

Article

Green Fleet: A Prototype Biogas and Hydrogen Refueling Management System for Private Fleet Stations

Antonio Martín-Márquez ¹, José Francisco Rangel-Serrano ^{1,*}, José Manuel Oyola-González ¹,
Adrian Talegón-Vázquez ¹, Mario Garrido ¹, Rodrigo García-Velayos ², Carla Patricia García-Heras ³,
David Bolonio ⁴ and Marcelo F. Ortega-Romero ⁴

- ¹ Aseproda Informática S.L. Paletina St. TEMPA S30 Building, 41007 Seville, Spain; amartin@aseproda.com (A.M.-M.); jmoyola@aseproda.com (J.M.O.-G.); atalegon@aseproda.com (A.T.-V.); mario.garrido@lafon.es (M.G.)
- ² Gas and Go Global Services S.L. (GAS&GO), Juan Caramuel Avenue 1, 28919 Leganés, Madrid, Spain; rodrigo.garcia@gasngo.es
- ³ Gasnam, CEXCO Building, Europa Avenue 10, 28821 Coslada, Madrid, Spain
- ⁴ Department of Energy and Fuels, Escuela Técnica Superior de Ingenieros de Minas y Energía, Universidad Politécnica de Madrid, Ríos Rosas 21, 28003 Madrid, Spain; david.bolonio@upm.es (D.B.); mf.ortega@upm.es (M.F.O.-R.)
- * Correspondence: jfrangel@aseproda.com

Abstract: Biogas and hydrogen (H₂) are breaking through as alternative energy sources in road transport, specifically for heavy-duty vehicles. Until a public network of service stations is deployed for such vehicles, the owners of large fleets will need to build and manage their own refueling facilities. Fleet refueling management and remote monitoring at these sites will become key business needs. This article describes the construction of a prototype system capable of solving those needs. During the design and development process of the prototype, the standard industry protocols involved in these installations have been considered, and the latest expertise in information technology systems has been applied. This prototype has been essential to determine the Strengths, Challenges, Opportunities and Risks (SCOR) of such a system, which is the first step of a more ambitious project. A second stage will involve setting up a pilot study and developing a commercial system that can be widely installed to provide a real solution for the industry.

Keywords: green hydrogen; biogas; heavy transport; fleet management; private refueling stations; facilities monitoring



Citation: Martín-Márquez, A.; Rangel-Serrano, J.F.; Oyola-González, J.M.; Talegón-Vázquez, A.; Garrido, M.; García-Velayos, R.; García-Heras, C.P.; Bolonio, D.; Ortega-Romero, M.F. Green Fleet: A Prototype Biogas and Hydrogen Refueling Management System for Private Fleet Stations. *Fuels* **2023**, *4*, 314–332. <https://doi.org/10.3390/fuels4030020>

Academic Editor: Evangelos Giakoumis

Received: 22 April 2023

Revised: 20 May 2023

Accepted: 17 July 2023

Published: 2 August 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Due to geopolitical strategics and the need to mitigate the impact of climate change, fossil fuels must be replaced. We must therefore find renewable energy sources for road mobility [1]. Electric Vehicles (EVs) seem to be the most relevant alternative to fossil fuels. But today, this technology is not suitable for all types of vehicles. Road freight by Heavy Goods Vehicles (HGVs), in particular those covering long distances, cannot clearly benefit from this technology [2].

Biogas and H₂ (see Figure 1 below) are emerging as promising energy sources for these vehicles [3]. However, developing a large public network to supply these renewable fuels will be a huge challenge that may take a long time to solve. In the meantime, fleet owners of such vehicles will need to build the required private infrastructure to refuel them. After some time, many of these private infrastructures can be converted into public facilities.

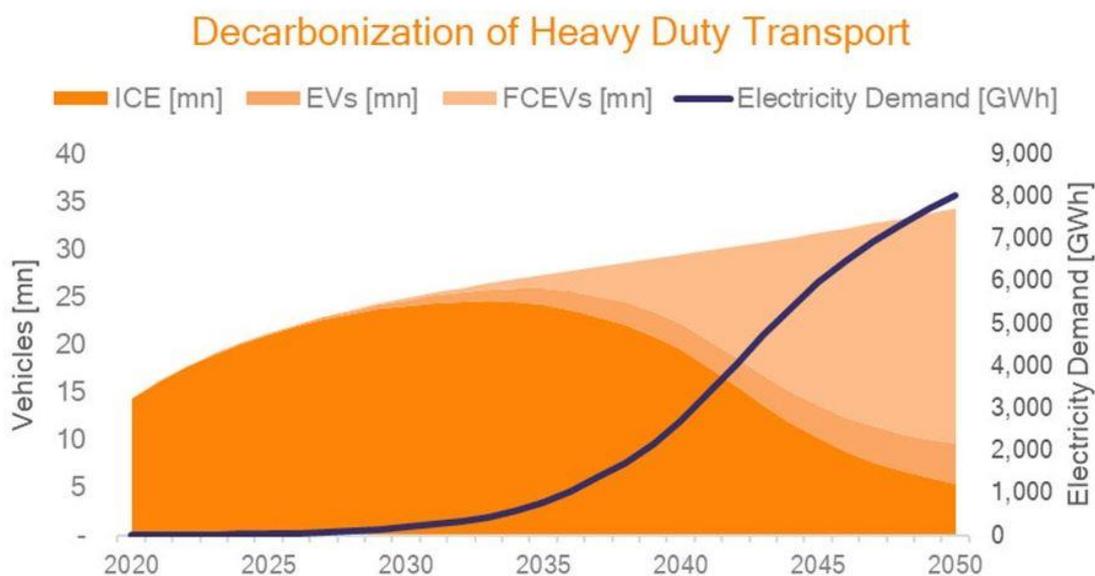


Figure 1. Hydrogen forecast to 2050. Source: <https://www.renewableenergymagazine.com/hydrogen/heavy-duty-transport-transition-will-rely-more-20220623> (accessed on 13 March 2023).

The biogas and hydrogen refueling sites sprouting across the EU territory are mostly owned by companies operating large HGV fleets. These refueling sites require adequate information systems to allow fleet refueling management and the remote monitoring of critical technical parameters [4]. Fleet management and stations monitoring will become important requirements within this sort of infrastructure, specifically for major companies with a large number of refilling sites distributed over a wide area. These carriers require information management systems tailored to their needs and peculiarities. Therefore, building a prototype to manage and control such facilities is an absolute necessity.

This paper deals with information systems engineering. It describes the prototype design process, from the requirements assessment through the design and implementation phases of hardware and software components to final validation. One of the main objectives of building a prototype is to analyze the SCOR for project improvement. In order to achieve this main objective, the two most significant specific goals of this research are as follows:

1. To show the needs and peculiarities derived from a completely new and innovative green fuel management system, which is not available in the market to date. This paper will also detail the main differences from existing information systems used to manage refueling stations for conventional Liquid Petroleum Products (LPPs), i.e., diesel and gasoline. Such systems are not suitable nor prepared for the complexity and diverse data managed by new green fuel technologies. Section 2 below lists arguments for not using the legacy technology still marketed and describes the main differences between conventional and biogas/hydrogen systems.
2. To describe the design and development process of an information system prototype offering solutions to these needs and peculiarities. The prototype will provide an opportunity to unveil the risks and improvement prospects associated with this type of system. Knowledge of devices commonly installed in these facilities—dispensers and sensors—is essential for this prototype. Their standard communication protocols also became a key issue. Current information systems architecture approaches were considered for this project as well. Concepts such as Internet of Things (IoT) or fog computing [5] are widely used in this prototype.

This project has been developed thanks to the partnership of the organizations Gasnam, Aseproda, and Gas&Go. Gasnam is a sustainable transport association which integrates the gas and hydrogen value chain in Spain. Aseproda and Gas&Go are two specialized companies that are members of this association. Gasnam provided organizational and management

expertise. Gas&Go is a company expert in the construction of this type of facility. It provided consulting and research services, including information on devices and protocols. Aseproda is a company focused on the design and development of hardware and software solutions for conventional fuel stations. It provided the design and development prototype features. The three organizations collaborated throughout the validation process.

2. Main Differences between LPP and Biogas/Hydrogen Private Refueling Facilities

Information systems for private fleet refueling management are not new. These systems have been in place for a long time to register diesel refueling. However, biogas and hydrogen sites show significant differences compared to LPP stations. The differences detailed below justify designing and implementing a completely new control and management system:

1. Biogas installations, and particularly H₂ sites, must be monitored more carefully and accurately. There are a large number of parameters to watch in order to ensure safe and productive facilities. Biogas and H₂ fuels are both gases under normal pressure and temperature conditions, which need to be compressed for storage in pressurized tanks or bottle racks. Additionally, hydrogen must be extremely cooled before delivery to any vehicle tank [6]. Pressure and temperature monitoring at these facilities are critical factors not required for diesel stations at all.
2. For most LPP facilities, refueling is measured through an analogue input from a square wave pulse generator linked to a liquid flow meter. This equipment is not available for current biogas or hydrogen dispensers. Thus, delivery data must be obtained from the dispenser itself through a protocol-based communication system.
3. The tank level system for measuring biogas or hydrogen stocks in pressurized containers is radically distinct from those used for LPP fuels and thus must be monitored differently [7].

These differences prevent LPP control systems from being suitable for biogas or hydrogen installations. There is another reason for developing new system architectures such as the one proposed in this paper. As mentioned above, the initial development of a biogas and hydrogen vehicle refueling network will largely be undertaken by fleet owners themselves. In the future, these facilities could become open-to-public dispensing stations, where third-party vehicles could be refueled. The proposed architecture, based on the IoT and fog computing, is different from those currently used in LPP-based systems. The architecture proposed in this paper can be easily upgraded to meet the requirements of public fuel stations.

Although it was not initially considered for analysis in this article, it would also seem reasonable to compare green fuel management and control systems against the existing ones for EV recharging stations. We have not considered such a comparison adequate because HGVs are not ideal for this powertrain technology. Furthermore, innovations applied to this prototype are completely different from the specific requirements used for an EV recharging station:

1. Green gas dispenser control differs completely from standard EV rechargers.
2. Monitoring systems for green gas sites are much more complex as data parameters are collected from very diverse control devices.
3. As a result, Single Board Computing (SBC) is the only available solution to manage this type of facility. SBC technology is not used for EV stations.
4. During the prototype phase, new requirements were discovered, especially related to green gas traceability management and legal compliance. Such requirements do not exist in the EV recharging market.

With these considerations in mind, it was decided to analyze the requirements in detail and develop a prototype system to meet such needs. The following sections outline the methods used to design the system and the results obtained.

3. Materials and Methods

The construction of the prototype required developing new hardware and software components. It was decided to follow suitable development methods for both cases. The Waterfall Model approach divided into sub-systems, originally developed by Dr. Winston W. Royce in 1970, was adopted [8].

The system design method progressed through the ordered sequence described below in Figure 2:

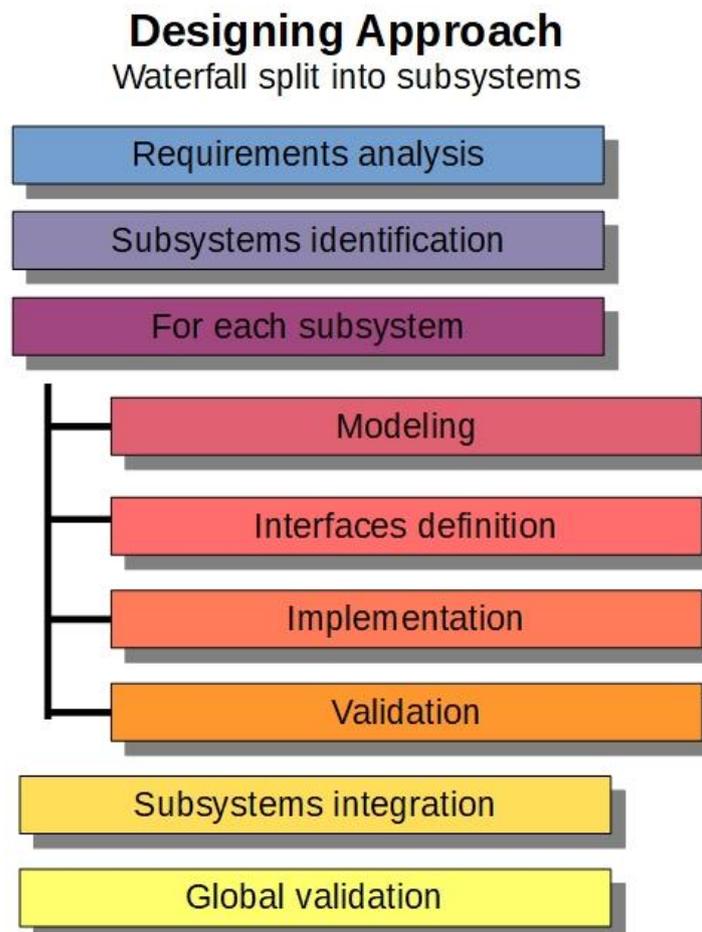


Figure 2. System design stages. Source: Martin, A. @ Aseproda, 2022.

A block representation method was applied to model the hardware. Standard components were used as much as possible. In some cases, designing and implementing certain hardware devices was required.

The classic 4 + 1 view method proposed by Philippe Kruchten [9] was not entirely used for the modelling software. The scenario view of this method was the only one employed. This view provides a unique perspective of the functional aspects to users, developers, and validation teams. The rest of the views were replaced by the C4 model proposed by Simon Brown [10]. This model shows four diagram types to depict software (context, containers, components, and classes). These diagrams were reinforced with the database design and the user interface prototype where deemed necessary.

The purpose of this article is to provide a system overview, intended as a solution to the demands of biogas or H₂ refueling stations. Therefore, certain design items which readers may need to replicate the prototype, such as component and class diagrams, will not be listed herein. Database design and user interface prototyping will also be omitted. Nevertheless, four interface screenshots will be displayed in the results section for illustrative purposes only.

3.1. Technologies

The technologies used in each of the identified sub-systems will be properly justified below. In any case, selecting a specific technology shall rely upon the following criteria [11]:

- Proven effectiveness in solving the needs for which they were created.
- The adequate balance between novelty and maturity. One of the designer's most demanding tasks is to select technologies that are innovative enough to add value to the project but mature enough to be stable and not end up as a fleeting trend.
- Widespread enough to find expert developers without too much difficulty.
- Inexpensive enough to keep the project and final product costs low. Most of the technologies used are publicly available or open source.
- Reputation: all technologies are commonly accepted by the developer's community and have been successfully used in large projects.
- Affinity with the activity sector: some of the selected technologies have a particularly relevant presence in the sector targeted by the project, as they solve specific issues.
- Alignment with the development team. Among the technologies meeting the above criteria, there is a tendency to select those where the design or the development team have a certain affinity or experience. This ensures greater stability of the final product and shorter development cycles.

3.2. Requirements

The list of requirements was drawn up based on the needs detected by experts for this type of facility (Gas&Go company) and on the experience provided by ASEPRODA with similar LPP fleet refilling stations. The following list details the main requirements decisively determining the system design:

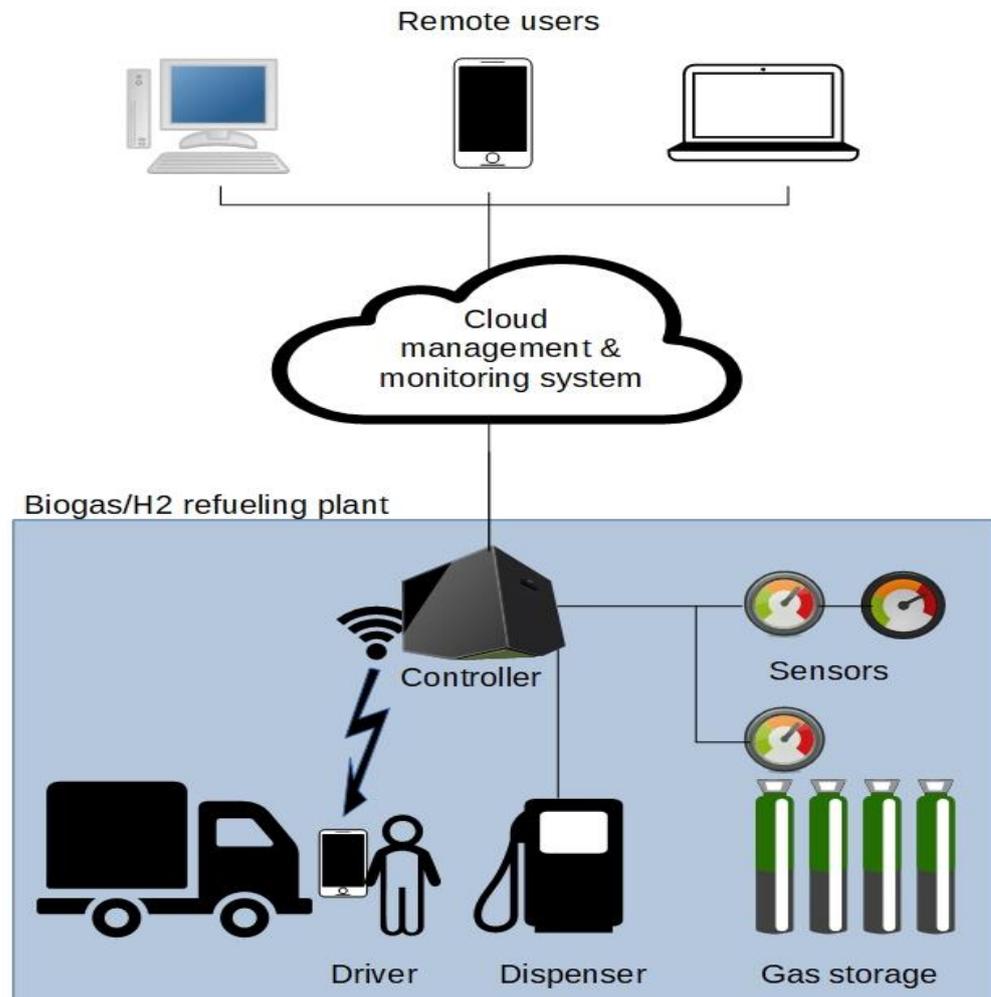
- Centralized control and monitoring of facilities and fleet yet remotely reachable from any location.
- Access to the refueling system by drivers, with no need for additional operator intervention during the refilling procedure.
- Identification of the driver and vehicle before refueling starts.
- Possibility to install more than one filling point at one single site and capability to use them simultaneously.
- Remote monitoring of technical parameters provided from site sensors.
- Use of industry standard protocols to communicate with different equipment—dispensers and sensors—in the refueling station.
- Fault tolerance. System downtime must be avoided in case of internet connection loss, server crash, or even any other device failure.

3.3. Sub-Systems and Their Interfaces

The requirements assessment has led to identifying the following sub-systems in order to achieve a global solution:

1. Hardware at each facility. This is needed to run the user interface and to communicate with gas dispensers during the refueling process. This hardware is also needed to run the software reading sensor values.
2. Software at each facility. This is software running on the above-mentioned hardware. It must serve as the user interface during the refueling process and read and synchronize sensor data with centralized cloud-based software. The hardware and software bundle installed on site is known as the "controller".
3. Software in the cloud. Centralized software allowing remote access. Its mission is to maintain the fleet's data, record its refueling, and obtain useful information from all of this. It must also register sensor values read by controllers from any site and offer tools for their correct monitoring. Finally, it must offer the functionality to synchronize data (fleet, refueling, and technical parameters) with the company's controller network.

Figure 3 below displays a typical system scheme:



Icons from publicdomainvectors.org Lic. Creative Commons Deed CC0

Figure 3. Context diagram of the complete system. Source: Martin, A. @ Aseproda, 2022.

Overall requirements and layout are suitable to apply certain current approaches to the system architecture:

- Internet of Things (IoT). The gas facilities' controllers should be as lightweight as possible. All of them working together and linked to their central system meet the IoT philosophy.
- Fog computing. This approach is a design philosophy derived from "Cloud Computing". It establishes that data used and generated by a system must be stored as physically close to the site as feasible. This philosophy is not always applicable and often leads to complications in software design, but it makes systems faster and more robust. One of the features that any system must meet is precisely its robustness. This system is especially critical, as noted in the requirements list. Should the controllers stop working, vehicles will not be capable of refilling and facilities will no longer be monitored. Site operations must not be dependent on connectivity to a central office, as facilities are often located in areas where internet coverage is poor. Hence, a fog computing approach is proposed. This concept does not need an online connection for its operation. The information needed for the controllers to operate is distributed throughout its own network via a synchronization process which keeps them updated when communication with the cloud is available.

- Redundancy. Given the critical nature of the system, its design must consider the option of having more than one controller at each refueling site. Thus, if one fails, another controller can take over.

3.4. Hardware Sub-System

The only hardware required by this project is the one installed at each refilling station. This sub-system must meet the following conditions:

- It must be able to interact with users (HGV drivers) to complete the refilling procedure.
- It must be light and robust, as it will often be installed outdoors.
- It must be able to communicate with on-site devices (gas dispensers and sensors).
- It must be able to maintain facilities in operation even if there is no connectivity to headquarters.

Among the different technologies available, Single Board Computing (SBC) was selected [12]. SBC is a type of computer built on a single circuit board, with a microprocessor, memory, input/output, and other features required of a functional computer. SBC was chosen because it has several advantages, such as the following:

- It reduces costs by decreasing the number of circuit boards required and eliminating the connectors and bus controller circuits that would otherwise be used.
- It saves space by having a compact and lightweight design which can be easily integrated into other systems or products.
- It facilitates the IoT (Internet of Things) by allowing various devices and sensors to be connected through standard or custom interfaces.
- It offers flexibility by being able to choose between different models and configurations of SBCs according to the needs of the project.

SBC also has certain limitations to be aware of, such as the following:

- Higher cost by requiring more engineering and design than ordinary computers.
- Not easily upgradeable because it is built as a fixed structure that does not allow adding or changing components.
- Intimidating when using Linux and non-removable hardware which can be difficult for inexperienced users to manage.

Technology based on fog computing demands storing and managing large and complex data structures, while Programmable Logic Controllers, known as PLCs [13], or micro-controllers are not powerful enough to meet such requirements. The latter also offers limited user interface capabilities. On the other hand, PC-based technology is inadequate for industrial environments, more complex, and expensive to support.

Cost and simplicity lead to using off-the-shelf components wherever possible. However, an additional device is needed to prevent the controller from improperly shutting down in the event of a power failure. There are no solutions on the market to properly solve this problem, so it proved imperative to develop a specific electronic circuit for this task. This new device, along with the metal cabinet housing all the controller devices, is the only hardware which required a custom design for this project.

One of the design decisions made was avoiding the user interacting directly with the hardware to perform the refilling procedure. This is to prevent outdoor-mounted screens, displays, readers, or keyboards. Using these components on the outside would make the system more fragile and expensive. The human-machine interface has been solved by using the driver's own mobile phone. Drivers would connect to a Wi-Fi access point provided by the controller itself and perform the necessary actions using the web browser on their smartphones. This way, users avoid installing Android or iOS applications on their devices. Hence, it is no longer essential to develop and update apps for either system every time a new version is deployed.

Figure 4 below describes the controller's hardware diagram:

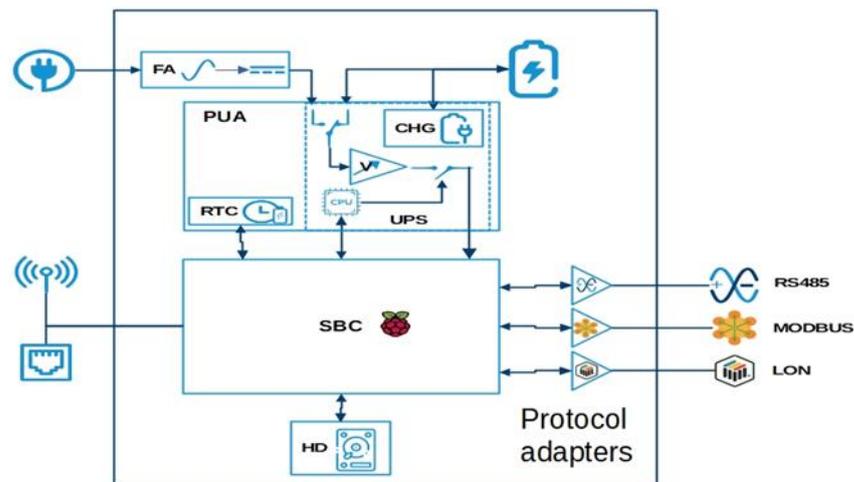


Figure 4. Block diagram of the controller's hardware. Source: Martin, A. @ Aseproda, 2022.

3.5. Controller Embedded Software

As mentioned above, the embedded software must meet the following main functional requirements:

- Provide a user interface for the refueling procedure.
- Communicate with on-site devices—biogas or H₂ dispenser and sensors—to control the refueling process and read parameter values from such devices.
- Keep the controller running (thus refueling activities) even without an internet connection to the monitoring office.

And the following non-functional requirements must be met:

- The driver's mobile interface for the refueling procedure must be easily accessible and user-friendly.
- It must be able to communicate with gas dispensers and the array of sensors on site, using the most common protocols in the industry. Therefore, the most popular dispenser protocol, IFSF-LON [14], and one of the most commonly used protocols for measurement devices and sensors, JSON over TCP/IP [15], have been selected for the prototype. However, software architecture must be flexible enough to incorporate more protocols in the future, thus supporting a variety of managed devices using different protocols at a specific location.

Regarding technology, it was chosen according to the criteria set out in Section 3 as follows:

- Linux operating system, Debian 10 distribution. Linux OS is suitable for running on an SBC platform. The Debian 10 distribution is lightweight and known for its robustness [16].
- C#/Net core framework as back-end programming language. It is a widespread and robust development tool, which has proven its usefulness for this type of project. The core version is used to ensure that the framework remains lightweight—it runs on an SBC device—and compatible with Linux OS [17].
- Maria DB database. Required since the device must be capable of storing substantial amounts of data by applying a "fog computing" approach. Maria DB is powerful enough for this purpose, and it is an extremely lightweight database server [18].
- NGINX web server. Essential to provide web pages comprising the user's refueling interface. NGINX is a well-known and dependable lightweight server which is ideal to run on an SBC device [19].
- HTML, CSS, and JavaScript as front-end languages. Used to build the web user interface. All of them are widespread, robust, and also lightweight enough to run on users' mobile devices [20].

- HTTP protocol and JSON data format to synchronize information with the cloud-based host server. JSON is remarkably simple and lightweight, ideal for communicating messages over the internet [21].

Figure 5 displayed below illustrates the containers diagram of the software running on the controller. It details the main software modules, as well as the technologies and protocols used by each of them.

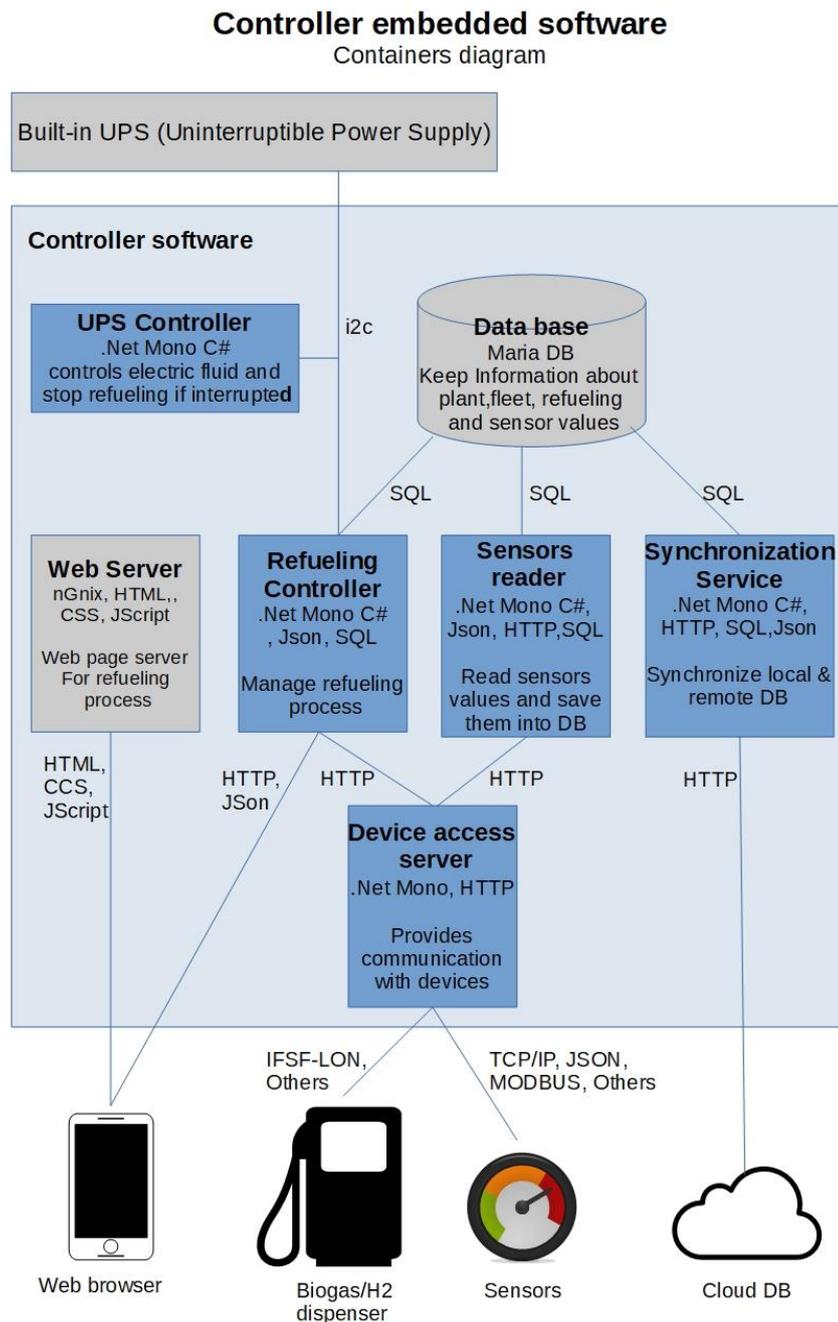


Figure 5. Diagram of containers for the controller’s software. Source: Martin, A. @ Aseproda, 2022.

Items outside the box do not belong to the software. They represent devices the controller has to communicate with (built-in UPS, driver’s mobile browser, dispensers, sensors, and cloud database).

Each container is labelled in bold font. Just below each container’s name, readers will find the technology used to build it and a brief description of its main functionality. Lines

connecting different items represent a communication link between them. Protocols used for each link are also indicated on these lines. The grey containers depict standard software containers used in the project (NGINX web server and Maria DB).

3.6. Centralized Cloud Software

As above, functional and non-functional requirements are listed below.

The main functional requirements are as follows:

- The fleet refueling management and site monitoring software must be accessible remotely and simultaneously by an undetermined number of users.
- Provide access to property configuration of each facility (fuel types, storage tanks or racks, controllers, dispensers, sensors, etc.) and fleet assets (vehicles and drivers).
- Allow access to site refueling operations with the capability to retrieve aggregated data from them.
- Display and monitor readings provided by sensors at each station.
- Control on-site fuel wet stock, including input management from suppliers or self-generated production.
- Regular communication with site controllers to keep fleet data and refueling and sensor values synchronized. Once per minute seems a reasonable frequency for this task.

The main non-functional requirements include the following:

- Simple and intuitive interface for users.
- Security against non-authorized access. Simplicity and maintenance reasons make it desirable to run one application with a single database for all companies using the system. Moreover, it must ensure that users can only access data pertaining to their company.
- Scalability. The system shall ensure efficiency even when managing large numbers of vehicles, refueling plants, sensors, and/or users.
- Regional customization. Selecting measurement units (metric or imperial) and language.
- Responsiveness. It must be designed in such a way that its interface meets browsers' resolution and appearance on a computer, mobile phone, or tablet.

From the above requirements, it will undoubtedly be a web application. The technologies used for this sub-system are widely used to develop web applications.

They have been selected based on the criteria set above in Section 3.1 as follows:

- Web server. NGINX or Apache will be used as they are robust, scalable, and well-known servers.
- C#/Net 5 as back-end programming language and framework. This is an extended and robust tool, which has proven useful for this type of project. Version 5 is used to ensure that the framework is compatible with Linux and Windows operating systems [22].
- Protocols and data format of the web API used to synchronize with controllers: HTTP and JSON. These are lightweight and powerful protocols suitable for transferring data over the internet.
- HTML, CSS, and JavaScript as front-end languages. Used to build the web user interface. All of them are widespread and robust. They are also light enough to run on any device (PC, tablet, or mobile phone).
- PostgreSQL database server. It is a widely used open-source database, robust, powerful, and easy to scale [23].
- JasperReports engine. It is essential for easy presentation of aggregated data. It is a well-known, free, and robust report generator, especially designed for use in web applications [24].
- HOST operating system: as a result of implementing the above technologies, this application could be installed on multiple platforms.

Figure 6 below details the aforementioned software containers:

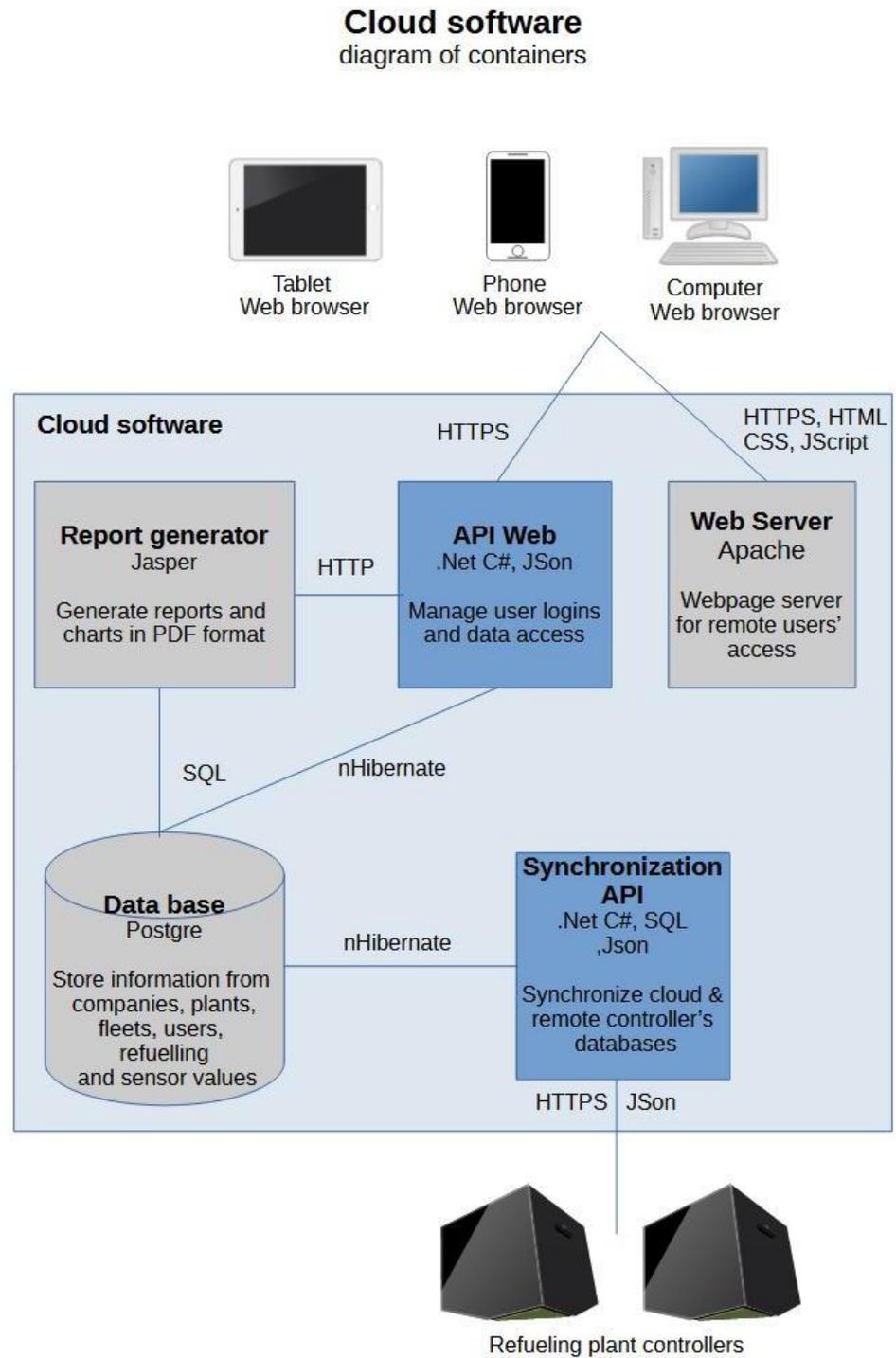


Figure 6. Diagram of containers for Cloud Software. Source: Martin, A. @ Aseproda, 2022.

The figure above is the best representation for high-level software architecture. It details the main software modules, as well as technologies and protocols used by each of them. Items outside the box do not belong to software. They represent items the software has to communicate with (remote user browser and site controllers).

Each container is labelled in bold font. Just below each container's name, readers will find the technology used to build it and a brief description of its main functionality. Lines connecting different items represent communication links between them. Protocols used for each link are also indicated on these lines. The grey containers depict standard software containers used in the project (Apache web server, JasperReport generator, and PostgreSQL).

4. Results

For each of the sub-systems described in Section 3, a series of validation tests were established to ensure that all of them meet the original requirements. The results obtained for each of these sub-systems are shown next.

4.1. Controller's Hardware

The appearance of the controller prototype is shown in Figures 7 and 8. It has been named the "Black Box" as it has no devices to interact with:

Controller's hardware

Outside look



1. Lock
2. Ventilation grill
3. Power supply
4. Wi-Fi antenna 1
5. Wi-Fi antenna 2

Figure 7. External view of the controller. Source: Martin, A. @ Aseproda, 2022.

Controller's hardware

Inside look



1. 5v power supply
2. Built-in Uninterruptible Power Supply (UPS)
3. SBC CPU. Only connectors can be seen as UPS is placed on top of it
4. USB Hub
5. SATA-USB Driver
6. SSD Hard disk
7. UPS battery
8. Wi-Fi 1
9. USB-LON adapter
10. Wi-Fi 2
11. Ethernet RJ45 connector
12. Terminal blocks
13. Fans

Figure 8. Internal view of the controller.

The hardware was evaluated in most cases without any need for embedded software, as described under Section 3.4, except for test # 2 which partially required it as follows:

1. System start-up. The controller does not have an on-off switch. The only way to turn it on was by plugging it into an external power supply (230 V AC). Then, the controller booted up successfully. The Linux's shell operating system appeared ready and all components (HD drive, Wi-Fi, USB-LON device, Ethernet, and i2c interfaces) were up and running.
2. System shutdown. As mentioned above, the only way to turn the system off is by disconnecting it from the power supply. However, this process is a bit more complex than powering up. The controller must shut down in an orderly manner. The custom-designed electrical circuit detects the power outage and uses its internal battery to keep the controller running for a certain period of time. A purpose-built software process detects this circumstance, triggering a command sequence to terminate any active refueling, saving information in the database, closing down the operating system, and cutting off power to the CPU and all other devices.
3. Temperature test. All hardware components have been selected to meet the extended range of ambient temperature requirements ($-20\text{ }^{\circ}\text{C}/+60\text{ }^{\circ}\text{C}$). This range is sufficient for equipment installed outdoors. Other than the CPU, no components require additional cooling systems. A fan has been included in the design to prevent the CPU from reaching temperatures exceeding its operating range ($+85\text{ }^{\circ}\text{C}$). During the prototype validation phase, a heat chamber was used to raise the ambient temperature to $60\text{ }^{\circ}\text{C}$, while the CPU did not exceed $70\text{ }^{\circ}\text{C}$ running highly demanding tasks. The certification of the prototype has not been considered. Nevertheless, any marketable equipment must be certified.
4. Dust- and waterproof. Equipment installed outdoors must meet at least an IP54 protection rating. The metal enclosure housing all the devices was designed to meet this requirement. No testing for this protection rating was carried out apart from installing the prototype outdoors for a one-month period. Prototype certification was not considered advisable, as the unit may still be subject to modifications before reaching a final commercial stage. As stated above, the marketable product must be certified to meet this requirement.
5. Electromagnetic compatibility. Although this requirement is not explicitly stated, all electronic devices manufactured in the European Union must undergo electromagnetic compatibility compliance tests to achieve a CE Marking. This obligation was considered during the design process, but it was not deemed appropriate to certify the controller at the prototype stage. The commercial product derived from it shall be certified to all applicable regulations.

4.2. Controller's Embedded Software

The controller software underwent a detailed battery of tests defined in the design phase. Specific tests were designed for each of the software modules that run on the controller: refueling, sensor reading, and cloud synchronization processes. All conducted tests were aimed at fulfilling the initial functional and non-functional requirements. Among the latter, special emphasis was placed on executing stress and performance tests.

The refueling process, which requires the driver's mobile phone to be carried out, was tested on different Android and iOS smartphone models. Refueling validation tests were performed using a real dispenser CPU with an IFSF-LON protocol. Sensor tests were also conducted using temperature sensors with a JSON data format over TCP/IP.

Figures 9–12 below illustrate four steps of the refueling interface:



Figure 9. Vehicle identification and odometer input.

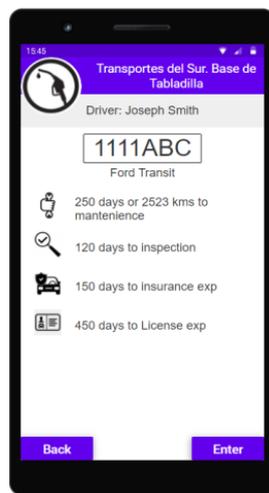


Figure 10. Vehicle and driver alerts.

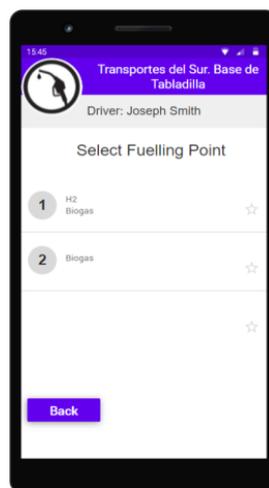


Figure 11. Fueling point and grade selection.

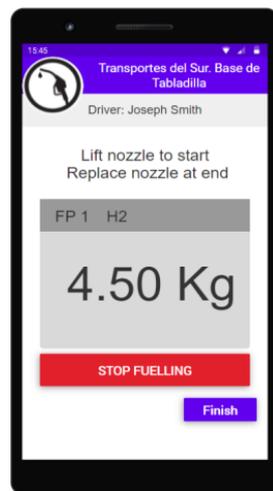


Figure 12. Fueling progress.

4.3. Cloud Software

Cloud-hosted software was subject to a battery of tests designed during the system modelling phase. Performance and security tests were also conducted on the refueling management, monitoring, and synchronization modules. The system was installed on a cloud server with a Linux operating system and an Apache web server.

Figures 13–16 below illustrate four user interface screenshots:

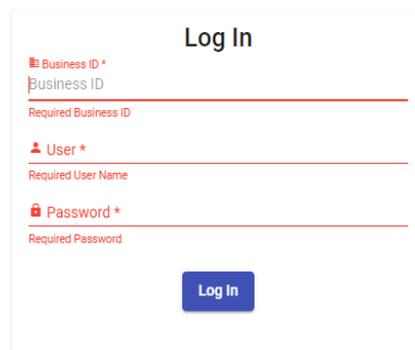


Figure 13. Login screen.

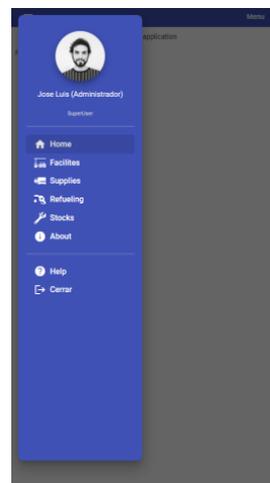


Figure 14. Menu in phone's portrait format.

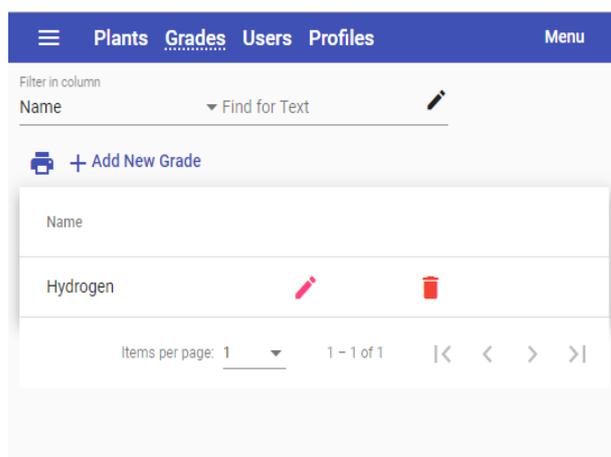


Figure 15. Company's grades definition.

Name	Actual value
Tank 1 Capacity	12851 L
Tank 2 Capacity	968 L
Tank 3 Capacity	4758 L
Pressure Pump 1 (In)	790 bar
Pressure Pump 1 (Out)	763 bar
Temperature Pump 1	-60.18 °C
Pressure Pump 2 (In)	751 bar
Pressure Pump 2 (Out)	680 bar
Temperature Pump 2	-63.68 °C

Figure 16. Sensor values.

5. Discussion

As explained in this article's introduction, one of the main objectives of building a prototype is to analyze the SCOR for project improvement. In this section, we will discuss each of these aspects, dealing separately with each of the sub-systems comprising the prototype.

5.1. Hardware

Certain opportunities for hardware improvement have been identified. The most obvious one is the convenience of reducing the cabinet dimensions housing the components. Not all hardware devices had yet been selected or engineered when the enclosure was conceived. It was designed to be large enough to fit bigger components than those actually used.

Another prominent issue to bear in mind is considering alternative communication protocols to those used in the prototype. Other dispensers or sensors may implement different physical-level protocols to the prototype's LON or Ethernet. For example, RS485 has been identified as a widespread industrial protocol [25]. The SBC's built-in USB ports are sufficient to accommodate future interfaces, as USB-RS485 standard adapters are available.

Finally, as mentioned above, the device's marketable version will require certification for electromagnetic compatibility and outdoors installation (IP protection class and extended temperature range approval). ATEX certification [26] will not be required, as the controller should be installed away from hazardous zones generated by gas dispensers or other refueling equipment.

5.2. Controller's Software

The prototype has shown the suitability of the system designed for the refueling process. Drivers can carry it out by themselves, using their mobile phone without having to download any apps.

However, the need for a faster and more reliable vehicle identification method has been recognized. With the prototype, the driver must enter the vehicle's registration number or license plate to identify it. A viable alternative system by reading a QR code affixed to the vehicle's windscreen has been discussed. This workaround has the additional advantage of automatically starting the refueling process by simply reading the QR code.

It should also be noted that more communication protocols for dispensers and sensors must be implemented in the future. Although two of the most common standard protocols in this industry have been developed, other protocols with a significant presence were also identified. Most likely, MODBUS over TCP/IP will be required soon [27]. This protocol is widely used by a large number of commercial sensors for PLC-based solutions. Luckily, the protocol management module included in the software has been designed to allow the seamless integration of new protocols.

Another detected opportunity for improvement is recording video during the refilling process. This feature can be implemented effortlessly thanks to SBC's capabilities. Its incorporation would enable vigilant monitoring of the refueling facilities, ensuring their proper and efficient usage.

Finally, establishing an automatic software update procedure for controller devices has been discussed. This would be useful to quickly deploy new functionalities and fix software bugs.

5.3. Management Software

Regarding the cloud management system, opportunities for improvement have also been recognized which could be added to future system versions.

One of them has to do with traceability management and Green Origin certificates for renewable fuels [28]. It would be useful to enter such information into the system for the product delivered to the station. This would enable road transport carriers using green energy sources to control the traceability of the consumed fuel.

Another need detected relates to improving the sensors' monitoring interface. For future system versions, it would be convenient to develop a dashboard where monitored sensor values and status can be shown in a clearer and more attractive way. The convenience of implementing an alert system to warn system administrators, by email or SMS, when any monitored sensor falls out of its normal working range has also been discussed.

Finally, the convenience of monitoring the controller's own parameters, such as CPU temperature, memory usage, etc., has been considered. Monitoring these parameters could be helpful for preventive maintenance.

6. Conclusions

The development of a management and monitoring system for biogas and hydrogen refueling in private fleet facilities is a key requirement for the deployment and expansion of these energies. It is essential to provide information systems which respond to the specific needs of this industry. Specifically, green gas fleets and facilities' investments require intensive optimization for business operations to ensure profitability. New tools and resources developed within this innovative management system shall contribute to a highly precise control of costs per vehicle and site maintenance.

The development of a prototype is a necessary step towards a marketable product. Further, the design and construction of this system has reached the objectives expected from any prototype:

1. It is viable and fully meets the demands for green refueling facilities.
2. New challenges and opportunities for its improvement were detected.

The work team has already set its goal for the next step, which is to set up a pilot project in an actual refueling site. This new project shall evaluate the system's efficiency and robustness to complete its fine-tuning in the field.

Author Contributions: Conceptualization, A.M.-M.; methodology J.F.R.-S.; head of software: J.F.R.-S.; head of hardware: J.M.O.-G. and A.T.-V.; requirements analysis: R.G.-V. and J.F.R.-S.; validation: C.P.G.-H., R.G.-V., J.M.O.-G. and J.F.R.-S.; writing—original draft preparation: A.M.-M. and J.F.R.-S.; writing—review: D.B., M.F.O.-R. and M.G.; visualization: A.M.-M., J.F.R.-S. and A.T.-V.; supervision: C.P.G.-H.; project administration: C.P.G.-H.; funding acquisition: C.P.G.-H. All authors have read and agreed to the published version of the manuscript.

Funding: This project was funded by the Spanish Ministry of Industry, Commerce and Tourism, with Next Generation funds from the European Union, within the framework of the Support Program for Innovative Business Groups, Grant no. AEI-010500-2021b-110.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: The authors thank Maria Teresa Ariza Gomez, Professor of Telematics at the University of Seville, for her advice in drafting this article and Jose Luis Jimenez Carranza for his assistance in seeking the devices and simulators needed to conduct validation tests.

Conflicts of Interest: The authors declare no conflict of interest of any kind.

References

1. Decarbonising Road Transport—The Role of Vehicles, Fuels and Transport Demand. Available online: <https://www.eea.europa.eu/publications/transport-and-environment-report-2021/download> (accessed on 13 March 2023).
2. Will Electric Trucks Be in It for the Long Haul? Available online: <https://www.commercialfleet.org/fleet-management/will-electric-trucks-be-in-it-for-the-long-haul> (accessed on 13 March 2023).
3. Europe's Truck Giants to Ditch Diesel, as Hydrogen's Benefits Come to Fore. Available online: <https://www.euractiv.com/section/energy/news/europes-truck-giants-to-ditch-diesel-as-hydrogens-benefits-come-to-fore/> (accessed on 14 March 2023).
4. Road Freight Decarbonization in Europe. Readiness of the European Fleets for Zero Emission Trucking. Available online: <https://theicct.org/wp-content/uploads/2022/09/road-freight-decarbonization-europe-sep22.pdf> (accessed on 14 March 2023).
5. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S.K. Fog computing and its role in the internet of things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing; ACM SIGCOMM 2012 Conference, Helsinki, Finland, 17 August 2012; pp. 13–16.
6. How a Hydrogen Filling Station Works. Available online: <https://www.roadandtrack.com/car-culture/g6917/how-hydrogen-filling-stations-work/> (accessed on 15 March 2023).
7. European Train the Trainer Programme for Responders. Lecture 12: Hydrogen Refueling Stations & Infrastructure. Chapter 5: Safety Features in HRS and Other Infrastructures; pp. 35–37. Available online: https://hyresponder.eu/wp-content/uploads/2021/06/L12_HyResponder_L4_210622.pdf (accessed on 15 March 2023).
8. Royce, W.W. Managing the Development of Large Software Systems. 1970. Available online: <https://blog.jbrains.ca/assets/articles/royce1970.pdf> (accessed on 16 March 2023).
9. Kruchten, P.B. The 4 + 1 View Model of Architecture. *IEEE Softw.* **1995**, *12*, 42–50. [CrossRef]
10. The C4 Model for Visualising Software Architecture. Available online: <https://c4model.com/> (accessed on 16 March 2023).
11. Bacon, C.J. The Use of decision criteria in selecting information systems/technology investments. *MIS Q.* **1992**, *1*, 335–353. [CrossRef]
12. Isikdag, U. *Internet of Things: Single-Board Computers. Enhanced Building Information Models: Using IoT Services and Integration Patterns*; Springer: New York, NY, USA, 2015; pp. 43–53, ISBN 978-3-319-21824-3.
13. Bolton, W. *Programmable Logic Controllers*; Newnes: Oxford, UK, 2015; ISBN 9780081003534.
14. IFSF—International Forecourt Standards Forum. Available online: <https://ifsf.org/> (accessed on 17 March 2023).
15. Nurseitov, N.; Paulson, M.; Reynolds, R.; Izurieta, C. Comparison of JSON and XML data interchange formats: A case study. *Caine* **2009**, *9*, 157–162.
16. Debian—The Universal Operating System. Available online: <https://www.debian.org/index.en.html> (accessed on 17 March 2023).
17. C# Documentation—Get Started, Tutorials, Reference. Microsoft Learn. Available online: <https://learn.microsoft.com/en-us/dotnet/csharp/> (accessed on 20 March 2023).
18. MariaDB Server: The Open Source Relational Database. Available online: <https://mariadb.org/> (accessed on 20 March 2023).
19. NGINX—Advanced Load Balancer, Web Server & Reverse Proxy. Available online: <https://www.nginx.com/> (accessed on 20 March 2023).

20. Dinh, D.; Wang, Z. Modern Front-End Web Development: How Libraries and Frameworks Transform Everything. Bachelor's Thesis, Turku University of Applied Sciences, Turku, Finland, 2020. Available online: <https://www.theseus.fi/bitstream/handle/10024/342325/Thesis-Duong-Wang.pdf> (accessed on 20 March 2023).
21. Bray, T. The JavaScript Object Notation (JSON) Data Interchange Format. RFC 8259, Last Updated December 2017. Available online: <https://datatracker.ietf.org/doc/rfc8259/> (accessed on 21 March 2023).
22. Price, M.J. *C# 9 and NET 5—Modern Cross-Platform Development: Build Intelligent Apps, Websites, and Services with Blazor, ASP.NET Core, and Entity Framework Core Using Visual Studio Code*; Packt Publishing Ltd.: Birmingham, UK, 2020.
23. Juba, S.; Vannahme, A.; Volkov, A. *Learning PostgreSQL*; Packt Publishing Ltd.: Birmingham, UK, 2015; pp. 38–43.
24. TIBCO JasperSoft—Reporting and Embedded Business Intelligence Software. Available online: <https://www.jaspersoft.com/> (accessed on 21 March 2023).
25. The RS-485 Design Guide (Rev. D). Available online: <https://www.ti.com/lit/pdf/slla272?keyMatch=RS-485> (accessed on 21 March 2023).
26. Directive 2014/34/EU of the European Parliament and of the Council of 26 February 2014. Available online: <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32014L0034> (accessed on 20 March 2023).
27. Swales, A. Open modbus/tcp specification. *Schneider Electr.* **1999**, *29*, 3–19.
28. EECS—AIB. Available online: <https://www.aib-net.org/eecs> (accessed on 22 March 2023).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.