Performance Evaluation of Electric Vehicle Chargers under Controlled Temperature Conditions

Final Report of the Integrative Project

Vasco Ferreira da Rocha Melo

U.PORTO

U.Porto - L.EIC - Capstone Project

Tutor at U.Porto: Rolando Martins
Institution Supervisor: Pedro Pascoal

Date: 2024/2025

Contents

1 Introduction

This report aims to present all the work carried out during the curricular internship, which was conducted within the scope of the Capstone Project course of the Bachelor's Degree in Informatics and Computing Engineering at the Faculty of Engineering of the University of Porto (FEUP) and the Faculty of Sciences of the University of Porto (FCUP).

1.1 Context

The curricular internship was conducted at the Smart Grids and Electric Vehicles Laboratory (SGEV) of the Power Systems Center (CPES) at the Institute for Systems and Computer Engineering, Technology and Science (INESC TEC). The internship was supervised by engineer Pedro Pascoal from the institution and by professor Rolando Martins from FEUP/FCUP.

The curricular internship was carried out in the second semester of the 2024/2025 academic year, starting on February 14 and ending on June 27, totaling 20 weeks of internship.

INESC TEC is a non-profit Portuguese research and development institute involved in European projects, specializing in areas such as robotics, artificial intelligence, energy, and telecommunications.

SGEV is a space dedicated to research and development of innovative technologies in the field of energy systems, with a primary focus on energy efficiency, electric mobility, smart grids, and renewable energy. With the growing demand for electric vehicle (EV) charging solutions, the efficiency and safety of chargers are crucial aspects. INESC TEC currently develops AC chargers that operate in outdoor environments under weather exposure, which can cause a significant increase in their internal temperature, especially during summer. Given this scenario, it is essential to conduct tests to evaluate the impact of external temperature variations on charger performance. These tests enable understanding of how temperature affects the operation and efficiency of chargers, ensuring product reliability under real operating conditions.

1.2 Objectives and expected results

To develop and implement a temperature measurement system using DS18B20 sensors and Raspberry Pi 3, which will be used to monitor the thermal performance of electric vehicle chargers under different temperature conditions, simulated in a climatic chamber.

At the end of the internship, it is expected that the measurement protocol will be implemented and validated, enabling the collection of precise data on the thermal performance of chargers, which will be presented in a dashboard.

1.3 Report structure

The report is structured as follows:

- **Chapter 1: Introduction** - Presents the internship context, objectives, and report structure;
- **Chapter 2: Methodology** - Describes the methodology used during the curricular internship, the stakeholders (their role and responsibility), and activities developed;
- **Chapter 3: Development** - Details the protocol development, requirements (functional and non-functional), system architecture and technologies, developed solution (divided into different modules), and validation;
- **Chapter 4: Conclusion** - Presents the achieved results, lessons learned, and future work.

2 Methodology and main activities developed

This section describes the methodology adopted during the internship, which includes iterative development, communication and collaboration tools, stakeholders and their responsibilities, and the activities that were carried out during the internship.

2.1 Methodology used

The internship was conducted based on an iterative development approach, characterized by the definition of daily objectives and the execution of activities in the laboratory environment. At the end of each day, feedback was received from the supervisor, which enabled continuous adjustments and improvements in the work developed.

Additionally, some tasks were performed outside laboratorial hours, with the purpose of continuing the project's progress. These activities included specific adjustments to previously developed tasks, documentation of developed solutions (e.g., UML diagrams), and conducting complementary research related to the topic.

Throughout the day, an average of three main briefings were held. The first took place in the morning, with the objective of conducting a project status update, providing feedback on work possibly developed during the week, and defining the main objectives to be achieved during the day. The second briefing occurred at the beginning of the afternoon, when the progress of activities carried out during the morning was discussed. The last briefing was held at the end of the day, with the purpose of reviewing the tasks performed and planning the next project stages for the following week. In this briefing, complementary work to be carried out outside internship hours was also defined.

The supervisor was always available to offer support and clarifications, both during in-person days at the laboratory and outside laboratorial hours, through different communication channels.

In more technical cases or those requiring deeper analysis, there was also always the possibility of consulting other laboratory engineers, who were always available to help.

All communication within the laboratory was conducted in person. In cases where remote communication was necessary, the institutional email was used as the official means.

Since the project covered different areas of knowledge (electronics and computer science), communication was fundamental for aligning expectations and tracking work progress. Throughout the internship, all established requirements were continuously adjusted to meet both the institution's needs and the intern's expectations. In this way, the entire project was developed collaboratively and adaptively, ensuring that the goals were achieved effectively.

All code developed during the internship was initially stored in a personal GitHub repository, and later shared with the team through a GitLab repository.

2.2 Stakeholders, roles and responsibilities

This internship involved collaboration from different participants, each with specific responsibilities:

- Institution supervisor:
  - Engineer Pedro Pascoal: Responsible for guiding the intern through continuous supervision and providing constructive feedback regarding the work developed. Also assisted in integrating the intern into the institution's facilities.
- FEUP/FCUP tutor:
  - Professor Rolando Martins: Responsible for guiding the intern from the academic side, representing the curricular unit's objectives and in preparing the final report.
- Client:
  - Engineer Pedro Pascoal: Responsible for representing INESC TEC's interests, establishing project requirements and expectations. Also responsible for validating the work developed.
- Intern:
  - Vasco Melo: Responsible for developing the different tasks proposed by the client, taking into account the quality and time proposed for their completion. Also responsible for reporting the entire internship through the different evaluation elements proposed by the curricular unit.

2.3 Activities developed

During the internship, different activities were developed, which included project planning, market research, and prototype development. As new requirements emerged, activities were adjusted to meet the project's new demands. Thus, the activities developed can be described as follows:

1. Project planning and market research:
   . Introduction of the intern to the facilities and their equipment;
   . Establishment of an action plan for project development;
   . Market research was conducted based on available equipment and expected objectives, which were being established/adjusted throughout the project.
2. Raspberry Pi 3 and DS18B20 Sensors:
   . Raspberry Pi 3 configuration;

. Development of a data reading system with DS18B20 sensors.

3. Creation of a GUI and communication system between the RPI and the MainPC:

. Requirements gathering for the graphical interface and market technologies;

. Development of different prototypes for the GUI;

. Establishment of an SSL/TLS protocol with the RPI;

. Creation of the temperature test protocol structure with the Raspberry Pi 3 and the MainPC.

4. Set up a management system for test storage:

. Study on data to be stored;

. Creation of the database schema;

. Implementation of a database control system.

5. Implementation of a database migration system:

. Creation of a database migration system between the MainPC and the RPI;

. Creation of a database migration system between the RPI and the EVSE.

6. Communication with the EVSE:

. Establishment of SSH communication between the RPI and the EVSE.

7. Create a dashboard for data visualization:

. Study on dashboard requirements and functionalities;

. Analysis of available technologies for implementation;

. Testing of different technologies;

. Dashboard implementation with defined functionalities.

8. Test migration scheduling system:

. Establishment of a scheduling system between the MainPC and the RPI;

. Establishment of a scheduling system between the RPI and the EVSE.

9. Communication with the thermal chamber and stabilization detection system:

. Study on thermal chamber functionalities;

. Establishment of communication between the RPI and the thermal chamber via serial port;

- Investigation of different mathematical formulations for thermal chamber temperature stabilization detection;
- Implementation of a temperature stabilization detection system with the thermal chamber.

10. Documentation:
    - Elaboration of different UML diagrams for the developed system;
    - Creation of technical documentation for the developed system.
11. Bootstrapping:
    - Elaboration of a dockerfile to facilitate the installation of the MainPC component;
    - Elaboration of an ansible system to facilitate the installation of the RPI component.

For better organization of the work developed, a Gantt chart was elaborated, which can be seen in Figure 1.

Figure 1: Gantt chart for project tasks

3 Solution Development

The solution development for the project was a process that involved several stages that were established over time. Thus, the solution was adapted according to new requirements. The development involved not only software but also hardware components, which made the protocol development in a laboratory environment essential to ensure system effectiveness. Since part of the development was carried out outside the laboratory, a component that simulates the implemented hardware was also developed. The solution took into account not only its usability, but also the difficulties and limitations encountered during the development process.

3.1 Requirements

This section will identify all functional and non-functional requirements established throughout the internship:

Functional requirements:

- Temperature data reading: The intern must develop a temperature data reading system;
- Graphical interface: The solution must include a graphical interface that will allow the user to control the test;
- Real-time visualization of obtained data: the developed system must allow the user to visualize the test data obtained in real time;
- Database: All values obtained from the test must be stored in a local database;
- Communication: The system must ensure communication between devices (between the RPI and the EVSE and between the RPI and the MainPC).

Non-functional requirements:

- Performance: The system must be capable of processing data in real time without significant delays;
- Scalability: The system must be capable of supporting an increase in data volume without compromising performance;
- Reliability: The system must ensure data integrity and availability during operation;
- Security: The system must ensure minimum data security standards and protection against unauthorized access;
- Maintainability: The system must be designed to facilitate maintenance and updates over time.

Constraints:

– The entire developed system must be limited to the material and resources available in the laboratory;
– The system must be tested in a laboratory environment to ensure result fidelity;
– The system must be capable of operating in different environments without losing its functionality.

3.2 Architecture and technologies

Throughout the internship, numerous technologies and architectures were considered to meet the client's requirements. As the requirements changed throughout the project, the project structure ended up being adjusted. This section will address the different architectures and technologies approached and their reasons for being selected.

3.2.1 Hardware components

The hardware components used were limited to the availability of resources and materials in the laboratory, which restricted the implementation options. For the solution implementation, a Raspberry Pi 3 and four DS18B20 sensors were used. For protocol validation, an EVSE charger developed by the Laboratory and a thermal chamber (incubator) IMP 400 from HERATERM (available in the Laboratory) were used.

The Raspberry Pi 3 is a low-cost and highly versatile microcomputer, ideal for prototyping and automation projects. Another advantage is its ability to connect to various devices and sensors, facilitating integration in IoT projects. The Raspbian operating system was installed on the RPI, which provides a user-friendly interface and support for various libraries necessary for project development, namely a set of drivers and configurations to use the Raspberry's GPIO input. The GPIO input was essential for communication between the DS18B20 sensors and the Raspberry Pi, allowing reading and control of connected devices. Another advantage of the Raspberry is the integration of a network card in the system, which facilitated communication with other devices and data exchange between them. For all configuration and establishment of communication between devices, the book [6] was used.

The DS18B20 sensors are used to measure temperature and are connected to the Raspberry Pi through the GPIO input, allowing precise reading of temperature data. This sensor was selected for the following reasons:

- Versatility: The sensor has a long wire which helps in positioning the sensors inside the thermal chamber. Additionally, the sensor is encapsulated, making it resistant to adverse environments.
- Digital temperature reading: The RPI's GPIO input only reads digital values and not analog values (very common in temperature sensors). The DS18B20 sensor reads temperature digitally, which facilitates data acquisition (does not require an analog module).

- Cost: The sensors are affordable and available in the market.

The EVSE (Electric Vehicle Supply Equipment) developed by INESC TEC is an AC (alternating current) charger, single-phase (230VAC), with a nominal power of 7.4kW = 32A, which has various installed sensors that allow measuring current, voltage, frequency, and temperature values (temperature values refer to the circuit). By reading voltage and current, power can be calculated and power consumption (W) over time gives us consumption or charging in energy (W/h), which is essential to see the efficiency of electric car charging over time. Inside the charger there is a Raspberry Pi that manages charging control and enables EVSE communication with an INESC TEC database for sending charging data.

The IMP 400 is a thermal chamber that allows controlling the internal temperature, ensuring controlled conditions for conducting tests. It has the capacity to maintain constant internal temperature between 5°C and 70°C (which covers the temperature range experienced in Portugal). Internally it has a temperature sensor that allows knowing the chamber temperature during the test. Finally, the thermal chamber also has an RS 232 input that allows serial port communication with the Raspberry Pi. Due to a limitation of the thermal chamber itself, only query commands are available (allows, for example, obtaining the temperature of the chamber's internal sensor), but set commands (for example, changing the chamber temperature) are not available. For better understanding of the thermal chamber functionalities, its manual [2] was used and for more specific questions, the HERATERM support team.

Additionally, other components such as resistors, cables, and connectors were also used to ensure communication between the devices and the RPI.

3.2.2 Software components

The software components used were the main focus in the study of available architectures and technologies on the market and their advantages. The main software components selected and the primary reasons for their choice were:

. Operating System: The operating system used on the RPI was Raspbian, which is a Linux distribution optimized for the Raspberry Pi. Raspbian provides a user-friendly interface and support for various libraries necessary for project development;

. Programming Language: The programming language used was Python, which is widely used in automation and IoT projects due to its simplicity and versatility. Python has various libraries that facilitate communication with sensors and manipulation of obtained data;

. Database: To store data obtained from sensors, MySQL was used, which is a widely used relational database management system. MySQL was chosen due to its robustness, scalability, and ease of use, in addition to being compatible with Python through the MySQL Connector library. Another important factor is that it is the main database system used in the laboratory, which facilitated integration with other existing systems;

. Graphical Interface: To create the system's graphical interface, the PyQt library was used, which is a graphical interface development library for Python;

. Dashboard: For real-time visualization of obtained data, Grafana was used, an open-source data visualization platform that allows creating interactive and customized dashboards;

. Bootstrapping: To ensure correct system initialization, Ansible was used for installation and configuration of necessary software components on the RPI and dockerfile with devcontainer integration on the MainPC.

In addition to these components, other software was used that helped in protocol development, namely:

. Git: For source code version control and to facilitate file transfer from the development computer to the RPI;

. Visual Studio Code: As IDE for source code development;

. MySQL Workbench: For MySQL database management, allowing creation and manipulation of tables, queries, and other database objects;

. RealVNC: For remote access to the Raspberry Pi, allowing visualization and control of the system's graphical interface from another computer;

. Drawio and Mermaid: For creating diagrams that help visualize system architecture and project planning;

. Fritzing: Electronic prototyping software that was used to draw the DS18B20 sensors connection circuit to the Raspberry Pi, facilitating circuit understanding and implementation.

3.3 Developed solution

A temperature test corresponds to a process of obtaining temperature data over a period of time that can present characteristics such as the initial temperature of the thermal chamber and its final temperature. The protocol's main objective is to ensure the necessary conditions for conducting the temperature test, in addition to considering other factors such as communication between devices, sensor data reading, and visualization of obtained data.

To help interpret the test data obtained, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) were used to calculate the moment when the temperature read by the thermal chamber control sensor and the control sensor stabilized.

The developed solution can be divided into two main components:

. **RPI**: Corresponds to the part of the system that is responsible for controlling the DS18B20 temperature sensors, obtaining temperature data from the thermal chamber and EVSE, and communicating with the MainPC about the test status;

. **MainPC**: Corresponds to the part of the system that is responsible for serving as a user interface through a graphical interface. Additionally, it is responsible for communicating with the RPI about actions to be performed, such as starting or stopping the test. It is on the MainPC that a dashboard (Grafana) is installed that allows visualizing obtained data in real time and a database (MySQL) that stores test data.

Both components have a MySQL database that stores test data, allowing subsequent data consultation and analysis. The schema of both databases is the same. To be able to see results obtained in real time, both components have a data migration system through scheduling. For this to be possible, communication between the RPI and MainPC is essential.

All subsystems are integrated with parallelism via threads, which allows different system components to function independently and simultaneously, in addition to being easy to deactivate via .env. On the MainPC, signals are also used (via pyqtSignal) for the graphical interface thread to communicate with the other MainPC threads.

Thus, the protocol scheme can be translated into the diagram in the Figure 2, which shows the communication between the RPI and MainPC, as well as the different components that make up the system.

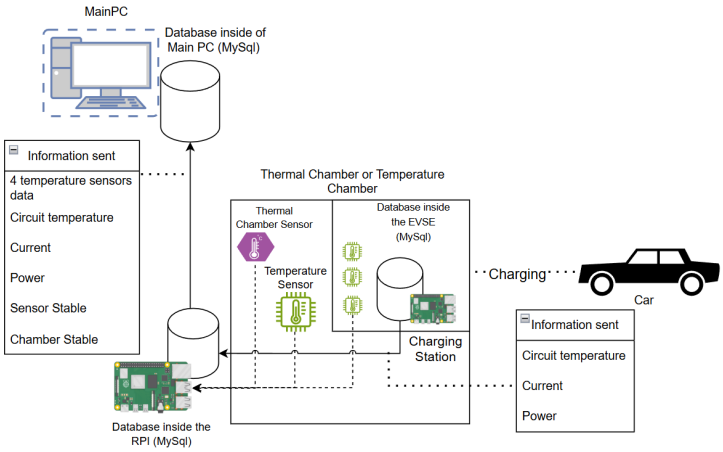


Figure 2: Communication protocol diagram between RPI and MainPC

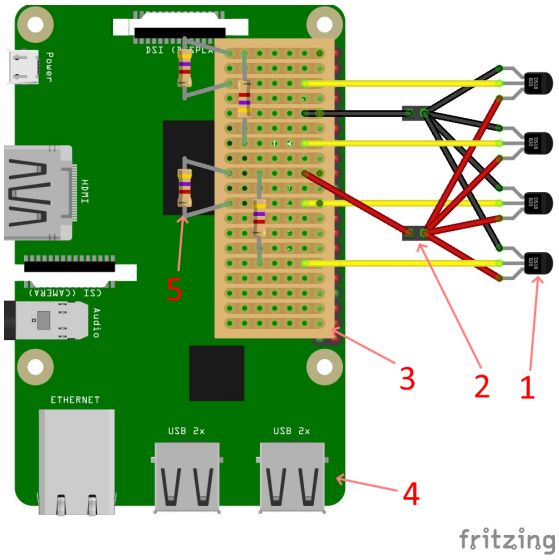
In the following sections, the different essential system components will be addressed, as well as their installation.

Another important point to consider is that the system is modular, which allows easily disabling features, changing configurations (IPs, ports, etc.), and modifying parameters. For this, .env files were used that allow defining system environment variables, facilitating configuration. More information can be found in the general protocol documentation [3].

3.3.1 DS18B20 temperature sensors and circuit

For reading temperature data from DS18B20 sensors, a circuit was developed that allows connecting the sensors to the Raspberry Pi through the GPIO input. Since the circuit can be soldered, it allows greater robustness and reliability in sensor connections.

In the Figure 3, the circuit diagram of the DS18B20 sensors connection to the Raspberry Pi is shown, which was drawn using Fritzing software. The circuit consists of a 4.7kΩ resistor connected to the data pin of the DS18B20 sensor, which is essential for correct sensor operation. The circuit also includes a connector that allows connecting the sensors to the Raspberry Pi through a busboard.



Legend:

1. DS18B20 Sensor
2. Connector
3. Connection busboard
4. RPI
5. 4.7 kΩ resistor

Figure 3: DS18B20 sensors connection circuit diagram to RPI

In the Figures 4 and 5, it is possible to see the circuit implemented in the laboratory.

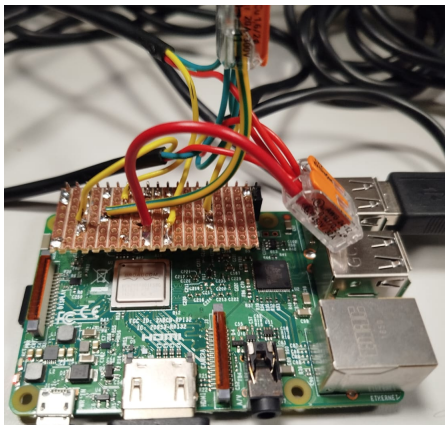


Figure 4: Top view of DS18B20 sensors connection circuit to RPI

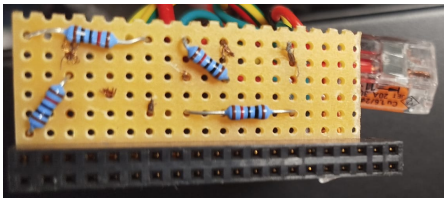


Figure 5: Resistors view of DS18B20 sensors connection circuit to RPI

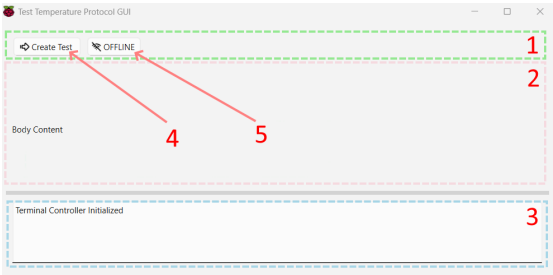
This circuit allows connection of more sensors, however we decided to install only 4 DS18B20 sensors, one of the sensors is used to measure the thermal chamber temperature (serving as chamber temperature control) and the other three are positioned inside the EVSE at different locations to measure the charger temperature at different points.

For its hardware and software implementation (sensor reading), the sites [4] and [5], and the DS18B20 sensor datasheet [1] were used.

3.3.2 Graphical interface

The graphical interface was developed using the PyQt library and has as its main objective to help the user control the temperature test, allowing to create (establish time, test temperature and test description), start the temperature test, forcibly stop the test, and extend the test duration. The interface was developed to be easy to implement other functionalities in the future.

In the Figure 6, the graphical interface can be seen, which consists of a main window with an upbar, body, and terminal. The upbar contains the connection button to the RPI and the Create a Test button, which allows creating a temperature test. The body contains the terminal that displays messages related to the test status and other information.



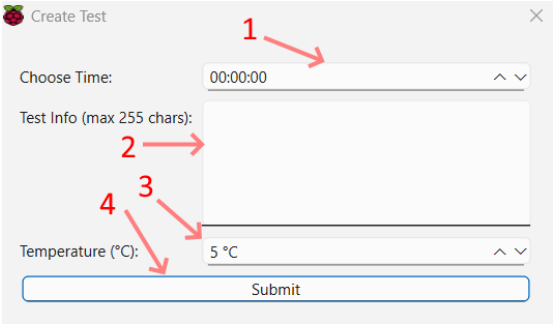
Legend:

1. Upbar
2. Body
3. Terminal
4. Create a Test button
5. RPI connection button

Figure 6: Protocol graphical interface

The Connection button allows the user to establish connection with the RPI (with IP and port parameters defined in the .env file). After the connection is established, the user can create a temperature test through the Create a Test button, with the window appearing as a pop-up.

The Figure 7 shows the temperature test creation pop-up, which allows the user to define the test time, test temperature, and test description.



Legend:

1. Test timer
2. Test description
3. Test temperature
4. Test submission button

Figure 7: Temperature test creation pop-up

If the form is filled correctly (test time is greater than 0), the test is created and communicated to the RPI (DS18B20 sensors). More information about the communication process in subsubsection 3.3.3.

To communicate user interactions with other MainPC threads (communication and scheduling), pyqtSignal (native to PyQt) was used, which allows sending signals between different MainPC threads.

3.3.3 Communication

Communication between the two devices is essential for creating/controlling temperature tests, but also as the main way to maintain synchronization of the scheduling system of the two devices (MainPC and RPI). It also helps to notify the MainPC scheduling system when the control sensor and thermal chamber sensor have stabilized.

Communication between the RPI and MainPC is done through an SSL/TLS protocol to ensure the security of transmitted data. For this, there is a need to create an SSL/TLS certificate that is used to establish the secure connection between the two devices.

Although communication works peer to peer, to facilitate implementation the RPI behaves as a server (opens a port where it waits for a connection) and the MainPC as a client (connects to the RPI). After the connection is established and certificate authentication is validated, the RPI does not accept any more connections, establishing a peer-to-peer system.

The sequence diagram in Figure 8 illustrates the communication between the MainPC (on the right) and the RPI (on the left). It highlights the various components involved in the process, including SSL/TLS communication, as well as the MainPC and RPI threads affected by this interaction.

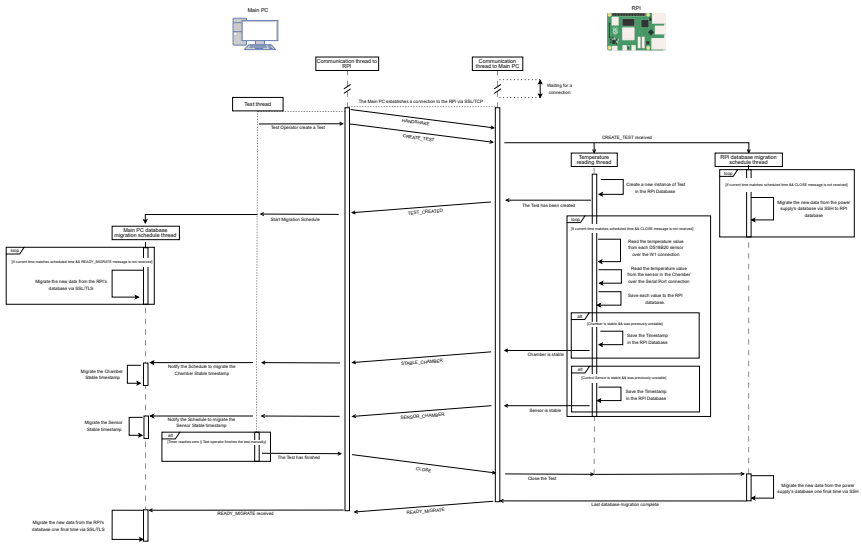


Figure 8: Communication protocol diagram between RPI and MainPC

More information about the communication process between RPI and MainPC can be found in the general protocol documentation [3].

3.3.4 Database

To safeguard the integrity of temperature test data, a MySQL database was configured on the MainPC and RPI with the same schema. The schema is based on the EVSE database tables (which already existed in the laboratory) and was adapted to include the test structure.

Figure 9 presents the database schema, highlighting the main tables used in the protocol. For confidentiality reasons, tables containing EVSE data are not included in this diagram.

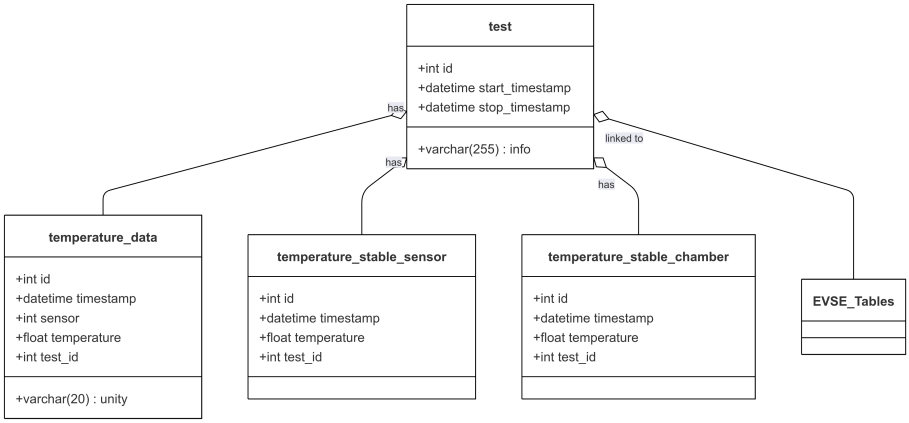


Figure 9: Protocol database class diagram

The databases are accessed through a Python library called MySQL Connector, which allows connection and manipulation of MySQL database data. The *test* table is the main table that stores temperature test data, while the *temperature_data* table is the table that stores temperature data obtained from DS18B20 sensors. The *temperature_stable_sensor* and *temperature_stable_chamber* tables are auxiliary tables that store the moment when the temperature read by the control sensor and thermal chamber sensor stabilized, respectively.

3.3.5 Data migration and scheduling

Scheduling is the automated planning and execution of tasks at specific times or regular intervals.

Since temperature tests can last several hours, it was necessary to implement a data migration system between the three system components (RPI, MainPC, and EVSE), to allow real-time visualization of obtained data. For this, it is important to ensure synchronization between different components through communication between them. A scheduling system through threads is used to perform data migration between different system components.

The scheduling system on the RPI is responsible for obtaining data from the EVSE. The scheduling system on the MainPC is responsible for obtaining data present on the RPI (DS18B20 sensor temperature values, thermal chamber control sensor, and EVSE data).

Additionally, after the RPI notifies the MainPC that the thermal chamber control sensor and control sensor have stabilized, the MainPC scheduling system is responsible for performing the migration of this data to the MainPC database on the next trigger.

The Figures 10 and 11 show the sequence diagrams of the scheduling system on the MainPC and RPI, respectively. These diagrams illustrate the interaction between the different components involved in the scheduling process, including the threads responsible for data migration and the actions performed by the MainPC and RPI during the scheduling process.

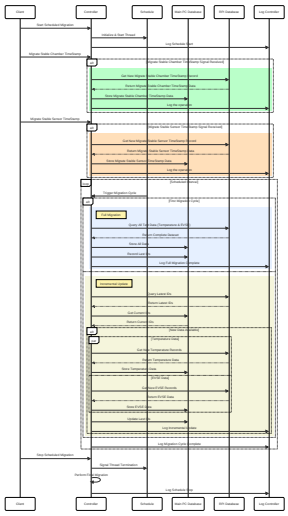


Figure 10: MainPC scheduling system sequence diagram

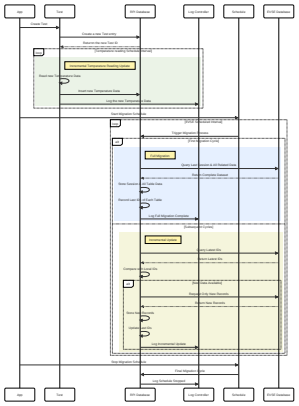


Figure 11: RPI scheduling system sequence diagram

3.3.6 Dashboard

A dashboard is a graphical interface that allows visualizing data in an interactive and customized way. The main objective of the dashboard is to allow the user to visualize temperature test data obtained in real time and charger data in real time.

Grafana not only allows real-time graphical visualization of obtained data, but also automatic creation of annotations to show the moments when temperature stabilized (both for the thermal chamber control sensor and EVSE control sensor).

Additionally, it is possible to insert the theoretical power value of the charger to calculate the efficiency of electric car charging over time, allowing more detailed analysis of charging performance.

Another important factor is the easy switching of test visualization, allowing the user to select the test they want to visualize and analyze the obtained data.

The Figure 12 shows the dashboard developed for visualizing temperature test data obtained in real time.

Figure 12: Test data visualization dashboard

Legend:

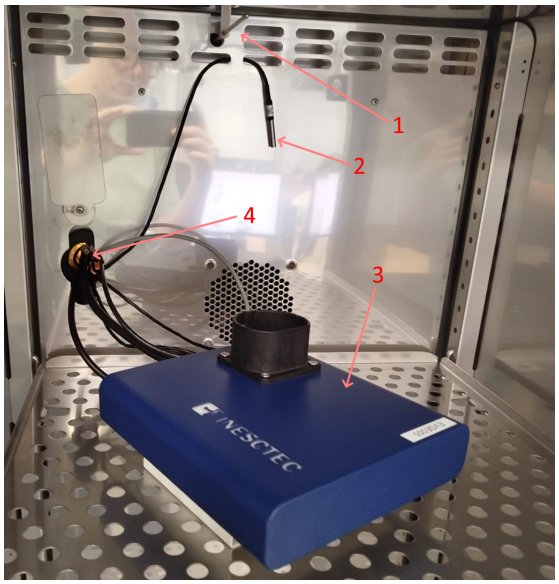
1. Example graph for visualizing test data obtained
2. Test selection dropdown (via *test_id*)
3. Input box for theoretical charger power value
4. Time interval selection for data visualization
5. Automatic annotation of thermal chamber sensor temperature stabilization
6. Automatic annotation of control sensor temperature stabilization

3.3.7 Installation and configuration

One of the tasks developed during the internship was system bootstrapping, to allow its rapid installation and configuration. For RPI component installation, Ansible was used while for MainPC component installation, a dockerfile with devcontainer integration was used (more information in [3]).

After initial setup and system configuration, the different hardware components must be properly connected and positioned.

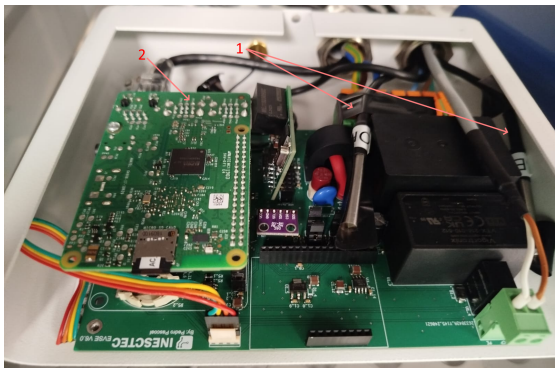
The Figures 13, 14, 15, and 16 show the installation of the EVSE inside the thermal chamber, as well as the positioning of DS18B20 sensors inside the EVSE and the thermal chamber. The figures also show the external rear part of the thermal chamber, where the electric vehicle charging cable connector is located (more information in [3]).



Legend:

1. Thermal chamber sensor
2. DS18B20 thermal chamber temperature control sensor
3. EVSE
4. Exit to the exterior of the thermal chamber

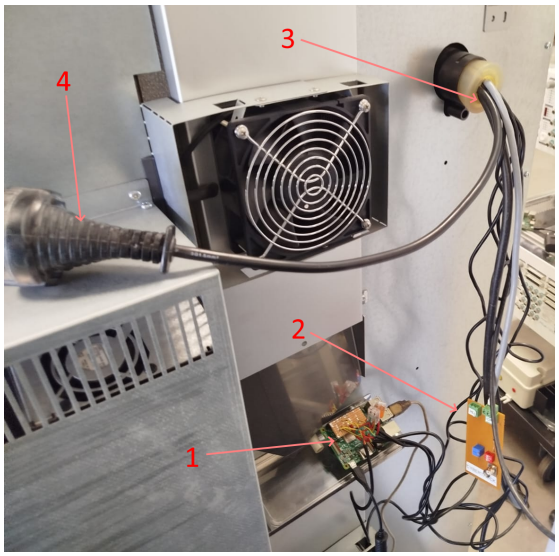
Figure 13: Image of EVSE inside the thermal chamber



Legend:

1. DS18B20 sensors positioned inside the EVSE
2. EVSE Raspberry

Figure 14: Positioning of DS18B20 sensors inside the EVSE



Legend:

1. Test control RPI
2. Control Pilot Simulator
3. Entry to the interior of the thermal chamber
4. Electric vehicle charging cable connector

Figure 15: External rear part of the thermal chamber

Legend:

1. IMP 400 thermal chamber
2. Car charging cable

Figure 16: External rear part of the thermal chamber

3.4 Validation

For protocol validation, several tests were carried out throughout the internship. The most relevant tests performed were the following:

- DS18B20 sensor precision and accuracy: Place DS18B20 sensors inside the thermal chamber and verify if the values read by the sensors are consistent with the thermal chamber temperature (several iterations were performed with different chamber temperatures);

- Scheduling: Verify if the scheduling system is working correctly with all system components (MainPC, RPI, and EVSE). For this, DS18B20 sensors were placed in the thermal chamber while the EVSE was outside the chamber sending data without any charging taking place;

- Logistics: Verify the arrangement of three DS18B20 sensors inside the EVSE inside the thermal chamber. It also served to verify our hypothesis that the charger would be at a lower temperature than the thermal chamber while the thermal chamber temperature was not stabilized (increasing temperature, for example), but that eventually the charger would reach a temperature higher than the thermal chamber;

- Test with an electric vehicle: To validate the protocol under real conditions, a test was performed with different electric vehicles that require different charging powers. The results obtained not only validated the protocol, but also allowed verifying the efficiency of electric car charging over time.

The results obtained from tests performed under real conditions revealed that the EVSE charger at high temperatures (above 60 degrees Celsius inside the thermal chamber) can maintain charging efficiency above 95% (which is considered acceptable for electric vehicle charging).

During the tests conducted at high temperatures over an extended period, it was observed that at a certain point, the EVSE stopped collecting performance data on the electric vehicle charging process. After conducting additional tests at high temperatures, it was discovered that the Raspberry Pi inside the EVSE, which is responsible for collecting charging performance data, would stop the data collection service when the CPU reached 85 degrees Celsius. This occurs due to a safety mechanism in the Raspberry Pi that shuts down non-essential services to prevent permanent damage.

Furthermore, this year, the highest temperature ever recorded by the external chargers since their installation was observed, reaching 79 degrees Celsius. The discovery of this charger limitation is important for the future, as it compromises the analysis of electric vehicle charging performance, which is essential for the research conducted in the laboratory. This limitation should be considered for a future redesign of the charger and for more in-depth studies on its performance.

4 Conclusions

4.1 Achieved Results

The temperature testing protocol developed achieved the proposed objectives, enabling the monitoring and analysis of electric vehicle charging, with real-time and easy monitoring of the obtained results. Furthermore, it was possible to verify the efficiency of electric car charging over time, allowing performance analysis of the EVSE charger under different temperature conditions. Additionally, the protocol was developed in a modular fashion, which allows it to be easily adapted to different types of devices, sensors, databases, etc. For a research laboratory context, the protocol is easily adaptable to different research contexts.

This protocol will be an important aid in the development of new electric vehicle chargers, as a means of validation.

4.2 Lessons Learned

The internship provided a deeper insight into research work in a laboratory context, where several projects are being developed simultaneously. Furthermore, since the proposed problem was applicable to a real context and was developed from scratch, it was possible to put into practice the various lessons about software development learned throughout the course, including both technical knowledge, such as parallelism and data management, and non-technical skills, such as the importance of good documentation, effective communication with the team, and good work organization.

4.3 Future Work

Future work would involve improving the protocol to provide more relevant information about charging that would help in the performance analysis of the EVSE charger, such as being able to perform multiple charging sessions during the same test. Another interesting feature would be the possibility of comparing the obtained results with results from other conducted tests. Furthermore, it would be interesting to implement a REST API to allow the protocol to be easily integrated with other systems and applications.

4.4 Acknowledgments

In this section, I would like to thank everyone who contributed to making the internship a success. First, I would like to thank INESC TEC for the opportunity to carry out the internship at the SGEV laboratory, where I had the opportunity to learn and grow as a professional. I would also like to thank my supervisor at the institution, engineer Pedro Pascoal, for his support and guidance throughout the internship and for providing me with the best means and resources for the development of the work, in addition to all his patience

and understanding throughout the process. I would like to thank my supervisor at FEUP/FCUP, professor Rolando Martins, for his support and guidance throughout the internship in the academic aspect and for helping me with the approval and suggestion of improvements to the project. I would like to thank all my internship colleagues, who helped and supported me throughout the internship, and who made the work environment more pleasant. Finally, I would like to thank the Faculty of Engineering of the University of Porto (FEUP) and the Faculty of Sciences of the University of Porto (FCUP) for the opportunity to develop this project and to learn from the best during my degree.

Bibliography and References

[1] Ds18b20 temperature sensor datasheet. URL: https://www.circuitbasics.com/wp-content/uploads/2016/03/DS18B20-Datasheet.pdf.

[2] Heratherm refrigerated incubators imp 180/ imp 400 manual [en]. URL: https://assets.thermofisher.com/TFS-Assets/LSG/manuals/LED-Heratherm-Refrigerated-Incubators-50150875-IMP180-400-Manual-EN.pdf.

[3] Project general documentation. URL: https://drive.google.com/drive/folders/1x0s7E_J1_cMlB3XS-bAqM-qdL6Pi9nff?usp=sharing.

[4] Brian, Mark, Benoit, Sara Santos, and Alessandro. Raspberry pi: Temperature readings with ds18b20 (python), Oct 2023. URL: https://randomnerdtutorials.com/raspberry-pi-ds18b20-python/.

[5] Scott Campbell. Raspberry pi ds18b20 temperature sensor tutorial, Mar 2025. URL: https://www.circuitbasics.com/raspberry-pi-ds18b20-temperature-sensor-tutorial/.

[6] Derek Molloy. *Exploring raspberry pi: Interfacing to the real world with embedded linux*. John Wiley and Sons, Inc, 2016.