a three-dimensional model, allowing the precise locations of each element to be visualized.

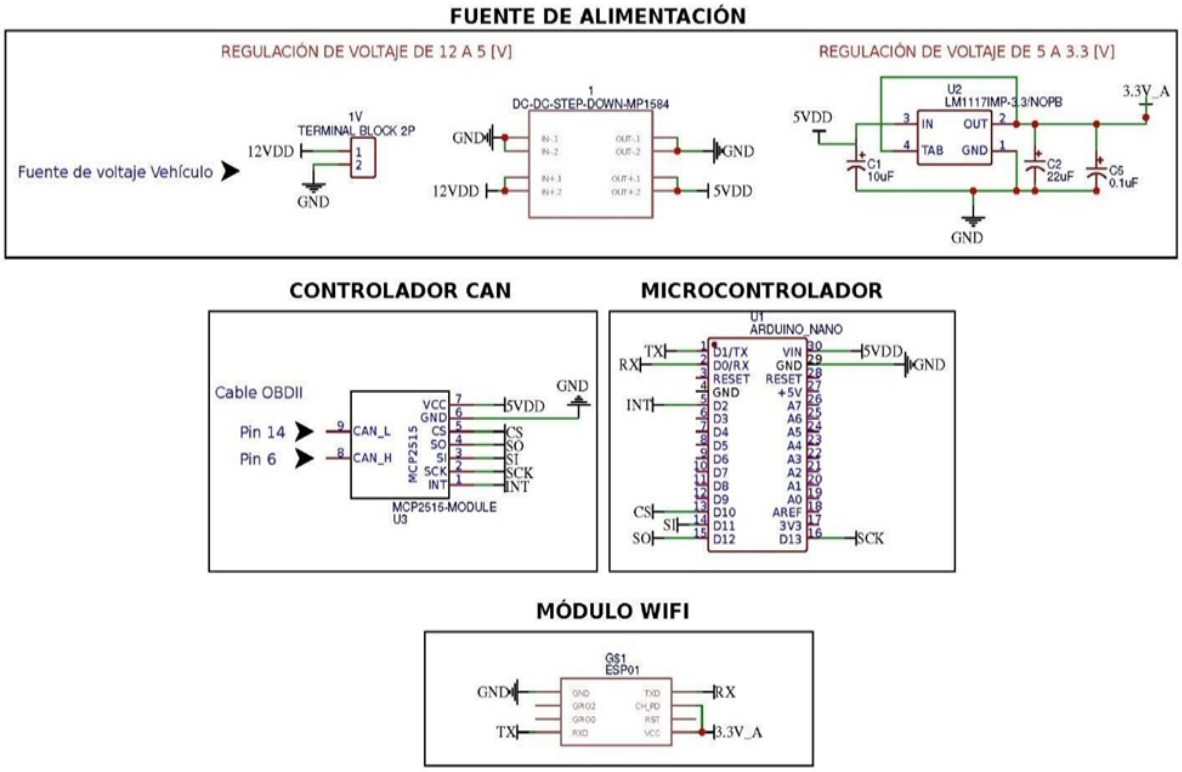

Figure 7. Schematic circuit of the Alexa Auto link system and the vehicle computer

Finally, this circuit has been put into operation and the respective circuit current has been measured, giving an average value of:

I = 146,2 [mA]

Identification of CAN messages used in the OBDII protocol

To establish communication with the CAN bus, it is necessary to know the structure of CAN messages and the services that can be accessed in this protocol. As part of the data field, the standard[21] defines nine diagnostic services, which must be specified in the query message. These services are detailed. The service that will be used to read sensor data is service 0x01, which allows current diagnostic information to be requested. In addition to these nine services, in April 2021, the SAE presented a successor to SAE J1979, called SAE J1979-2, which specifies 26 additional Unified Diagnostic Services (UDS), starting at address 0x10 and ending at 0x2F.[22]

Functional addresses are used to perform a standard query or response on the vehicle's CAN bus. For a diagnostic query, ID 0x7DF is used, which is a r received by the ECU and responds with ID 0x7E8. These specifications are defined in the SAE J1979 standard, as well as the structure of the message data field and the structure of the respective response.

To find out the parameter IDs (PIDs) available in the Toyota truck's computer, a request for PID 00 is made. This allows the PIDs compatible with the vehicle to be displayed in a 4-byte data format, where each of the bits that make up the response represents one of the following 32 PIDs established in SAE J1979, specifying whether or not that PID is compatible.

The request made is as follows: 0x7DF020100, where the bytes sent to the CAN bus are detailed in table 1 below:

Table 1. PID 00 request – Show available PIDs

			Data bytes		
11-bit identifier (Request)		Number of additional data bytes	Service (01 = show current data)	PID code (00 = Display PIDs)	Not used (ISO 15765-2 suggests CCh)
7DF		02	01	00	

The response to this request is as follows: 0x7E8064100BE1FA813, where the bytes received correspond to

the parameters specified in table 2:

Table 2. PID 00 response – Show available PIDs

Data bytes								
11-bit identifier (Response)	Number of additional data bytes	Query service, plus 40h (41h =show)	PID code (00 = Show PIDs)	Data byte 0	Data byte 1	Data byte 2	Data byte 3	Not used
7E8	06	41	00	BE	1F	A8	13	

Using the bytes from the previous response, they are decoded into a total of 32 bits. Details each of these bits and their relationship to the respective PIDs. This makes it possible to determine which PIDs are available or not in the range from 0x01 to 0x20.

Following this same methodology, two additional requests are made for PID 0x20 and PID 0x40 to find out what other parameters are available in the vehicle in the range from 0x21 to 0x40 and from 0x41 to 0x60, respectively. With the responses obtained from these PIDs, decodes and details all PIDs that are compatible with the 2020 Toyota Hilux CS vehicle.

In the case of the Mazda CX3, identifying the actuator IDs involves a different process. One option is to use the hardware designed above to read the data from the CAN bus and present it in an interface that groups and sorts it for better analysis. In this case, the open-source application "CANSniffer," available on GitHub,[23] is used. This desktop application is specifically designed to read CAN messages from a computer's COM port. To do this, the microcontroller is connected to the computer via USB, and all messages read from the CAN bus are printed to the serial port. The application will then connect to the serial port and begin displaying the messages in the interface shown in figure 8:

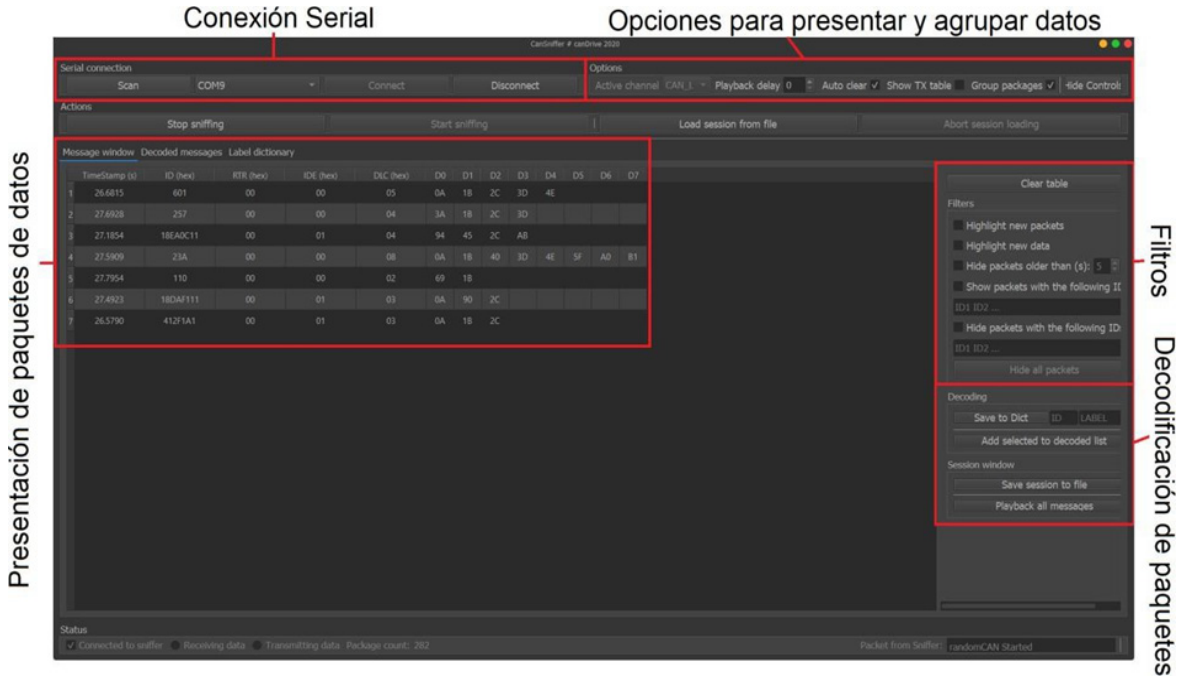

Figure 8. CANSniffer graphical interface

This interface provides access to various functions such as packet grouping, ID filtering, packet decoding, and others. Once the desktop application has been launched, the process for identifying the actuators is empirical or reverse engineering, as an actuator in the vehicle is activated manually and the new packets that have been deployed are verified in the interface. By repeating this process several times, the CAN message related to the activation of a specific actuator can be identified. This process is necessary because there is no official information or documentation specifying the CAN packets associated with the different vehicle actuators.

An unofficial alternative for obtaining more information about CAN messages associated with configurations of brands such as Mazda are user forums, which provide valuable information regarding the decoding of messages. An Excel template has even been found available in which several data packets have been identified. It also has tools for converting binary to hexadecimal in the format required to send the message to the CAN bus, making it much easier and more accessible to modify certain ECU parameters.[24]

Variable selection

Based on the information detailed above, the following variables have been selected, both for sensors according to the PIDs available in the Toyota vehicle and for actuators according to information extracted from Mazda user forums.

Sensors in the 2020 Toyota Hilux CS vehicle

Based on the data presented, the following variables related to the vehicle's sensors have been selected, as detailed in table 3.

Table 3. PID request 00 – Show available PIDs

PID	Description	Min	Max	Formula	Units
03	Fuel system status	-	-	-	Table 5
05	Engine coolant temperature	-40	215	A- 40	°C
0E	Advance time	-64	63,5	A() − 64 2	° Before PMS
24	Oxygen sensor 1 (Air-fuel ratio)	0	<2	2/(256A+B) 65536	Ratio
	Oxygen sensor 1 (Voltage)	0	<8	8/(256C+D) 65536	V

PID 03 – Fuel System Status

PID 03 returns two bytes of data, the first describing the vehicle's primary fuel system and the second describing secondary fuel systems, if any. The data values that each byte can take are described in table 4:

Table 4. Fuel system statuses

Value	Description
0	Engine is off
1	Open circuit due to insufficient engine temperature
2	Closed circuit, using feedback from the oxygen sensor to determine the fuel mixture
4	Open circuit due to engine load or fuel cutoff due to deceleration
8	Open circuit due to system failure
16	Closed circuit, using at least one oxygen sensor, but there is a fault in the feedback system

Under normal conditions and when the car is running, the value obtained should be 2, indicating that the fuel system is in good working order and uses the oxygen sensor to determine the fuel mixture.

PID 05 – Engine coolant temperature

Shows the engine coolant temperature. The measurement of this variable is used in the calculation of the air-fuel mixture. The relationship is inversely proportional, since lower temperatures require more fuel and vice versa. As specified in HELLA[12], the coolant temperature should be between 85°C and 95°C when the engine is at full load and between 95°C and 110°C at partial load for a 1,6L engine. It should be noted that these ranges may vary slightly depending on the engine.

The response to the PID 05 request has 1 byte of data which, when converted to a decimal number, can represent a temperature range between -40°C and 215°C.

PID 0E – Advance time

The advance time, or ignition advance, corresponds to the moment when the air-fuel mixture must be ignited to achieve maximum combustion pressure after top dead center (TDC). To do this, ignition must occur at a crankshaft angle of approximately 5° to 10° before TDC. This range will depend on the physical characteristics of the engine and the speed at which it is running.[25] If this value is high, the spark will occur earlier than necessary and cause the mixture to ignite in an uncontrolled manner, causing what is known as detonation. The consequences can include damage to the pistons, connecting rods, and other components, as well as reduced engine efficiency. Conversely, if the advance time is delayed or occurs after TDC, incomplete combustion of the mixture occurs, affecting engine power and performance since the maximum amount of energy released in combustion is not utilized.[26]

The PID 0E request returns 1 byte of data that can be used to describe a range between -64° and 63,5° before TDC. In this case, a normal range between 0° and 25° has been considered, as suggested in CERVANTES GAS[26] based on tests carried out on different vehicle models.

PID 24 – Oxygen Sensor 1

The oxygen sensor determines the composition of the exhaust gases by measuring the oxygen concentration in the exhaust pipe. The signal provided by this sensor is used to determine the optimal air-fuel ratio. Depending on the value of the ratio, it is possible to determine whether a mixture is rich or lean. If there is more air than fuel in the mixture, it is considered a lean mixture, and conversely, if there is more fuel than air, the mixture is considered rich.[27] The PID 24 request returns 4 bytes of data, the first two bytes A and B are used to calculate the air-fuel equivalence ratio called lambda (λ). According to SAE J1979 specifications, the values that $\boldsymbol{\lambda}$ can take are between 0 and less than 2.

For this application, the lambda factor is used, which if $\boldsymbol{\lambda} > \mathbf{1}$ indicates a lean mixture, and if $\boldsymbol{\lambda} < \mathbf{1}$ the mixture is rich. When the value is $\boldsymbol{\lambda} = \mathbf{1}$, the air-fuel ratio (AFR) is considered to be exactly what is needed for efficient combustion, considered a stoichiometric mixture. In the case of gasoline engines, this air-fuel ratio is 14,7:1.[28] Thus, to determine the air-fuel ratio (AFR), the following formula is used:

$$\boldsymbol{\lambda} = \frac{\boldsymbol{AFR}}{\boldsymbol{AFR_{estequimetrica}}} \qquad (\mathbf{1})$$

From which we obtain:

$$\boldsymbol{AFR} = \boldsymbol{\lambda}\ \mathbf{x}\ \boldsymbol{AFR_{estequimetrica}} = \boldsymbol{\lambda}\ \mathbf{x}\ \mathbf{14.7} \qquad (\mathbf{2})$$

AFR values below 14,7 indicate a rich mixture, and higher values indicate a lean mixture. Depending on the engine's performance, the AFR value may vary. However, as explained in Fernández F.[28], when the load and acceleration are moderate, the ratio should be close to the stoichiometric ratio (AFR = 14.7:1). If the load is higher or the power is at maximum, this ratio can reach up to AFR = 13:1. and conversely, if the load and acceleration are low, the ratio can be up to AFR = 16:1. Higher or lower values in this range may indicate a problem with the vehicle's performance.

Mazda CX-3 vehicle actuators

To select the actuator to be activated, an Excel template developed by Litnevskyi[29] is used, which organizes much of the information from the data packets handled on the CAN bus for Mazda vehicles. This template details each of the modules that the vehicle may have and the data blocks in each one. In addition, he has decoded several of the codes, associating them with vehicle variables or actuators, such as automatic light adjustment and automatic door locking, among others.

For this application, the option that allows the door locks to be activated or deactivated has been selected. To do this, the messages that must be sent to the CAN bus have been identified:

- To deactivate or unlock the doors, the following message is used, which is associated with the module with ID 7B7, in block 02 and the first frame. The complete message is: 0x7B70201400000000001.
- To activate or lock the doors, a message is sent to the same address or ID, changing the bits corresponding to the activation of the respective actuator. The complete message is: 0x7B702011000000000D1

According to the information provided by Litnevskyi[29], these messages maintain the following general structure, which is presented in table 5, for one of the above messages:

Table 5. CAN message structure for door lock activation in Mazda

ID		No. Block	Message No.	Data bytes			Checksum	
07	B7	02	01	40	00	00	00	01

In this type of message, each bit of the data bytes is assigned or related to a specific vehicle function. Due to limited official information or data, the operation of the vast majority is unknown, so it is not recommended to modify this data without being certain of its function.

Software design for reading and writing data in the vehicle computer

The "Arduino-CAN" library is used to read and write messages on the CAN bus of the vehicle's computer. It is compatible with the MCP2515 module and several models of Arduino boards. It has the necessary functions to write messages to the bus, as well as to read and filter the messages found on it. This makes it possible to send requests to the bus to read data from the vehicle's sensors, and when the respective module returns a response with the desired variable value, it is possible to filter the corresponding message for processing. On the other hand, to activate actuators, instead of making a data request, only the respective message specifying the information necessary to activate the desired actuator is sent to the bus.

General functions of the Arduino-CAN library

To communicate with the CAN bus, the "Arduino-CAN" library has various functions that allow both reading and writing messages. Several of the functions necessary to achieve this purpose are described below:

- CAN.begin (500E3): this function initializes the CAN bus at 500 kbps, which is the standard speed used for a high-speed communication bus, which is used in the vehicle for the interconnection of important systems.
- CAN.filter (ID): this function allows you to add a filter to receive only messages with a specific ID. This way, only messages with that ID will be read, and not all messages on the bus.
- CAN.beginPacket (ID): initializes a new CAN packet or message with the specified ID. ID 0x7DF is used for reading data, indicating that a request for information is being made.
- CAN.write (Data): sends a CAN message to the bus with the previously specified ID.

With this function, each of the message parameters or data must be specified byte by byte.

- CAN.endPacket(): ends the packet that was written to the CAN bus.
- CAN.parsePacket(): this function checks if there is a new message in the receive buffer and if it is available to be read.
- CAN.read(): allows CAN bus messages to be read as a defined structure that stores the data that make up the message.

Using these functions, the code is designed to read various vehicle variables and activate any of the actuators available in the vehicle.

Reading vehicle sensor variables

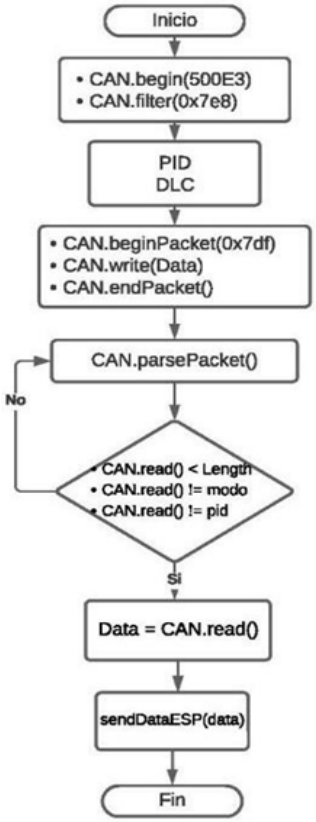

Figure 9. Flowchart for reading ECU sensor information using Arduino Nano

Figure 9 describes the procedure for making an information request to the ECU to read variables from the sensors connected to the vehicle. First, the CAN bus is initialized at a speed of 500 kbps, and a filter is added to read only the messages containing the requested variable data. The CAN.filter(0x7E8) function is used to filter messages with this ID, which corresponds to the responses to an information request.

Next, it is necessary to define the PID of the variable to be read and also the data length (DLC) of the message to be written to the bus. With this information, a new message is started using the CAN.beginPacket(0x7DF) function. Next, the data is written to the bus, specifying the PID of the desired variable, as well as other parameters necessary for the message, and it is finalized with the CAN.endPacket() function.

Once the message has been written to the bus, a response message is awaited that corresponds to the previously configured filter ID, which is done by the CAN.parsePacket() function. When a message is available, a check is performed to ensure that the message corresponds to the requested information.

In this process, the length of the message, the PID, and the service used are verified to be correct; otherwise, the system continues to wait for the correct message. When the verification is successful, the message data is read, which will correspond to the variables of the requested sensors. Finally, this data is sent to the ESP01 via serial communication. There, it will be processed and sent to the Blynk platform cloud.

This procedure is performed in a loop to read each of the vehicle sensor variables. The only parameters that are modified each time it is executed are the PID and the DLC, as they depend on each desired variable. The respective code for the Arduino Nano is available.

In the ESP01 WIFI module, the data sent by the Arduino Nano serial port is read and processed. This process is shown in figure 10, which begins with the configuration for the connection to the Blynk platform and also uses the timer.setInterval() function to set the time interval in which certain functions will be executed, such as reading data or sending it to the Blynk platform.

Next, the data received at the serial port is read, where each data frame received has an identifier that corresponds to each of the selected variables. It should be noted that the data received has not been pre-processed, i.e., it corresponds to the hexadecimal data returned by the ECU. Therefore, once it is read and identified, it is processed as appropriate, since the data for the sensor variables must be transformed to a known scale. Finally, the values obtained for each sensor are sent to the Blynk platform. This is done using the Blynk.virtualWrite(V0, Data) function, which allows the data from each sensor to be written to the platform's virtual variables.

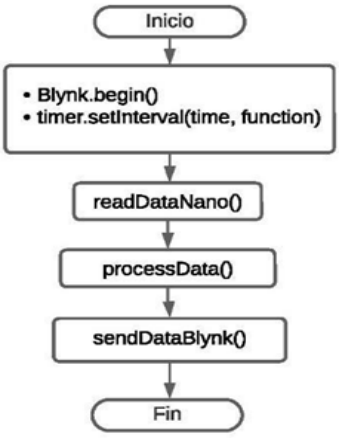

Figure 10. Flowchart for reading data from the ESP01 module's serial port and sending it to the Blynk platform

In this case, the function responsible for reading data from the serial port will run every 500 ms, which will allow the data to be constantly updated on the Blynk platform. The respective Arduino code for the WIFI module is available.

Activation of vehicle actuators

In the case of actuators, the user gives the command to activate one. To do this, the skill is responsible for registering the command and associating it with a specific variable, either for activation or deactivation. These variables are created on the Blynk platform, and from the skill, a call is made to the Blynk API to update the value of one of the variables associated with a specific actuator.

Figure 11 shows the process implemented in the ESP01 module when an actuator is to be activated. The first step is to initialize communication with the Blynk platform using the Blynk.begin() function. The program then uses the BLYNK_WRITE(V0) function to monitor any changes in a specific variable, in this case the virtual variable V0. Thus, once the skill updates the value of the variable, this function is executed, which is responsible for

sending the frame with the ID of the actuator to be activated via serial communication. It will then wait again for any change in any virtual variable on the Blynk platform. The Arduino code for the WIFI module is available.

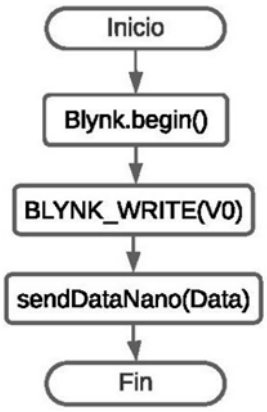

Figure 11. Flowchart for monitoring changes in virtual variables

The Arduino Nano follows the procedure shown in figure 12, where the CAN bus is initialized at a speed of 125kbps. The data sent by the ESP01 from the serial port is then read to identify the frames received and the corresponding actuator. With this information, the respective message is written to the CAN bus, specifying the actuator ID and the data needed to activate it.

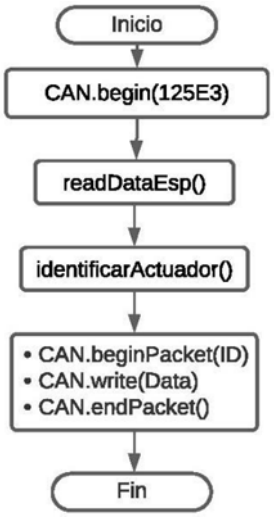

Figure 12. Flowchart for reading and identifying frames for actuator activation

The corresponding code for the Arduino Nano is available.

Configuration of the Blynk IoT platform

To use Blynk, you must register and create an account on the platform. You will then receive an authentication token by email, which is used to access data from the variables or devices configured on the platform. In this case, the device will be the ESP01 module, which acts as a WiFi module and connects to the Arduino Nano microcontroller.

When programming the WiFi module, you must add the necessary libraries and the information required to establish communication with the Blynk platform. As shown in figure 13, you need to define the project ID, the token, and the libraries for connecting to the platform data:

Once the device has connected to the platform, the corresponding control panel is created with each of the variables. In this instance, six variables have been configured. The first variable is a timer that indicates the seconds elapsed since the device connected to the IoT platform. The second variable reflects that the WiFi module connection to the platform has been established. The remaining four variables represent the values from the vehicle's sensors with their established ranges.

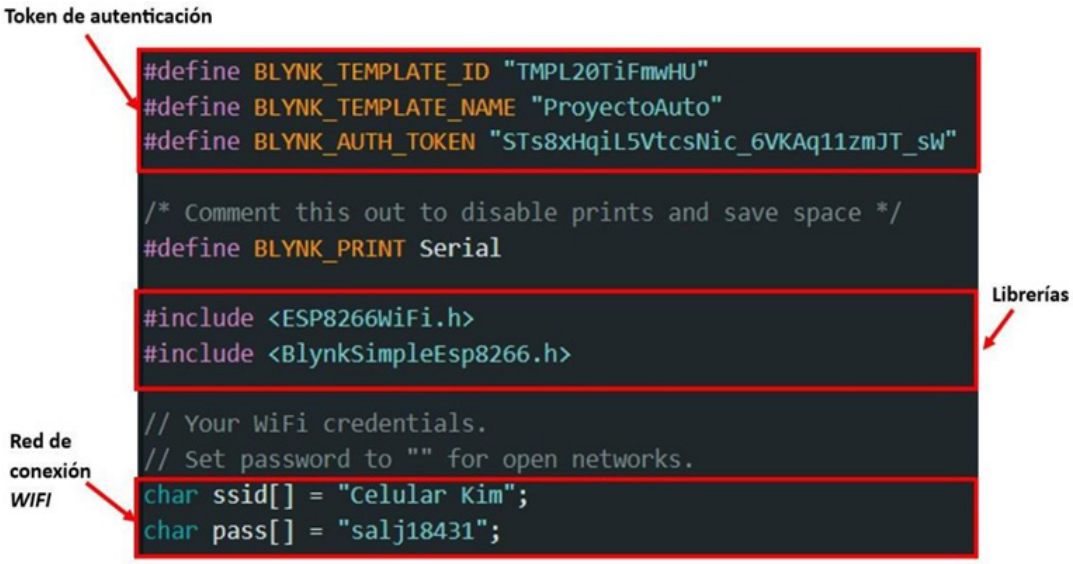

Figure 13. Configuration of libraries, authentication token, and WiFi connection network

Figure 14 shows the interface with the selected variables, where their status can be constantly monitored.

Figure 14. Display panel for the selected variables on the Blynk platform

Development of the Skill for Alexa

The skill is designed and programmed on the Voiceflow platform, where block programming is used to develop each of the functionalities that the skill may have. Additionally, this platform allows all the necessary configurations and data required by the skill to be transferred to Amazon's own development console, so that it can be used on Alexa devices.

Reading sensor variables

Figure 15 shows the flowchart of the programming in Voiceflow for reading the variables of the selected sensors. In this case, the skill's activation phrase will be "Alexa, open my Toyota." This will start the conversation, where the assistant will ask the user for the variable they want to know, and depending on their response, several cases will be presented. These correspond to each of the variables, as the user's response is compared with the available options, and the one that matches will be given as the answer.

If the user does not know how the skill works, there are two alternatives. The first is the "Help" option, which will explain the purpose of the skill and its functionalities. The second alternative is "Options," which will simply remind the user of the functionalities they can select.

Once Alexa has stated the value of the variable, a check is performed to see if this value is within the normal ranges; if not, a message is displayed to alert the user. If everything is in order, the user is asked if they want any additional information. If the answer is yes, the user can choose another variable, and if they do not want more information, the conversation ends.

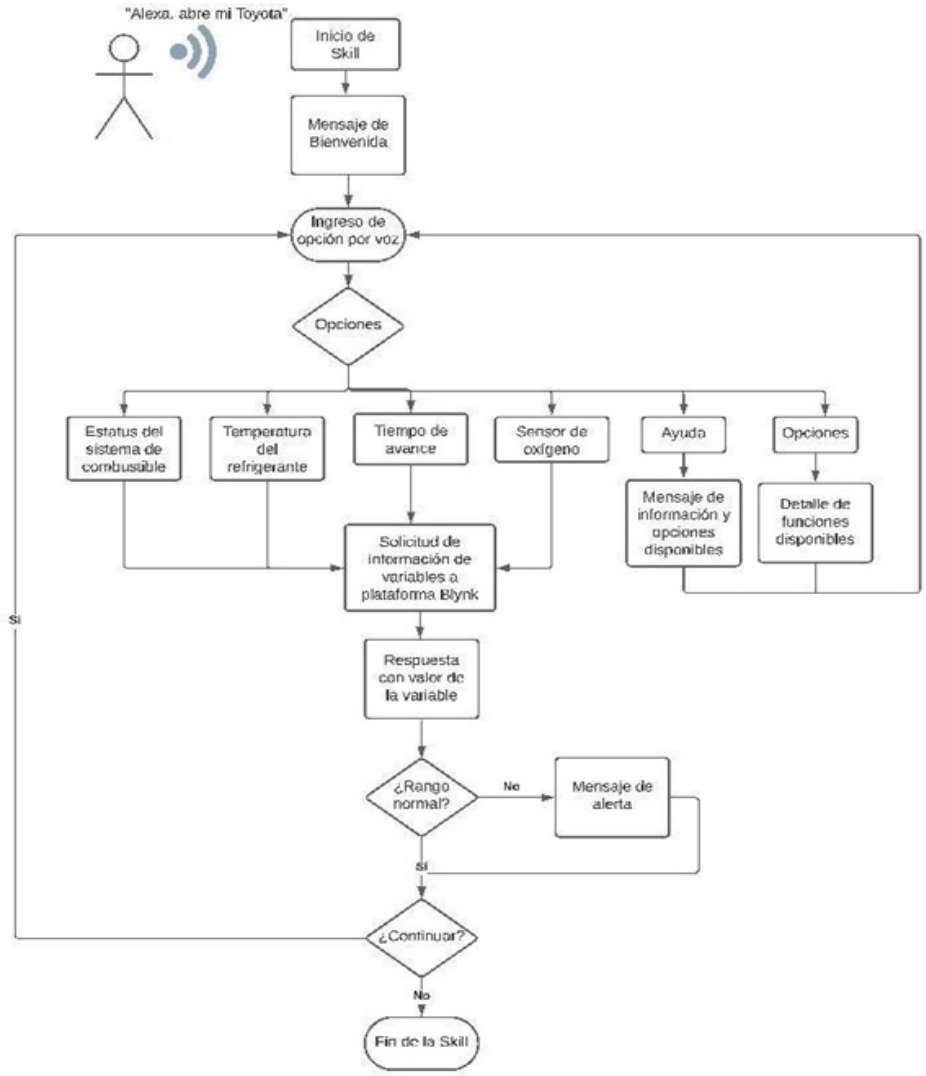

Figure 15. Flowchart of the skill designed to read variables from the sensors of the Toyota Hilux CS 2020 vehicle

- Fuel system status: responds with one of the alternatives presented in table 5, which refers to the different states of the fuel system.
- Coolant temperature: Informs the user of the temperature of the coolant.
- Advance time: informs the user of the advance time or ignition advance before TDC.
- Oxygen sensor: informs the user of the air-fuel ratio (AFR).
- Help: provides more information about the skill and its features.
- Options: informs the user only of the features it has.

Actuator activation

In the case of actuators, the skill program flow is similar to that of sensor reading. However, instead of sending a request for information to the Blynk platform, an update of data on the respective variables is performed. Figure 16 shows the corresponding flowchart for activating the vehicle door locks.

In this case, the Mazda CX3 vehicle is used, as mentioned above, because it has two CAN buses. However, due to the limited information about the messages that must be written to the CAN bus to activate any of its actuators, the option for the user to activate or deactivate the vehicle's door locks has been selected.

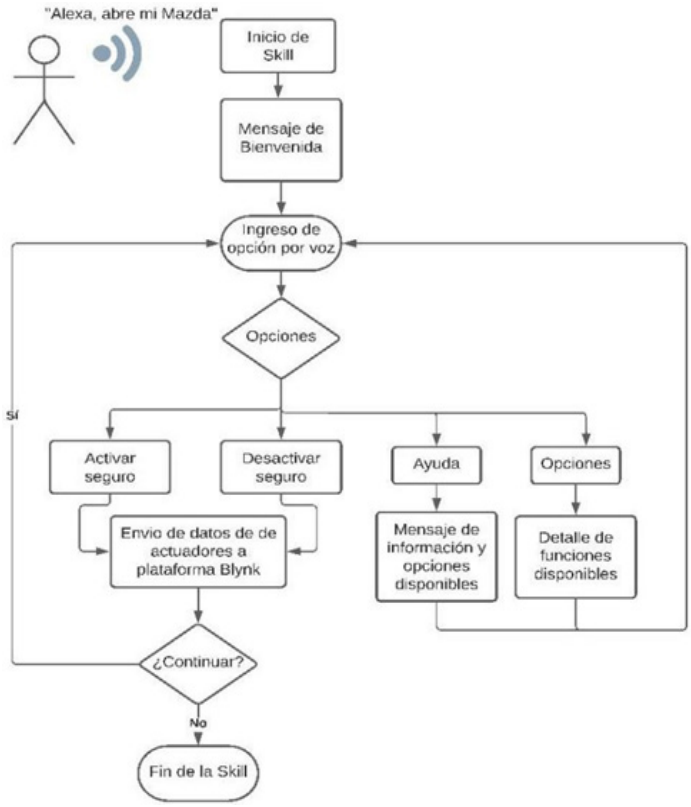

Figure 16. Flowchart of the skill designed to activate the vehicle's locks

The flow that follows the conversion between Alexa and the user is very similar for both sensor reading and actuator activation. In this case, the programmed functionalities are:

- Activate lock: updates a variable on the Blynk platform related to the activation of the door locks.
- Deactivate lock: this updates a variable on the Blynk platform related to the deactivation of the door locks.

The "Help" and "Options" alternatives have the same functionality as in sensor reading.

General system connection diagram

The main objective of the system is to connect the vehicle to the Alexa Auto voice assistant. Priority is given to the acquisition of variables in real time and their integration into a cloud server to manage the information efficiently. Figure 17 shows the connection diagram for the devices and platforms used.

Figure 17. Diagram for interconnection of Alexa Auto and the vehicle computer

For the diagram shown in figure 17, the MCP2515 module is first connected to the vehicle computer via the standard pins (CAN_H pin 6 and CAN_L pin 14) of the OBDII port, which enables the transmission and reception of data packets via the CAN bus. On the one hand, this allows a request to be made for data from the sensors

with the respective PIDs and also to receive the corresponding data in response. On the other hand, it allows messages to be sent on the CAN bus to activate specific actuators, provided that the module ID and the bits to be modified are known.

The CAN controller uses an SPI communication interface to connect to the Arduino NANO microcontroller, which is responsible for both monitoring the information to be sent to the vehicle and receiving the corresponding data from the sensors.

The data from each of the sensors or actuators is then sent or received via serial communication to the ESP01 WiFi module. At this point, the connection to the Blynk platform is established, to which the data read from the sensors is constantly transmitted, and in return, it sends information if the user wishes to activate an actuator.

On the Blynk platform, a panel is created to display the variables received, where the sensor values can be constantly checked, as well as whether the WIFI module is connected. On the other hand, the Voiceflow platform allows the creation of the skill, where the logic that will guide the user's interaction with the Alexa voice assistant is implemented. And through the API blocks, the data from Blynk is collected. To do this, the requests made have the following structure:

- To read data from a variable: https://{server_address}/external/api/getAll?token={token}
- To update values in a variable: https://{server_address}/external/api/update?token={token}&{pin}={value}

It should be noted that when reading a variable, it must be stored in a local variable of the skill so that it can be given to the user as an audible response. This allows the necessary information to be obtained and the user's questions to be answered.

The skill will include actions that the voice assistant will execute in relation to the information previously collected from the vehicle variables. It is important to note that within the skill's programming logic, the data obtained from the sensors is verified to be within normal ranges, which allows the user to be alerted to abnormal vehicle behavior.

Implementation of the vehicle communication system hardware and software

The implementation of the system involves the use of an OBDII cable connected to the vehicle's port, a 3D-printed box that houses the PCB board, and a button to power the circuit. The Echo Auto device is also part of the system, as it is responsible for receiving voice commands from the user and transmitting them to the different platforms. The box housing the PCB board is 3 [mm] thick and measures 10,3 [cm], 9,3 [cm] long, and 4 [cm] high, excluding the lid structure. As shown in figure 18, there is an opening in the lower left corner of the box for connecting the OBDII cable to the PCB board, while a slot has been created on the front to insert an ON/OFF switch that will enable the system to be activated.

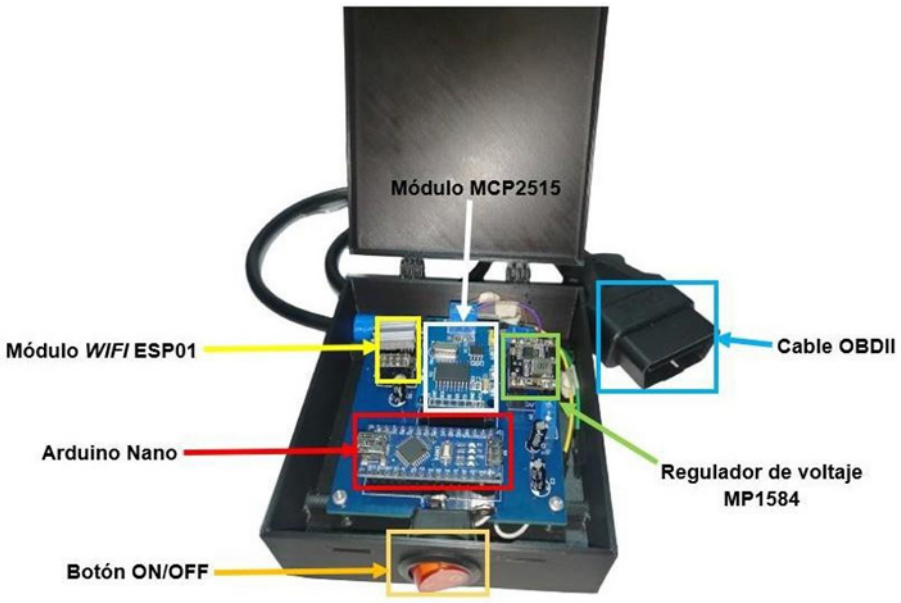

Figure 18. Device implementation

It is important to note that these devices will be strategically located so as not to cause discomfort to the driver.

Tests and Results

Introduction

The functional tests of the developed skill focus on reading sensors and activating the actuator in the respective vehicle, in order to evaluate that the values given to the user are correct and that the door locks are controlled. These tests will be carried out in different scenarios or situations, including: the vehicle at rest with the engine running, and the vehicle in motion.

Functional testing of the skill for reading sensors

These tests were carried out based on the requests programmed in the skill, which refer to each of the selected variables of the 2020 Toyota Hilux CS vehicle. Table 6 describes these variables and the respective voice command, through which the tests will be carried out in various environments. It should be noted that various alternatives for each command were considered in the programming of the skill, meaning that the user can use similar words to select one of the alternatives.

Table 6. Variables and voice commands for testing the skill's functionality

Selected variable	Voice command
Fuel system status	Car status
Engine coolant temperature	Antifreeze temperature
Advance time	Car advance timing
Oxygen sensor	Oxygen sensor
Help	Help
Options	Options

Each of the recorded values was checked using the ELM327 module, a 16-pin automotive scanner that connects directly to the OBDII interface. It is used to read error codes and also allows all vehicle parameters to be viewed using a mobile or desktop application to which it connects wirelessly via Bluetooth.[30] This device (ELM327) will be connected to the vehicle at the same time as the device developed in this project to collect information in real time, using a duplex OBDII cable. As for the activation of the door locks, the tests carried out are similar, as they verify that the voice commands are correctly recognized and that the requested action is executed.[31,32]

Sensor reading tests with the vehicle at rest

To perform tests with the car at rest, we considered the situation in which the engine is on but not running. In other words, the car is not moving.[33,34] The aim is to determine the Echo Auto's ability to respond in situations with little noise in the environment. For each case, a total of five tests were considered for each of the available interactions.[35,36]

The following considerations were taken into account during the test:

- The engine is running, but the vehicle is not moving.
- The approximate ambient noise is 61 dB.
- Revolutions per minute: 656 rpm
- The car is in neutral.
- The accelerator is not being pressed.

Based on these considerations, the values for each of the selected sensors have been recorded.

Fuel system status

Table 7 shows the results obtained from reading the fuel status with the engine running:

Table 7. Results of tests reading the fuel system status variable

Test	1	2	3	4	5
Car status	2	2	2	2	2
ELM327 value	2	2	2	2	2

In this case, the value obtained of 2, according to table 8, indicates that the fuel system circuit has been closed, using feedback from the oxygen sensor to determine the fuel mixture. Both the value obtained from the skill and the ELM327 coincide in each of the attempts made.[37,38,39]

Antifreeze temperature

To record the temperature values in table 8, the engine was kept running for approximately 10 min. As a result, the temperature rose to approximately 45°C.[40,41,42]

Table 8. Results of the tests to read the vehicle's antifreeze temperature variable

Test	1	2	3	4	5
Antifreeze temperatura antifreeze	45°C	45°C	45°C	46°C	46°C
ELM327 value	45°C	45°C	45°C	46°C	46°C
Error	0	0	0	0	0

In this test, the values provided by both the skill and the ELM327 scanner match 100 %. In this case, it should be noted that after a period of time with the engine running, the temperature has stabilized at approximately 45°C.[43,44,45]

Advance time

The results of the advance time reading are presented in table 9. In this specific case, with the engine running but not in motion, there are no major variations in the recorded advance time.[46,47,48]

Table 9. Results of the advance time variable reading tests for the vehicle

Test	1	2	3	4	5
Advance time	10	10	10	11	10
ELM327 value	9	10	10	10	10
Error	11,11	0	0	10	0

In this case, the average error obtained is 4,22 %. Since the car is not running, the advance time value is set at approximately 10° before TDC.[49,50,51]

Oxygen sensor

Table 10 shows the air-fuel ratio (AFR) values obtained from the oxygen sensor:

Table 10. Results of the vehicle oxygen sensor variable reading tests

Test	1	2	3	4	5
Oxygen sensor	14,72	14,75	14,73	14,71	14,75
ELM327 value	14,72	14,76	14,73	14,72	14,76
Error	0	0,06	0	0,06	0,06

The average error found is 0,036 %, which does not indicate a significant variation from the ELM327 reference values.[52,53]

In this test, with an additional source of noise, two main points have been identified that generate variations in the recording of sensor data.[54,55,56] On the one hand, the pronunciation of long commands can cause Echo Auto to not record the audio correctly. On the other hand, in variables such as advance time or air-fuel ratio (AFR), the values provided by the skill with respect to those of the ELM327 show greater error.[57,58] This is mainly because these are variables that change rapidly over time and generate small variations in the data since both systems do not read the variables at exactly the same time.[59,60]

Sensor reading tests with the vehicle in motion

To perform these tests, a situation in which the vehicle is in motion was considered. In this case, there is a considerable difference in noise inside the vehicle. Similarly, for the recording of sensor data, a total of five tests were considered for each of the available interactions.[61,62,63,64]

The following considerations were taken into account when carrying out this test:

- The engine is running and the vehicle is in motion.
- Windows are closed.
- The approximate ambient noise is 63 dB.
- The car is traveling at an average speed of 30-40 km/h

Under these conditions, the sensor variables are recorded and compared with those provided by the ELM327 scanner.[65]

Fuel system status

Table 11 shows the results obtained from reading the fuel status while the vehicle is in motion:

Table 11. Results of the fuel system status variable reading tests

Test	1	2	3	4	5
Car status	2	2	2	2	2
ELM327 value	2	2	2	2	2

In this case, the value of 2 remains the same as in the previous tests, since the engine is running and the fuel system is not malfunctioning.

Antifreeze temperature

The antifreeze temperature values are shown in table 12. While the vehicle is running, the temperature values have risen considerably but remain within normal ranges.[66,67,68]

Table 12. Results of the tests to read the vehicle's antifreeze temperature variable

Test	1	2	3	4	5
Antifreeze temperature	81°C	81°C	80°C	81°C	81°C
ELM327 value	81°C	80°C	80°C	80°C	81°C
Error	0	1,25	0	1,25 %	0

In this test, the average error obtained is 0,5 %. Given that the system generally maintains an intermittent internet connection, there are errors in the reception of data from the moving vehicle.[69,70,71]

Travel time

Table 13 shows the results of the advance time data readings.[72,73]

Table 13. Results of the vehicle advance time variable reading tests

Test	1	2	3	4	5
Time of advance	29°	41°	37°	42°	42
ELM327 value	30	40	37	42	41
Error	3,33 %	2,5	0	0	2,44

In this case, there is an average error of 1,65 %. Since the vehicle is in motion, the advance time value varies greatly as it depends on factors such as load, speed, and acceleration throughout the journey. However, the values read do not vary greatly from the values read by the scanner.[74,75,76]

Oxygen sensor

Table 14 shows the air-fuel ratio (AFR) values obtained from the oxygen sensor with the vehicle in motion. In this specific case, as the vehicle is in motion, these values vary according to the driving conditions, such as the speed or load of the vehicle:

Table 14. Results of the vehicle oxygen sensor variable reading tests

Test	1	2	3	4	5
Oxygen sensor	14,82	14,60	14,77	14,77	14,56
ELM327 value	14,81	14,62	14,71	14,76	14,54
Error	0,06	0,14	0,41	0,06	0,14

In this situation, an average error of 0,16 % is shown. Since the values are less than 1 %, it is considered an acceptable error between the data provided by the proposed circuit and the value obtained with the ELM327 device.[77,78,79]

The errors presented in the sensor reading tests with the vehicle in motion are less than 2 %, which indicates that the existing difference can be considered acceptable in this application. It is important to consider that there are multiple factors external to the circuit that can affect the response given.[80,81,82] These factors may originate from the vehicle, such as speed, load, acceleration, and other physical characteristics during driving, as well as factors external to the vehicle, such as the WIFI module's internet connection and the Bluetooth link

between the mobile device and the Echo Auto, which may affect the device's operation. However, the recorded results for each variable do not show significant errors.[83,84]

Actuator activation skill tests

To perform the actuator performance tests, the following conditions were considered with respect to the state of the car (Mazda CX3) and the environment:

- The engine is running, but the vehicle is not in motion.
- The approximate ambient noise is 50 dB.
- Revolutions per minute: 750 rpm
- The car is in parking mode
- The accelerator is not being pressed

Under these conditions, table 15 shows the results regarding compliance with the requested command, whether to activate or deactivate the vehicle door locks.

Table 15. Results of the tests of the skill for activating vehicle actuators

Test	1	2	3	4	5	6	7	8	9	10	Total [%]
Activate insurance	C	C	C	C	C	X	C	C	C	C	90
Disable lock	C	C	X	C	C	C	C	C	C	X	80 %

This test achieved 85 % effectiveness, as in the tests that presented errors, the system did not correctly recognize the voice command to activate or deactivate the vehicle door lock.[85,86] As the commands are very similar, the user may make pronunciation errors, and additional background noise may cause this type of error when registering the command with the voice assistant.[87,88] It should be noted that this activation was performed in parking mode, as in this mode the actuator is in automatic operation and can be interacted with using the programming described above.[89,90] It is important to mention that this mode of interaction with the vehicle was implemented due to the vehicle's own safety measures to prevent the doors from being accidentally opened while the vehicle is in motion.[91]

CONCLUSIONS

A system has been implemented that is capable of interconnecting a vehicle's computer to the Alexa voice assistant via the Echo Auto device. This system allows access to sensor information and the activation of specific actuators. For this purpose, hardware capable of connecting to the vehicle's OBDII port and, in turn, to the CAN communication network has been selected, with the aim of being able to send messages to the bus for reading or writing a variable.

The hardware designed includes an Arduino NANO microcontroller, which is responsible for specifying which messages should be sent and what information the CAN messages should contain, thus defining the data necessary for obtaining specific variable data. In addition, it sends this data to the WIFI module, which is responsible for processing it and sending it to the Blynk platform, where all this information will be available for the user to monitor.

IoT platforms offer a great advantage, whether for managing IoT devices or for accessing information on variables from these devices. This has enabled the optimal development of a skill for Alexa, since HTTP requests make it possible to access information from the Blynk platform and give the user an audible response on available sensor data.

This system presents an important opportunity for the development of applications that increase passenger comfort inside a vehicle and also anticipate possible engine malfunctions. Voice commands can be used to access car diagnostic information or even control certain actuators, all without the driver having to take their hands off the wheel or divert their attention from the road.

Reading vehicle variables or parameters in the 2020 Toyota Hilux CS pickup truck has involved an exhaustive study of the OBDII protocol, since for this purpose it is necessary to know the structure of the messages used and the regulations governing the communication system. This has made it possible to identify which standard parameters are or are not compatible with the vehicle and thus select the most appropriate ones. With this information, the software that will write or read the messages from the CAN communication bus has been successfully developed.

The limited official information about the parameters or messages handled within the CAN communication bus of vehicles greatly limits the possibilities for actuator control. Added to this are the characteristics of the communication network architecture, as not all vehicle control modules are connected to the CAN network, and therefore access is restricted. Despite this, information has been extracted from unofficial sources on several decoded message packets related to the control functions of certain actuators for the Mazda CX-3. This

has made it possible to activate and deactivate the door locks using voice commands through the developed system.

The tests carried out indicate a high level of effectiveness in voice command recognition both when the vehicle is stationary and in motion. The values of the variables read do not show more than a 4 % error with respect to the reference data provided by the ELM327 scanner when the car is stationary. When in motion, the error is less than 2 %. These errors can be explained by the fact that the advance time and AFR are variables that change rapidly, and the designed system does not read the variables at the same time as the ELM327, thus generating small variations.

With regard to the tests carried out for the activation of the actuator in the Mazda CX-3 vehicle, 85 % effectiveness was achieved, which indicates a high level of recognition with respect to voice commands. However, the implementation of this function was only achieved under specific conditions, due to the characteristics of the vehicle's configuration, which limit the modification of certain parameter values.

RECOMMENDATIONS

Seek official information from the vehicle's communication network before attempting to modify any vehicle parameters.

Do not exceed a length of 50 cm from the OBDII port to the physical system developed to avoid interference or significant voltage drops.

Verify the vehicle's protocol before connecting the system, as the system described is only compatible with the CAN protocol.

Before reading the PID, check the PIDs that are available in the vehicle used to prevent the system from sending messages to the data bus requesting variables that are not available and that may cause errors in the program.

Provide the system with a stable Wi-Fi network, as data is constantly sent to the cloud and, in the event of connection interruptions, the variables are not updated and the user may obtain results that do not correspond to the moment they were requested.

BIBLIOGRAPHIC REFERENCES

1. Ortiz A, Dávila R. Implementación de un asistente virtual para los estudiantes de pregrado de una universidad peruana. Rev Conrado. 2023;19(92):121-8.

2. Gonzales P. Los coches que mejor te hablan. 2022. https://hackercar.com/los-coches-que-mejor-te-hablan/

3. Fernández R. Amazon Alexa Auto Tech: The ultimate guide to Alexa in your car. MUO. 2022. https://www.makeuseof.com/alexa-auto-tech-ultimate-guide/

4. Mihale-Wilson A, Zibuschka J, Hinz O. User preferences and willingness to pay for in-vehicle assistance. Electron Markets. 2019;29:37-53.

5. Amazon. ¿Qué es una Skill de Alexa? 2024. https://developer.amazon.com/es-ES/alexa/alexa-skills-kit

6. Amazon. Design Your Skill. 2024. https://developer.amazon.com/en-US/docs/alexa/design/design-your-skill.html

7. Villén J. Simulador de la ECU de un vehículo con protocolo ISO 9141-2. Sevilla: Universidad de Sevilla; 2016.

8. ELM Electronics. OBD Interpreter: Protocols Support - ELM327 (OBD Interpreter ICs). 2023. https://www.elmelectronics.com/products/ics/obd/

9. Weis O. Conector OBD2 explicado. Flexihub; 2023. https://www.flexihub.com/es/oobd2-pinout

10. Toyota Venza. Toyota Venza: system diagram. 2016. https://www.tovenza.com/system_diagram-1256.html

11. Opinautos. ¿Qué es el sensor de oxígeno (o sonda lambda) en un Hilux? 2024. https://www.opinautos.com/ec/toyota/hilux/guias/sensor-de-oxigeno

12. HELLA. Sensor de temperatura del refrigerante. 2024. https://www.hella.com/techworld/mx/Informacion-Tecnica/Sensores-y-actuadores/Revision-del-sensor-de-temperatura-del-refrigerante-4277

13. AUTODOC CLUB. Sensor del cigüeñal: fallas, función, síntomas. 2021. https://club.autodoc.es/magazin/sensor-del-ciguenal-fallas-funcion-sintomas

14. Mazda. Mazda CX-3. 2024. https://www.mazda.com.ec/mazda-cx-3-360

15. Transductor.net. Apertura eléctrica de puertas actuadores. 2023. https://transductor.net/apertura-electrica-de-puertas-actuadores/

16. MACTRONICA. Convertidor DC-DC 12V a 5V 1A reductor. 2024. https://www.mactronica.com.co/convertidor-dc-dc-12v-a-5v-1a-reductor

17. AV Electronics. Módulo WiFi ESP8266EX. 2024. https://avelectronics.cc/producto/modulo-wifi-esp8266ex/

18. Arduino. Arduino Nano. 2024. https://arduino.cl/arduino-nano/

19. AV Electronics. Arduino Nano. 2024. https://avelectronics.cc/producto/nano-v3/

20. Arduino Spain. Conectar módulo WiFi ESP8266 ESP-01 Arduino. 2023. https://arduino-spain.site/modulo-esp8266/

21. SAE. SAE J1979: Surface Vehicle Standard. 2017. https://img.antpedia.com/standard/files/pdfs_ora/20200926/SAE%20J1979-2017.pdf

22. Softing. SAE J1979-2 OBDonUDS – diagnostic standard. Softing; 2023. https://automotive.softing.com/standards/protocols/obdonuds-sae-j1979-2.html

23. Varga A. CANDRIVE. GitHub; 2023. https://github.com/adamtheone/canDrive

24. FORScan forum. Configuration and programming. 2021. https://forscan.org/forum/viewforum.php?f=16

25. E-AUTO. Avance de encendido. 2012. https://www.e-auto.com.mx/enew/index.php?view=article&id=3429

26. CERVANTES GAS. Consecuencias de un auto con avance excesivo: ¿qué debes saber? 2022. https://cervantesgas.com.ar/que-pasa-si-el-auto-esta-muy-avanzado/?expand_article=1

27. Freudenburger B. Sensores de oxígeno 101. 2020. https://www.walkerproducts.com/wp-content/uploads/2020/05/OXYGEN-SENSORS-101-INFORMATION-BOOKLET-Spanish.pdf

28. Fernández F. La sonda lambda o sensor de oxígeno. 2012. https://www.ea1uro.com/eb3emd/Sonda_Lambda/Sonda_lambda.htm

29. Litnevskyi S. Manual de usuario de Excel Mazda SkyActiv OBD-II calc (FORScan). 2021. https://www.youtube.com/watch?v=N9si40XZbcg

30. AutomotrizEscaner. Escáner Automotriz ELM327. 2024. https://www.automotrizescaner.com/escaner-automotriz-elm327-usb

31. Amazon. Conector macho de 16 pines J1962 OBD2 OBD-II para abrir el cable, cable de extensión de diagnóstico OBD para bricolaje. 2020. https://www.amazon.com/dp/B07F16GPMB?starsLeft=1&ref_=cm_sw_r_cso_wa_apan_dp_KM170K9HRAWAJMJKM2WX&th=1

32. Amazon. For the first time, Amazon enables companies to access Alexa's advanced AI to build their own intelligent assistants with Alexa Custom Assistant. Fiat Chrysler Automobiles is the first Automotive OEM to implement in vehicles. 2024. https://developer.amazon.com/en-US/blogs/alexa/alexa-auto/2021/01/Amazon-Announces-Alexa-Custom-Assistant

33. Amazon. ¿Qué es el Alexa Skills Kit? 2024. https://developer.amazon.com/es-ES/alexa/alexa-skills-kit

34. Amazon. Build Your Skill. 2024. https://developer.amazon.com/en-US/docs/alexa/build/build-your-skill-overview.html

35. Amazon. Certify and Publish Your Skill. 2024. https://developer.amazon.com/en-US/docs/alexa/certify/certify-your-skill.html

36. Amazon. Develop Your First Alexa Skill With the ASK SDK for Node.js. 2024. https://developer.amazon.com/en-US/docs/alexa/alexa-skills-kit-sdk-for-nodejs/develop-your-first-skill.html

37. Amazon. Echo Auto - Pon Alexa en tu coche. 2024. https://www.amazon.es/dp/B078YP59TT?tag=xtk-pivot-21&asc_refurl=&asc_source=xataka&asc_campaign=everlasting

38. Amazon. Lleva a Alexa en tu vehículo: Todo lo demás. 2024. https://www.amazon.com/-/es/generaci%C3%B3n-modelo-Lleva-Alexa-veh%C3%ADculo/dp/B09X27YPS1?th=1

39. Amazon. Monitor Your Skill Metrics and Earnings. 2024. https://developer.amazon.com/en-US/docs/alexa/monitor/monitor-your-skill.html

40. Amazon. Skill development workflow. 2024. https://developer.amazon.com/en-US/docs/alexa/ask-overviews/what-is-the-alexa-skills-kit.html

41. Amazon. Test and Debug Your Skill. 2024. https://developer.amazon.com/en-US/docs/alexa/test/test-your-skill-overview.html

42. Amazon. Vehicles with Alexa. 2024. https://www.amazon.com/b?node=17744356011&ref_=ALEXA_AUTO_HP_VEHICLES%C2%A0&tag=muo-v2-519n9pv-20&ascsubtag=UUmuoUeUpU2013241&asc_refurl=https%3A%2F%2Fwww.makeuseof.com%2Falexa-auto-tech-ultimate-guide%2F&asc_campaign=Evergreen

43. Android Auto. Compatibilidad con Android Auto. 2017. https://www.android.com/intl/es_es/auto/compatibility/

44. Arévalo F, Ortega A. Desarrollo de una interfaz para la visualización y adquisición de datos provenientes de la ECU a través de OBD-II mediante un dispositivo de comunicación serial y del analizador de gases Qrotech 6000. Universidad Politécnica Salesiana; 2016. https://dspace.ups.edu.ec/bitstream/123456789/12029/1/UPS-CT005836.pdf

45. Blynk.io. Introduction - Blynk Documentation. 2024. https://docs.blynk.io/en/

46. Braun M, Mainz A, Chadowitz R, Pfleging B, Alt F. At Your Service: Designing Voice Assistant Personalities to Improve Automotive User Interfaces. Conf Hum Factors Comput Syst (CHI '19). 2019;1-11.

47. Bret K. Consumers will use voice assistants more while driving in 2019 and they are already influencing new car buyer decisions. Voicebot. 2019. https://voicebot.ai/2019/01/22/consumers-will-use-voice-assistants-more-while-driving-in-2019-and-they-are-already-influencing-new-car-buyer-decisions/

48. Caicedo K, Bodero L. Diseño e implementación de un sistema de conectividad para el monitoreo en tiempo real de los sistemas funcionales de un automóvil. Guayaquil: Universidad Politécnica Salesiana; 2021. https://dspace.ups.edu.ec/bitstream/123456789/21754/1/UPS-GT003585.pdf

49. Calleja J. On the use of virtual assistant for smart cities parking service deployment. Santander: Universidad de Cantabria; 2021. https://repositorio.unican.es/xmlui/bitstream/handle/10902/22078/435274.pdf

50. Contreras J. Usos del puerto OBD2 para diagnóstico del motor de un vehículo desde un dispositivo móvil. Huejutla: Instituto Tecnológico de Huejutla; 2020. https://rinacional.tecnm.mx/bitstream/TecNM/1127/1/JOSE%20ANTONIO%20CONTRERAS%20RAMIREZ.pdf

51. Domínguez J. Desarrollo de una skill educativa sobre el asistente virtual Alexa: estudia conmigo. Elche: Universidad Miguel Hernández; 2022. http://hdl.handle.net/11000/26573

52. Edu J, Ferrer X, Such J, Suarez G. SkillVet: Automated Traceability Analysis of Amazon Alexa Skills. IEEE Trans Depend Secure Comput. 2021;20:161-75.

53. Electronilab. Módulo CAN bus MCP2515. 2024. https://electronilab.co/tienda/modulo-can-bus-mcp2515/

54. ElectrOnline. MINI ELM327 V2.1 Bluetooth HH OBD OBDII avanzado calidad. 2024. https://www.electronline.cl/mini-elm327-v21-bluetooth-hh-obd-obdii-avanzado

55. ENDADO. Sensor temperatura del refrigerante HELLA 89422-35010. 2024. https://www.endado.com/p/sensor-temperatura-del-refrigerante-hella-6pt009107481

56. Falch M. OBD2 explained - a simple intro. 2023. https://www.csselectronics.com/pages/obd2-explained-simple-intro

57. Google Cloud. Cómo implementar un chatbot de IA con Dialogflow. 2020. https://www.cloudskillsboost.google/focuses/634?locale=es&parent=catalog

58. HELLA. Refrigeración del vehículo: conocimientos básicos para el taller. 2013. https://gruasytransportes.files.wordpress.com/2014/09/bhs_2013_refrigeracion_del_vehiculo.pdf

59. Huang J. La norma SAE J1962 OBD II, el conjunto de cable de diagnóstico automotriz. 2024. https://es.made-in-china.com/co_edgarwireharness/product_SAE-J1962-OBD-II-Automotive-Diagnostic-Cable-Assembly_esiineheu.html

60. IONOS. Programación visual: la entrada más sencilla al mundo digital. 2020. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/programacion-visual/

61. Apple. iOS - CarPlay. España: Apple; 2021. https://www.apple.com/es/ios/carplay/

62. International Organization for Standardization. ISO 15031-3: Road vehicles: Communication between vehicle and external equipment for emissions-related diagnostics. 2016. https://cdn.standards.iteh.ai/samples/64636/635b127e64d54bfd90d498c272008392/ISO-15031-3-2016.pdf

63. JBL. JBL Link Drive Optimized for the Google Assistant. 2024. https://ca.jbl.com/LINK+DRIVE.html

64. Katreddi S, Kasani S, Thiruvengadam A. A review of applications of artificial intelligence in heavy duty trucks. Energies. 2022;15:7457.

65. Mazda CX-5 Service & Repair Manual: Controller Area Network (CAN) System. 2016. https://www.mcx5.org/controller_area_network_can_system-1163.html

66. Motores Auto. Sistema de inyección: partes, funcionamiento, tipos. 2023. https://www.motoresauto.com/sistema-de-inyeccion/

67. Muñoz C. Cómo crear tus skills de Alexa, aunque no sepas de programación. Tuexperto.com; 2022. https://www.tuexperto.com/2022/03/24/como-crear-tus-skills-de-alexa-aunque-no-sepas-de-programacion/

68. National Instruments. ¿Cuál es la longitud máxima del cable para un bus CAN? 2023. https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z000000P7ikSAC&l=es-EC

69. Navarro A. Creación e implementación de una skill de Alexa para laboratorios farmacéuticos. Barcelona: Escola Tècnica Superior d'Enginyeria Industrial de Barcelona; 2020.

70. Naylamp Mechatronics. Módulo CAN MCP2515. 2023. https://naylampmechatronics.com/alambrico/253-modulo-can-mcp2515.html

71. PiEmbSysTech. CAN protocol. 2021. https://piembsystech.com/can-protocol/

72. Portilla X. Mi primera Alexa Skill sin escribir código. 2020. https://planetachatbot.com/alexa-skill-sin-codigo/

73. Ramírez M. Asistentes de voz se perfilan como aliados de la revolución automotriz. Forbes México. 2021. https://www.forbes.com.mx/forbes-life/auto-asistentes-de-voz-aliados-revolucion-automotriz/

74. Sánchez L, Molano M, Fabela M, Martínez M, Hernández J, Vázquez D, et al. Revisión documental del protocolo CAN como herramienta de comunicación y aplicación en vehículos. 2016. https://imt.mx/archivos/Publicaciones/PublicacionTecnica/pt474.pdf

75. SEPLN-PD. InLIFE: Tecnologías del Lenguaje aplicadas al envejecimiento activo. In: Annual Conference of the Spanish Association for Natural Language Processing 2022: Projects and Demonstrations; 2022. Coruña. p. 21-3.

76. Smith G. ¿Qué es el bus CAN (red de área del controlador) y cómo se compara con otras redes de bus de vehículos? 2021. https://dewesoft.com/es/blog/que-es-el-bus-can

77. Smith S. Digital voice assistants in use to triple to 8 billion by 2023. Juniper Research; 2019. https://www.juniperresearch.com/press/press-releases/digital-voice-assistants-in-use-to-8-million-2023

78. STP. Sistema de combustible y aditivos de combustible. STP; 2023. https://www.stp.com/es/sistema-de-combustible-y-aditivos-de-combustible

79. Tahsim A. Una descripción general de los conceptos básicos del diseño en Voiceflow. Voiceflow; 2023. https://learn.voiceflow.com/hc/en-us/articles/9174803155341-What-are-Steps-Blocks

80. Tahsim A. Paso API: ¿Cómo hago llamadas API? ¿Cómo conecto mi Voiceflow a las API? Voiceflow; 2023. https://learn.voiceflow.com/hc/en-us/articles/9262787577613

81. Tobisch V, Funk M, Emfield A. Dealing with input uncertainty in automotive voice assistants. In: Automotive user interfaces and interactive vehicular applications. 2020. p.161-8.

82. Toyota Guatemala. ¡Toyota Hilux 2020 versión full equipo! Toyota Guatemala; 2021. https://www.youtube.com/watch?v=GPbdY-u3HYw

83. UBUY. Dorman 937-700 motor de actuador de cerradura de puerta lateral del conductor delantero compatible con modelos seleccionados de Mazda. UBUY; 2023. https://www.ubuy.gt/es/product/2CRNZRDI-dorman-937-700-front-driver-side-door-lock-actuator-motor-for-select-mazda-models

84. Villamar I. Estudio y análisis de los sistemas de diagnóstico en los automóviles modernos sistemas OBD. Cuenca: Universidad de Azuay; 2018.

85. Voicebot. Voice assistant influence on car purchase decision. Voicebot; 2019. https://voicebot.ai/google-home-google-assistant-stats/#voice-asst-infl-car-desc

86. Voiceflow. Quick start guide. Voiceflow; 2024. https://developer.voiceflow.com/docs/get-started

87. Williams K, Peters JC, Breazeal C. Towards leveraging the driver's mobile device for an intelligent, sociable in-car robotic assistant. In: IEEE Intelligent vehicles symposium. 2013.

88. Yadav A. Creación de una habilidad de postre aleatoria personalizada utilizando NodeJS alojado en Alexa. C# Corner; 2020. https://www.c-sharpcorner.com/article/creating-custom-random-dessert-skill-using-alexa-hosted-node-js/

89. Rudrawar K, Choudhar N, Meshram A. Voice assisted bots for automobile applications. In: Proceedings of the 3rd International Conference on Advanced Technologies for Societal Applications. Springer; 2021. p. 489-97.

90. Spectra Premium. Sensores de posición de árbol de levas y de cigüeñal. Spectra Premium; 2024. https://www.spectrapremium.com/es/aftermarket/north-america/camshaft-and-crankshaft-position-sensors

91. Vistrónica. Regulador LM1117 3.3V. Vistrónica; 2023. https://www.vistronica.com/componentes-activos/regulador-lm1117-3-3v-detail.html

FINANCING

None.

CONFLICT OF INTEREST

Authors declare that there is no conflict of interest.

AUTHORSHIP CONTRIBUTION

Conceptualization: Alexis Javier Villacis Ninasunta, Mariana Pinargote Basurto.
Data curation: Alexis Javier Villacis Ninasunta, Mariana Pinargote Basurto.
Formal analysis: Alexis Javier Villacis Ninasunta, Mariana Pinargote Basurto.
Drafting - original draft: Alexis Javier Villacis Ninasunta, Mariana Pinargote Basurto.
Writing - proofreading and editing: Alexis Javier Villacis Ninasunta, Mariana Pinargote Basurto.