Requirement Analysis and HW/SW Modelling for an Ethanol-based SOFC Systems

Vinícius Slovinski, Eduardo Forster Beathalter, Anderson Wedderhoff Spengler, Giovani Gracioli and Rafael Camargo Catapan
Software/Hardware Integration Lab (LISHA)
Laboratory of Applied Catalysis and Combustion (LAC)
Universidade Federal de Santa Catarina (UFSC)

ABSTRACT

The increasing demand for decarbonization in the transportation sector calls for innovation in the current powertrains, including new propulsion strategies. Bioethanol is a well-known alternative fuel, since its carbon cycle is approximately neutral. However, its use in internal combustion engines still decreases the vehicle's thermal efficiency. The use of fuel cells such as Solid Oxide Fuel Cell (SOFC) integrated with an ethanol reformer is a promising path for new and efficient powertrains. In order to achieve the maximum ethanol conversion and selectivity towards hydrogen, it is fundamental to develop an autonomous system to perform the whole process. Thus, Hardware and Software integration to sense, control, and actuate the system is required.

This paper presents the functional and nonfunctional requirements, hardware specification, and software modeling for an ethanol-based reformer, following a well-defined mechanical diagram, as the hydrogen processes complexity and available documentation are limiting factors. The work is being carried out within a consortium with the Federal University of Santa Catarina, BMW, and AVL in the Rota 2030 program.

INTRODUCTION

Concerns about greenhouse gasses emissions have been increasing in modern days. Recent studies documenting the reduction of the discovery of new petroleum sources in association with the rising global energy demand demonstrate an urgent need for low carbon technologies. In this context, hydrogen presents itself as a promising alternative for the future of mechanized transportation [1][2].

Currently, in the automotive industry, there are a few examples of cars using hydrogen as fuel. It certainly represents an important innovation, given that our world is still dominated by internal combustion engines. However, problems such as hydrogen storage and the lack of appropriate fuel stations still undermine its potential. Despite the previously mentioned issues, this breakthrough opened doors for the research of alternative solutions [3]. One of the aforementioned alternatives is the production in situ of hydrogen through a fuel reformer. This alternative eliminates the need for specialized refueling infrastructure that is usually associated with hydrogen [4].

In this context, the Federal University of Santa Catarina, together with BMW and AVL, started in 2021 a research project within the Rota 2030 program to produce the equivalent of 5 kW of electrical power in hydrogen. This goal will be accomplished using a reactor that employs a catalytic, autothermal reaction to convert water, ethanol, and air into hydrogen [5].

Such a process requires a strictly controlled environment to ensure stability and maximum efficiency. This corresponds to the reactants flow rate, temperature, and pressure control. Furthermore, the system should be able to handle a wide range of inputs and ambient conditions, allowing flexibility for the tests that will be conducted.

Considering the purposes of automation and effectiveness, this paper aims to describe the development progress of an autonomous ethanol reformer system, focusing on fuel and air injection functionalities. Moreover, every project step is discussed, from requirements and analysis to electronic circuit design and software modeling.

PROJECT DESCRIPTION

A system of such complexity involves numerous areas of knowledge, thus a research team divided into cells has been working on the project. The chemistry cell is tasked with the research of the catalyst composition that will result in the best hydrogen conversion and selectivity rates possible. The mechanical cell focuses on the development of the reformer unit and the design of a testbench, ensuring that the other cells have an adequate platform to validate their results on. Finally, the automation cell has the objective of integrating the hardware and software, accomplishing the automation of the testbench and guaranteeing compliance with the other cells' requirements.

Therefore, much of the work presented in this article comes from the coordination and alignment with other cells. The following section describes the major guideline for the automation process: the mechanical diagram. Using this diagram, it is possible to understand the workflow and visualize all the sensors and actuators that will be necessary.

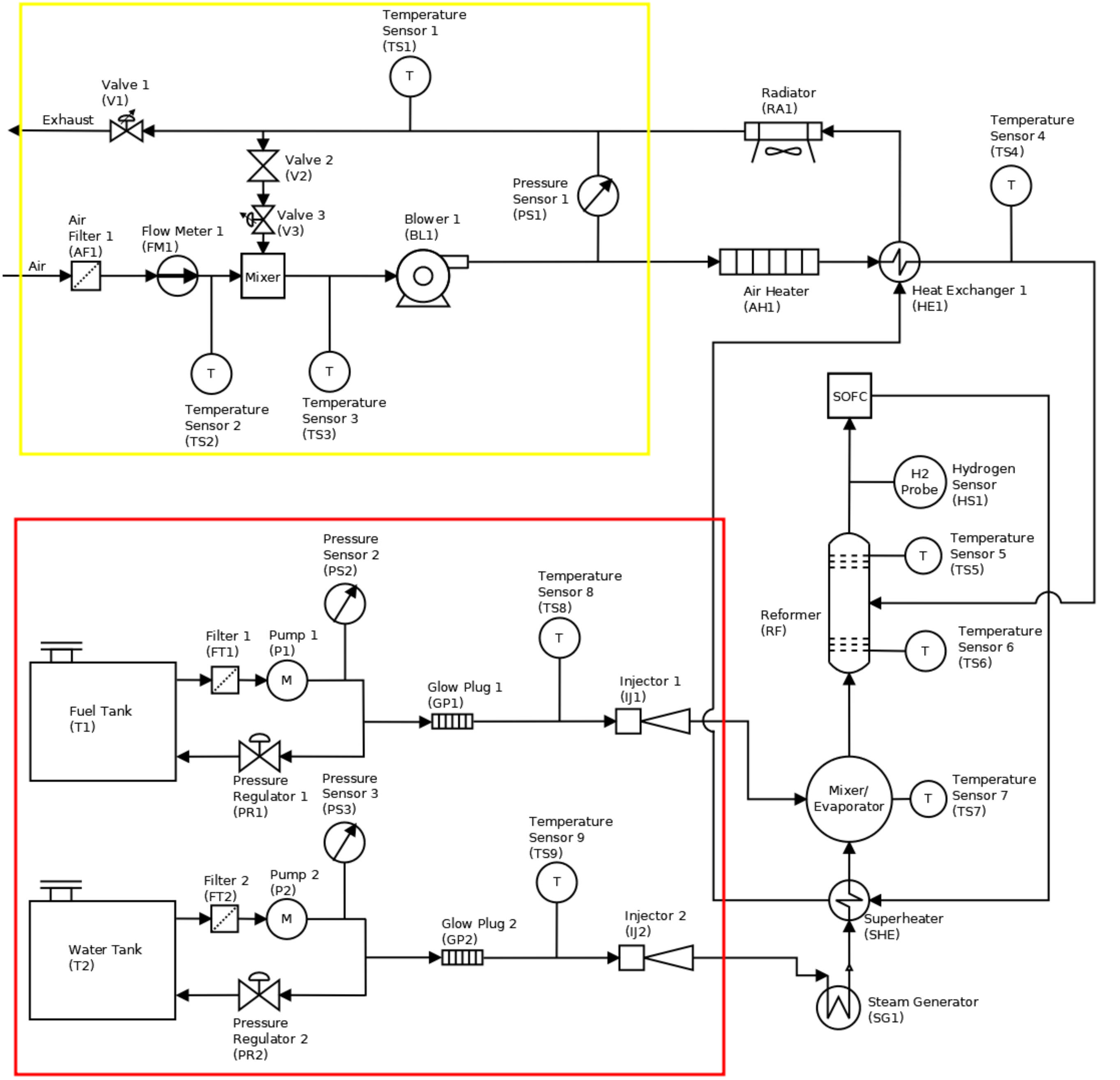

Figure 1. Complete Mechanical Diagram with subsystem divisions..

MECHANICAL DIAGRAM - Considering that the reactor unit will be developed from scratch, no direct references to automate the proposed powertrain will be used, other than the mechanical diagram. For this reason, the strategy employed to reduce the complexity was by sectioning it into subsystems. Figure 1 presents the whole diagram and its divisions.

Inside the red rectangle lies the Fuel Subsystem. It is responsible for supplying the reformer with the right mixture of ethanol and water. The yellow region contains the Air Subsystem. Its duty is to provide the amount of air required by the reaction. Additionally, this subsystem controls the recirculation of the exhaust gasses. All reformer inlets have specific temperature and pressure parameters associated, which will be discussed in the following sections.

The remaining elements of the diagram compose preheating and Reformer-SOFC subsystems. However, those subsystems are out of the scope of this paper, and thus will not be discussed in detail. Despite this, they might be referenced in the electronic design and software modeling, since these tasks must consider their functionalities.

The diagram is the main tool used to understand all the necessary control loops, working steps and how the subsystems interconnect. As a result, the acquisition of hardware and software requirements becomes possible. Such requirements will guide component specification and software modeling processes.

PROJECT REQUIREMENTS

Defining the functional and non-functional requirements is a fundamental step in a project's development, as it is the foundation of every other task and serves as a guide to the developer. This makes it easier to detect any differences in relation to the original concept, as poor documentation could open margin for conceptual errors [6].

The requirements engineering process involves elicitation of the cell members' needs and goals, as well as the analysis and specification, followed by the validation of the requirements. Group interviews was the method employed for this process. Notes were taken documenting the topics discussed, ensuring a general understanding of the project [7][8].

In order to develop an autonomous system, the functional and non-functional requirements were collected according to the team members' needs and project restrictions [9]. Thus, a detailed document containing hardware, software, and interface requirements was created. Such requirements are presented in the following subsections.

HARDWARE - In the interests of the current project, the hardware embraces all physical components that have any kind of electrical connection. This includes actuators, sensors, ICs (Integrated Circuits), microcontrollers, circuit boards, power sources, and eventually a computer. Naturally, there are constraints over them that must be taken into account while choosing the actual components.

<u>Functional Requirements</u> – This subsection describes the main functionalities and elements related to hardware that must be present in the system. It outlines the services to be executed, the system capabilities and features [8]. The functional requirements are listed below:

1.1. flow rate and temperature control in the fuel and air injection;
1.2. exhaust gasses recirculation;
1.3. actuators must supply the system with necessary reactants;
1.4. temperature sensing;
1.5. pressure sensing;
1.6. sensors must be deployed in a way that facilitates the creation of closed control loops;
1.7. data processing must be performed using a microcontroller;
1.8. the microcontroller should be able to communicate with every sensor and send signals to every actuator;
1.9. a computer should be able to receive data from the microcontroller to act as a human-machine interface;
1.10. workbench visual signaling.

<u>Non-Functional Requirements</u> – This subsection lists how the functional requirements can be implemented, the performance parameters should and constraints that must be respected:

2.1. preference for automotive components whenever possible to facilitate the process of embedding the system into a vehicle;
2.2. fuel pumps working at 4 bar;
2.3. the fuel flow rate regulation must be done with fuel injectors;
2.4. glow plug for fuel heating;
2.5. recirculation needs to be performed using a controllable valve;
2.6. air injection must be performed using a blower;
2.7. the microcontroller must support C/C++ programming languages;
2.8. microcontroller support for UART, CAN, SPI and I2C communication protocols;
2.9. the secondary microcontroller should support UART and internet communication protocols;
2.10. fuel pressure sensor resolution must be at least 0.01 bar;
2.11. K-type thermocouple resolution of 0.1°C;
2.12. differential pressure sensor resolution must be 0.01 kPa;
2.13. multiple voltage regulators: 1.8, 3.3, 5 and 10V;
2.14. external A/D (Analog to Digital Converter) capable of at least 14 bits of resolution;
2.15. actuators compatibility with PWM (Pulse Width Modulation) control when possible;
2.16. system electricity supply using one or more automotive batteries;
2.17. LEDs (Light Emitting Diode) for visual indication of status.

SOFTWARE - Software design and implementation has an important role in system automation. Parameters such as temperature, air, and fuel concentration need to be controlled to ensure the reactor works properly. Moreover, an initialization sequence must be respected. For that reason, the system's status should be displayed on an interface allowing the user to follow them.

<u>Functional Requirements</u> – This section describes the tasks that should be accomplished via software implementation. The software requirements are listed below:

3.1. automate the system, ensuring proper operational steps without the user's action;
3.2. implement closed-loop control of reactants, temperature, and recirculation;
3.3. provide an interface for the user.

<u>Non-Functional Requirements</u> – As a consequence of such complex software activities, constraints and performance parameters must be set, in order to guide the developer's work. The description of limiting factors and how the software will achieve the previously outlined functionalities are described below:

4.1. implement I2C and SPI protocols to communicate with sensors;
4.2. implement UART communication between microcontrollers;
4.3. implement internet communication between secondary microcontroller and computer;
4.4. contain a software structure designed to facilitate quality verification;
4.5. support a 10 Hz minimum sampling frequency;
4.6. implement a shutdown procedure to ensure safety;
4.7. display data in real-time, making it possible to alter parameters and step durations before initialization.

HARDWARE SPECIFICATION

After the requirements definition, the next phase in automation is the technical specification of the system itself. This phase demands special attention as there are various compatibility constraints, such as temperature and pressure working parameters, electrical demands, chemical resistance, control schemes, and size.

FUEL INJECTION SUBSYSTEM - Since fuel injection is frequently used in the current automotive powertrains, it was the first developed subsystem. In the present project, most components are commercial automotive parts such as fuel tanks, pumps, filters, injectors, glow plugs, and pressure regulators. By attending the requirement 2.1., the result should get closer to a real automotive system.

There are two important parameters to be controlled in the fuel line: temperature and mass flow. Therefore, it requires two sensors (temperature and pressure), and two actuators (fuel injector and glow plug) to perform a closed-loop control. The parameter values adopted are based on the LAC member studies, which brings the following results:

- fuel temperature around 70 - 80 °C;
- water flow of 0.47 g/s;
- ethanol flow of 0.81 g/s.

Using the previously described requirements, it becomes possible to choose the actual components, as now all the constraints and desired parameters are defined.

AIR SUPPLY SUBSYSTEM - Following the diagram, the next section is dedicated to the Air Supply, the subsystem responsible for feeding air into the reformer and controlling the recirculation of the exhaust gasses. Air is necessary for the reaction to become autothermal. Without it, the reactor would operate in the steam reforming reaction, which requires external heat to properly function [5].

With this in mind, closed-loop control of the air supply requires a blower with controllable speed and an air flow meter. The closed-loop for the exhaust will be made using a proportional control valve and temperature sensors. Through analysis of the chemical equilibrium from the working parameter of the air subsystem is:

- air flow of 1.22 g/s.

The values of pressure and temperature in the air line are unknown factors. The theoretical maximum ratings were estimated in order to avoid element incorrect use, as it could lead to early failure of the element and system. These parameters will be determined experimentally as they rely on physical reactor characteristics.

ELECTRONICS

As a consequence of the previous definition of parameters, it becomes possible to design the PCB (Printed Circuit Board) containing the circuits that provide an effective integration with each component. This is an essential step as there are multiple devices operating in different voltages. In addition to that, the PCB would allow individual and integrated test phases, signal conditioning and reliable data processing by the microcontroller.

The hardware and software design must accommodate changes in the system, such as the addition of new sensors or actuators. For this reason, it is very important to prepare the PCB for future improvements. With this in mind, every circuit has been designed to be as generic as possible, allowing different sensor outputs, different frequency actuators and different voltages to be added without compatibility issues. The stipulated over-dimension is at least 150% of the current quantity of inputs/outputs.

SENSORS - The intention behind the signal conditioning is to develop a solution that allows a large variety of analog sensors to be connected, regardless of its output range. In fact, the only circuit variation between different sensor outputs is the gain needed, which itself is determined by the value of the resistors on the gain loop of the amplifier. Luckily, the resistors can be easily swapped, resulting in generic signal conditioning.

In order to perform the gain and adequate signal for the A/D ratings, Operational Amplifiers will be employed. Figure 2 shows the final circuit schematic for the sensors. The A/D configuration consists of three steps:

- Gain, so that the voltage range value matches the A/D reference;
- Common mode gain, so the signal will be around ideal voltage value;
- Pre-filtering the signal, to achieve signal stability before the conversion.

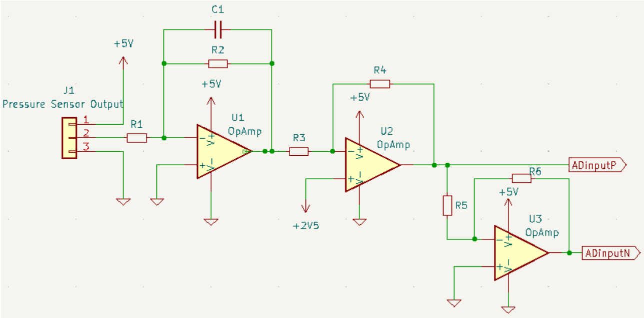

Figure 2. Generic circuit of signal conditioning.

Moreover, a capacitor is utilized as a low pass filter. The presented solution is called fully differential since there are two analog signals as positive and negative inputs of the A/D. Practicing this improves the quality of data acquisition, working around the ideal input voltage of the A/D converter.

ACTUATORS - Translating the microcontroller commands into physical actions is the task of the actuators. PWM has been chosen as the signal form to be sent, due to its simple implementation, precision and good acceptance by most components. However, there are two main constraints in this type of circuit: frequency and voltage compatibility. Figure 3 presents the circuit schematic.

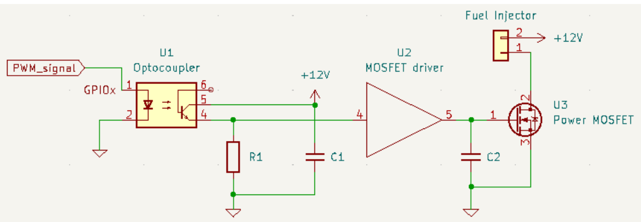

Figure 3. 12V PWM generating circuit example.

To overcome the previously mentioned issues, Power MOSFETs (Metal Oxide Semiconductor Field Effect Transistor) seem to be the perfect solution, given that they are able to work as a switch while isolating the microcontroller's PWM signal from high power circuitry. This component allows PWM control over the actuators while offering a wide frequency range. In addition, the same circuit can be used to perform digital triggering, turning elements on and off by setting the duty cycle at 100% and 0% respectively.

There are two categories of actuator PWM control: signal and power. Therefore, the over-dimension must be applied to both. For power, the elements have been chosen to match the standard value of 12 Volts. For signaling, however, there are different values among the actuators. In this case, voltage regulators will be employed as the source. A voltage regulator's output is dependent on the resistor association, calculated using a simple voltage divider formula and thus can be easily altered if needed.

MICROCONTROLLERS - The primary microcontroller must provide an operating surplus, such as pin availability, wide frequency range for PWM generation, communication protocol buses, and processing performance. Additionally, it is essential that it exhibits a reliability comparable to the one encountered in automotive applications. Another requirement is that the microcontroller possesses real-time performance. On the other hand, the secondary microcontroller's only task is to receive data from UART and send it via the internet to the computer, and for that reason its requirements are less demanding.

SOFTWARE MODELING

For modeling purposes, the UML (Unified Modeling Language) has been employed. This tool is a graphical language that helps to introduce organization in software development. By using UML diagrams, the developer is concerned mainly with software quality, reusability, and the satisfaction of user requirements [10].

USE CASE DIAGRAM - The Use Case Diagram (UCD) describes the same functional requirements presented in previous sections, but now from a software perspective. Therefore, the UCD serves as an intuitive tool to keep the user and the developer on the same page. The person icons symbolize the actors, the circles are the use cases, and the rectangular regions represent the packages [11].

In Figure 4 it is possible to visualize the project's UCD. Note that this diagram provides only a macroscopic vision, presenting dynamic subsystems' behavior and the actors involved inside their equivalent packages. Despite its simplicity, this visualization becomes especially useful in the creation of the following diagrams.

CLASS DIAGRAM - Considering that the software is centered around an object-oriented implementation, the class diagram becomes essential. Such a diagram focuses not on the services themselves, but rather on the objects that will execute them and their interactions. The class diagram is useful to specify and visualize the basic static elements and how they would work together [10].

Figure 5 shows the system class diagram. Each rectangle represents a class, which itself represents a set of objects that have similar characteristics. Inside each rectangle, its attributes and operations are described. It should be noted that a given set of classes should perform the use cases previously defined [11].

To illustrate the importance of understanding the interfaces that allow the controller to communicate, take the example of the GPIO (General Purpose Input Output) class. Through it, the controller should receive signals from the sensors and command the actuators, and thus communication protocols and PWM signals for each case must be defined.

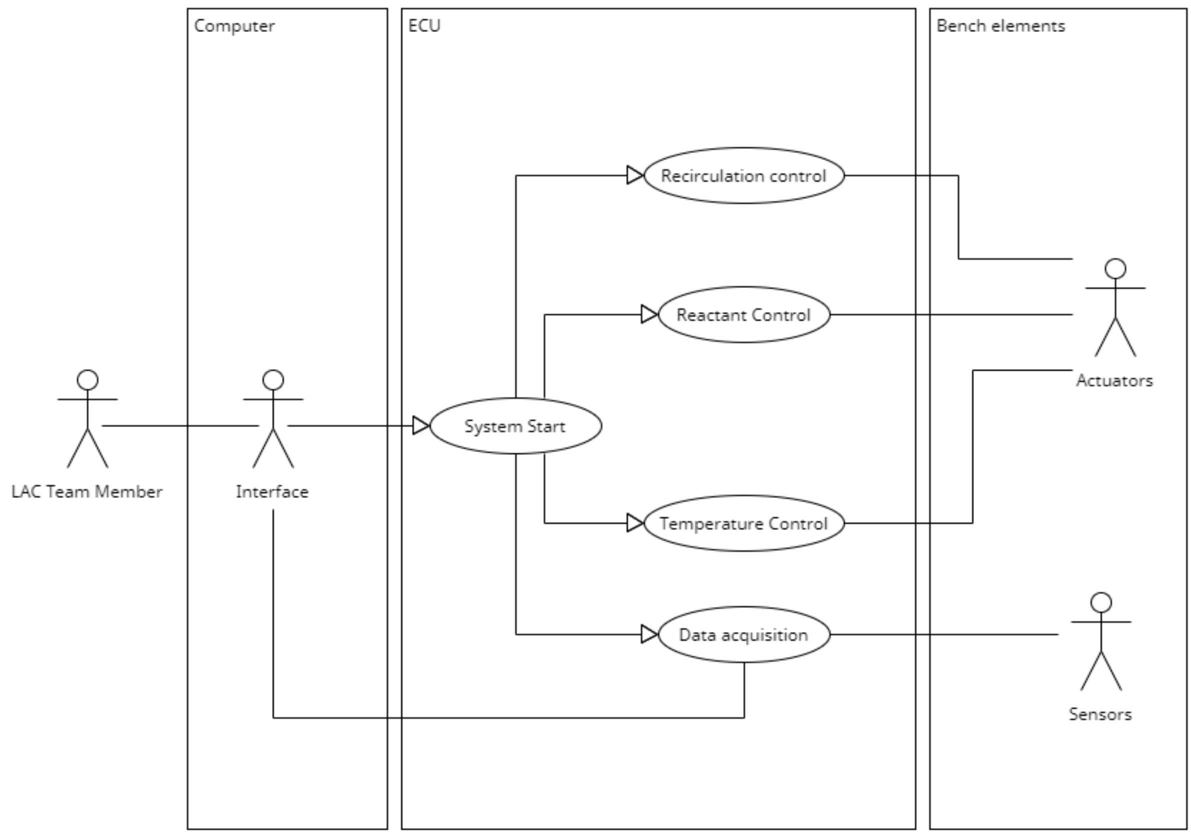

Figure 4. Use Case Diagram.

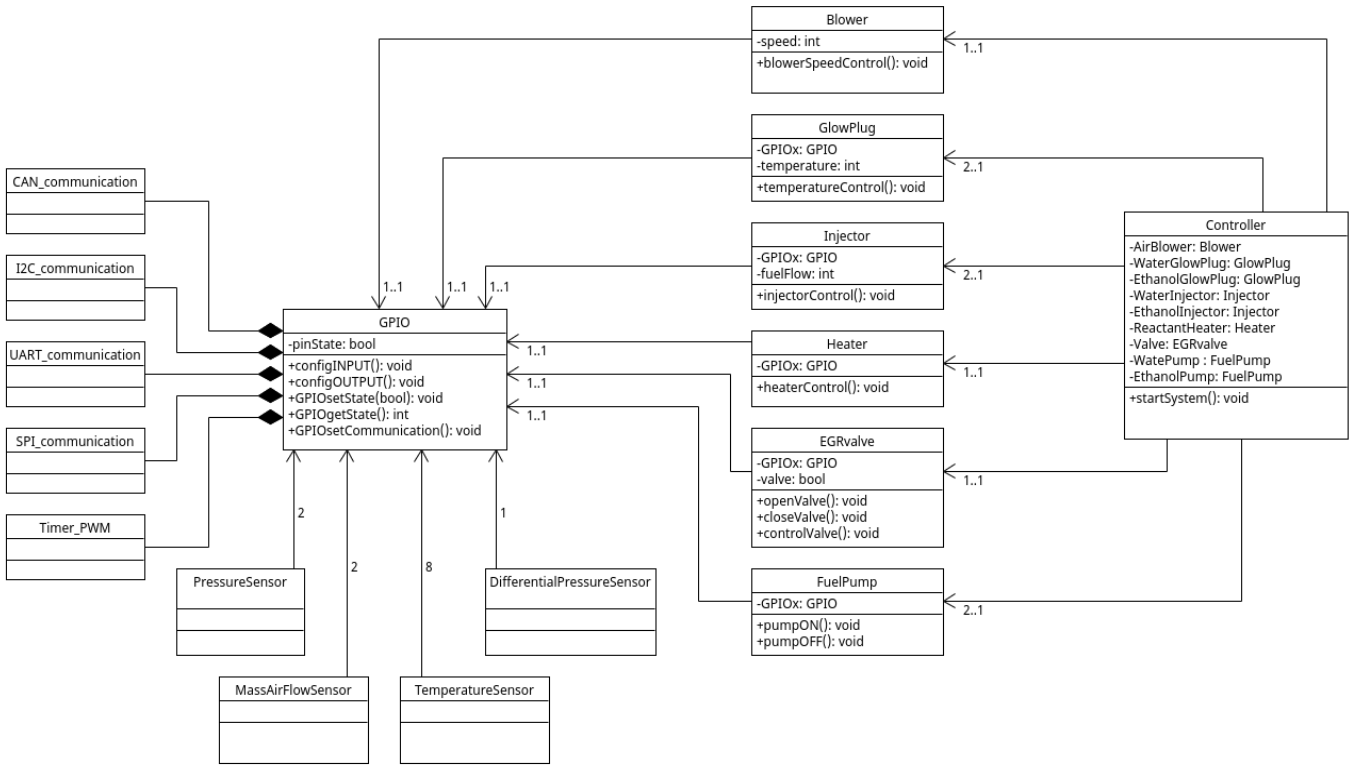

Figure 5. Class diagram.

Furthermore, the class diagram will be directly used in the coding phase as it represents the structure of the object-oriented software. It is good practice to start a code following the class diagram, as it promotes a more fluid work while ensuring software quality, preventing poor organization, and improving the software reusability.

SEQUENCE DIAGRAM - The illustration of the system's dynamic view and workflow is given by the diagram that will be discussed in the current section. This diagram emphasizes the modeling of object behaviors and the time ordering of messages. Sequence diagrams show the construction and destruction of objects, as well as their relationships. The sequence diagram of the proposed project is presented in Figure 6.

During an object's life, there are messages being exchanged. This communication is represented by the arrows between the lifelines of two objects. For example, a method call is considered a message. Through the combination of sequence and class diagrams it becomes possible to understand the structure of objects, their sequential steps and interaction in an object-oriented program. Therefore, their creation visually describes exactly how the code should function.

Vertical lines from top to bottom are used to symbolize the object's lifeline. Note from the diagram that the system starts when the parameters are set, and the start button is pushed. From this point on, the data acquisition becomes active. On a side notice, the sensor objects have been left implicit to improve diagram visualization. Moreover, the recommendation for such a complex system is the creation of the sequence diagram by use case, which would result in a better explanation of each individual case. However, only a single diagram was used, as its intended purpose in this paper is solely to highlight the general objects' behavior.

After the initialization signal, the first object to be created is the heater, whose task is to increase the air

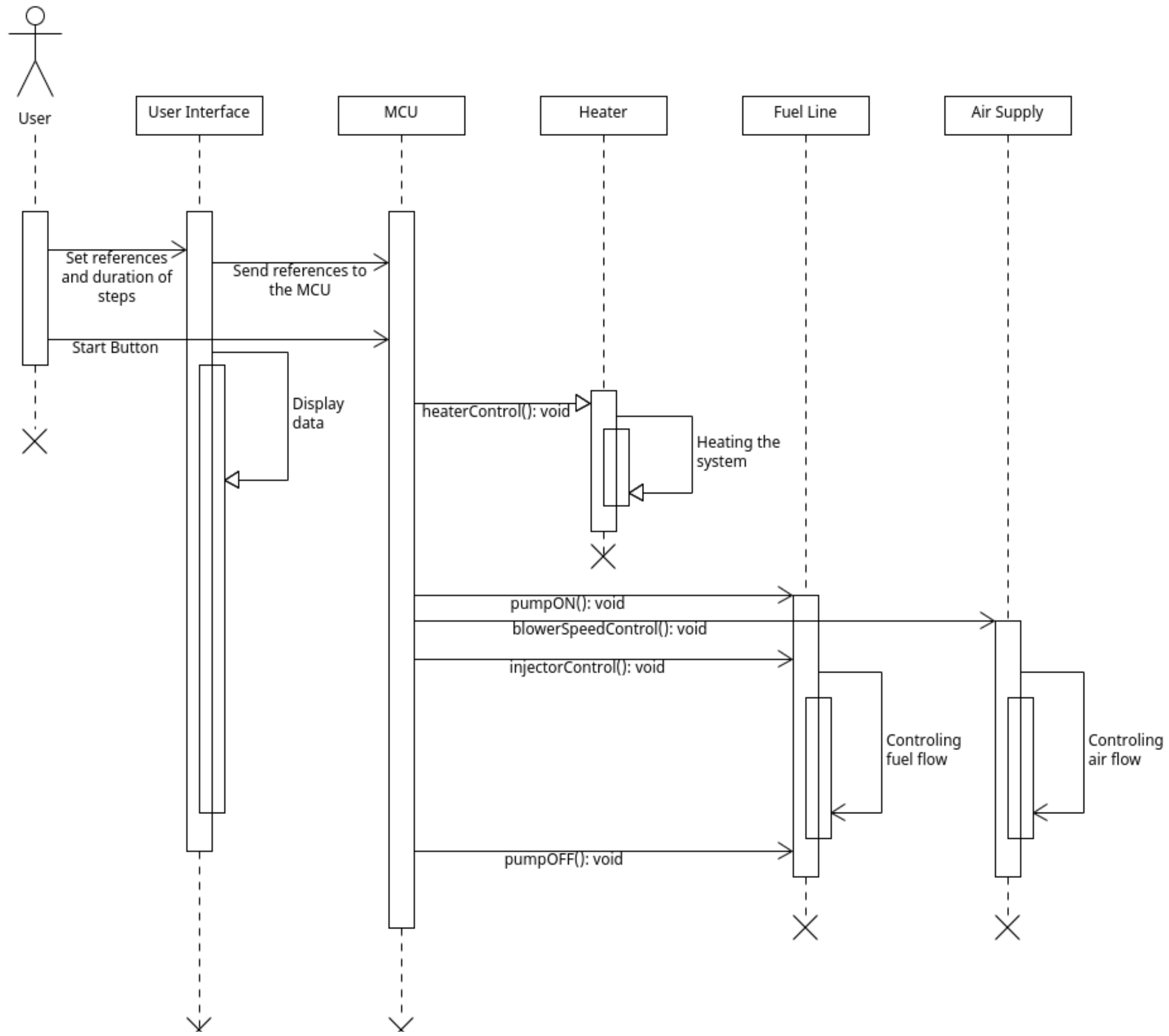

Figure 6. Sequence diagram.

temperature above 400 °C, thus the MCU (Microcontroller) holds while the heating process isn't finished. Once the heater object is destroyed, the microcontroller calls for the fuel injection and air supply methods. The duration of this step is given by the user input. In the end the action comes back to the MCU that must shut down the whole system.

Despite the simplifications adopted in the sequence diagram, there is an important characteristic that must be emphasized: the simultaneous control and acquisition processes. As an example: the controls of fuel injection, fuel temperature and air injection as well as the communication with the sensors are all happening concurrently. For this reason, care must be taken when dealing with simultaneous tasks, as they need to be schedulable so that the MCU can successfully perform its duties.

CONCLUSION AND PERSPECTIVES

Amidst the climate change crysis, the research and development of low carbon technologies has had two major motivations: the reduction of the world's dependence on fossil fuels and the minimization of CO2 emissions. Among the proposed solutions, hydrogen stands out due to its high energy density and the possibility of a neutral carbon cycle. The combination of hydrogen fuel with direct chemical energy to electricity conversion technologies, such as fuel cells, has the potential to reach up to 60% efficiency. Despite the elevated efficiency in comparison with piston engines, the use of hydrogen in the automotive sector is still hindered by its storage and the availability of refueling stations. As an alternative to the previously described scenario, this paper outlined the development of an autonomous system capable of converting ethanol, water and air into a hydrogen rich gas, therefore drastically reducing or even eliminating current issues that prevent widespread use of hydrogen in the mobility sector.

Despite the lack of similar proposals in the literature, the mechanical diagram provided an excellent basis for the structuring of the automation duties. This paper introduced the essential steps in the process of hardware and software integration, as well as project management concepts. As a result, the following sequence of activities was established:

- user requirements evaluation by interview method;
- actuator and sensors specification on the fuel line and air supply subsystems;
- description of electronic designing for signal conditioning and PWM control;
- static and dynamic software modeling, including use case, class and sequence UML diagram.

The accomplishment of such tasks enabled the creation of a diagram with exclusive respect to the interests of the automation cell, which is exhibited in Figure 7. Such an illustration reiterates the mechatronics concepts' relevance for the system progression. It highlights the data processing unit connections to the sensors and actuators, as wells as their position in the fuel line and air supply. It is also worth noting that through the mechanical connection line, the closed loops become clear. This considerable amount of information in a single diagram, and its visual appeal has turned it into the new model guide for prospect advancements.

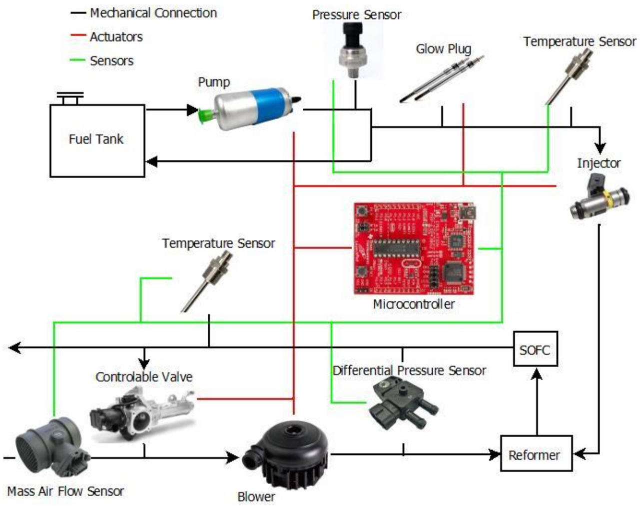

Figure 7. Workbench from Hardware and Software perspective.

In spite of the documented project evolution, many practical challenges still need to be solved before a real, working autonomous ethanol processing unit can be constructed. Consequently, the possibility of further improvements and innovation remains open. Such changes could come in the form of improved test phases, system safety procedures, instrumentation techniques, control methods, real time application, software quality and user interfacing to name a few. Some of these topics will certain be explored over the course of the project, and even more so in future works.

ACKNOWLEDGEMENT

This work was supported by Fundação de Desenvolvimento da Pesquisa - Fundep Rota 2030/Linha V 27192.01.01/2020.01-00.

REFERENCES

[1] BALAT, Mustafa. Potential importance of hydrogen as a future solution to environmental and transportation problems. International Journal of Hydrogen Energy, v. 33, n. 15, p. 4013-4029, 2008.

[2] Hirsch, R L, Bezdek, Roger, and Wendling, Robert. Peaking of world oil production: Impacts, mitigation, & risk management. United States: N. p., 2005.

[3] GOEBEL, S. G. et al. Fast starting fuel processor for automotive fuel cell systems. International journal of hydrogen energy, v. 30, n. 9, p. 953-962, 2005.

[4] MARTIN, Stefan; WÖRNER, Antje. On-board reforming of biodiesel and bioethanol for high temperature PEM fuel cells: Comparison of autothermal reforming and steam reforming. Journal of Power Sources, v. 196, n. 6, p. 3163-3171, 2011.

[5] HOU, Tengfei et al. Hydrogen production from ethanol reforming: catalysts and reaction mechanism. Renewable and Sustainable Energy Reviews, v. 44, p. 132-148, 2015.

[6] MAGUIRE, M.; BEVAN, N.. User Requirements Analysis. In: Hammond, J., Gross, T., Wesson, J. (eds) Usability. IFIP WCC TC13 2002. IFIP — The International Federation for Information Processing, vol 99. 2002. Springer, Boston, MA.

[7] Ehn, Maria; Derneborg, Mattias; Revenäs, Åsa; Cicchetti, Antonio. User-centered requirements engineering to manage the fuzzy front-end of open innovation in e-health: A study on support systems for seniors' physical activity. International Journal of Medical Informatics, v. 154, 2021.

[8] P. Bourque and R.E. Fairley, eds., Guide to the Software Engineering Body of Knowledge, Version 3.0, IEEE Computer Society, p. 32-45, 2014; www.swebok.org.

[9] Maguire, Martin; Kirakowski, J.; Vereker, N. RESPECT: User centred requirements handbook. Loughborough University, v 3.3, p. 93-114, 1998.

[10] Booch, Grady & Rumbaugh, James & Jacobson, Ivar. Unified Modeling Language User Guide, The (2nd Edition) (Addison-Wesley Object Technology Series). J. Database Manag.. 10. 1999.

[11] Martin Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd. ed.). Addison-Wesley Longman Publishing Co., Inc., v. 3, p. 48-88, 2003. USA. ISBN 9780321193681.