Martín Gállego Casals

Design of a SCADA system with CAN communication for a portable DC microgrid

Treball de Fi de Grau

Dirigit pel:

Msc. David Alejandro Zambrano Prada

Grau d' Enginyeria en Electrònica Industrial i Automàtica

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2023

Table of contents

List of Tables

List of Figures

List of Abbreviations and acronyms

Abbreviation	Definition
AC	Alternating Current
ACK	Acknowledgment
BMS	Battery Managment System
CAN	Controller Area Network
CAN FD	Controller Area Network Flexible Data-Rate
CRC	Cyclic Redundancy Check
CSMA/CD-A	Carrier-Sense Multiple Access with Collision Detection using Arbitration
DC	Direct Current
DLC	Data Length Code
DLL	Data Link Layer
ECU	Electronic Control Unit
EOF	End Of Frame
ESS	Energy Storage System
EVB	Electric Vehicle of Battery
FPGA	Field-Programmable Gate Array
ID	Identifier
IDE	Identifier Extension
IOT	Internet Of Things
ISO	International Organization for Standardization
LIN	Local Interconnect Network
LV	Low Voltage
MV	Medium Voltage
NI	National Instruments
NRZ	Non-Return-to-Zero
OSI	Open Systems Interconnection
PL	Physical Layer

RIO	Reconfigurable I/O
RTR	Remote Transmission Request
SCADA	Supervisory Control And Data Acquisition
SOF	Start Of Frame
SST	Solid State Transformer
UFCEV	Ultra-Fast Charging of Electric Vehicles
VI	Visual Instrument

1. Introduction

1.1 Background

This project is part of the Ultra-Fast Charging of Electric Vehicles (**UFCEV**) research project, which is sponsored by Ministry of Science and Innovation of Spain and is being developed at the university [1]. In order to understand the goals of this project, it is necessary to give some context and background on that research project too.

1.2 UFCEV

1.2.1 Context and state of the art

Transportation based on electric vehicle of battery (**EVB**) is one of the most attractive solutions among the usual scenarios that are often proposed to mitigate the rate of global warming. This is mainly due to the positive consequences that its massive introduction could produce in both economy and environment.

The underlying hypothesis is that EVBs could play a significant role in both transportation and people mobility under certain circumstances benefiting specific groups of users. Although significant advances in the field of technologies for energy storage can be still expected, it can be affirmed that the existing EVBs are capable of performing specific tasks of transportation satisfactorily.

Moreover, there exists a clear consensus in admitting that the main obstacles that complicate the market penetration of EVBs are the so called range anxiety (fear of not having enough charge in the vehicle), customer price, charging time and charging availability, all of them related to the battery. To overcome the range anxiety and decrease the charging time, the so called fast charging technique has emerged in the last years, which represents a short time of charging for both battery and energy supply system.

Although there are around half a dozen fast chargers in the market, there is not a clear model of charging station that can be used as a reference to calculate investments cost for such an infrastructure, which can be derived, among others, from ground surface conditioning, length of the conduits from the power service to the service transformer and from the transformer to the fast charger, material cost, permits, and administration.

At the same time, the idea of mimicking a conventional gasoline station appears repeatedly in the professional forums for transport electrification. This essentially consists in providing a service to several users simultaneously. The latter would be connected through their respective points of charge by means of appropriate adaptors to the main supply line, which will play the role of the reservoir in a conventional gasoline station.

This analysis reveals that, in spite of the existence of some particular cases, the systematic design and the associated implementation of fast charging stations for EVBs are in an incipient phase and many alternatives are open for solving both problems.

The electrical architecture of the charging station offers numerous open aspects to investigate. In this sense, the low-frequency service transformer could be substituted by a solid-state transformer with the aim of increasing the efficiency and reducing the volume and weight. The notion of solid-state transformer is used here in a broad sense, which means that it can be considered as such any conversion structure implemented with switching converters that is capable to transform MV into DC.

In addition to possible implementations of the solid-state transformer, the electrical architecture of the charging station is an open research subject in itself. On the one hand, both AC and DC distribution can be implemented through different solutions regarding the

type and number of converters and in what concerns the control of the interconnected blocks. In that context, a third actor is also possible such as a hybrid solution, i.e. a power distribution system with an AC bus and a DC bus. This type of microgrids could facilitate the integration of renewable energy sources of different nature, e.g. PV and wind energies, and ensure the mutual reinforcement between the AC and DC lines by transferring energy between them when necessary.

Other relevant issues that are open to investigation are the storage energy mechanism, the power electronics interfaces and the dynamic study of the interconnection of the different conversion blocks that would constitute the fast charging station.

1.2.2 General objectives

The main goal of the project is to propose solutions to the problems caused by the absence of fast charging infrastructure for EVBs.

In particular, the project aims to perform a study of realistic scenarios to implement fast charging stations that:

- can be easily integrated in the MV or LV networks,
- allow the inclusion of energy storage,
- permit the insertion of renewable energy, and;
- are able to supply simultaneously several vehicles.

Complementarily, as a proof of concept, the research project builds a demonstrator of an ultrafast battery charger that would be supplied by a hybrid microgrid of modular type based on the following electrical structure.

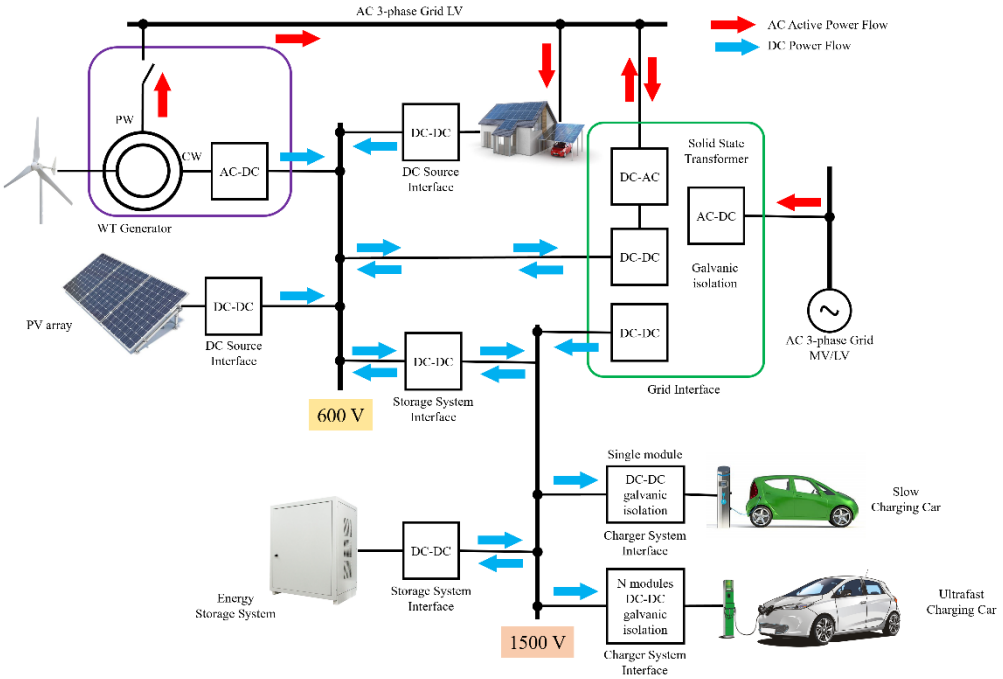

Figure 1. Block diagram of the proposed electrical architecture of the hybrid microgrid for the implementation of 1:10 demonstrator for ultrafast charging of batteries. [1]

1.3 Demonstrator

Being a scaled electrical structure based on the Figure 1, the demonstrator has a scale of 1:4 in voltage and current and 1:16 in power. That meant that the bus voltage level was scaled from 1500 VDC to 380 VDC in the power bus in charge to supply the EVBs. Currently, it has just implemented this bus in order to prove the ultrafast charging process, but considering the incorporation of renewable sources bus later in the design. So, the current implemented model of the demonstrator had the following structure:

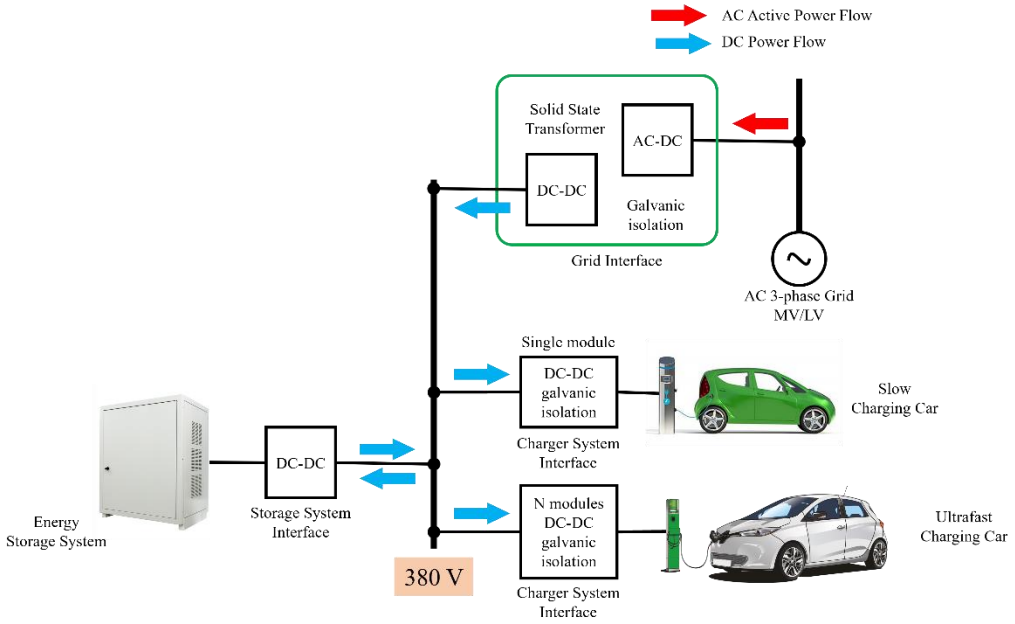

Figure 2. Block diagram of the modified electrical architecture of the demonstrator.

For the sake of clarity, the block diagram is simplified and not all the elements present in the demonstrator are shown. For example, the supercapacitors, the FPGA, and some of the converters are not shown, and we also added a battery pack for testing purposes. The following table details the full list of elements present in the demonstrator:

Table 1. Elements present in the demonstrator, with their respective voltage levels.

Element	Voltage level	Notes
SST	LV 3-phase grid to 380 VDC	LV grid is 220 VAC
Bus	380 VDC	
Battery packs	200 VDC	LFP pack that makes up the Energy Storage System
	100 VDC	LTO pack that emulates a car battery
	48 VDC	LTO pack that emulates a motorbike battery
Supercapacitors	160 VDC	5,8 F
	102 VDC	88 F
DC/DC converters	380 VDC to 100 VDC	From bus to car battery pack
	380 VDC to 48 VDC	From bus to motorbike battery pack
	380 VDC to 200 VDC	From bus to Energy Storage System battery pack
	380 VDC to 102 VDC	From bus to 88 F supercapacitor
	380 VDC to 160 VDC	From bus to 5,8 F supercapacitor

At the time of writing this report, the state of the demonstrator is the following:

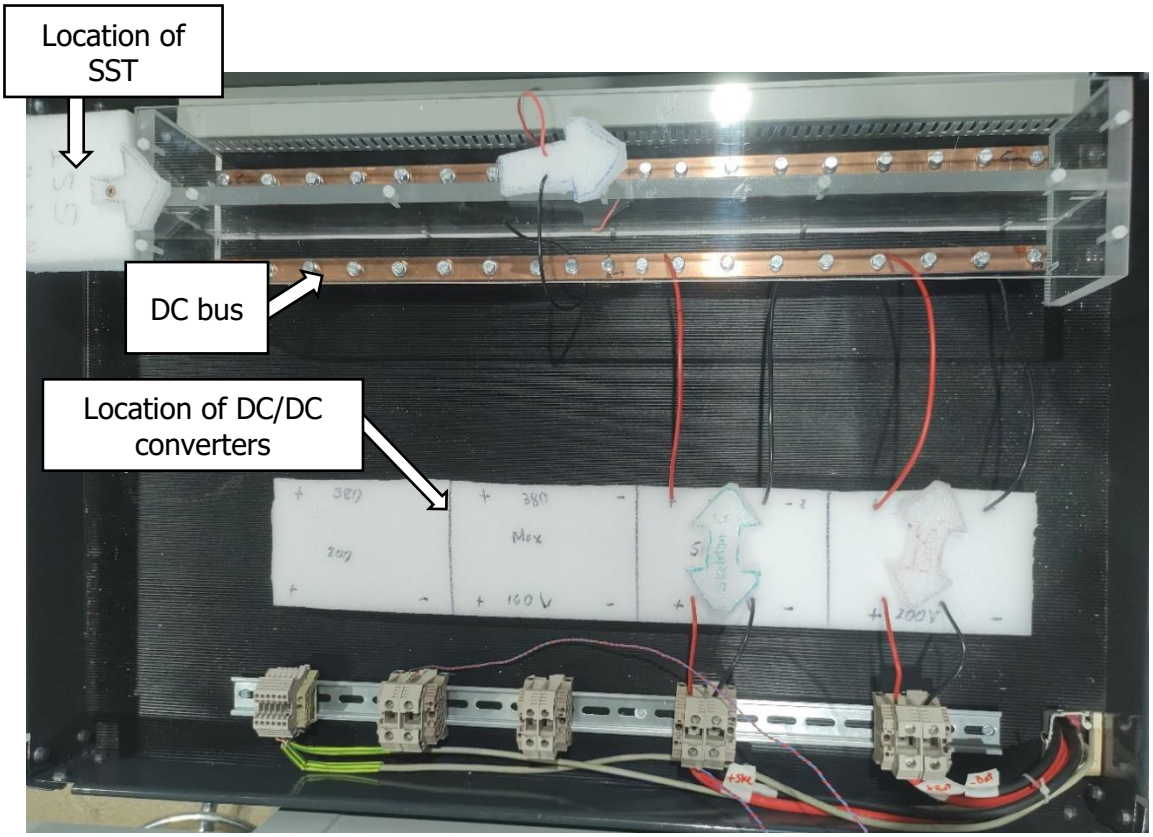

Figure 3. Top view of the Energy Storage System.

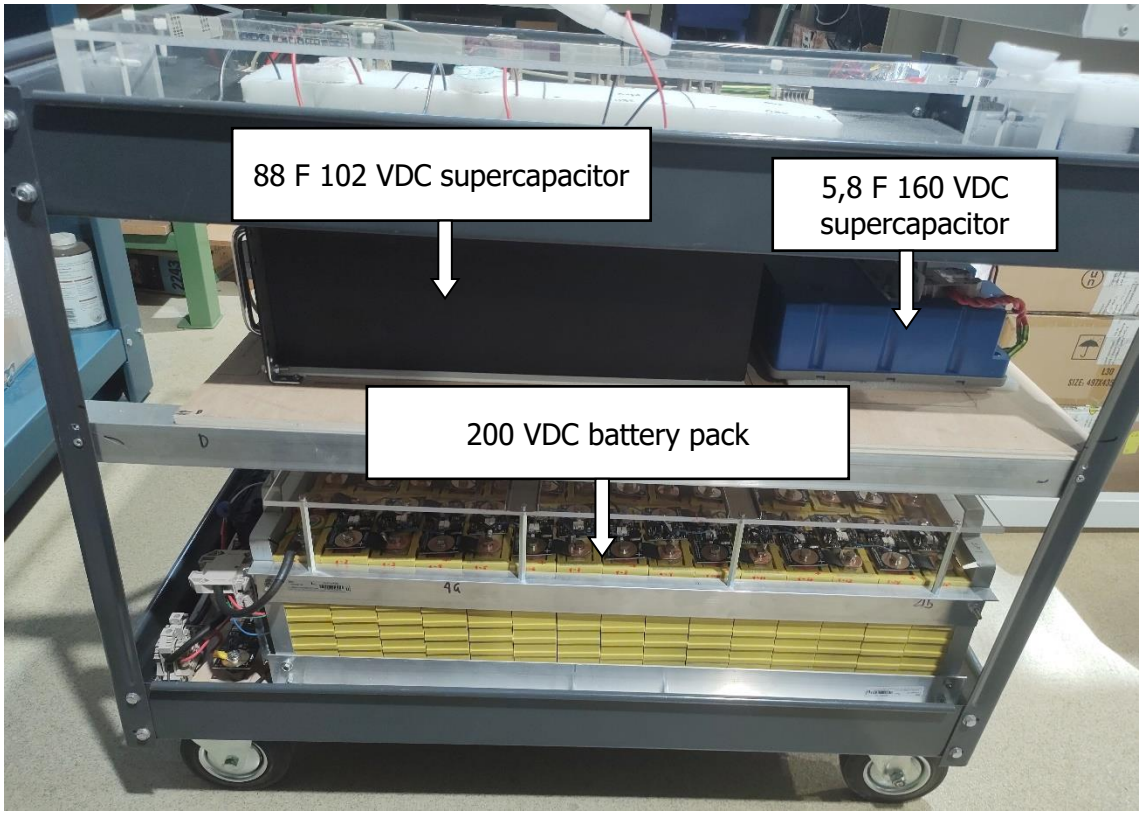

Figure 4. Side view of the Energy Storage System.

1.4 Objectives and scope

As we have seen, this demonstrator consists of several elements, which will be interconnected, that need to work simultaneously. To achieve a proper and optimal functioning, it is required to have a good coordination between all the devices, which in turn requires a platform to manage and visualize all the interactions between the devices.

This project aims to develop a program that can gather electrical–and in some cases also thermal–data from all devices in the system. Additionally, it has to be able to visualize this data on a screen and then save it to a file.

With this goal in mind, two big questions arise. Firstly, how do we establish a connection between all the devices, and secondly, how do we gather the data, display it on a screen and save it.

1.4.1 Communication

The first issue we encounter is the connection and communication between all the devices. There are several communication protocols that we could use in this project:

CAN

CAN FD

LIN

FlexRay

We considered many variables when comparing these options, but one of the deciding factors was each protocol's ability to provide flexibility and scalability to the system. All the aforementioned protocols are well known in the automotive sector, but CAN is arguably the most used one [2]. This facilitated the process of connecting and communicating all the devices.

We also took bit speed into account, and CAN provides a wide and comfortable range that goes from 125 kbit/s and up to 1 Mbit/s. Other protocols are much slower (LIN works around 20 kbit/s), and others offer speeds that are higher than what this project requires (CAN FD can go up to 5 Mbit/s, and FlexRay up to 10 Mbit/s).

Another determining factor was that some of the devices that we were going to use, such as the supercapacitor, already had an implemented CAN bus port to transmit data.

Keeping in mind these considerations, we added an FPGA board to the demonstrator to be able to collect the CAN data from all the devices and then send it to an external computer. The board used is the NI sbRIO-9627, which has a CAN port and an Ethernet port, among others.

Figure 5. FPGA NI sbRIO-9627, in which the CAN port has been marked in yellow and the Ethernet port in red [3].

1.4.2 Data management and visualization

To fulfill the second requirement mentioned before, we needed to develop a SACDA system, and for that, we had to choose a software that allowed us to have a scalable and modular program.

The software chosen was LabVIEW from National Instruments, due to the FPGA board we had chosen. Additionally, this software uses a visual programming language, which provides high modularity and scalability.

2. CAN communication

2.1 An introduction to CAN

2.1.1 Overview

CAN is a message-based protocol, originally designed for multiplex electrical wiring within automobiles to save on copper, but it can also be used in many other contexts. Messages are divided into frames, which contain an 11-bit identifier (or 29-bit for the extended format), data of up to 8 bytes and various other bits for control information and error checking.

This protocol uses a priority-based arbitration system called CSMA/CD-A. This means that for each device, the data in a frame is transmitted serially but in such a way that if more than one device transmits at the same time, the highest priority device can continue while the others back off. Frames are received by all devices, including by the transmitting device [4].

Devices connected to a CAN network are called nodes or ECUs and are able to communicate with other ECUs by broadcasting data via the CAN bus (consisting of two wires, CAN high and CAN low).

During this introduction we will also mention some of the ISO norms that specify certain parameters that have to be followed when building a CAN network.

2.1.2 OSI Model Layers

The Open Systems Interconnection (OSI) model describes seven layers that computer systems use to communicate over a network, where each layer has its own function.

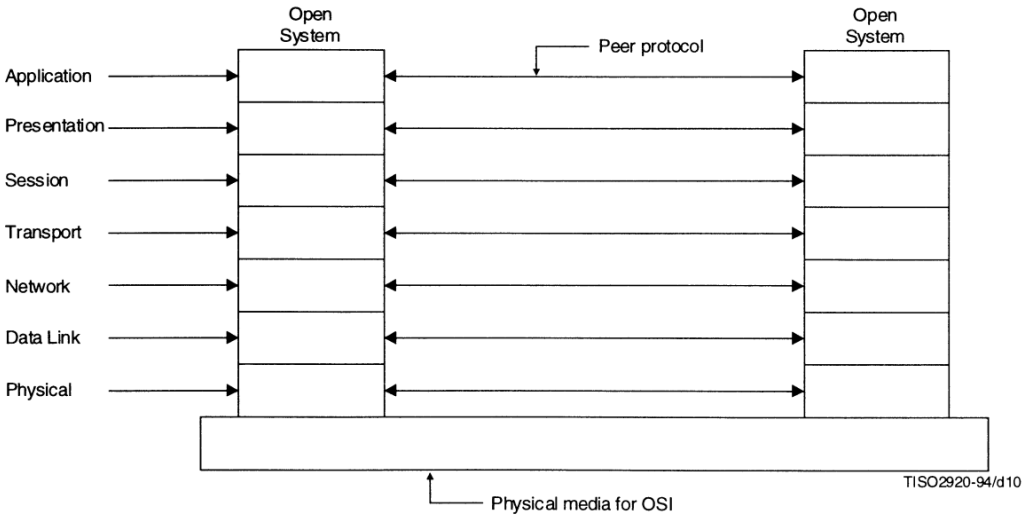

Figure 6. Visual representation of the OSI model and its layers, as well as the virtual connection between two systems.

In more technical terms, the controller area network is described by a DLL (data link layer) and a PL (physical layer). In the case of high-speed CAN, ISO 11898-1 describes the DLL, while ISO 11898-2 describes the PL. The role of CAN is often presented in the 7-layer OSI model as per the following illustration.

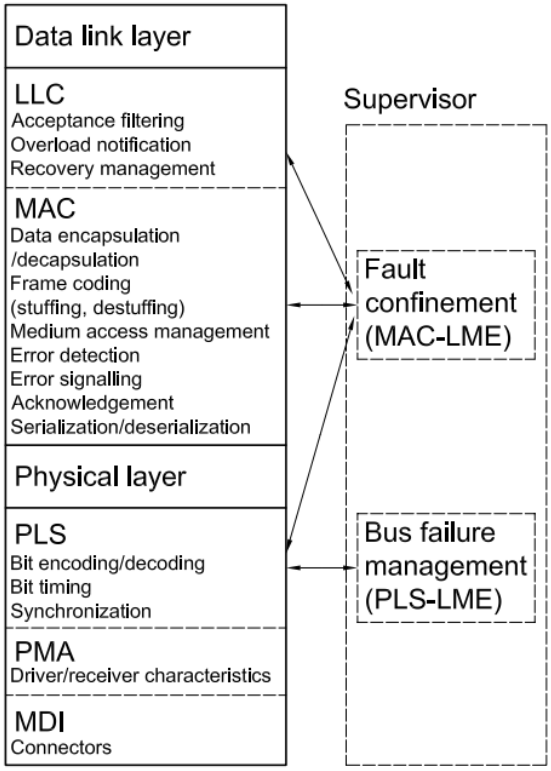

Figure 7. Detailed representation of the Data Link and Physical Layers.

2.1.2.1 Physical Layer

ISO 11898-2 dictates several parameters, including: bus voltage, baud rate, termination resistor and network and connector topology [5].

- Bus voltage:

  The bus can have one of the two logical states: recessive or dominant. For the recessive state, the nominal value of both VCAN_H and VCAN_L is 2,5 V. For the dominant state, the nominal value of VCAN_H is 3,5 V and VCAN_L is 1,5 V.

- Baud rate:

  This norm specifies the high-speed CAN protocol, which can reach transmission rates of up to 1Mbit/s.

- Termination resistor:

  Its value must be between 100 Ω and 130 Ω, with a nominal value of 120 Ω.

- Network connector topology:

  Specifies 3 parameters. Firstly, the bus length must be less than 40 m. Secondly, the cable stub length must be less than 0,3 m. And lastly, the distance between nodes must be less than 40 m.

2.1.2.2 Data Link Layer

ISO 11898-1 comprises the functions and rules related to: encapsulation/ decapsulation of the transmit/receive data, error detection and signaling, and codification of the transmitted message [6].

- Encapsulation/ decapsulation of the transmit/receive data:

  Messages are divided in frames, which have the following structure:

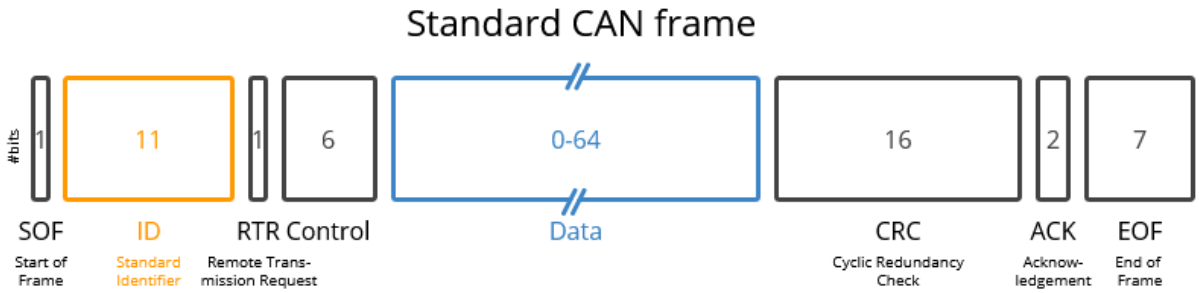

Figure 8. Standard CAN frame and its fields [7].

Table 2. List of fields in a CAN frame.

Name	Bit length	Description
Start Of Frame	1	The SOF is a 'dominant 0' to tell the other nodes that a CAN node intends to talk.
Identifier	11	The ID is the frame identifier - lower values have higher priority.
Remote Transmission Request	1	The RTR indicates whether a node sends data or requests dedicated data from another node.
Identifier Extension	1	The IDE is a 'dominant 0' for 11-bit IDs, and a 'recessive 1' for 29-bit IDs.
Reserved	1	Reserved bit.
Data Length Code	4	The DLC specifies the length of the data bytes to be transmitted (0 to 8 bytes)
Data Field	0 - 64	The Data Field contains the data bytes or payload, which includes CAN signals that can be extracted and decoded for information.
Cyclic Redundancy Check	16	The CRC is used to ensure data integrity.
Acknowledgment	2	The ACK slot indicates if the node has acknowledged and received the data correctly.
End Of Frame	7	The EOF marks the end of the CAN frame.

- Arbitration:

  The message priority is address-based. If a collision is detected between two or more nodes, the one with the highest priority (lowest address) gets access to the bus, and the other nodes are set to reception mode.

- Codification and bit stuffing:

  The SOF segments such as frame, arbitration field, control field, data field and CRC sequence shall be coded by the method of bit stuffing. Whenever a transmitter detects five consecutive bits (including stuff bits) of identical value in the bit stream to be transmitted, it shall automatically insert a complementary bit in the actual transmitted bit stream. The bit stream in a frame shall be coded according to the NRZ method. This means that the generated bit level is constant during the total bit time.

2.1.3 Benefits

Some of the benefits of using CAN are the following [8]:

- Simple & low cost:

  CAN provides an inexpensive, durable network that helps multiple CAN devices communicate with one another. An advantage to this is that ECUs can have a single CAN interface rather than analog and digital inputs to every device in the system. This decreases overall cost and weight in automobiles.

- Broadcast communication:

  Each of the devices on the network has a CAN controller chip and is therefore intelligent. All devices on the network see all transmitted messages, and each device can decide if a message is relevant or if it should be filtered. This structure allows modifications to CAN networks with minimal impact. Additional non-transmitting nodes can be added without having to modify the network.

- Priority:

  As mentioned in the previous chapter, every message has a priority. This arbitration is non-destructive and results in non-interrupted transmission of the highest priority message. This also allows networks to meet deterministic timing constraints.

- Error capabilities:

  The CAN specification includes a CRC to perform error checking on each frame's contents. Frames with errors are disregarded by all nodes, and an error frame can be transmitted to signal the error to the network. Global and local errors are differentiated by the controller, and if too many errors are detected, individual nodes can stop transmitting errors or disconnect itself from the network completely.

- Easy access:

  The CAN bus provides a 'one point-of-entry' to communicate with all other network ECUs, thus enabling central diagnostics, data logging and configuration.

- Extremely robust:

  The system is robust towards electric disturbances and electromagnetic interference, which is ideal for safety critical applications (e.g. vehicles or machines).

2.1.4 Applications

- Logging/streaming data from cars:

  Data from cars can e.g., be used to reduce fuel costs, improve driving, test prototype parts and insurance.

- Heavy duty fleet telematics:

  Data from trucks, buses, tractors etc. can be used in fleet management to reduce costs or improve safety.

- Predictive maintenance:

  Vehicles and machinery can be monitored via IoT CAN loggers in the cloud to predict and avoid breakdowns.

- Vehicle/machine blackbox:

  A CAN logger can serve as a 'blackbox' for vehicles or equipment, providing data for e.g. disputes or diagnostics.

2.2 CAN specifications of the demonstrator

Within the project, all the devices in the demonstrator that use CAN to transmit data can be organized in three main categories:

- Batteries
- Supercapacitors
- Others

The devices in each of these categories have a different way to structure data and CAN messages.

2.2.1 Batteries

The elements that follow this category are the four battery packs. They use a BMS and an additional module that gathers data from each individual cell and then sends it to the CAN bus. The BMS used is a '123\SmartBMS gen3', by 123electric, and the additional module is the '123\SmartBMS extended module', also by 123electric.

Figure 9. 123\SmartBMS gen3 [9].

Figure 10. 123\SmartBMS extended module [10].

The data collected by the BMS is divided into eight 8-byte messages, which are sent with a starting ID of "N". As this ID can be modified, we thought of a simple way to assign an ID to each battery pack to make them easy to understand and interpret: the first pack will have an ID of 0x10 (the number with this description are hexadecimals), the second one of 0x20, and so on. This also allowed us to have a scalable and flexible system.

The structure of the CAN bus messages is the following [11]:

Table 3. Structure of the CAN bus messages from the BMS.

Address	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
N+0	Total voltage		Current IN		Current OUT		Current Battery	
N+1	Energy stored				Battery capacity		SOC	-
N+2	Energy today collected				Energy today consumed			
N+3	Total energy collected				Total energy consumed			
N+4	Cell voltage MIN		Cell voltage MAX		Cell volt bypass	-	-	-
N+5	Cell voltage Lowest		Low Nr	Cell voltage Highest		High Nr	Sbyte 1	Sbyte 2
N+6	Temp Lowest	Low Nr	Temp Highest	High Nr	Min charg temp	Min discharg temp	Max temp	-
N+7	Current cell voltage		Current cell temp	Current cell Nr	Cell count	-	Isolation resist	-

Out of all these signals, only the ones highlighted in blue are necessary to get an overview on the state of the batteries, but the ones highlighted in yellow give more detailed information on each cell of the pack and general characteristics of the pack, and we will use them later as well.

2.2.2 Supercapacitor

Both supercapacitors follow the same message structure. They send two messages based on the J1939 format [12].

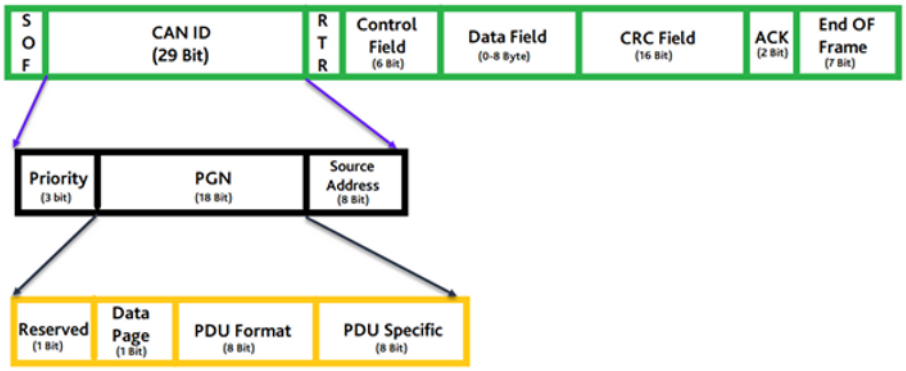

Figure 11. Structure of the J1939 message format.

Instead of the ID assignation method that we used for the BMS of the battery, the IDs of the supercapacitor are fixed [13].

Table 4. IDs used by the supercapacitors.

ID	Priority	PGN	Source Address	Frequency	Description
0x0CF091xx	3	61585	xx	10 Hz	Electrical data
0x0CF092xx	3	61586	xx	1 Hz	Thermal data

For the first supercapacitor, the Source Address will be 01, and for the second one it will be 02.

The structure of the CAN bus messages is the following:

Table 5. Structure of the CAN bus messages from the supercapacitors.

PGN	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
61585	Electrical Status Flags	Voltage Sum		Highest Voltage		Average Voltage		Life Counter
61586	Thermal Status Flags	Highest Cell Temp	Lowest Cell Temp	Life Counter	-	-		

Similar to the case with the batteries, only the signals highlighted in blue are necessary, and the ones highlighted in yellow can be used to provide more detailed information. Nonetheless, we don't have information on the current value, so we will have to get it in another way. A possible solution is installing a current sensor at the output of the converter and sending that signal to a transceiver to then send it to the CAN bus.

2.2.3 Others

The previous two categories already had a defined message structure, because the device itself had a transceiver and a protocol to create these messages using all the signals it could gather.

This category encompasses all the other devices that we needed to gather data from but did not incorporate a way to send CAN messages, e.g., the DC/DC converters, the DC bus, the SST, etc.

These devices will send only one message with an ID of "M", which follows the same pattern as the battery packs. As an example, the first device of this category will have an ID of 0x50.

Table 6. Structure of the CAN bus messages of the devices from the third category.

Address	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
M	Voltage		Current		-	-	-	-

2.3 Description of the CAN network

We built the physical CAN network and buses following the instructions given by ISO 11898-2:

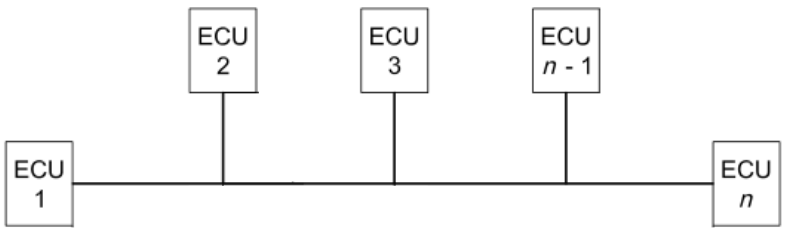

Figure 12. Wiring network topology, according to ISO 11898-2.

The bus was built with a pair of twisted wires, and at both ends of it, we added a termination resistor of 120 Ω.

All the devices have a CAN_High and CAN_Low output, but in this network, we used two kinds of connectors: a 20-pin Minifit Junior connector and a D-sub DE-9 connector. The battery packs use the 20-pin connector, and the sb-RIO and the supercapacitors use the 9-pin connector.

For the 20-pin connector, CAN_High and CAN_Low can be found in pins 17 and 7, respectively:

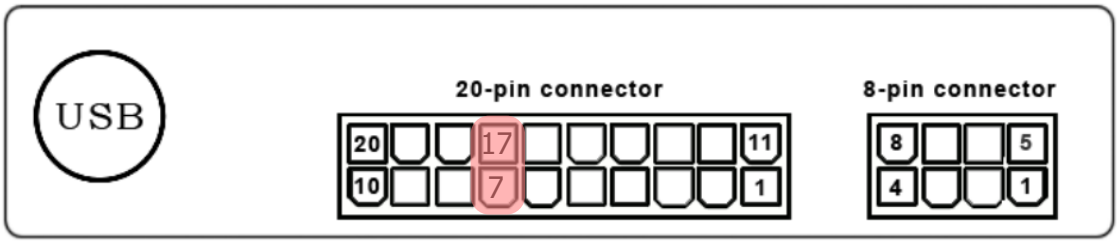

Figure 13. Diagram of the 20-pin connector and location of both CAN pins.

For the 9-pin connector, CAN_High and CAN_Low can be found in pins 7 and 2, respectively:

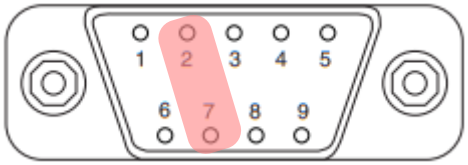

Figure 14. Diagram of the 9-pin connector and location of both CAN pins.

3. SCADA system

3.1 Design methodology

The methodology used for designing the SCADA system of the demonstrator is module-based. This methodology allows for high versatility because each module is replicable and independent of the rest. The modularity also allows for scalability, as a master module from each category mentioned in the previous chapter (batteries, supercapacitors, and others) can be copied and used for another device from the same category.

Hardware-wise, the system runs on two platforms: an FPGA board that reads the frames from the CAN bus and converts them into signals, and an external computer that permits the visualization and recording of data.

There are four types of modules: communication, conversion, visualization, and recording.

The communication and conversion modules are located on the FPGA board. The communication modules read the CAN messages from the CAN bus and send the payload–the data carried by each message frame–to the next module, which is the conversion module. These modules are responsible for transforming hexadecimal, non-scaled, and non-dimensional values (called frame values) into decimal, scaled, and dimensional values (called signal values).

Table 7. Example of a conversion of a voltage value from the BMS, from a frame value to a signal value.

Frame value	Scale (Step)	Signal value
0x15FF	0.1 V/bit	563.1 V

The visualization and recording modules can be found on the computer. The visualization modules are responsible for creating and updating the user interface with the most recent values. The recording modules only activate when the program stops running, either due to a manual stop or an error, and take the data that has been received during the program's runtime and put it into an Excel file, which then is saved in the computer.

The workflow of the SCADA system is represented in a flowchart below:

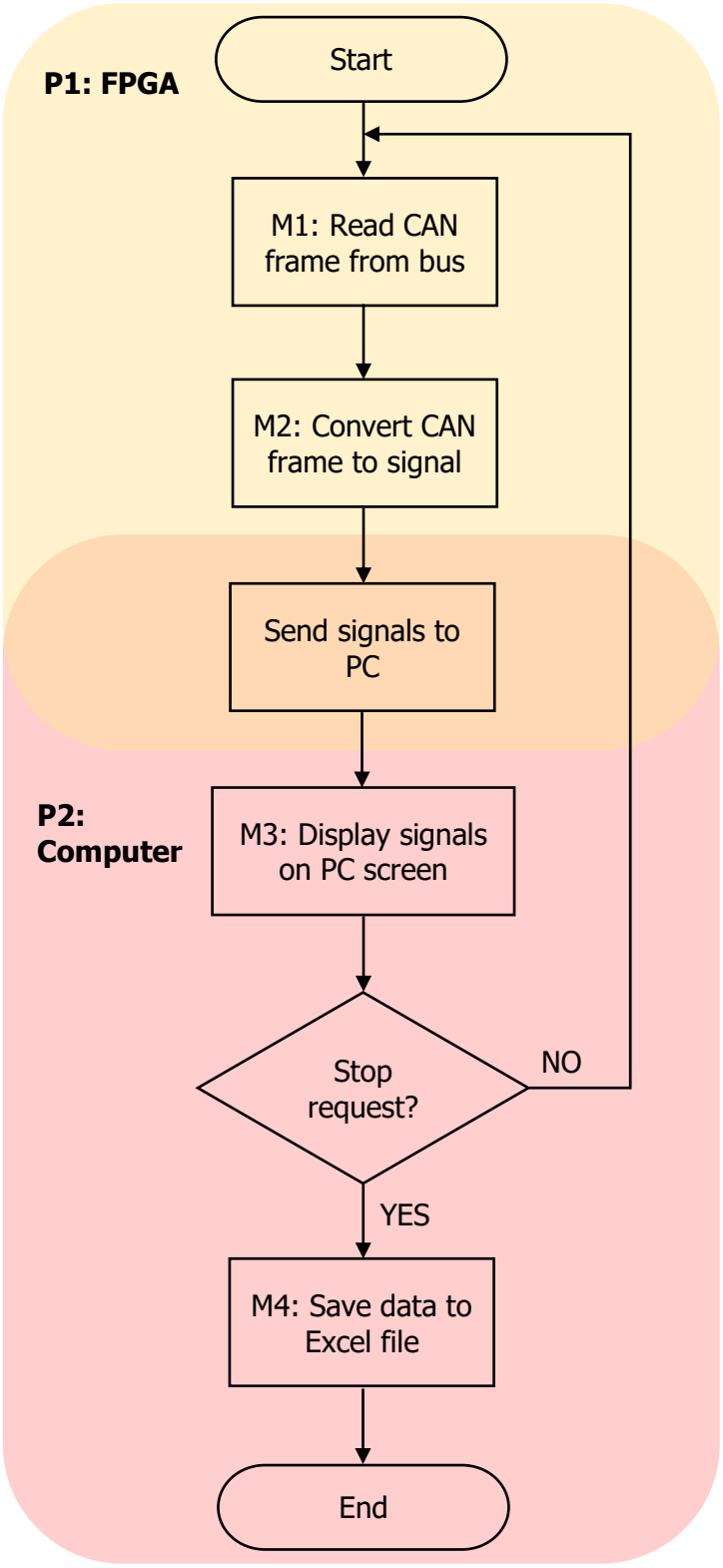

Figure 15. Flowchart that represents the workflow of the SCADA program.

Looking at this flowchart we can clearly see each of the four modules, the two hardware platforms, and how they interact with each other.

3.2 Description

The user interface of the SCADA system is comprised of two different types of panels: the main panel and the individual panels. From the main panel, the user can open an individual panel by clicking the image of a device. Then, the user can close the individual panel window and go back to the main panel (without stopping the individual program) by clicking the "Close" button.

3.2.1 Main panel

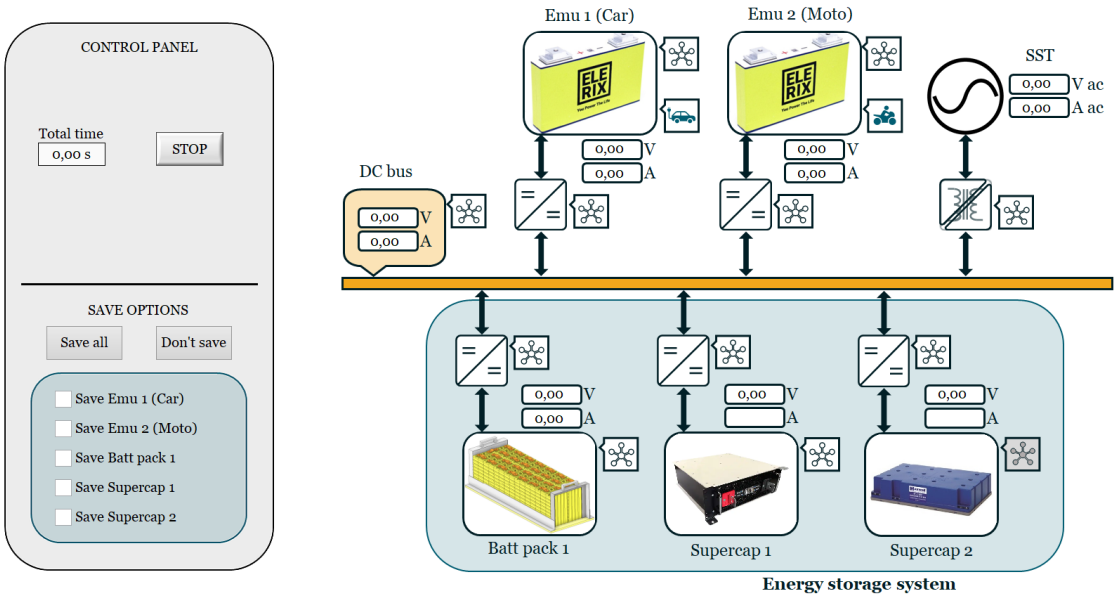

Figure 16. Screenshot of the main control panel of the SCADA program on the computer on standby.

Figure 16 shows the user interface of the SCADA system, which is divided in two parts. On the left there is the control panel, and on the right, there is the visualization panel.

The control panel contains all the user controls. These controls include a counter of the program runtime, a stop button to end the execution of the program, and several buttons to control data recording (to choose which devices will have their data recorded).

The visualization panel displays on the screen all the information necessary. The information is displayed in two formats: values and icons. Values are used to display the current and voltage values, which are measured either from the batteries or supercapacitors, or from the power converters. Icons are used to show the status of the CAN connection of each device to the network, according to Table 8.

Table 8. List and of connection status icons and their meaning.

Icon	Meaning
	The device can be connected to the CAN network but is currently disconnected.
	The device can be connected to the CAN network and is currently connected.
	The device cannot be connected to the CAN network.

3.2.2 Individual panels

The individual panels are used to show detailed data from the batteries and supercapacitors. Due to the high amount of data that we receive from each device, we created several pages that include diagrams, graphs, and counters to be able to display all the data in one panel.

Additionally, as the individual panels can be run separately, we added a small control window. From it the user can choose whether to save the data, see the program runtime and close or stop the program.

The first page is the overview. In this page we displayed general information of the device such as total and average values, configuration values, and links to three documents with useful information about the program and the device.

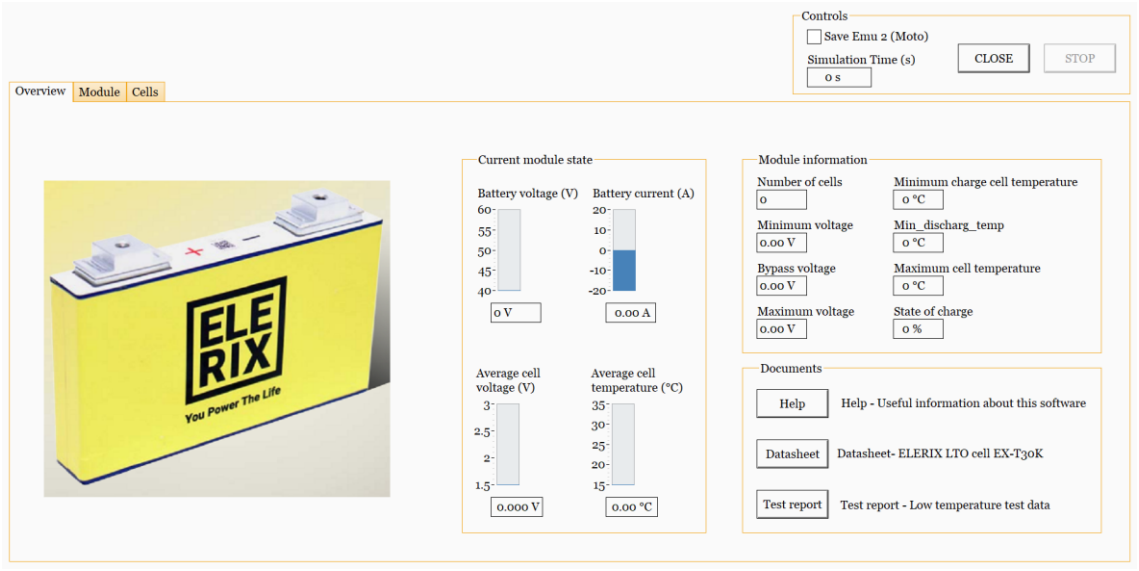

Figure 17. Screenshot of the overview page of the individual panel of the SCADA program on the computer on standby.

The second page is the module information, in which can be shown the current and voltage values for the whole pack. The data is displayed in charts, to allow for a more visual clarity. Additionally, the user can modify the axis ranges to facilitate viewing a specific range of data. On the bottom left there is a table that shows other values for the individual cells (highest, lowest, and average).

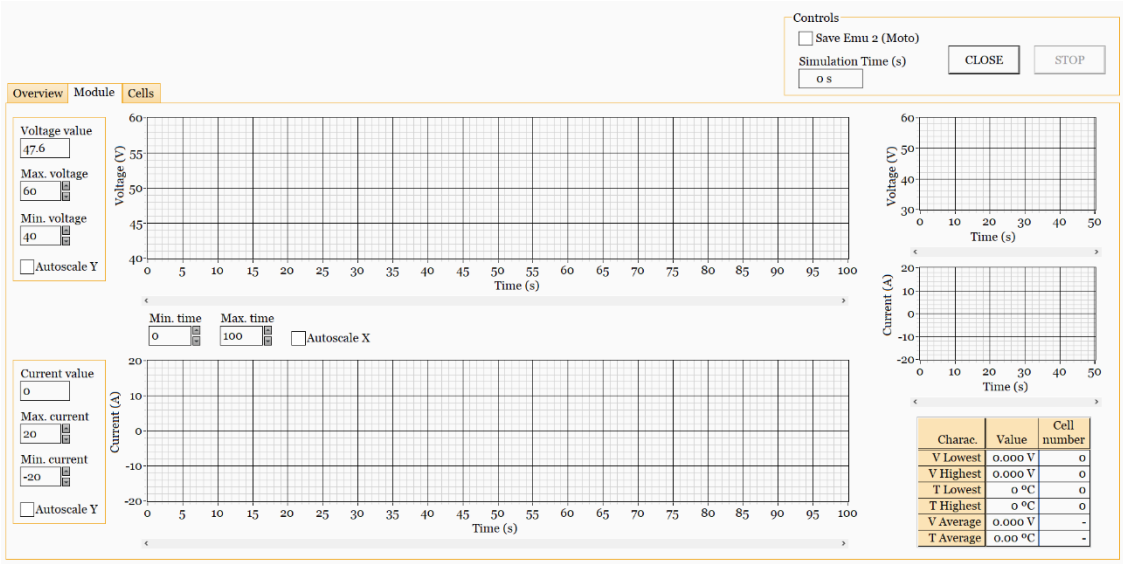

Figure 18. Screenshot of the module page of the individual panel of the SCADA program on the computer on standby.

The third page is the cell information. In this page we displayed the voltage and temperature values for each cell. We also displayed the data using a bar graph, where each bar represented a cell, and using a color coding (red, orange, and green) to visually represent the state that each cell is in (far of the nominal value, not too far of the nominal value, close to the nominal value).

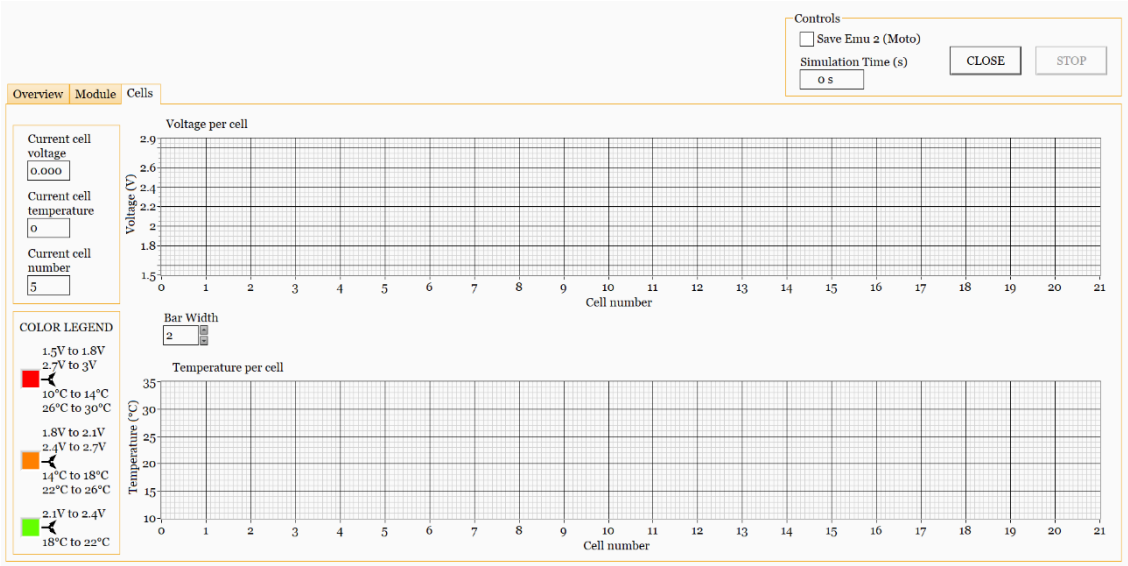

Figure 19. Screenshot of the cell information page of the individual panel of the SCADA program on the computer on standby.

3.3 Characteristics

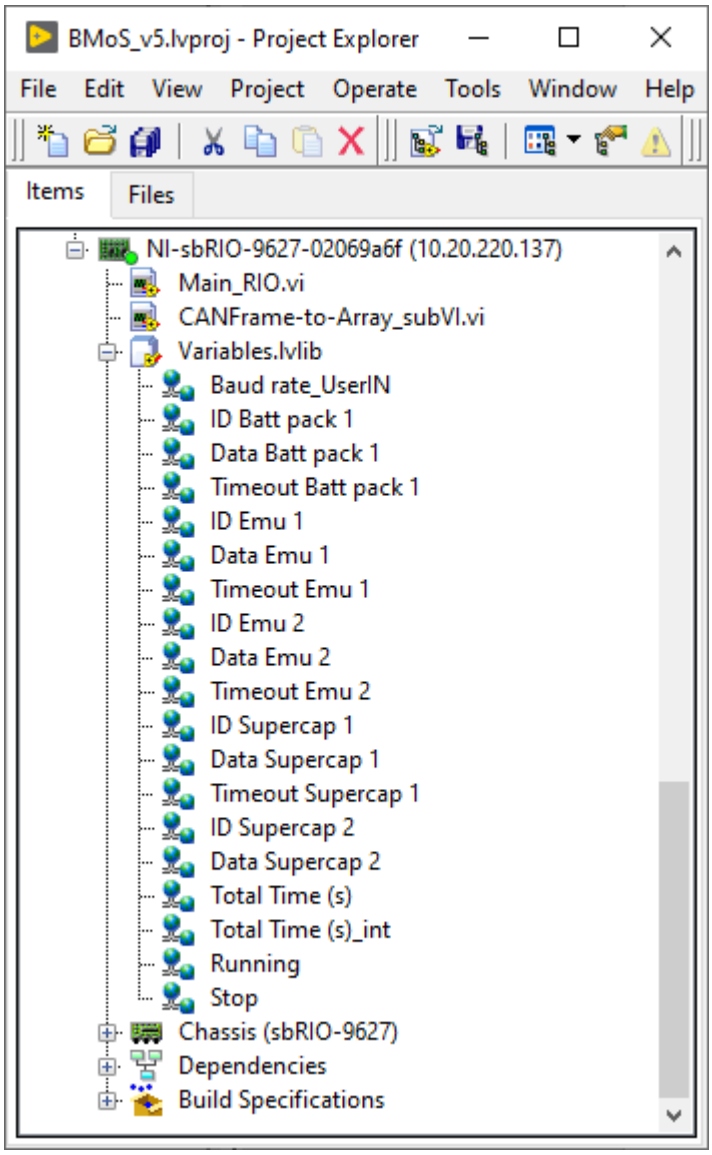

Figure 20. Screenshot of all the files present in the program, on the FPGA platform.

As we mentioned before, the program runs on two platforms. To run the program properly, the user must execute the main Vis on each platform. Figure 20 shows all the files that run in the FPGA, and also the variables used. When the user runs the program on the FPGA, the main VI ("Main_RIO.vi") is executed. This VI can call other subVIs (such as "CANFrame-to-Array_subVI.vi"), which act as functions. The variables are used to transfer data from one VI to another.

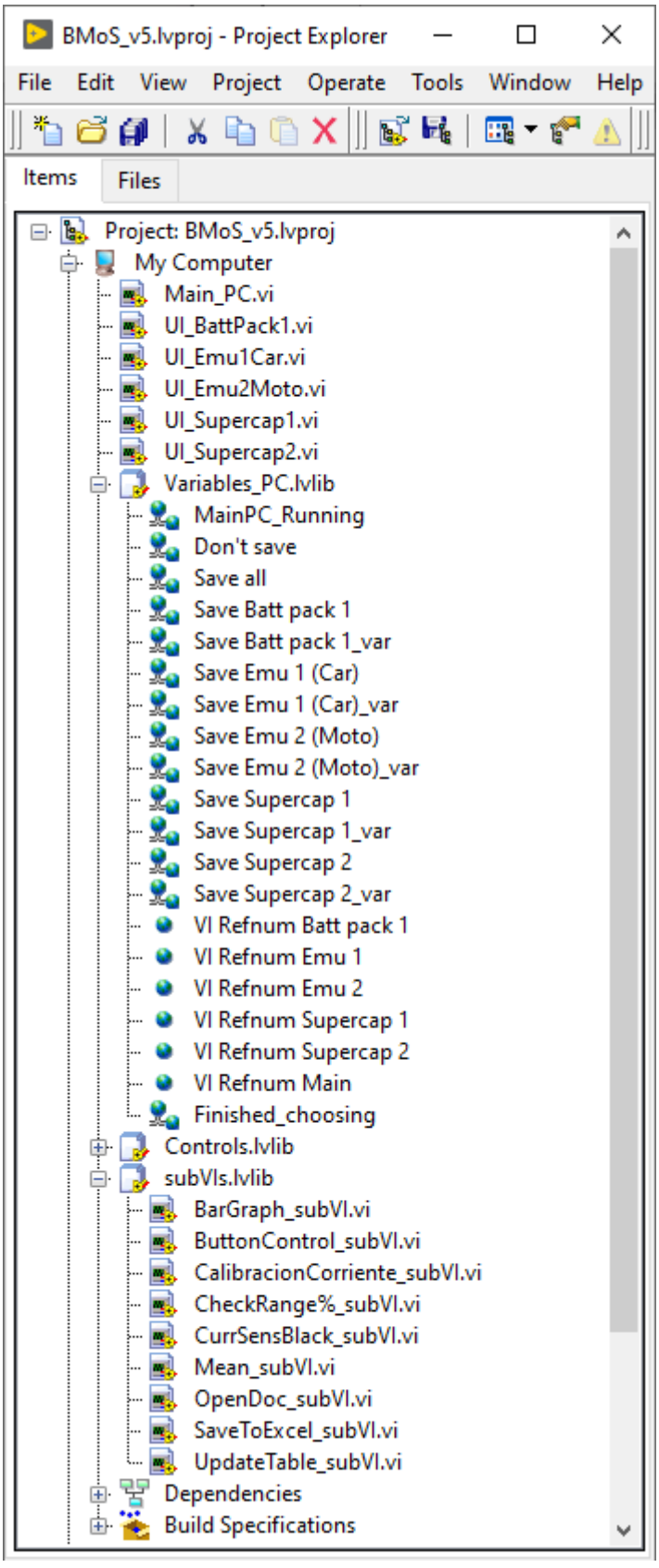

Figure 21. Screenshot of all the files present in the program, on the computer platform.

Figure 21 shows the files and variables used on the computer platform. To execute the program on the computer, the user has two options. The program can be used to display data

for all the devices or for just an individual device. For all the devices, the user must execute the main VI ("Main_PC.vi"), which will automatically execute all the individual VIs. For one device, the user must execute the individual VI with the name of that device (e.g., "UI_BattPack1.vi"). These VIs can call other subVIs as if they were functions and use variables to send data to other VIs.

This program uses timed structures to synchronize between panels, and as a result, some timing limitations are present in the program, as seen in Table 9.

Table 9. Actions that present timing limitations.

Action	Time limitation
Program reads data from the bus	Every 5 ms
Device sends data to the bus	Every 1 s, approximately
Update data on the screen, Main Panel	Every 50 ms
Update data on the screen, Individual Panel	Every 25 ms

3.4 Program structure

All the main and individual VIs are sequential, and for that we used a sequential structure. This structure divides the program into several parts. Some of the most notable ones are:

- Header:

It is present in all VIs. It serves the purpose of initializing variables and setting constant values. See Figure 22.

- Main loop:

This loop will contain the majority of the program for each VI. There are three types of loops, depending on the period that each one has —5 ms, 25 ms, and 50 ms—. See Figure 23, Figure 24, and Figure 25.

- Database selection:

This part can be found in the main VI on the FPGA, and selects which database will be used for each conversion. The database contains the frame structure and the position of each signal on a frame. See Figure 26.

- Individual VIs execution:

It is located in the main VI on the computer. It is responsible for running the individual VIs when the main VI is executed. See Figure 27.

- Error management:

It is present in all VIs. The main purpose of this part is managing the errors when the connection is lost between the computer and the FPGA. Figure 28 is an example of this part on the main VI on the computer.

- Data recording:

Can be found only on the individual VIs. It takes the data gathered during the simulation and saves it to an Excel file. See Figure 29.

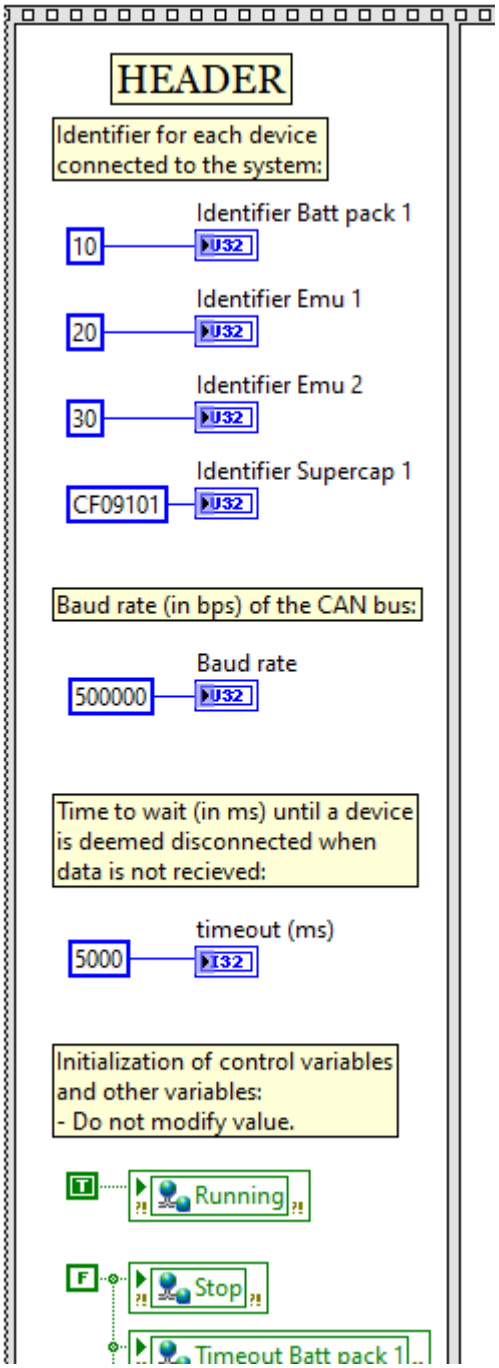

Figure 22. Header section of the main VI on the FPGA.

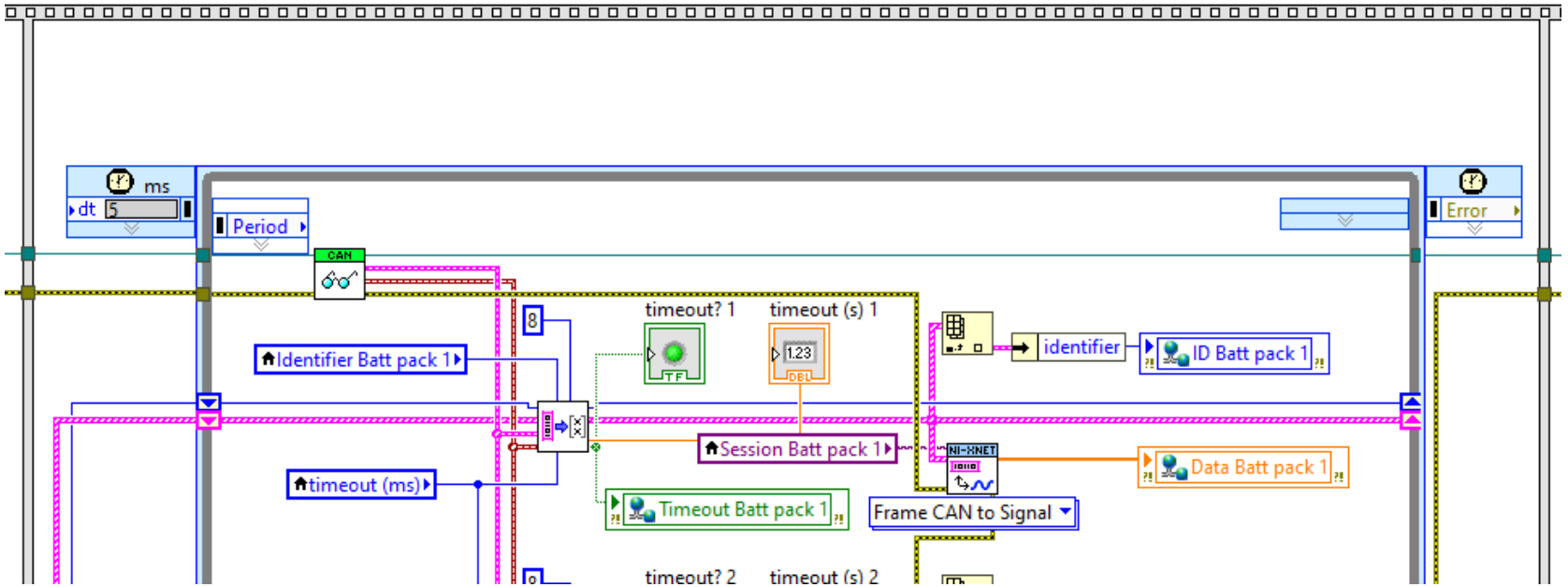

Figure 23. Top part of the main loop of 5 ms, located on the FPGA.

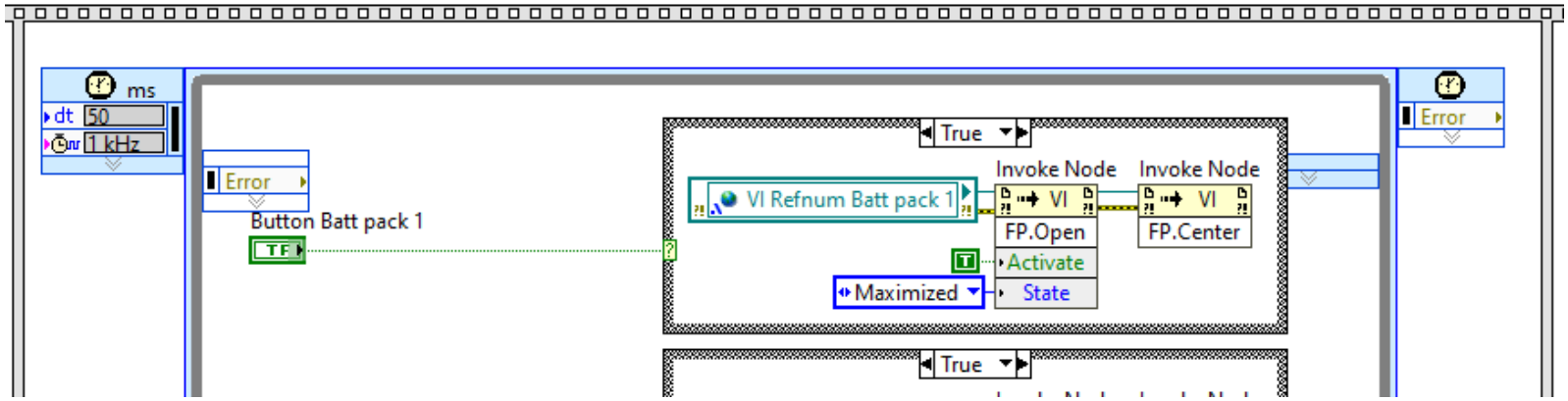

Figure 24. Top part of the main loop of 50 ms, located on the main panel on the computer.

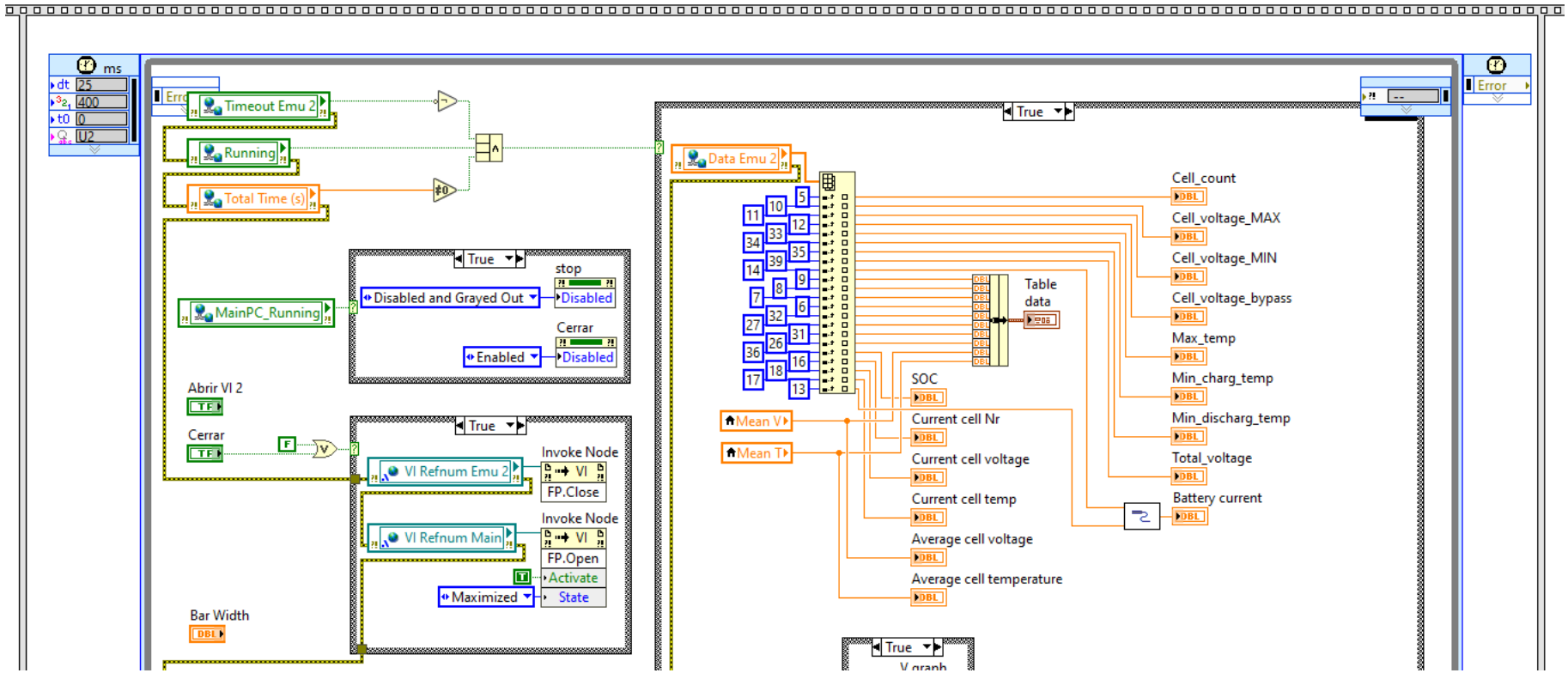

Figure 25. Top part of the main loop of 25 ms, located on one of the individual panels.

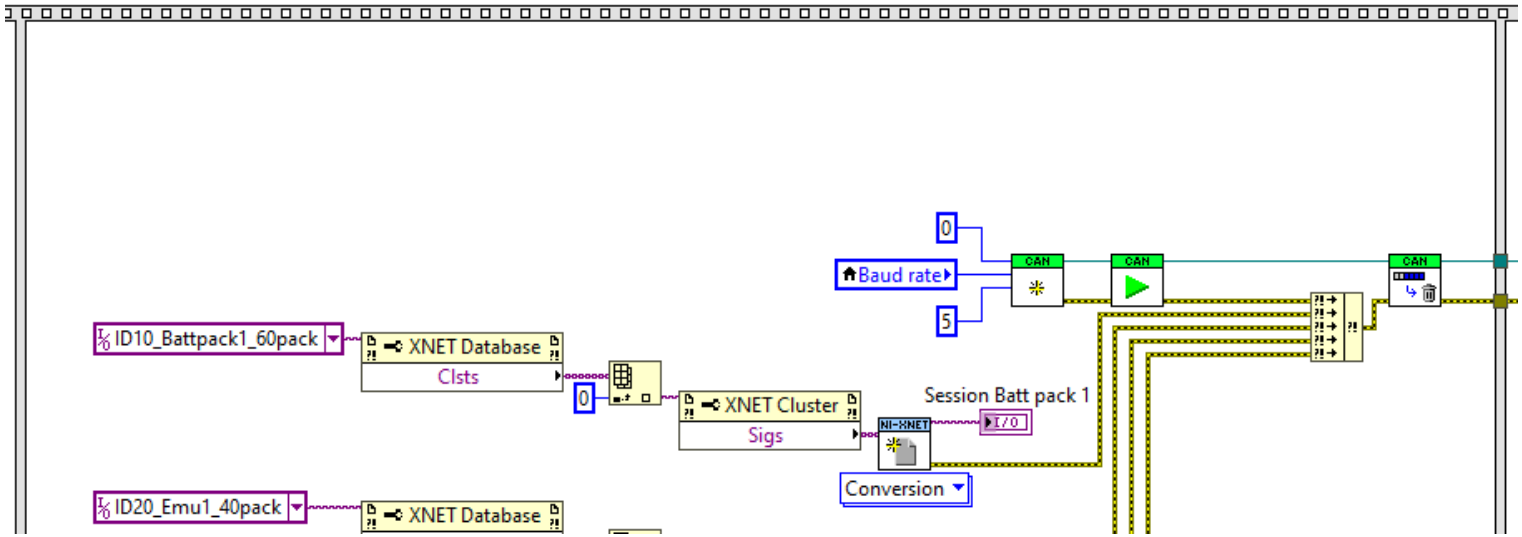

Figure 26. Database for the frame structure and CAN bus reading initialization.

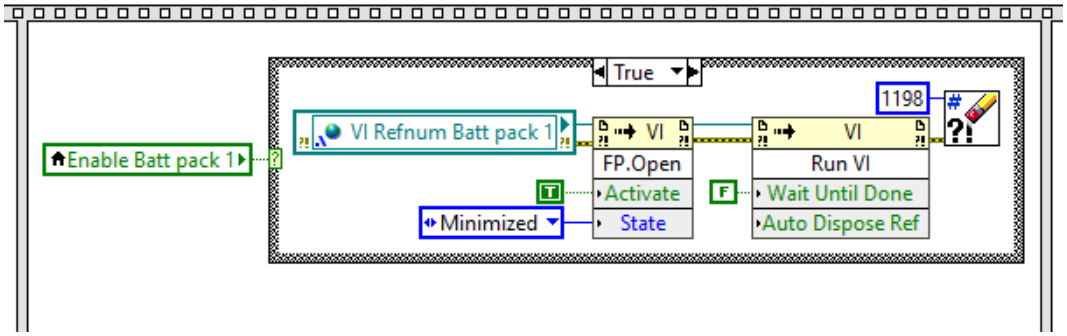

Figure 27. Sequence for executing the individual VIs, on the main VI.

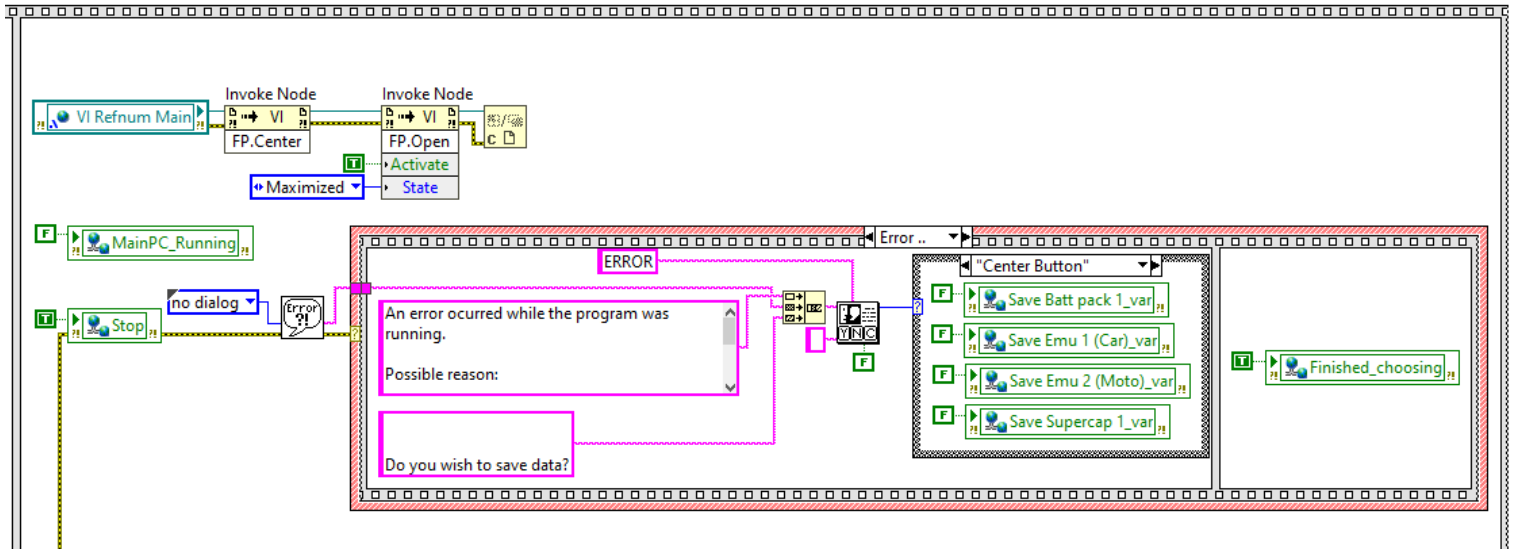

Figure 28. Error management on the main VI.

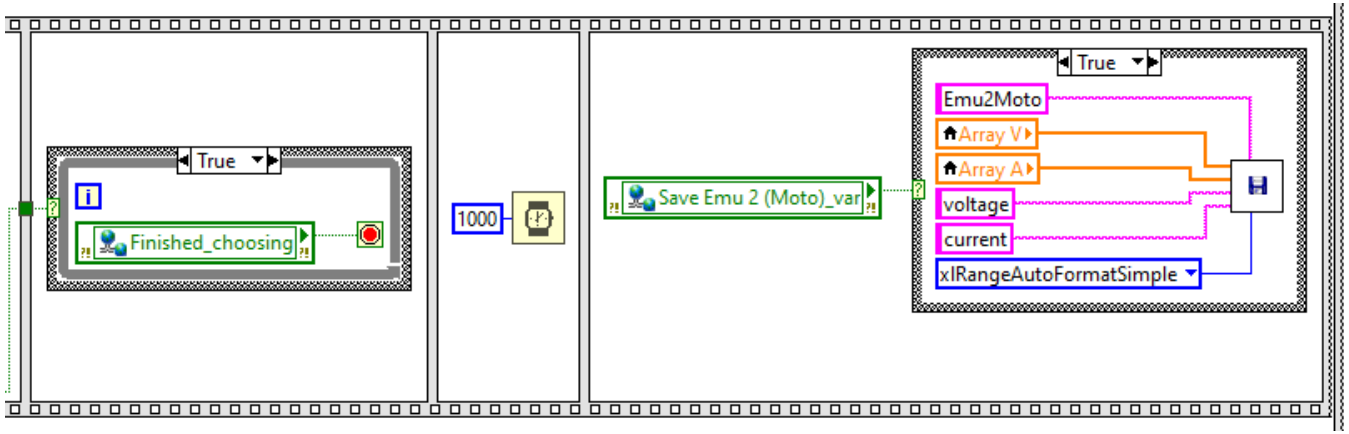

Figure 29. Sequence for saving the data, on one of the individual VIs.

4. SCADA testing

4.1 Preamble

Although the demonstrator building process has not been finished yet, the software can be tested by adapting the hardware present.

Firstly, we will test the communication between the devices, then we will do a charge and discharge cycle for each device individually, and lastly, we will do error testing.

4.2 Testing infrastructure

The elements that will be tested are the 200 VDC battery pack, the 48 VDC battery pack, and the 102 VDC 88 F supercapacitor. Additionally, in order to charge and discharge the devices, we needed a power supply, an electronic load, and a bidirectional power supply. These devices were mounted as shown in Figure 30, Figure 31, and Figure 32.

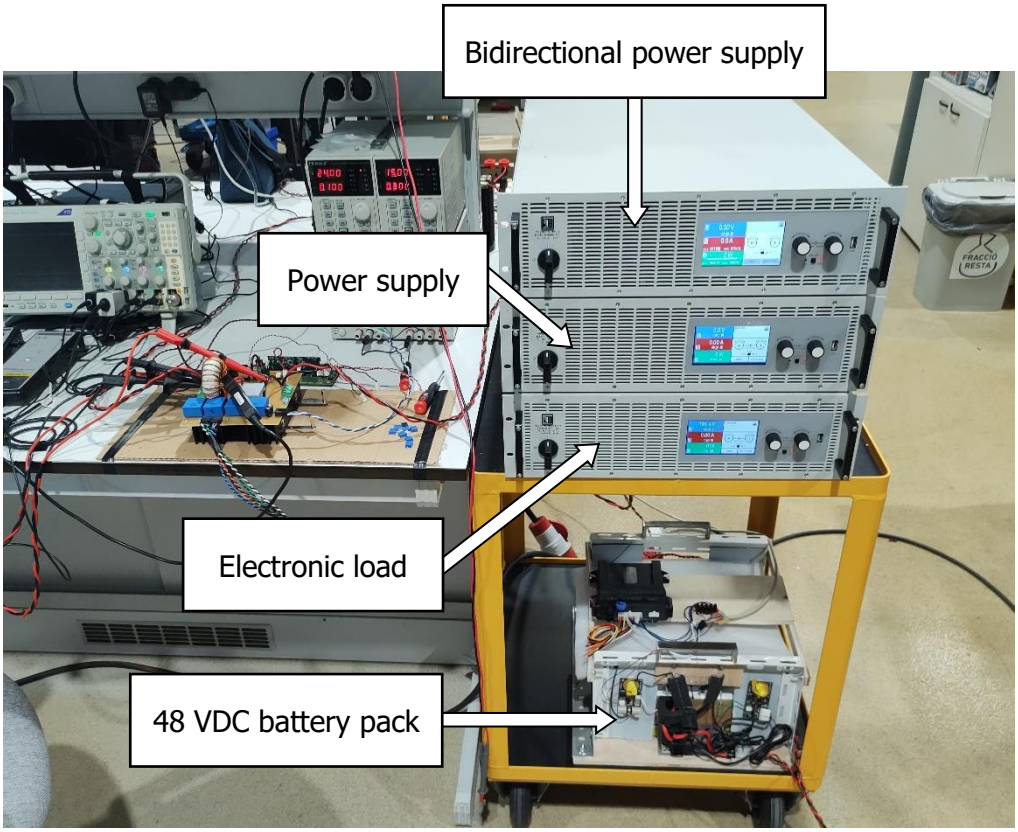

Figure 30. Testing setup. Power supplies and battery pack.

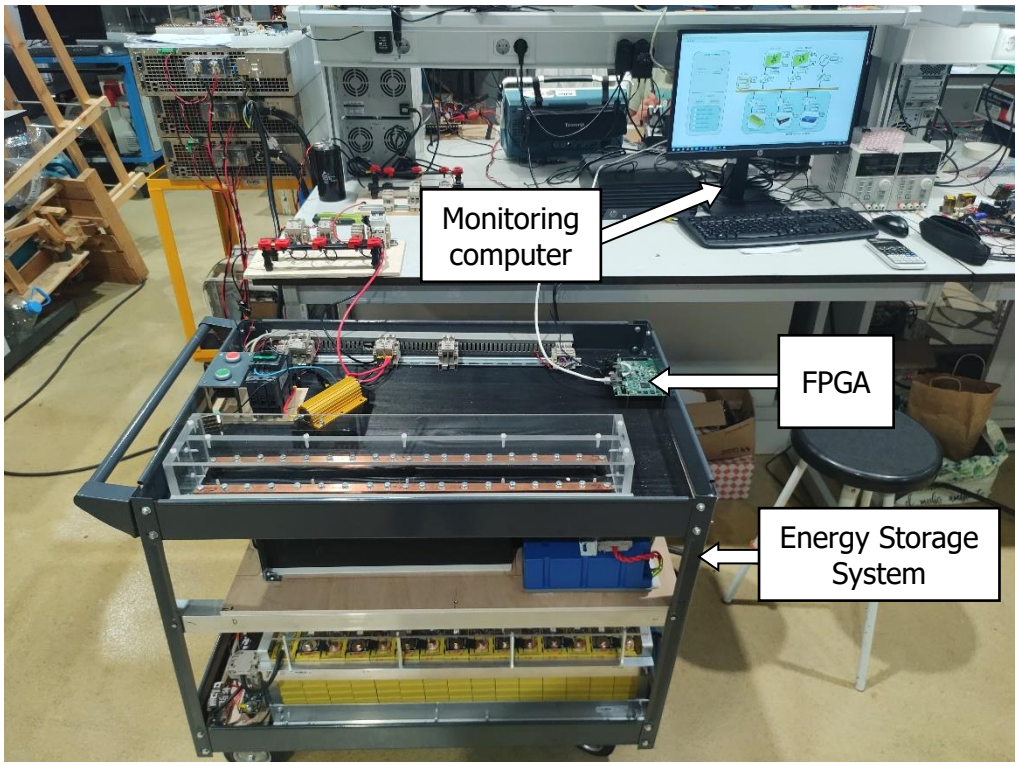

Figure 31. Testing setup. Monitoring computer, FPGA, and Energy Storage System (shown in Figure 3, Figure 4).

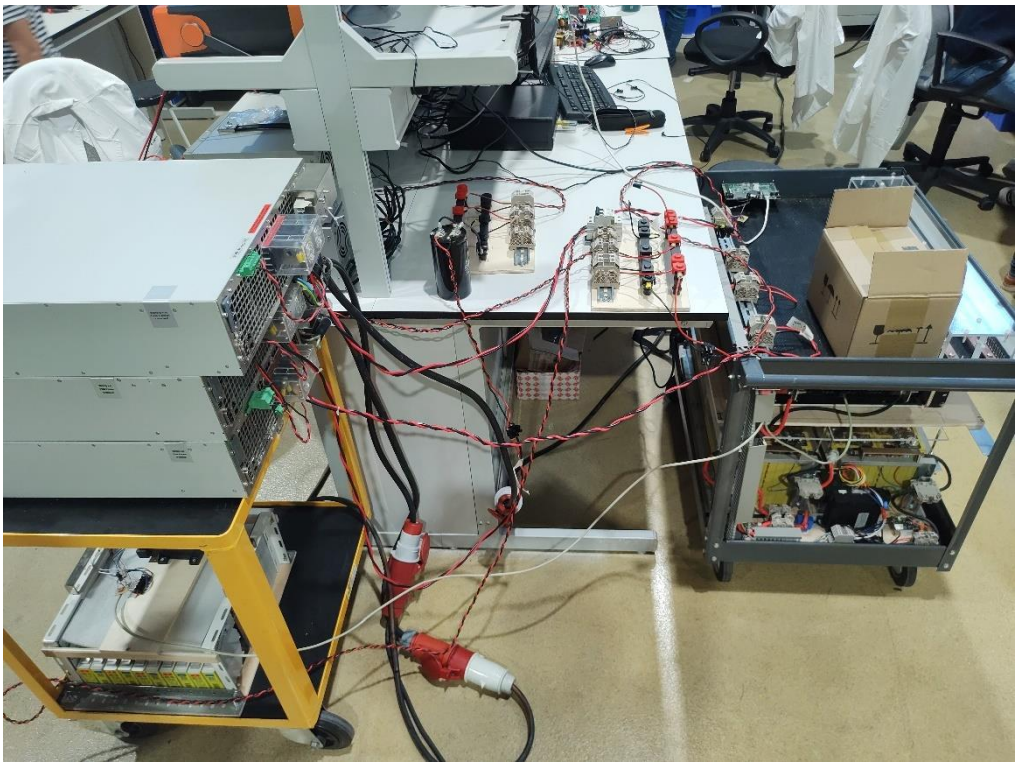

Figure 32. Testing setup. Connections between the power supplies and the Energy Storage System.

4.3 Communication test

The goal of this test is checking if the CAN communication between all the devices and the FPGA is working properly. For this, we just run the program and check if the values shown in the main panel are the same as the values measured with a multimeter. We will also check if the general information shown in the individual panels is correct.

On the main panel, we should see only three devices connected–two battery packs and one supercapacitor. For voltage values, the two battery packs should be charged at their nominal values. One battery pack should be at around 48 V, and the other one at around 200 V. For security, the supercapacitor should be discharged while it is not working (at around 0 V). For current values, the two battery packs should be at 0 A, because the power supplies are disconnected. The supercapacitor is not able to measure current, so there should not be a measurement.

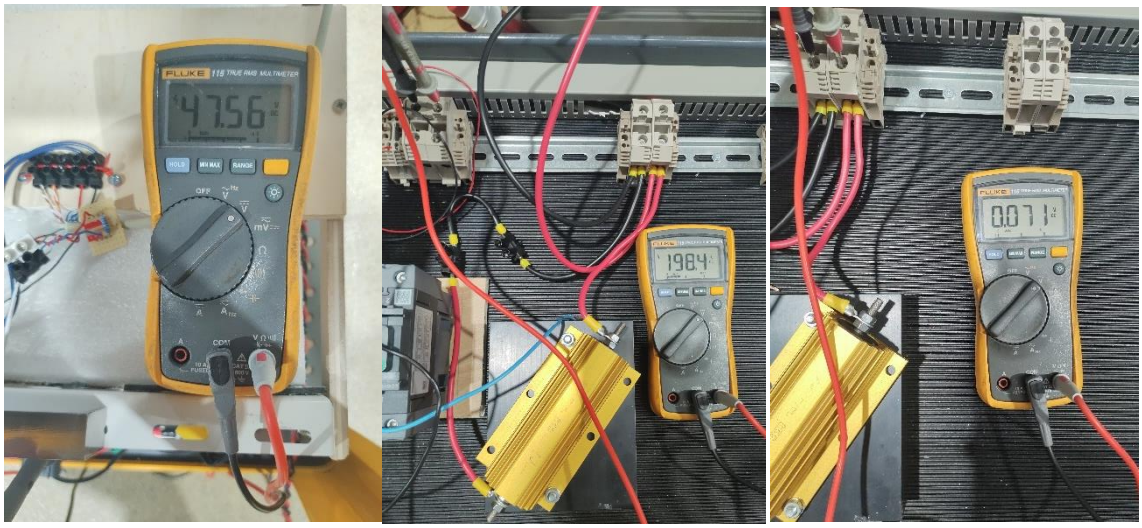

Figure 33. From left to right, voltage values of the 48 VDC battery pack, the 200 VDC battery pack, and the supercapacitor.

¡Error! No se encuentra el origen de la referencia. Figure 34 shows that only the three devices that we predicted have a working CAN communication, indicated by the orange icon next to the image of each device. All the other devices have a white icon, which indicates that there is no CAN communication established. The results obtained in Figure 33 verify the voltage values from Figure 34, which are very similar for the two battery packs. For the supercapacitor, we should take into account that the minimum value that it is able to send to the CAN bus is of 1 V, so the value shown in Figure 34 is correct.

For the individual panels, as all of them are created from the same pattern, we only test one of them. Figure 35 is the result obtained from the individual panel of the 48 VDC battery pack. The values in the "Current module state" section are the same as the ones in the main panel, and we can verify with the datasheet that the values displayed in the "Module information" section are correct.

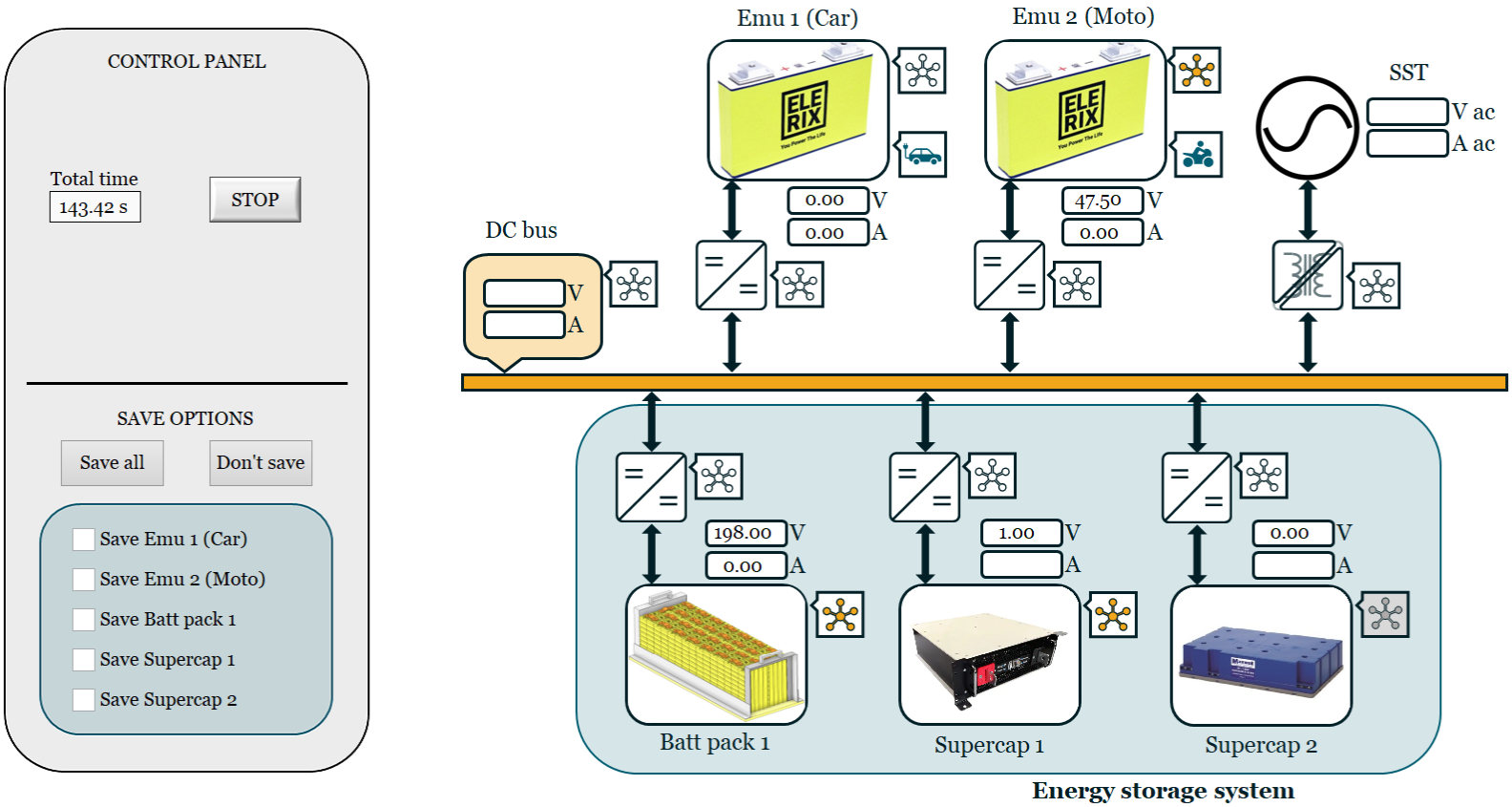

Figure 34. Test result of the communication test for the main panel.

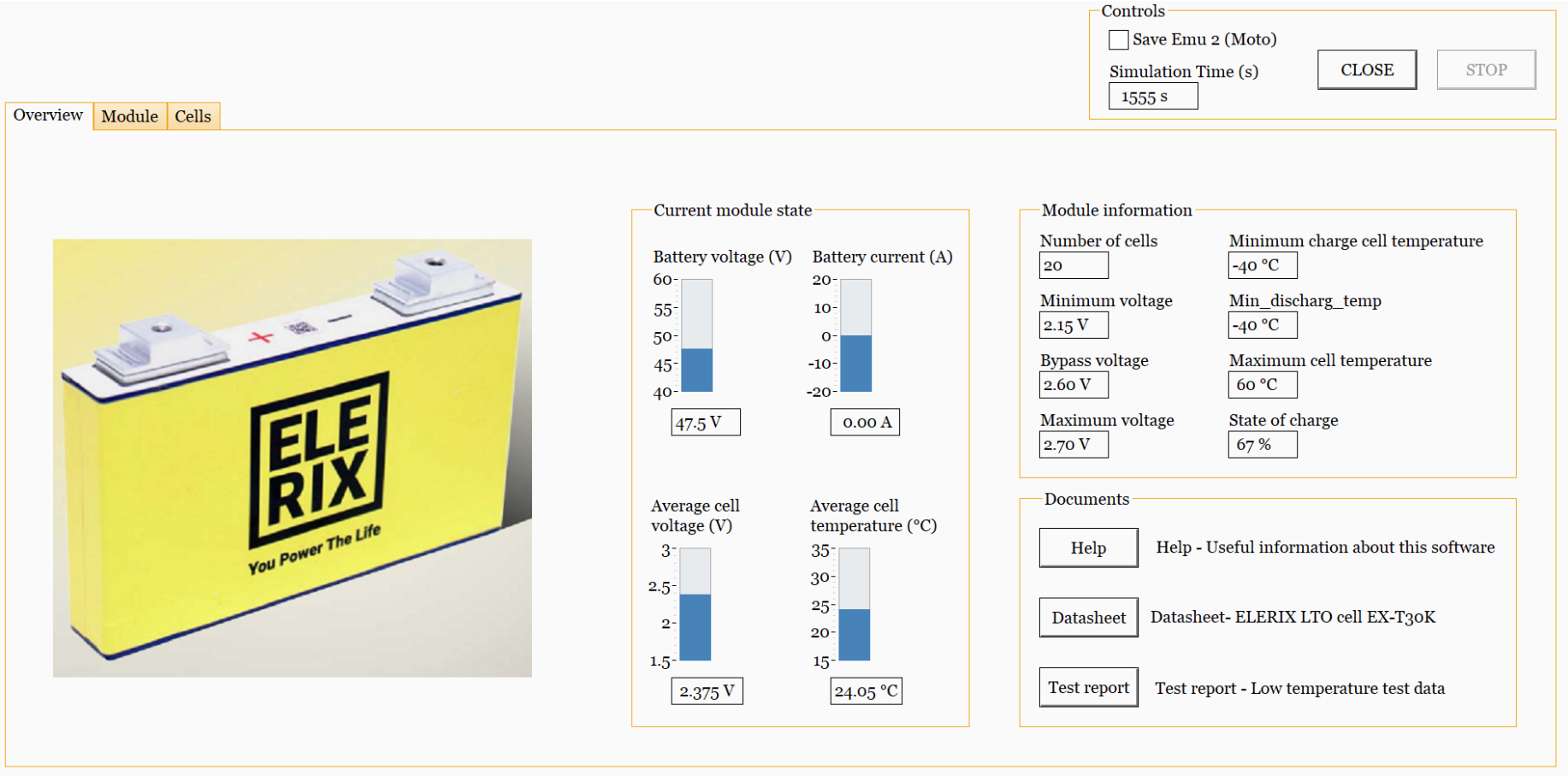

Figure 35. Test result of the communication test for an individual panel.

4.4 Charge/discharge test

This test aims to check the program's ability to gather and save the data received. For this, we will do one charge and discharge cycle for each of the three devices. We will also check whether the two other pages of the individual panels–module and cells–are working properly.

4.4.1 Supercapacitor

The easiest way to verify the results is with the supercapacitor. If we charge it, we can experimentally obtain the curve defined by (1). The discharging curve can be obtained with (2). After that we can use the approximation (3) to calculate the theorical time required to charge or discharge the supercapacitor and check if it corresponds with the simulation runtime.

$$V_{supercap}(t) = V_{PS}(1 - e^{-\frac{t}{RC}}) \quad (1)$$

$$V_{supercap}(t) = V_{PS}(e^{-\frac{t}{RC}}) \quad (2)$$

$$t_{sim} \approx 5\tau \quad (3)$$

where $V_{supercap}(t)$ is the voltage value in relation to time, V_{PS} is the voltage of the power supply, and t_{sim} is the approximate time required for $V_{supercap}$ to reach V_{PS}.

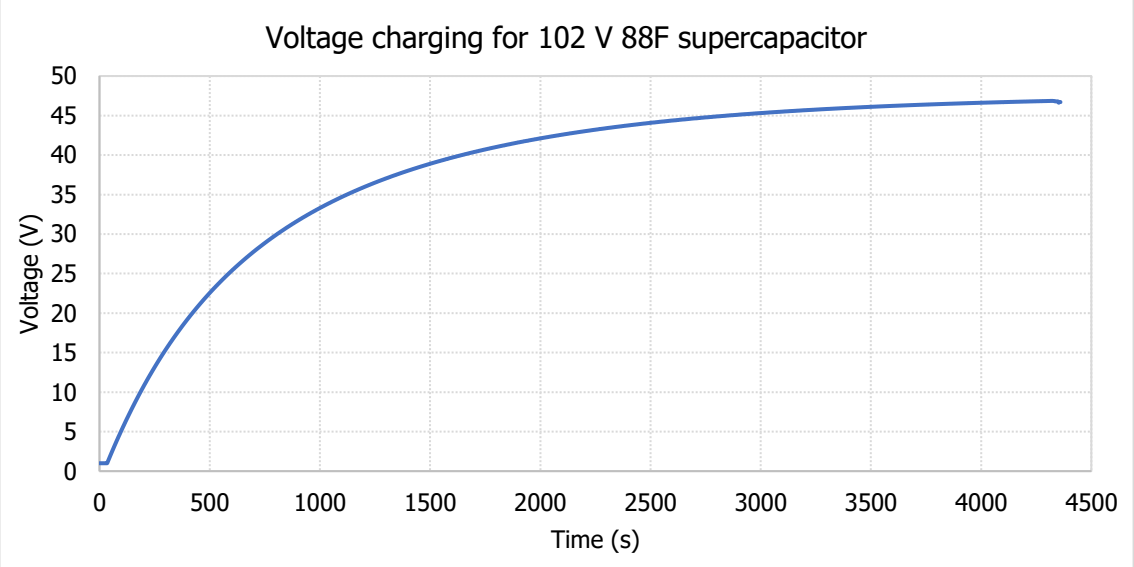

Figure 36. Experimental voltage charging graph for the supercapacitor.

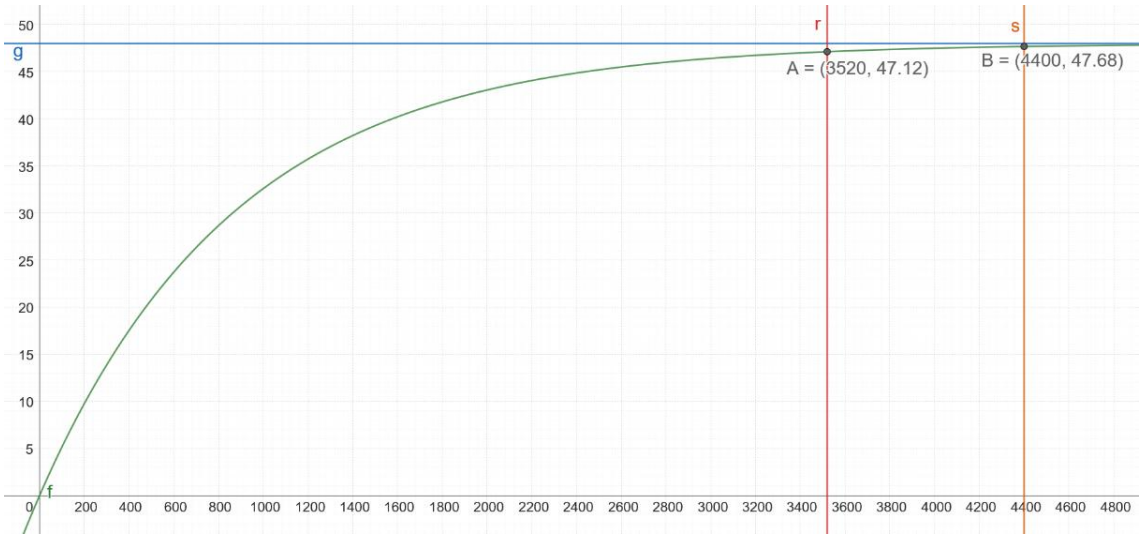

Figure 37. Theoretical voltage charging graph for the supercapacitor.

Using a voltage level of 48 V in the power supply and the data gathered with the program, we obtain Figure 36. The experimental data show that $V_{supercap}$ takes around 4300 s to reach 98 % of the final value. We can compare this to the theoretical values, shown in Figure 37. Lines r and s indicate when $t = 4\tau$ and 5τ, respectively. Checking the experimental data, we can see that function f takes between 3500 s and 4400 s to reach 98 % and 99 % of the final value.

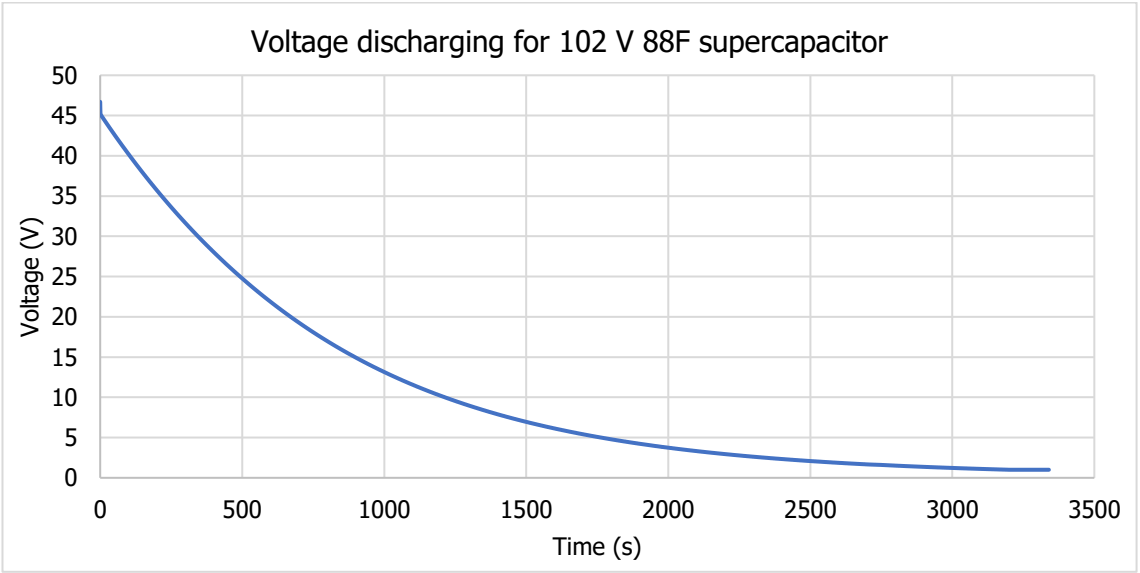

Figure 38. Experimental voltage discharging graph for the supercapacitor.

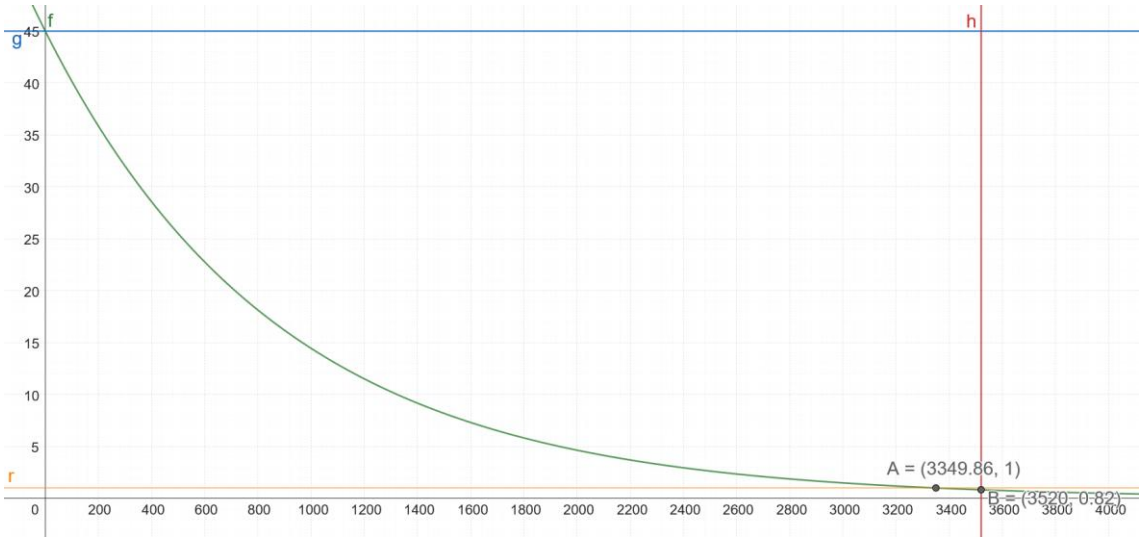

Figure 39. Theoretical voltage discharging graph for the supercapacitor.

Similar to the case with the charging values, we obtain the discharging curve presented in Figure 38. From it we obtain that it takes around 3200 s for $V_{supercap}$ to reach 1V, which as mentioned before is the minimum value that the supercapacitor can send to the bus. If we take a look at the theoretical values, shown in Figure 39, we can see that the function takes around 3350 s to reach 1 V.

Additionally, during the test, we verified that the module page of the individual panel is working properly:

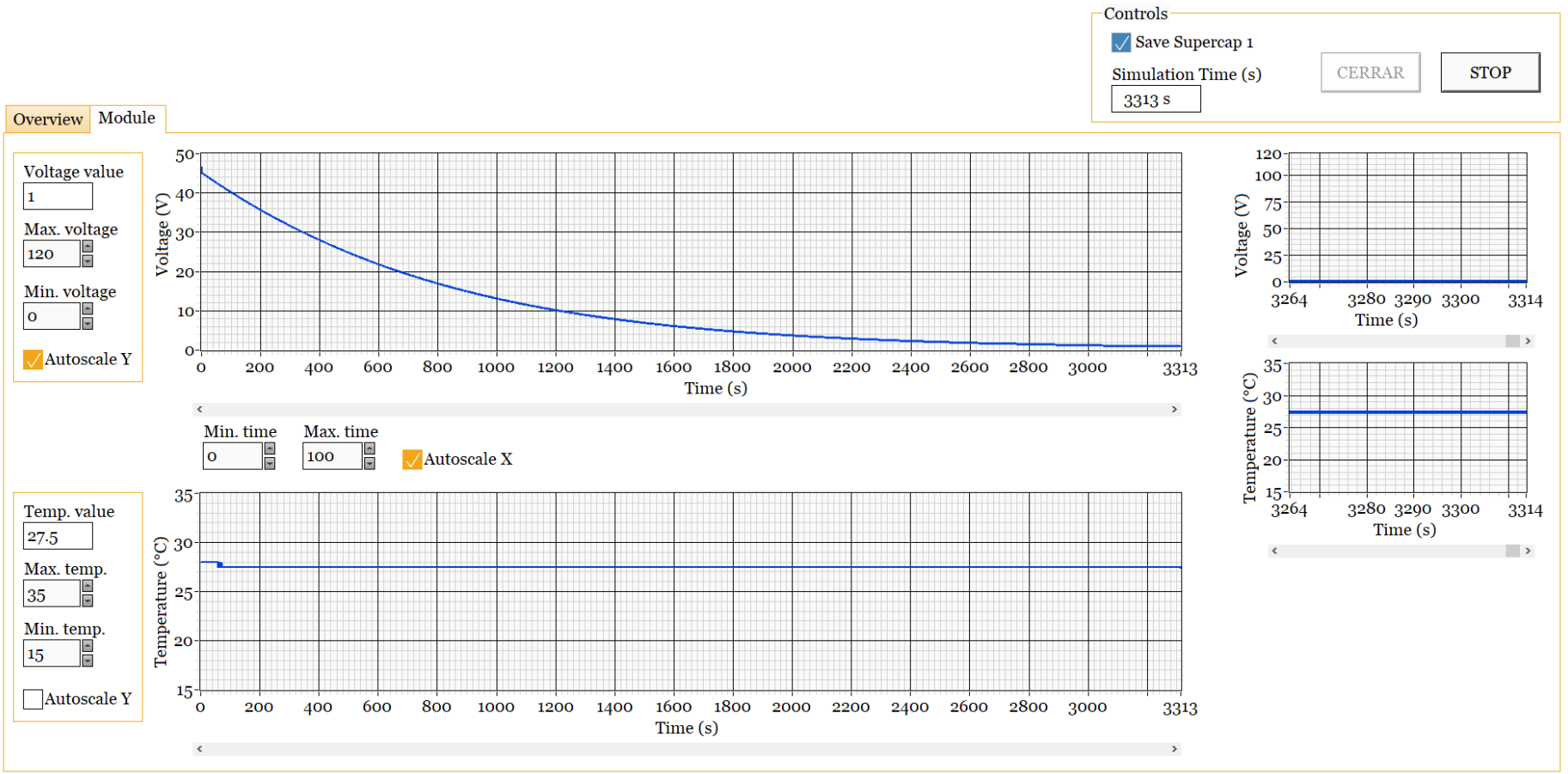

Figure 40. Screenshot of the module page during the discharge test of the supercapacitor.

4.4.2 Battery packs

For the two batteries, we gathered the voltage and current values during the discharge – charge tests. For the 48 V battery pack, the battery was discharging to a constant current of 7,5 A until reach the minimum value of 42,5 V, and charged with the standard constant current – constant voltage protocol with an initial current of 15 A.

When performing this test, we noticed that the current sensors did not read the same current values as the power supply and that they should be calibrated in order to display the correct current value, therefore we had to carry out a calibration test.

For this calibration test, we noted the current values from both the power supply and the current sensor on an Excel sheet:

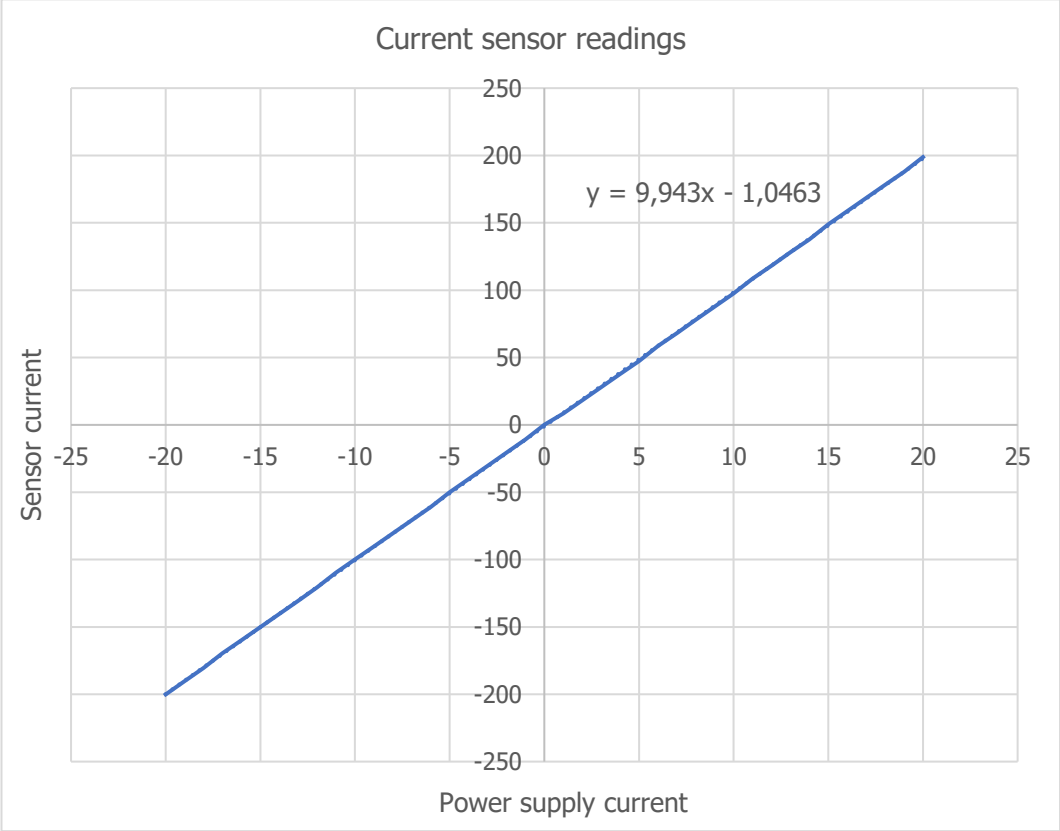

Figure 41. Current sensor readings before calibration.

We can see that the values are almost directly proportional with a proportionality constant of 10, but we are looking for a proportionality with a constant of 1. After applying the constant of 9,943 and the offset of 1,046 to the current sensor values, we obtain Figure 42.

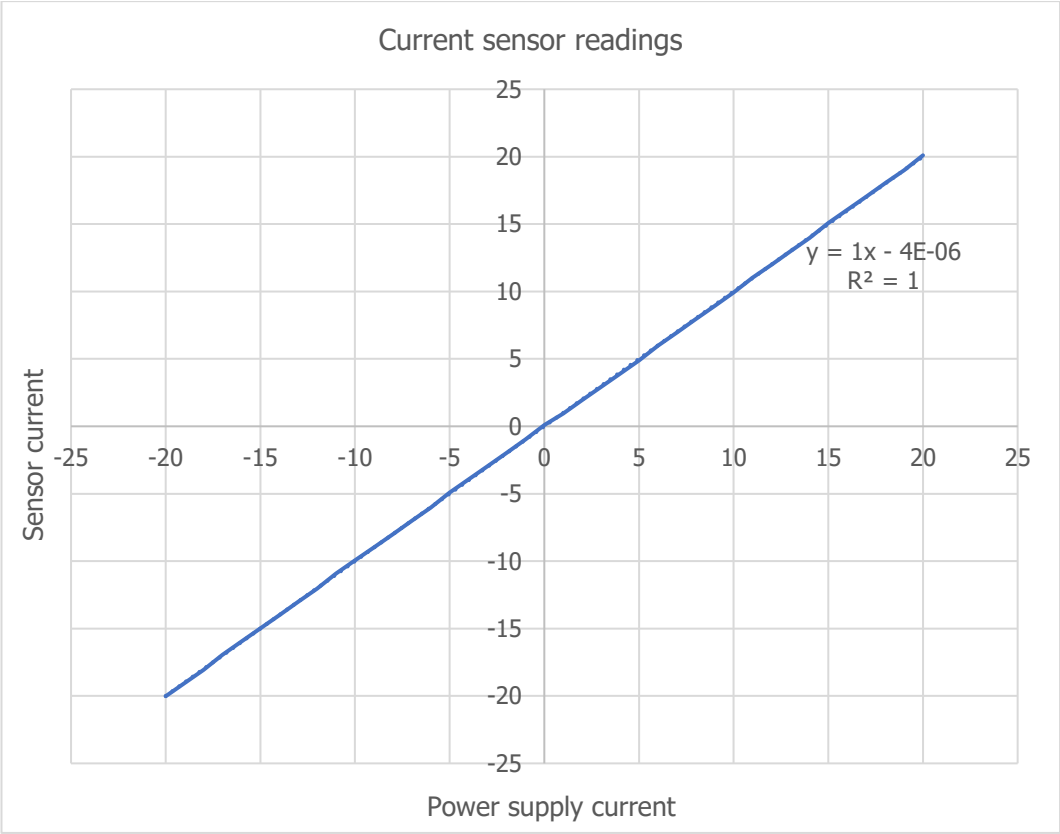

Figure 42. Current sensor readings after calibration.

After the calibration test, we can begin testing with the batteries. The results for the tests are represented in graphs:

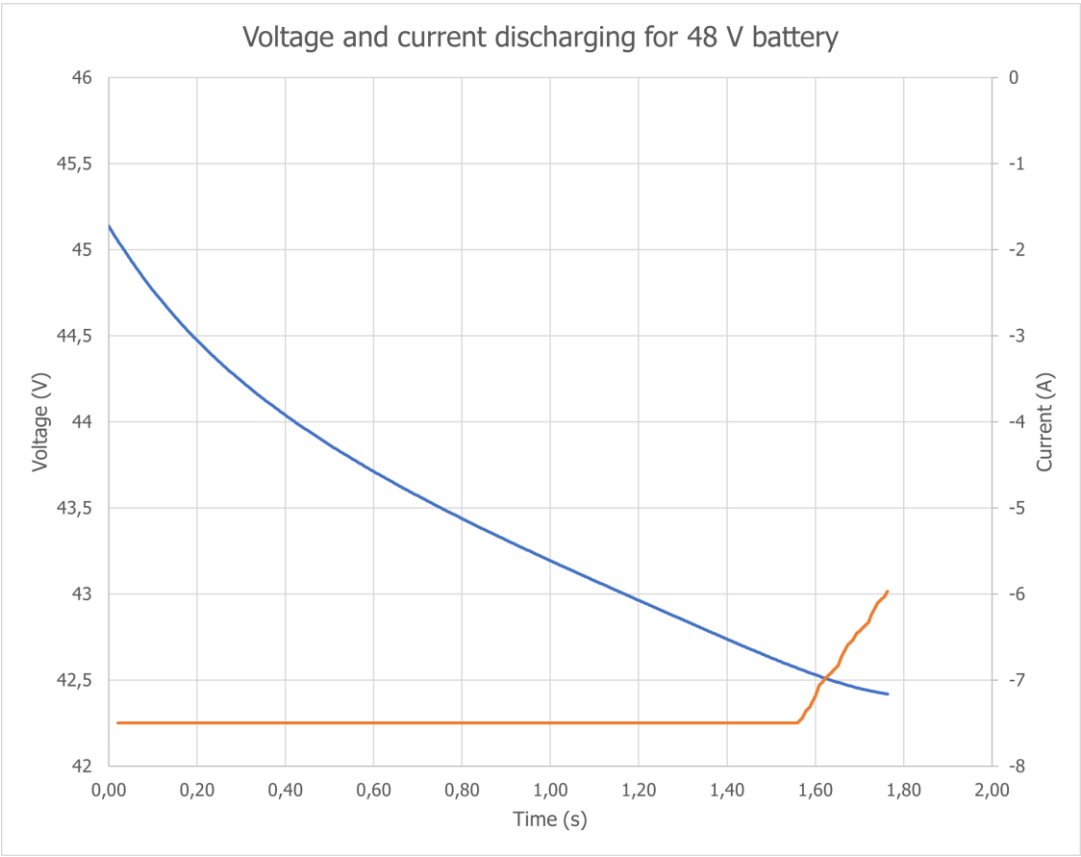

Figure 43. Experimental data of the discharge test for the 48 V battery.

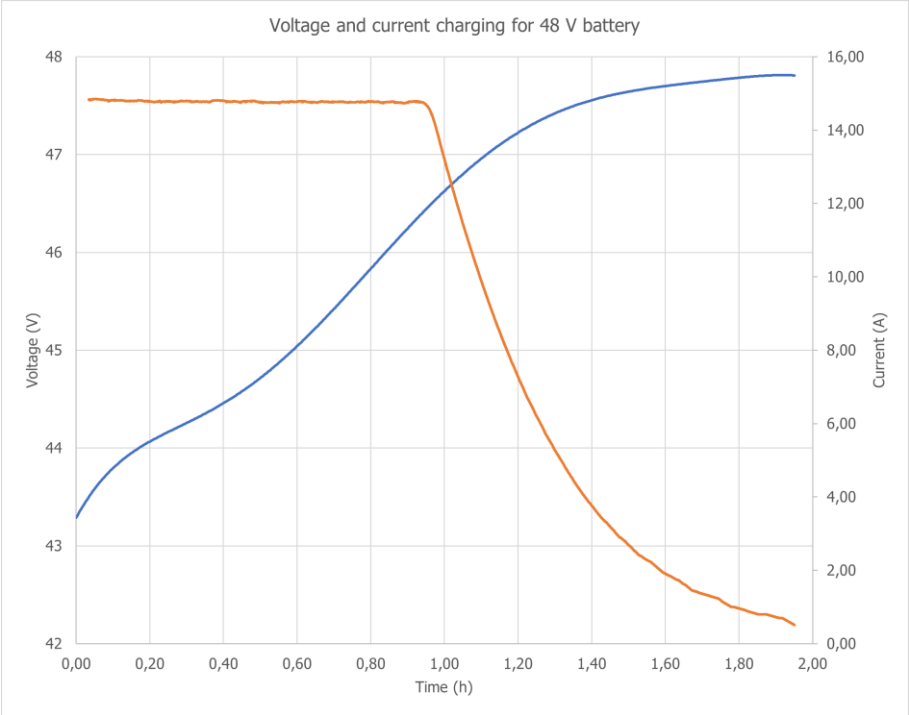

Figure 44. Experimental data of the charge test for the 48 V battery.

The same protocol was employed for the 200 V battery pack, where a constant current of 10 A was applied in both tests.

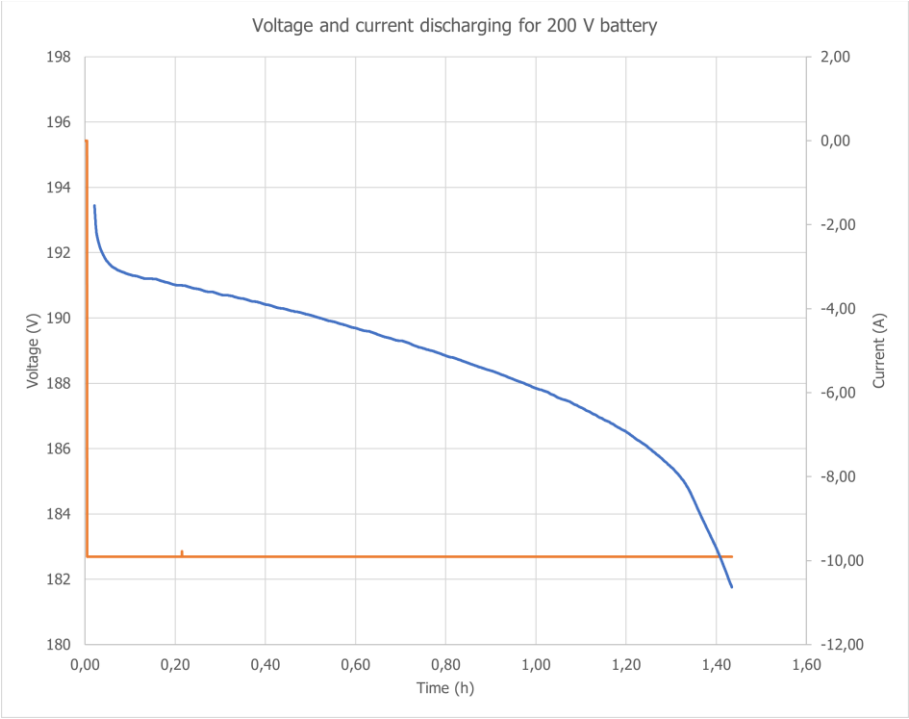

Figure 45. Experimental data of the discharge test for the 200 V battery.

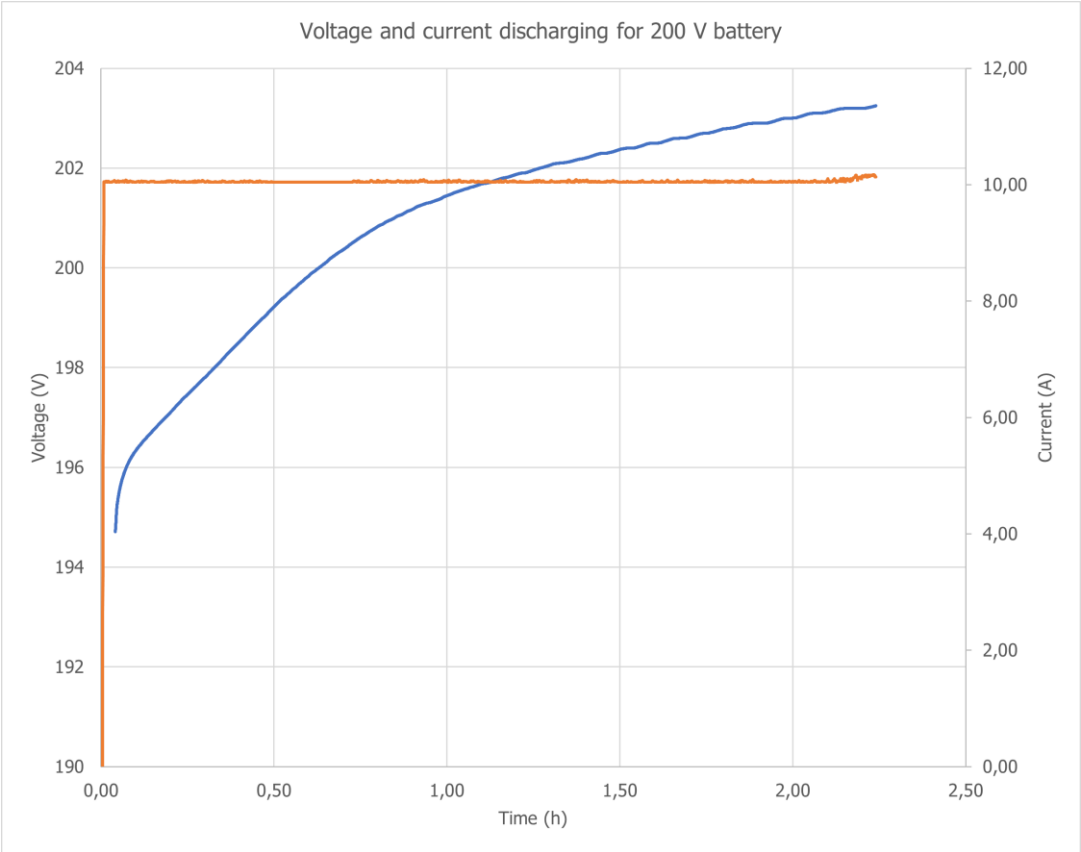

Figure 46. Experimental data of the charge test for the 200 V battery.

We also tested if the module and cells pages of the individual battery panels were working as expected:

Figure 47. Screenshot of the module page during the charge test of the 200 V battery.

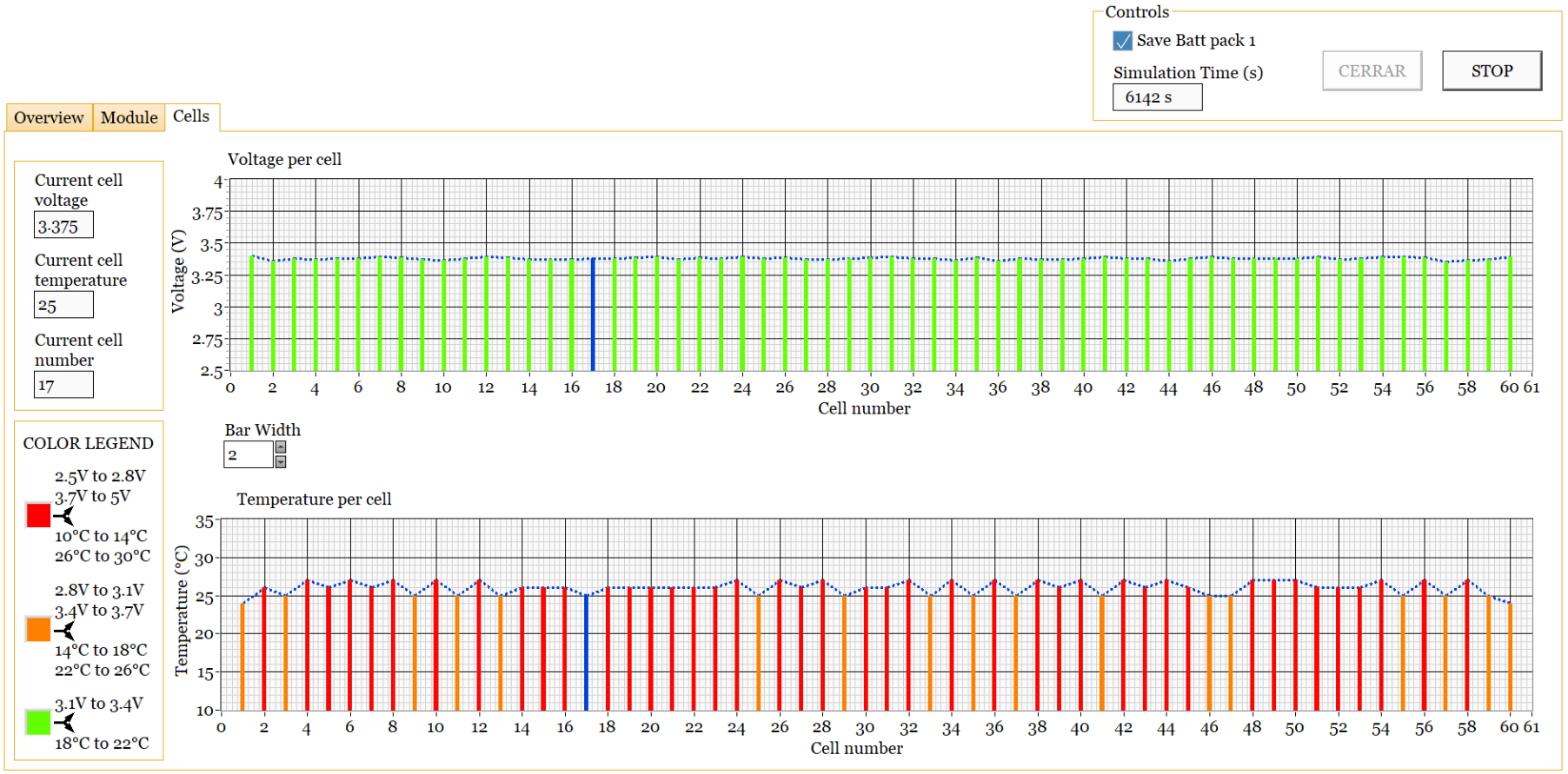

Figure 48. Screenshot of the cells page during the charge test of the 200 V battery.

4.5 Error test

4.5.1 FPGA is disconnected

This is arguably the most common error, and on most occasions is caused when either the computer or the FPGA:

- get disconnected from the local network.
- lose power and shut down.

In these circumstances, the program should give the user the choice to either save the data recorded up to that point, or simply stopping the execution of the program without saving the data. This solution will be implemented with a pop-up window that displays the general information about the error and allows the user to choose either option.

To test this error, on one occasion we disconnected the Ethernet cable from the FPGA, and on another we disconnected the power cable from the FPGA. On both occasions we obtained the same result:

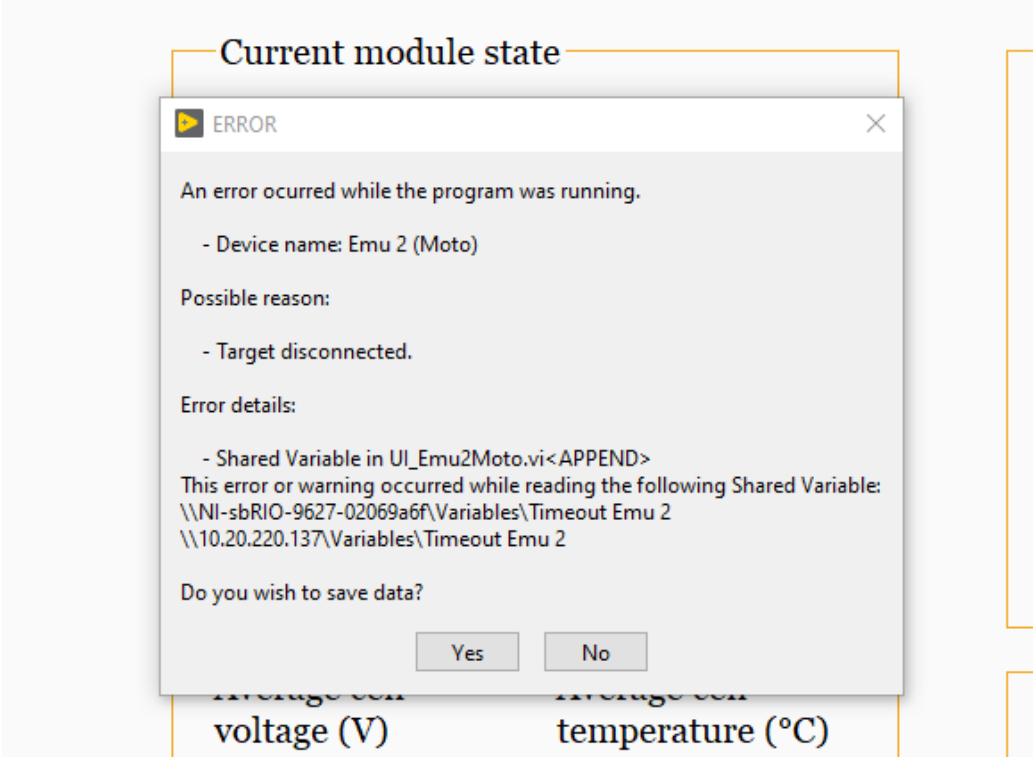

Figure 49. Screenshot of the error pop-up window.

4.5.2 Target folder for data files does not exist

If this problem was not treated properly, the program could get stuck before saving the data, and thus it would be lost.

The way we solved this error was by saving the data in a folder located in the same directory as the program, and if said folder does not exist, it creates one.

4.5.3 Executing a panel that is already running

This error happens when the user tries to execute the main panel on the computer while an individual panel was already running. The main panel tries to automatically execute the individual panel at the start but finds that it is already running.

If this happens, we can just disable the program block that tried to execute the individual panel a second time.

5. Conclusions

5.1 Goal fulfillment

The main objective of this project, which was to develop a program that is able to gather data from all devices in the system, visualize this data on a screen, and then save it to a file, has been accomplished. The program fulfills all the requisites, and it has been tested on several devices to prove its proper functioning.

It is worth mentioning that the modular methodology used for designing the SCADA system was very helpful, in the sense that it allowed us to basically only have to design one of the panels from scratch. The other panels were implemented using the first panel as a pattern, thus reducing the amount of time spent in the process.

Another conclusion that we can take away from this project is that the decision to use the CAN protocol was probably the correct one. It allowed us to have a very flexible and lightweight communication network, since we could add or remove devices from the bus without impact on the network.

5.2 Future work and improvements

5.2.1 Adding more devices

Due to the limited number of devices available during this project, the program has only been able to be tested on a few devices. Nonetheless, one of the biggest advantages of the modular methodology used on this project is that it allows for easy expansions and, as such, more devices can be added in the future. This means that more batteries and supercapacitors can be added to the SCADA system, as well as the devices mentioned in chapter 2.2.3, and others, e.g., power supplies.

5.2.2 Controlling and managing the system

Another topic that is left open to research is adding the capability for the SCADA to manage and control the system. Currently the program can only monitor the system, which encompasses visualizing the data and creating data backups. In the future, the program could be able to send orders to the devices, such as initiating a charging or discharging process, or remotely disconnecting devices from the system. It could also be used to test energy management techniques and its optimization.

6. References

[1] D. Zambrano-Prada, A. El Aroudi, L. Vazquez-Seisdedos and L. Martínez-Salamero, "UltraFast Charging of Electric Vehicles (UFCEV)," *XXVIII Seminario Anual de Automática, Electrónica Industrial e Instrumentación,* ser. SAAEI '21, Ciudad Real, Jul 2021 (in Spanish).

[2] R. Mathur, R. Saraswat and G. Mathur, "An Analytical Study of Communication Protocols Used in Automotive Industry," *International Journal of Engineering Research & Thecnology (IJERT),* vol. 2, no. 3, p. 6, 2014.

[3] National Instruments, "sbRIO-9627," [Online]. Available: http://www.ni.com. [Accessed 12 October 2022].

[4] Texas Instruments Incorporated, "Introduction to the Controller Area Network (CAN)," August 2002. [Online]. Available: http://www.ti.com. [Accessed 21 March 2023].

[5] International Organization for Standardization, "Road vehicles — Controller area network (CAN) — Part 2: High-speed medium access unit". ISO 11898-2:2003, 1 December 2003.

[6] International Organization for Standardization, "Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling". ISO 11898-1:2003, 1 December 2003.

[7] CSS Electronics, "CAN Bus Explained - A Simple Intro," 2023. [Online]. Available: http://www.csselectronics.com. [Accessed 17 February 2023].

[8] National instruments, "Controller Area Network (CAN) Overview," 24 May 2023. [Online]. Available: http://www.ni.com. [Accessed 30 May 2023].

[9] 123electric, "123SmartBMS gen3," [Online]. Available: http://123electric.eu. [Accessed 7 November 2022].

[10] 123electric, "123\SmartBMS Extended Module," [Online]. Available: https://123electric.eu. [Accessed 7 November 2022].

[11] 123electric, "123\SmartBMS Extended Module Manual," [Online]. Available: https://123electric.eu. [Accessed 7 November 2022].

[12] Influx Technology, "Understanding SAE J1939," [Online]. Available: https://www.influxbigdata.in. [Accessed 14 April 2023].

[13] Skeleton Technologies GmbH, "The ultimate building block for modular systems - SkelMod 102V," [Online]. Available: https://www.skeletontech.com. [Accessed 24 February 2023].