Communication and Control in Power Electronics Systems

Vladimir Mitrovic

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
In
Electrical Engineering

Dushan Boroyevich
Rolando P. Burgos
Igor Cvetkovic

December 17th, 2021
Arlington, VA

Keywords: Communication Protocols, Ethernet, MODBUS, IoT, Real-Time Operating Systems, Real-Time Communication protocols, EtherCAT, Distributed Systems, Distributed Control, Synchronization, Modular Multilevel Converters

Communication and Control in Power Electronics Systems

Vladimir Mitrovic

ABSTRACT

The demands of a modern way of life have changed the way power electronics systems work. For instance, the grid has to provide not only the service of delivering electrical energy but also the communication to enable interactions between customers and enable them to be producers of electrical energy, too. Thus, the smart grid has come into existence. The consequence of the smart grid is that consumers could be "smart." The most obvious consumers are households, so the houses have to also be smart and must be equipped with various power electronics devices for producing and managing electrical energy. Again, all those devices have to communicate somehow and provide data for managing electrical energy in the house. Zoomed in further, novel, state-of-the-art measurement equipment could have been built from different power electronics devices, and communication among them would be necessary for good operation. Zoomed further in, communication among different pieces of power electronics devices (such as converters) could offer benefits such as flexibility, abstraction, and modularity.

This thesis provides insight into different communication techniques and protocols used in power electronics systems. A top-down approach presents three different levels of communication used in real-life projects with all the challenges they bring, starting with the smart house, followed by the state-of-the-art impedance measurement unit, and finalizing with internal power electronics building block (PEBB) communication.

In the case of a smart house, where the house is equipped with solar panels, charge controllers, batteries, and inverters, communication allows interoperation between different

elements of the power electronics system, enabling energy management. Results show the operation of the system and energy management algorithm. A house of this type won first prize at an international competition where energy management was one of the disciplines.

The impedance measurement unit consists of different power electronics devices. In this case, too, communication between devices enables the operation of the impedance measurement unit. Communication techniques used here are shown together with measurement results.

Finally, inter-PEBB communication has been shown as an approach for interaction among the different elements inside the PEBB, such as controller, GDs, sensors, and actuators. Real-time communication protocol, including all challenges, is described and developed. This approach is shown to enable communication and synchronization among different nodes inside the PEBB. Communication enables all internal elements of the PEBB to be transparent outside the PEBB in the sense that data gathered from them could be reused anywhere else in the system. Also, this approach enables the development of distributed event (time) driven control, hardware and software, abstraction, high modularity, and flexibility. A very important aspect of inter-PEBB communication is synchronization. A simple technique of sharing a clock among the parts of a 6 kV PEBB has been shown.

Communication and Control in Power Electronics Systems

Vladimir Mitrovic

GENERAL AUDIENCE ABSTRACT

This thesis provides insight into different communication techniques and protocols used in power electronics systems. A top-down approach presents three different levels of communication used in real-life projects with all the challenges they bring, starting with the smart house and a custom device designed and developed to be a communication interface among different power electronics devices from different vendors, such as charge controllers or inverters, but with capabilities not only to communicate but to also provide a platform for the development of energy management algorithms used to make houses grid zero if not grid positive.

Aside from the smart house, this thesis describes communication protocols and techniques used in the impedance measurement unit (IMU). This complex measurement device provides valuable and accurate impedance measurements and consists of different power electronics devices that need to communicate.

Finally, at the power electronics building block (PEBB) level, real-time communication protocol with all challenges is described. Developed communication protocol provides communication and synchronization among different nodes such as GDs, sensors, and actuators inside the PEBB. This intra-PEBB communication and synchronization combined with inter-PEBB communication and synchronization provide the foundation for the development of truly distributed event- (time-) driven control as well as hardware and software abstraction.

Acknowledgments

I'm very grateful to my advisor Dr. Dushan Boroyevich and co-advisor, Dr. Rolando Burgos. This work could not have been done without their support and wise guidance throughout my research.

I want to express my sincere gratitude to Dr. Milan Mijalkovic and Dr. Victor Stefanovic, who provided me with the idea and opportunity to continue my education at Virginia Tech.

I want to thank all faculty, students, and visiting scholars who worked with me on all my research, especially Dr. Igor Cvetkovic, prof. Joe Wheeler, Dr. Bo Wen, Dr. Sizhan Zou, Dr. Zhiyu Shen, Dr. Boran Fan, Yu Rong, Slavko Mocevic, and Jianghui Yu. I enjoyed working with them and learned a lot from them. Special thanks to Adrien Arsuffi, who knew how to turn gloomy days into joyful days.

This work could also not have been done without the support of my fellow students, Narayan Rajagopal, Taha Mohaz, He Song, Jack Knoll, Arash Nazari, Haris Ashraf and Jayesh Motwani for giving me immense energy boost while working on this thesis.

Finally, I would like to thank my family. They were here when needed. And, I want to express my huge gratitude to Dragana Stajic, my leading star, energy source, and bright light.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Research Motivation and Challenges

Power electronics technology is ubiquitous and obvious in the current era. It's hidden in almost every electrical device used in normal life. For example, power-hungry mobile phones require a capable supply to provide enough power for everyday tasks. On the other hand, those powerful but tiny power supplies sitting inside the thin mobile phone have to be very dense and efficient, so that the battery can provide enough energy for a whole day of operation or even more. Our current way of life requires an operational mobile phone even if the battery is empty, but fast chargers that could recharge the phone in about 15 minutes are required. It's even better if it could be wirelessly charged while just sitting on the desk without being plugged into the wall.

Electric cars are very similar to phones but on a larger scale. Users are accustomed to conventional gas vehicles, with how quickly they can be refilled, and how far they can reach with a full tank. So, electric vehicles must have similar performance if users are to accept them. As part of a car's power electronics system, a motor drive must be extremely efficient to provide the speed, torque, and range comparable to that of a conventional vehicle. In addition, the electric car must be refilled as soon as possible, or appropriate distraction during the charging time has to be provided to users to give us researchers a bit more time to figure out how to charge the car in less than two minutes. Very powerful charging stations are needed to achieve that. Those stations have to be connected to a grid that could support them.

This example is just what comes from scratching the surface of current power electronics demands, and as illustrated, power electronics enable the development of next-generation systems such as electric vehicles, but also smart grids, smart houses, or power-hungry server farms. New systems require high-efficiency, high-density, and high-quality electrical power conversion.

1.1.1 Motivations

The current grid in the US was built in 1890 when demands were not nearly what they are today [1]; it was designed for utilities to deliver electric energy to homes. It is a one-way interaction where the utility knows only how much energy has been delivered, and it is tough for the grid to respond to modern energy demands. Smart grid development introduces two-way communication in which data and energy can be exchanged between customers and utilities. The smart grid is a network of different devices, such as controllers and power electronics. The synergy of new technologies and tools unite to make the smart grid clean, reliable, efficient, and secure. It enables newer technologies to be integrated, such as wind and solar for energy production and electric vehicle charging. Also, smart homes could be integrated both as producers and as consumers. Communication is in common among different system nodes, and this plays an important role in the system's overall functionality.

Smart homes are part of the smart grid, and they have a double role in the system, as consumers and as producers. The only way to enable the “smartness” of the house and its double role is to seamlessly interconnect all producers and consumers in the house and figure out the best way of arranging it so that the house can reliably manage available energy no matter if it comes from solar panels installed on the roof, batteries, the grid itself,

or any other source of energy. To assure smart interconnections, uninterrupted data flow, and reliable control, some type of communication is needed.

Modern life, as mentioned, involves, for example, electric vehicles. Among other things, they are equipped with batteries and chargers. Some of them, like the Ford F-150 Lightning [2], use batteries to provide intelligent backup power for the house. Thus, it's a complete system that consists of the increasing number of electronic components, such as chargers, inverters for motor drives, or even a front panel or air conditioner: it is a system very similar to that in a smart house, but on a smaller scale. All those devices work in harmony because of secure and reliable communication [3]. Usually, the controller area network (CAN) is used for this purpose. The new trend of autonomous vehicles brings new challenges in reliable and high-bandwidth communication networks used for sensor systems, which open the door to a new communication protocol like 5G [4]. Similar to vehicles, there are other devices consisting of several smaller units that need communication in order to operate.

When focused on vehicles, one of the main parts of the vehicle is the drive -- in the case of electric vehicles, this is an electric motor. To operate, it must have an inverter [5]. The inverter has several gate drivers (GDs), depending on the topology. In most cases, they are directly driven from the local controller without any communication protocol. What would be possible if elements of the drive could communicate?

Even though this work is not directly related to electric vehicles, they are used as a basic example of how the electric system could and is organized. The idea is to define the smallest element of the power electronics system, the least common denominator, as is done with the power electronics building block (PEBB) [6]. Connected using a

communication interface, larger power electronics devices like converters could be built out of PEBBs. Further, those power electronics devices could be combined into larger systems like smart houses or modular multi-level converters. Communication between each element in the system will allow the platform to have distributed control [7] as well as hardware and software abstraction. PEBBs could be arranged in different ways, which brings flexibility. Communication among PEBBs will increase flexibility even further. Communication could bring additional layers of abstraction, hardware, and software. Since communication between the furthest system elements exists, the number of sensors could be reduced because some sensors could be reused. For example, the current sensor used in the GD) for overcurrent (OC) protection could be used for phase-lag current reconstruction.

1.1.2 Challenges

The communication protocols between devices from different vendors are often not compatible, and even though devices might be partially capable, they are not able to communicate. Industrial devices often use MODBUS as a communication protocol and de facto standard. This protocol was developed in 1979 by Modicon, now Schneider Electric. It could run over RS485 or a local area network (LAN), and it is well established and accepted, but devices using it are not always compatible. Most likely, their vendors didn't want them to work together. Hence, an additional device must be developed to act as a bridge between them. If carefully designed, the same device could be used for other purposes, such as control or energy management. If a unified communication protocol could be used, devices from different vendors could work together, and energy management could be distributed among them.

Tight synchronization and fast data transfer are the main challenges in real-time communication, such as what happens between GDs, when synchronization is crucial for safe operation. For example, if looked at from the perspective of the half-bridge, the non-synchronized operation of top and bottom switches will cause shoot-through and will damage the device. Developing good synchronization is so delicate that even poorly designed field-programmable gate array (FPGA) code for a first-in-first-out (FIFO) buffer could cause synchronization issues.

1.2 Thesis Outline

Using a top-down approach, this thesis provides insight into some communication techniques and protocols used in power electronics systems. Three different levels of communication used in real-life projects, with all the challenges they bring, are shown. Starting with the smart house and a custom device designed and developed to be a communication interface among different power electronics devices from different vendors (such as charge controllers or inverters), but with capabilities not only to communicate but to provide a platform for the development of an energy management algorithm used to make the house grid zero if not grid positive.

Aside from the smart house, communication protocols and techniques used in the impedance measurement unit (IMU) are described. This complex measurement device consists of different power electronics devices that need to communicate in order to provide valuable and accurate impedance measurements. The IMU is an example of a basic distributed system where different nodes in the system (inside of the unit) have different roles and work in harmony because of seamless communication.

Finally, on the PEBB level, a real-time communication protocol with all challenges is described. The developed protocol provides communication and synchronization between different nodes such as GDs, sensors, and actuators inside the PEBB. The combination of intra-PEBB communication and synchronization with inter-PEBB communication and synchronization provides the foundation for the development of a truly distributed event (time) driven control as well as hardware and software abstraction.

Chapter 2 Energy Management in the Smart House

2.1 Introduction

With a clear goal of achieving a net-positive energy balance in the FutureHAUS [8], whose cartridge concept, an advanced electronic energy system comprised of state-of-the-art equipment is installed. Figure 1 shows this power electronics system. It consists of five solar arrays, aggregately contributing close to 14 kW of peak power. Each solar array features its dedicated charge controller for increased reliability as well as independent, per string, maximum power point tracking (MPPT). The lithium ferro phosphate (LFP) battery is currently the safest and least polluting rechargeable battery on the market, built with very high environmental standards, and this product is also safe to be stored indoors with no need for venting or cooling.

Furthermore, the efficient and contemporary 8 kW power inverter interfaces photovoltaics and batteries with the utility grid. It serves as the main generator of clean energy, utterly minimizing, if not eliminating, utility grid dependence. In addition to the above-described renewable energy system, FutureHAUS features an information infrastructure and an advanced, decision-making energy management algorithm that goes beyond traditional residential system control. For instance, if instantaneous harvested power from the photovoltaic (PV) array is higher than what the house load demands at that instant in time, an algorithm sends a command to the inverter to sell the excess power back to the grid (net-positive mode). As this algorithm also has access to the weather forecast (short and long term), it will decide whether it is better to sell all excess energy or reroute some percentage to charge the battery (e.g., if the rain will start soon, the majority of that

energy will go into the battery). If, however, power from the PV is lower than what the house load demands, an algorithm switches to the net-zero energy balance mode.

Additionally, inspired by the recent trends of power distribution where contemporary loads process energy through power electronic converters and use direct current (DC) as the primary source, the FutureHAUS has implemented low-voltage DC power distribution in a rail that seamlessly delivers power to the flex-space walls as they move from end to end [9]. The main motivation for demonstrating this concept comes from the fact that none of the contemporary home appliances (ranging from portable electronics and laptops to induction cooktops and air conditioners) directly use an alternating current (AC) supply [10]. All of them first rectify the alternating input to DC before supplying their internal circuitry for operation, hence rendering it completely unnecessary to impose the energy loss associated with AC-DC conversion. Unfortunately, the slow development of regulations and standards significantly impacts the widespread use and fast adoption of this technology.

2.1.1 Overview of the FutureHAUS

FutureHAUS presents a new way to build houses using prefabricated modules/cartridges. Modules are prebuilt in the factory and assembled on site. A module is a functional unit of the house; for example, a bedroom, bathroom, kitchen or power electronics (Figure 1). From the standpoint of electrical engineering, each module is an independent unit because it is equipped with its own breaker panel and electrical installation. Modules are connected to the so-called spine module from an architectural

perspective, a corridor, and from the electrical engineering perspective, a backbone. The main power supply from the main panel goes through the “spine” to the cartridge.

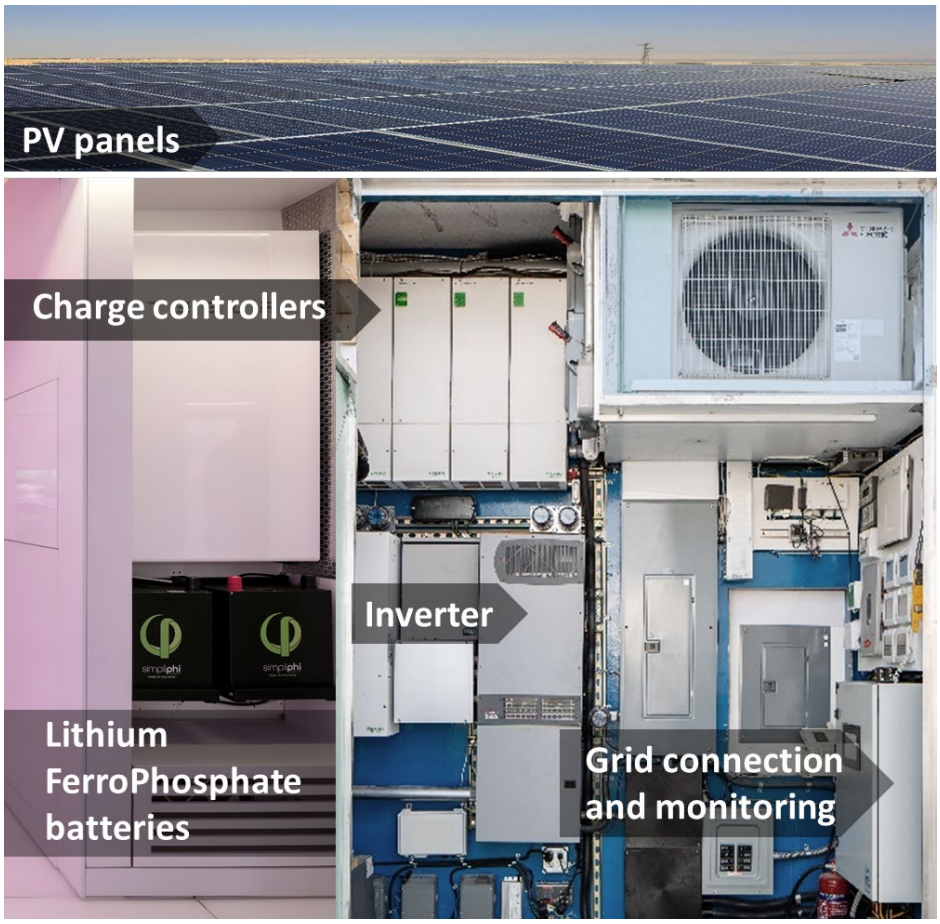

Figure 1: Power electronics cartridge(s) shown together

An innovative concept applied at the house is called “flex space.” Moving walls separate the bedroom, living room, and office space. They move to rearrange space for the best possible utilization of unused space. Walls hang from rails affixed to the ceiling. The third rail-like mechanism powers walls with low-voltage (LV) DC (48 V), without the need for cables, as shown in Figure 2. DC is inverted back to AC to provide power to different devices in the wall (TV, stereo, etc.).

All power electronics equipment is stored in a power electronics cartridge, which is shown in Figure 1. The idea is to have a unique design for the power electronics cartridge such that all needed elements are tightly integrated and that will fit every house. The power electronics cartridge consists of charge controllers, inverter, light controls, DC breaker box, PV breaker box, main breaker box, monitoring panel, and different instruments.

Figure 2: DC-powered moving wall (flex space)

2.1.2 Elements of the Electrical System in the House

The overview of the FutureHAUS power electronics system is shown in Figure 3. PV is the primary energy source in the FutureHAUS, and there are 50 PV panels installed on the roof, fixed in a horizontal arrangement. Each panel has 50 high-efficiency LG NeON® R [11] module cells. The efficiency of these PV panels is rated at 20.8%. Their

improved temperature coefficiency was a significant factor in the choice of these panels, since they have been used in a hot climate area.

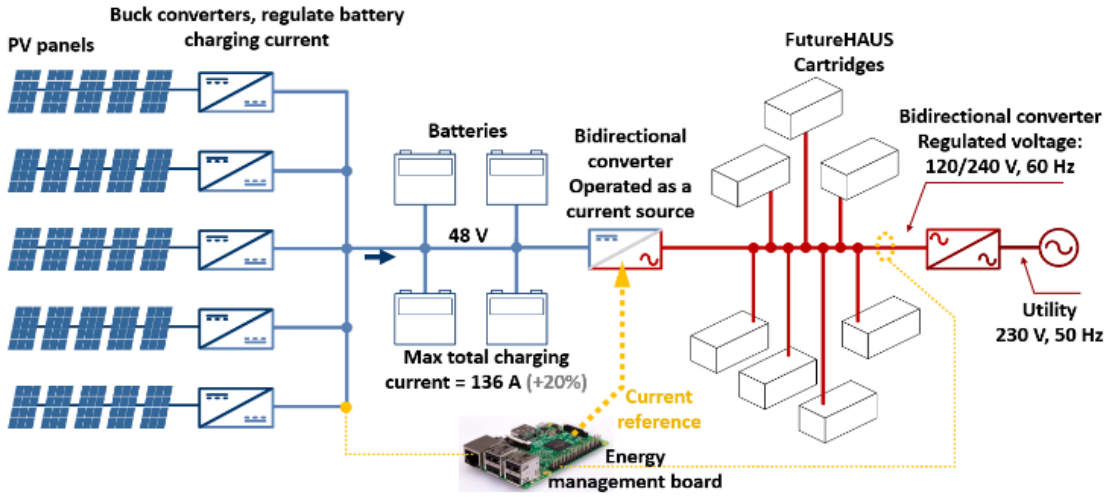

Figure 3: FutureHAUS power electronics system

Those 50 panels are divided into five strings, each of which is provided with ~270 W, which totals ~13 kW for the whole array. From an architectural standpoint, each string is one PV cartridge. Every string goes to a separate charge controller. The benefits of this concept are modularity, redundancy, and MPPT per string.

Solar charge controllers used in the system are Schneider Conext MPPT 80 600 [12]. They have a power efficiency of 96% at 48 V, which is an essential characteristic of overall system efficiency.

Chargers charged four parallel-connected Simpliphi PHI 3.5 48V lithium ferro phosphate (LFP) batteries [13]. The rated capacity of one battery is 3.5 kWh, and the depth of discharge (DOD) is 100%. A very important characteristic of batteries is that they do

not have thermal runaway, which means, there is no liquid evaporation, so they are safe for indoor use.

An inverter with a low voltage (LV) link (48 V) and reliability in a dusty and high-temperature environment is needed. The inverter used in this project is an Outback Radian series GS8048A-01 [14]. Its nominal output is 8 kW, but the real-life capability is 7.2 kW. Its typical efficiency is 93% [14].

As stated before, the inverter has a maximum output power of ~7 kWh, and the PV array has a maximum output power of ~14 kWh. That allows better solar energy utilization of up to 7 kWh (Figure 4, green and yellow area) as compared to 7 kWh PV panels (Figure 4, yellow area). With precise tweaking of the charge controllers, a decent portion of energy above 7 kWh has been used (Figure 4, red zone) for charging the batteries.

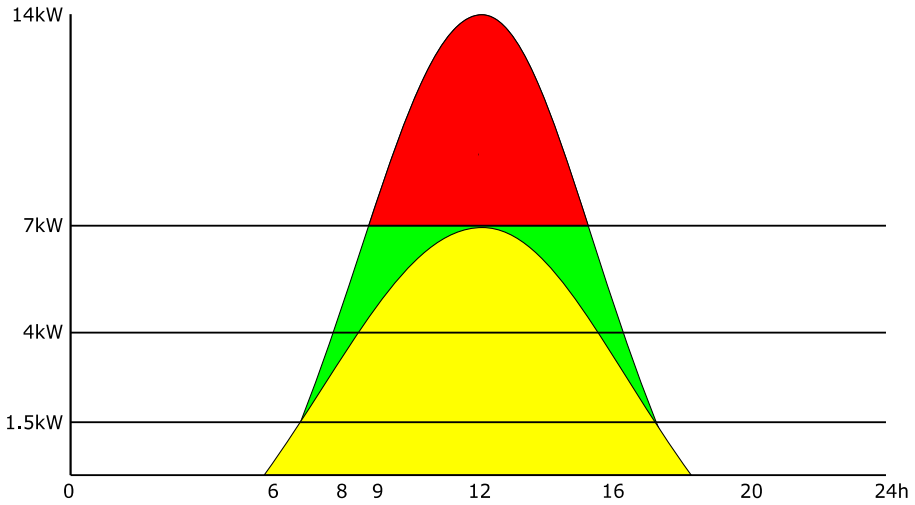

Figure 4: Advantage achieved with 14 kWh PV

The charge controller and inverter have their own communication interfaces. For internal communication, they use proprietary protocols, but for communication with third-

party devices, Modbus [15] [16] over LAN has been used. This is a vital feature of both the charge controller and the inverter. It brings a level of control flexibility that is further used to develop the energy management/control algorithm. The measurement technique described in [17] is used as a reference.

The heart of the local controller is the RaspberryPI computer with SmartPI [18] as an expansion module. SmartPI has an interface for measuring voltages and currents in a contactless manner (Figure 5). The interface is based on Analog Devices ADE7878, a multiphase, energy-metering integrated circuit (IC). It enables the measurement of electricity production and consumption and turns the RaspberryPI into a smart meter.

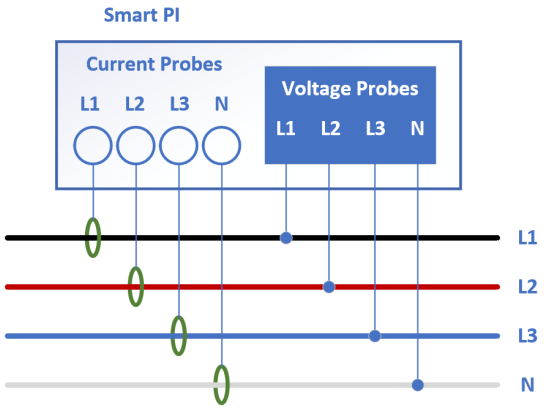

Figure 5: SmartPi connection diagram

RaspberryPI (model 3B+) is a credit-card-sized computer equipped with a Quad-Core 64-bit Cortex-A53 CPU that runs on 1.2GHz, 1GB of RAM, 100 Base Ethernet, and a 40-pin GPIO port [19]. It runs a fully-featured Debian OS-based Linux distribution.

The controller has two roles in the system: make a communication bridge between the charge controller and inverter and run an energy management algorithm.

2.2 Communication Bridge

One of the biggest challenges is controlling the inverter based on the data received from the charge controller and the mains measurements. Information collected from the charge controller is the amount of energy utilized from PVs and battery charges. The Schneider charge controller cannot deliver the data to the inverter, because even though both devices use MODBUS over LAN, they cannot work without an interpreter/bridge between them.

The role of the bridge is given to the RaspberryPI controller. It collects data from the charge controller and operates the inverter. Also, since it is equipped with voltage and current sensors, it uses that information, too. The PI (proportional, integral) [20] control (Figure 6) manage the output power of the inverter so the system is in grid-zero mode, whenever possible. For the grid zero mode the current reference (i_{REF}) is set to 0 and is then compared to the current consumed from the grid. Inverter output current is then set accordingly. All other modes of opperation (described in this document) are achieved by changing the reference current.

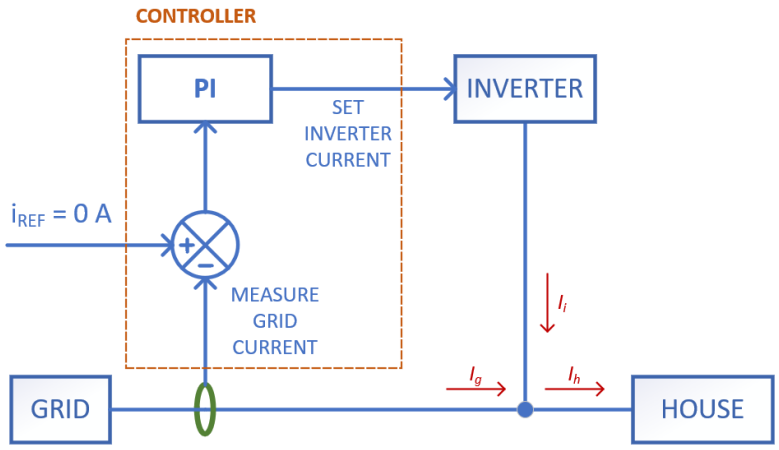

Figure 6: Control scheme

Code for energy management has been written using the programming tool Node-RED for wiring together the hardware devices [21]. The lightweight runtime is built on Node.js, taking full advantage of its event-driven, non-blocking model. This makes it ideal for running at the edge of the network on low-cost hardware such as the RaspberryPI. It provides a browser-based flow editor that makes it easy to wire together flows using the wide range of nodes in the palette. The flow used in this project is shown in Figure 7.

Now, once the charge controller and inverter control have been established, the energy management algorithm can run.

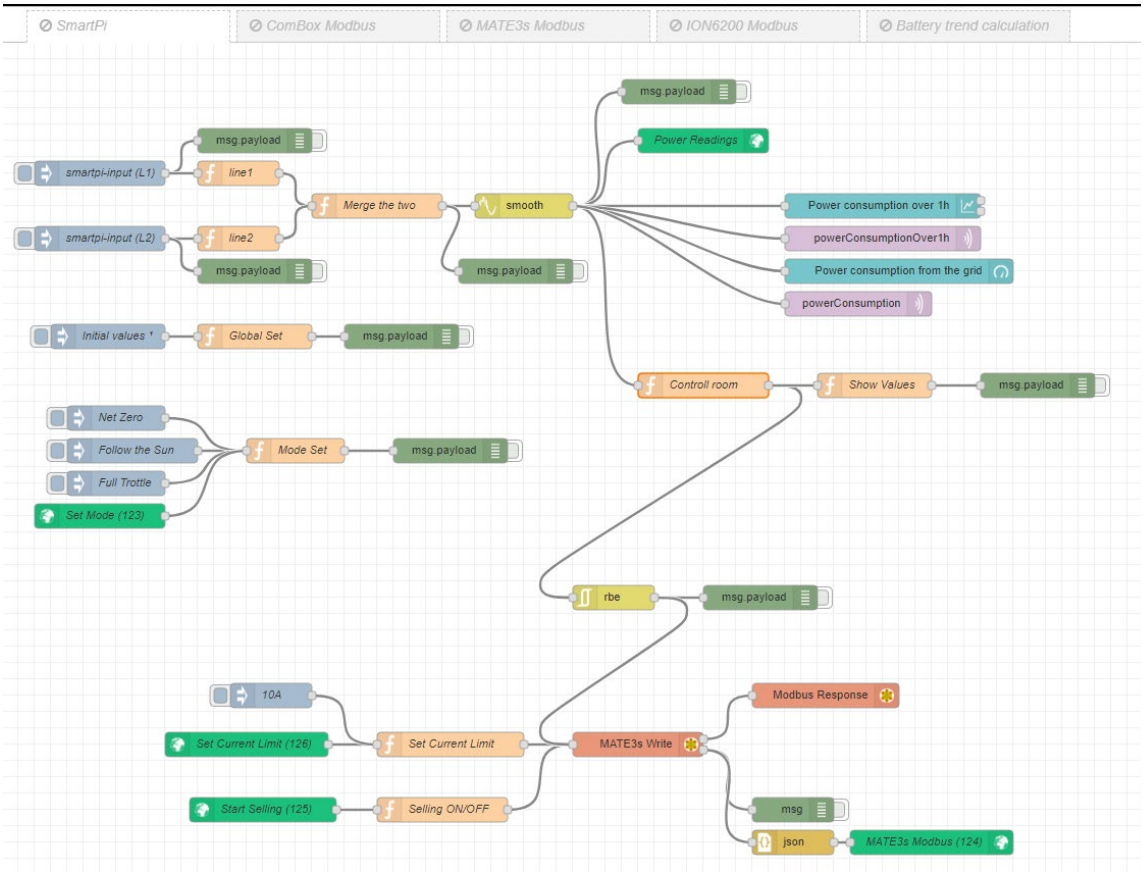

Figure 7: Energy management flowchart

2.3 Energy Management Algorithm

This work is operating under the assumption that solar energy is available for 10 hours a day, from 7 am until 5 pm (Fig. 8), which is the case during November. The first and last hour of the day are used only for maintaining grid zero, since there is not enough solar energy for anything else. The experiments illustrated that in this case, the permanent load of the house is ~1.5 kWh, which is the case in hot and humid areas where HVAC has to maintain a specific temperature and humidity in the home. Also, in this case, the minimal inverter power was ~720 W (3 A x 240 V), so that was the minimum possible.The goal is to have enough energy stored in the batteries to be net-zero during the night and to have enough solar power for all daily activities (cooking, dishwashing, watching TV, etc.) for one average household.

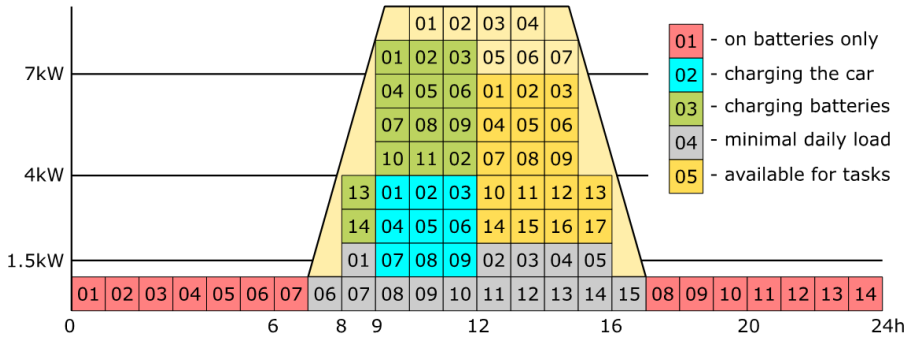

Figure 8: Schema of available energy (each square is 1 kWh)

The day is divided into three main tasks. Charging batteries (which include car batteries) is the priority task, and is presented as the green and blue squares (each square is 1 kWh). This task starts at the beginning of the day and will last as long as needed to recharge batteries before the end of the day. Thus, one issue is how to balance the available

energy between tasks and still have fully charged batteries by the end of the day. A 14kWh battery in the house and a 9 kWh battery in the car are available. The maximum electrical enery for charging the in-house battery is 4 kWh, and the car is 3 kWh, so four sunny hours are needed to recharge batteries. If needed, the charge controllers could be pushed to extract 7 kWh more.

The second task is to obtain enough solar energy to cover a minimal daily load and provide enough power to cover the daily needs of an average family. Covering minimal daily needs (Figure 8, gray squares) is the second priority, just after charging the batteries, and covering the daily need of inhabitants (Figure 8, yellow squares) is the third priority. Energy reserved for the minimum daily load was ~15 kWh, and energy reserved for everyday tasks like laundry, cooking, dishwashing, and entertainment is 17 kWh.

The third task was to provide electrical energy during the night. As much as possible, power should come from batteries. Since 14 kWh battery capacity is available for 14 hours of the night, and the minimum expected permanent power consumption is 1.5 kW, and the minimum possible power production is 0.7 kW (the inverter does not go lower than this), this is not such an easy task.

These three tasks shaped the energy management algorithm.

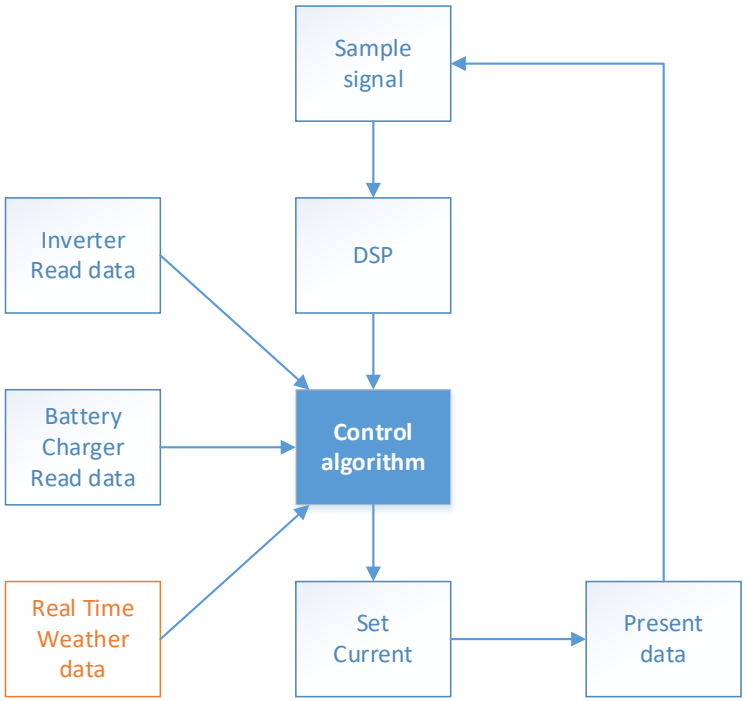

Figure 9: Simplified energy management flowchart

The control algorithm (Figure 9) that runs on RaspberryPI receives data from the inverter and charge controllers using communication lines and measures the power consumption at the mains using SmartPi. It reads battery voltage and harvested PV power from the charge controller, reads voltage and current on the mains, and reads voltage and current that the inverter is selling/producing. It also receives data from an internet-based or local weather station. Based on collected data, it "decides" in which mode to operate and how much current it should sell to maintain at least grid-zero operation. Weather data is not mandatory, but based on it, the algorithm can predict production and adapt itself to future weather conditions -- mainly cloudy weather.

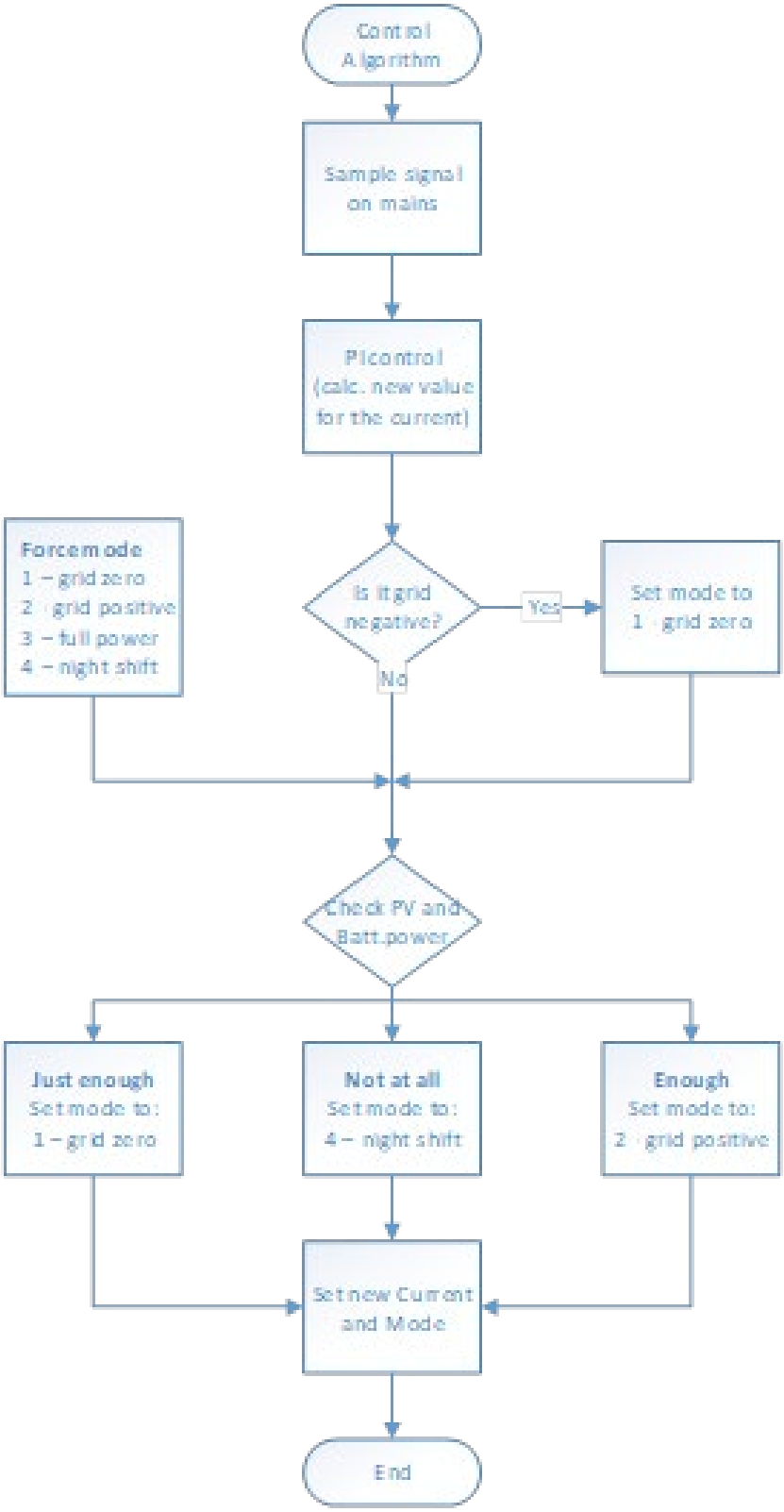

Figure 10: High-level inverter current control flowchart

The high-level inverter current flowchart is shown in Figure 10. The controller reads the difference between produced and consumed power. Based on sampled values, the basic PI control calculates a new value for the inverter's output current. Suppose the consumption is higher than production (i.e., grid-negative). In that case, the controller will force the inverter into a grid-zero mode (production equals consumption) by setting the newly calculated value for the inverter's output current. If there is no solar power, the inverter will use battery power to maintain grid-zero, as is the case during the night. That ensures that the system will always be at least grid-zero if there is enough power in the PVs or batteries.

The next step is to choose a working mode based on the tasks that should be performed. From the charge controller, SmartPI knows the PV power and battery status. It will maintain grid-zero until the internal battery is fully charged. When it is fully charged and if there is enough solar power to be grid-positive, the controller will bypass PI control and sell all available energy back to the grid. That mode is called grid-positive. In that case, it will use the grid as energy storage since later during the night, in some cases, it will be able to resume utilizing that energy.

In rare cases, if there is no PV power and the battery level is critical, it will switch to so-called night shift mode. In this mode, the inverter will produce just enough energy for the system to operate, even if it is grid-negative. This is usually the case during the night when overall power consumption is low.

Overall, during the extended testing period, especially during the two weeks of the 2018 Middle East Solar Decathlon, this approach was approved as the right solution for maintaining a net-zero.

2.4 Experimental Data

Experimental data has been recorded during the Solar Decathlon Middle East [22] (SDME), which is an international competition created by the U.S. Department of Energy in which universities from all over the world meet to design, build, and operate a grid-connected, solar-powered house.

There were system limitations for the competition. Maximal output power for the inverter was limited to 8 kW. Battery storage was limited to 15 kWh. The system must be grid-tied, so the grid is used as a voltage source and the inverter as a current source. All loads must be connected to the AC side. There was no limitation to PV production, and thus having as many PV panels as possible was beneficial.

From an electrical engineering perspective, the goal is to be at least net (grid) -zero as much as possible, which means that electrical energy consumption should be smaller than production. In this way, the system will feed both batteries and the grid with generated power. Later, during the night, the system will utilize energy from the batteries.

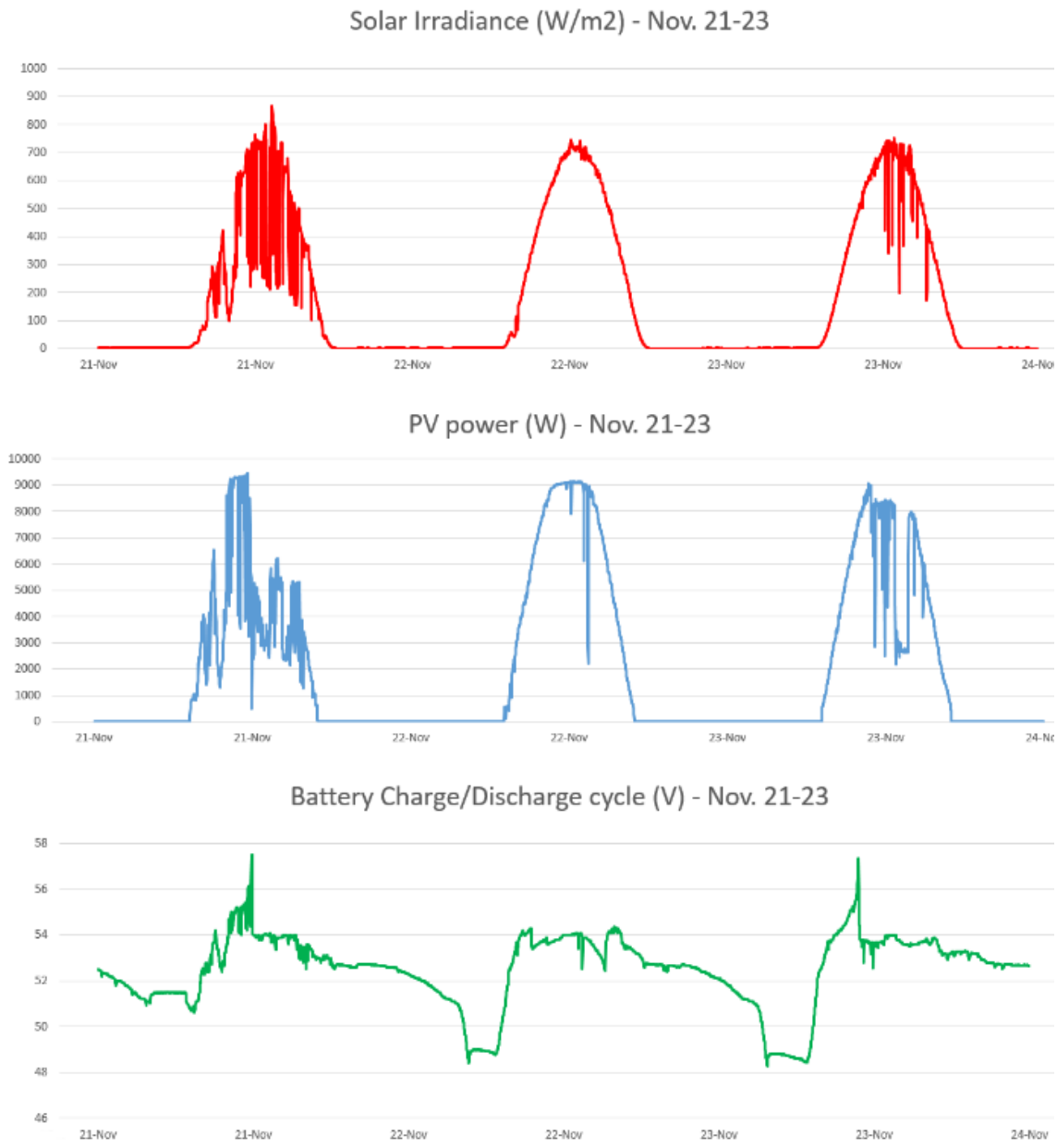

Figure 11: Solar irradiance, PV power, and battery terminal voltage

Figure 11 shows data from three days of regular house operation during the competition. The solar irradiance plot shows that the first day was cloudy. Day two was nice and sunny, and the third day had a cloudy afternoon. The utilization of solar power

followed the sun's irradiation. From all three days, it can be seen that peak power production was around 9 kW. From the first day, it can be seen that the system charged the batteries for the first couple of hours and then stayed in net-zero mode since there was not enough energy for selling. Since the second day was bright and sunny, there was plenty of solar energy, and the PV curve looked almost perfect. The system decided to charge batteries over the day and to sell as much as possible. At the beginning of the sunset, a dip occurs in solar energy consumption. The system will recognize that moment as the evening had begun and switched to net-zero mode to charge the batteries for a short period, and then decided to go back to selling. Since the day was hot, the night was hot, too. The HVAC system maintained the temperature and humidity in the house all night long and discharged the batteries before dawn. The system will first recharge batteries and then, from time to time, tried to sell as much as possible and, at another moment, went into the safe/net-zero state to collect as much energy as possible for the following night.

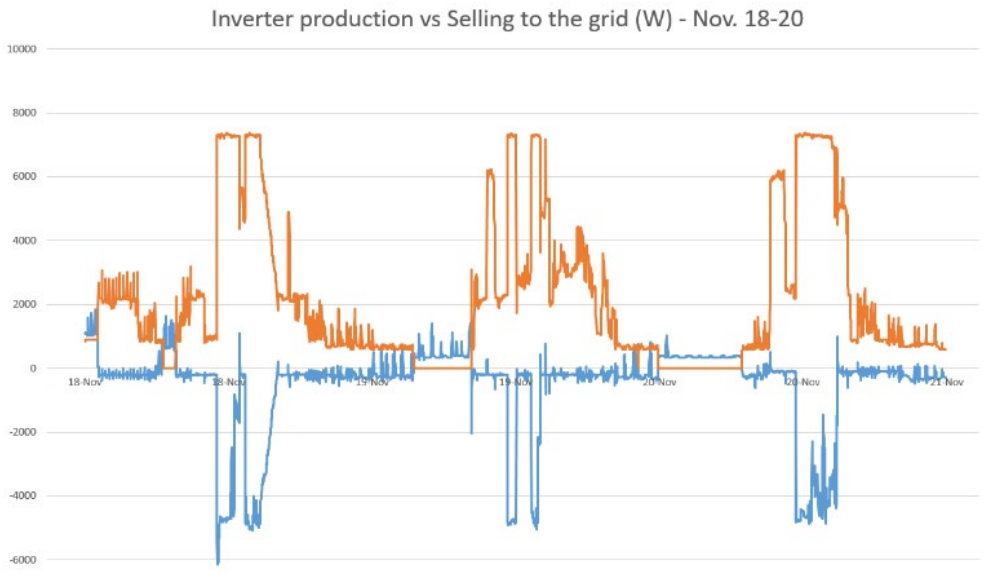

Figure 12: Inverter's energy production vs. consumption over a three-day period

Figure 12 shows the energy that the inverter produced and that the house consumed over a three-day period. The reddish plot shows the overall power generated by the inverter, and the bluish line is energy sold to the grid. The plot illustrates that a decent portion of solar energy was sold to the network. Also, Figure 13 shows the cumulative production and consumption measured by SDME staff. Across the 12 days, the house consumed ~330 kWh and produced ~440 kWh. The energy stored in the batteries was not measured by the SDME staff, and that was an additional ~140 kWh, so roughly, the house produced 580 kWh and was able to sell ~250 kWh during a given time frame.

2.6 Conclusion

This chapter describes an advanced electrical energy system designed and implemented in FutureHAUS. The system is comprised of solar arrays, charge controllers, batteries, and a grid-tied inverter serving as the main interface between the FutureHAUS and the utility grid. The developed system features an advanced information infrastructure and decision-making energy management algorithm, critical for the net-positive energy balance operation achieved in this project. The concept, architecture, system components, communication, and energy management algorithm have been described. Experimentally collected data have been shown as well.

As illustrated, the communication bridge enabled the functionality of this system. The biggest challenge was figuring out a communication protocol so that data from the charge controllers and the inverter could be collected and that the inverter could be controlled.

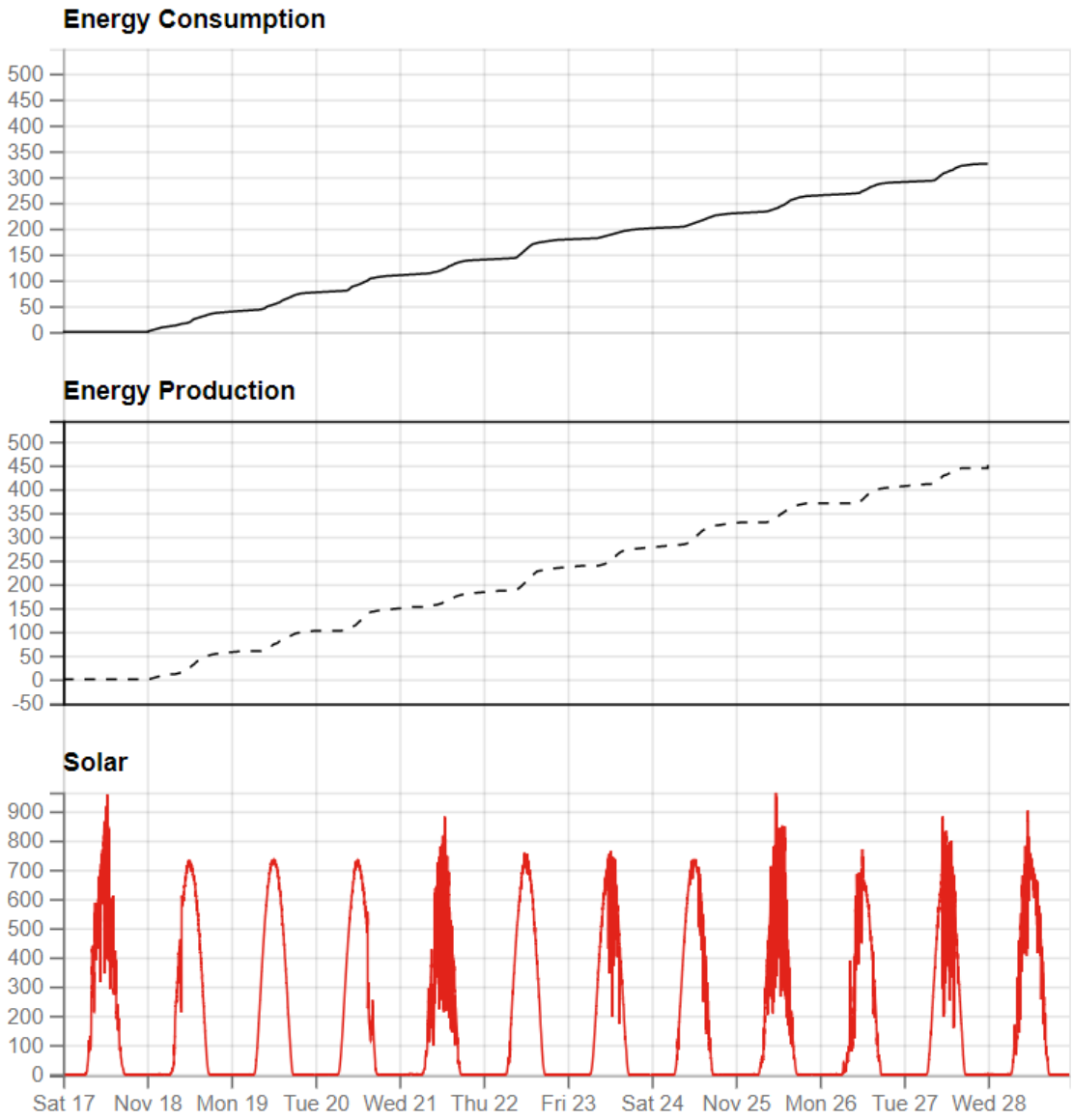

Figure 13: Example of energy utilization

Chapter 3 Communication and Control of an IMU

3.1 Introduction to Impedance Measurement Unit (IMU)

Small-signal impedances of components in an electronic power system can be used to measure the system's stability [23] [24]. Measuring impedances of the system requires a sophisticated control and software system [25]. Such a system consists of a state-of-the-art data acquisition unit (DAU), several high-precision sensors, and a PC with the software for data storage, data processing, perturbation control, and user interface [26] [27] [28].

A sensed signal flows through the signal conditioner unit (SCU) to the DAU. The primary purpose of the DAU is to sample the signal and improve it by filtering noise, addressing aliasing issues, and removing DC offset that occurs due to imprecise components in the signal conditioner. The digitally conditioned signal is also used for phase and frequency identification, which is then sent to the PC for data storage and offline data processing; examples of this include impedance calculation and system stability estimation.

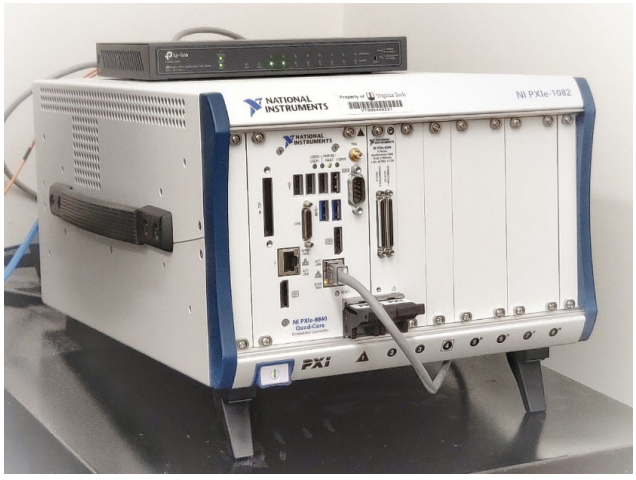

Figure 14: Data acquisition unit (PXI)

The hardware of the DAU is a National Instruments (NI) quad-core PXI computer (Figure 14) with a high-performance data acquisition (DAQ) card installed [ref]. The main design challenge for this unit is the software. To coordinate a real-time task, the PXI computer is set to run a real-time operating system. The LabWindows/CVI [ref] with a real-time support module is used for real-time software development. LabWindows/CVI is also used for the development of high-level measurement control PC software. Using the same development platform for PXI and the PC software makes development easier, since code can be easily shared between platforms, saving the overall cost of development. Since NI LabWindows/CVI is a C language development environment, many useful third-party libraries are available.

The measurement control application controls the perturbation injection unit (PIU), DAQ, storage, and analysis. It was developed on the host computer, which has a non-real-time operating system (MS Windows).

Different software techniques are used to ensure that the acquired digital signal won't be lost on its way to store and preserve the required precision. The software solutions used to develop this system will be discussed later.

3.2 Data Acquisition, Processing, and Streaming

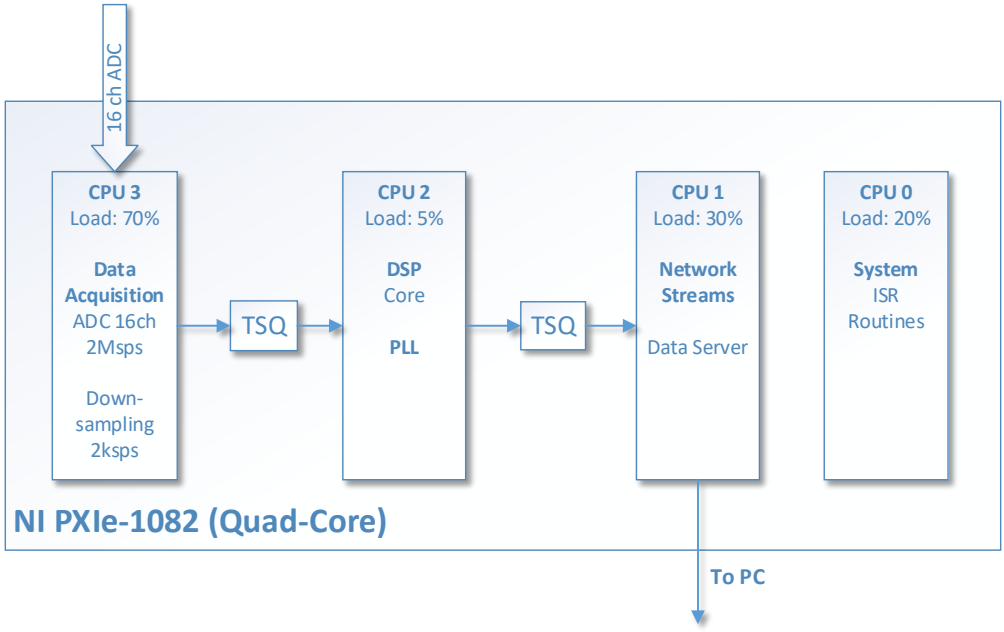

Figure 15: Software on the DAU (block diagram)

Analog signals from the sensors are sampled in the DAU and are then sent to the PC (personal computer) for further usage. The DAU also conducts some real-time signal processing tasks. Digitized and processed data is sent through the LAN network using the NI Network Streams protocol [29], which operates on top of a TCP/IP (Transmission Control Protocol/Internet Protocol).

Figure 15 shows a block diagram of the software on the DAU. The software was designed to run as a multithread application to maximize data throughput. The PXI computer has a four-core CPU, so threads are distributed between those four physical cores. Each CPU core, except for system core zero, has one thread locked to it, giving full control over processes executed on the core.

A very similar concept could be found, for example, at PLECS's RT Box 3 [30], hardware in the loop (HIL) device, where one core is used for maintaining HIL operations, while the other three cores are used for simulation purposes. In the RT Box case, the user has to define the relationship between physical cores and what part of the simulation that will run on that core, while here, the tasks for each CPU core are predefined to achieve the highest efficiency.

Analog data flows from the SCU to CPU 3 for digitalization/sampling and downsampling, then CPU 2 for phase calculation. Information from the analog to digital converter (ADC) to data acquisition is combined with PLL data and sent to CPU 1, preparing and streaming the data to the PC. Threads are connected with thread-safe queues (TSQs). The TSQ is a first-in, first-out (FIFO) buffer with an included mechanism for safe inter-thread data exchange.

3.2.1 Communication Between Threads and Physical CPU Cores - TSQ

A multithreaded application requests carefully processed and transferred data between different threads [31]. Data can be safely passed among threads using the TSQ from the LabWindows/CVI Utility Library [32]. A typical use of a TSQ is when the program contains one thread that generates an array of data and another thread that must operate on the array of data. This scenario is best known as the producer-consumer design pattern [33]. A TSQ handles all data locking internally. It uses a locking scheme in which one thread can read from the queue while another thread is writing to the queue, and they work without blocking each other. A TSQ can be configured for event-based access, which

means that a reader callback can be registered and then called when a certain amount of data is available in the queue. A TSQ can also be configured for writer callback, called when a specified amount of space is available in the queue [34]. Further, a TSQ can be configured to grow automatically if data is added when it is already full.

3.2.2 Data Acquisition and Downsampling (CPU #3)

The dedicated National Instruments (NI) DAQ card executes data acquisition. The NI-DAQmx [35] framework and application programming interface (API) functions are used for interfacing with the DAQ card, which simplifies the DAQ tasks. NI-DAQmx can work in a multithread environment, which is necessary for this design. Before sampling the data, the channel for analog to digital conversion and sampling frequency has to be set to successfully execute DAQ. Since DAQ is time-critical, a dedicated CPU core has been reserved. The NI-DAQmx has two roles. The first is to scale and offset sampled data. This will simplify data usage. The second role is to transfer sampled and scaled data from the DAQ card buffer to the memory of the PXI.

The sampled data is processed in the PXI before sending out to the PC. Analog to digital conversion, in this configuration, runs at a maximum data transfer rate of 2 Msps per channel, and the raw data rate is roughly 1 Gbps (2 Msps per channel x 16 channels x 32 bits per channel). So, sending unprocessed, raw data from the PXI to the PC requires a large bandwidth. The highest frequency of interest is 1 kHz. So, according to Nyquist, a sample rate of at least 2 ks/s is needed to represent such a signal. But, instead of 2 ks/s, the sampling rate of 2 Ms/s is used for an oversampling technique. This way, the analog to

digital conversion resolution is increased, and noise can easily be reduced. Each analog signal is sampled at a frequency significantly higher than twice the bandwidth or highest frequency of the sampled signal. This approach helps avoid aliasing issues, improves resolution, and reduces noise [26].

The sampled data is then averaged internally in the PXI. This way, lower-sampling-rate data flow is achieved. The most significant benefit of the down-sampling filter is that background noise is significantly reduced in the averaging procedure. Since averaging is applied, in this case, over 100 samples of each channel, the output is a data flow of a sampling frequency of 20 ks/s. The required data rate is significantly reduced from 1 Gb/s to 10 Mb/s.

Averaging data is a time-consuming job. Usually, the most efficient way to average data is to use a platform-specific routine. Lab Windows/CVI provides an analysis library [36], which includes fast and optimized functions for 1D and 2D array manipulation, complex and matrix operations, and statistics. One of those functions [37] is used to calculate the average value of the input array. This task has been done entirely in CPU 3, simultaneously with the sampling. Synchronization has been achieved through the TSQ. Since the amount of data is significantly reduced, the buffers used to carry data between threads and the CPU load have also been reduced.

3.2.3 Signal Processing (CPU #2)

The phase identification of the sampled signal is carried out in a separate CPU core. The phase of the system voltage is used for the ABC to DQ coordinates transformation

[38]. Tracking the real phase is very important for ABC to DQ transformation; otherwise, the spectrum of the signals in the DQ coordinates will be changed. The system frequency is recorded, too, and is used in the upcoming data processing. The CPU 2 core is reserved for PLL calculations.

3.2.4 Network Streams (CPU #1)

The down-sampled raw data and the phase and frequency information received from the PLL block are passed to another core that will send it to the PC. The NI Network Streams protocol is used for data streaming implementation. The data streaming server is in charge of communication with the PC that acts as an MCU. The data streaming server runs as a separate thread, locked to CPU 1, and sends acquired data out to the PC.

The NI Network Streams [39] function is an easy-to-configure, well-integrated, dynamic communication protocol for transferring data between two applications. It works on top of the TCP/IP, and it is designed to provide efficient data streaming at speed similar to that achieved with raw TCP or UDP (User Datagram Protocol). The NI Network Streams function provides an easy-to-use high-level abstraction to handle connections and dataflow control. It has connection management that automatically restores network connectivity after a network outage or other system failure. The NI Network Streams function uses a buffered lossless communication strategy, which ensures that data written to the stream is never lost. The one-to-one, unidirectional communication channel has a writer and a reader endpoint, similar to that in the TSQ. Since network streams are unidirectional, if there is a

need to pass bi-directionally between two endpoints, two streams should be opened. Multiple streams should be opened to pass data to multiple targets.

The dedicated thread reads the data from the input buffer using the TSQ API to which the signal processing thread writes data. NI Network Streams protocols have error detection, and a data-resending mechanism inherited from the TCP/IP. This way, the data will reliably arrive at the target. To verify that data doesn't get lost if the application runs out of receiving buffer (buffer overflow) and fails to read the data out from the NI network streams FIFO buffer, block numbering is used to detect buffer overflow and data loss.

3.2.5 Console View

The PXI computer runs a real-time operating system with dedicated software for DAQ. The PXI software runs headless, so it does not need a monitor, keyboard, or mouse to operate. Information about the PXI's operation is shown on standard output. In this case, the standard output is a monitor connected to a port on the front of the NI computer chassis. If an extra monitor is not present, the desktop PC application, called "ConsoleView," runs remotely on the desktop PC screen.

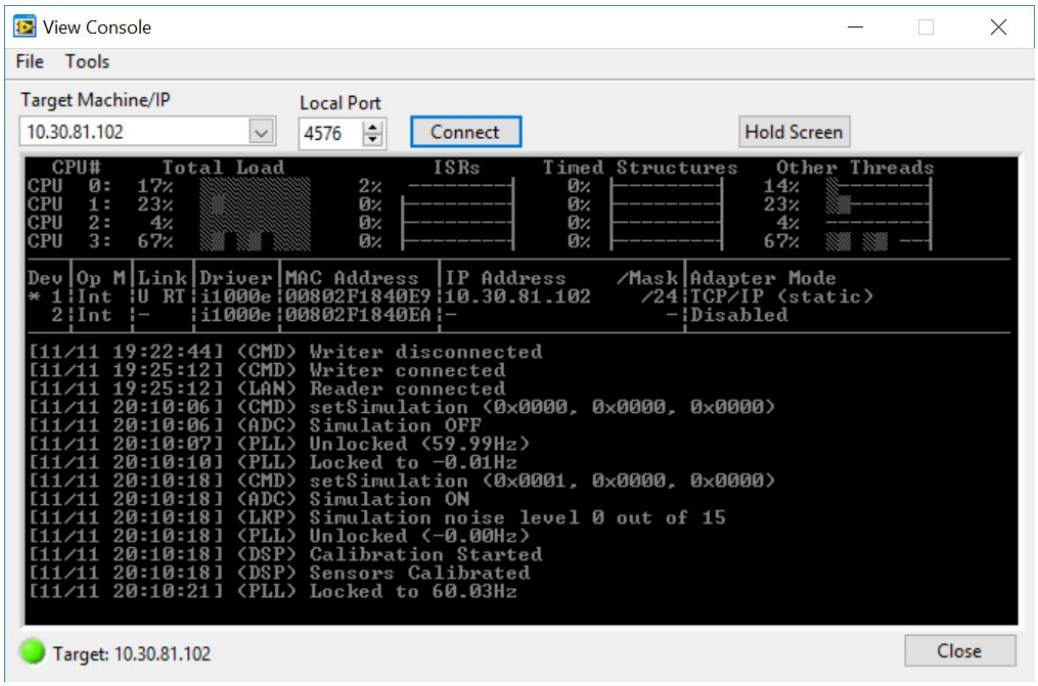

Figure 16: NI PXI ConsoleView application running remotely on the desktop PC

When the application is running on the PC and is connected to the PXI, a lot of useful data is shown. Figure 16 shows the CPU load and network interface status in the upper half of the screen and different messages from the real-time application in the bottom half.

The CPU load is important information. Percentages from Figure 16 show normal CPU operation. CPU 0 is an all-purpose/system CPU, and in normal conditions, its total load is up to 20%. CPU 1 is used only for streaming data to the PC. In normal operation, its load is not more than 30%. If the load is bigger than 30%, there is some problem with network communication. The data rate used for streaming is 28.8 Mb/s. A smaller or larger data rate is a good indication that something is wrong in communication between the PXI and PC. Generally, since the data frame is a fixed size, the data rate should not vary more than 1% for a longer time frame (more than a couple of seconds).

CPU 2 is used only for digital signal processing (DSP) functions. This core handles only PLL (frequency and phase) calculations, so the total load shouldn't be more than 5%. CPU 3 is used for sampling and downsampling the data. Data is sampled at 2 Msps/ch (mega samples per second per channel) and downsampled to 20 ks/s per channel. This is a very time-consuming job, so its load is around 70% in normal operating conditions. Since 2 Ms/s is the maximum sample rate achievable by the PXI, and since the application is sampling all available analog channels (16 channels), a load greater than 80% is not expected. If a total load of CPU 3 reaches 100%, the Real-Time application does not have enough time to sample the data, which indicates an internal problem.

3.3 Host Computer Software

The PC application called “IMU Control Center” is a graphical user interface (GUI) of the IMU.

Three main tasks of the software on the host computer are:

- control of the perturbation injection unit (PIU),
- receiving data from the PXI computer, and
- impedance calculation.

These tasks are divided into two applications: Measurement Control and Result Inspector. Figure 17 gives an overview of the software architecture designed for the host computer.

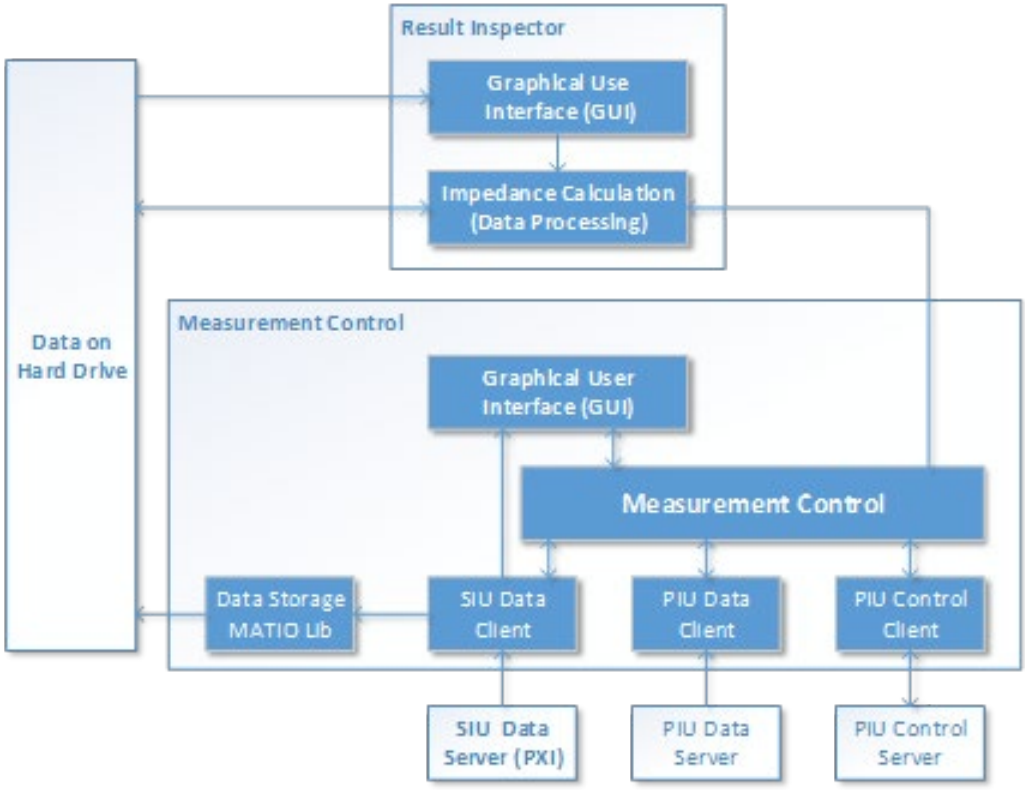

Figure 17: Software architecture on the host computer

3.3.1 Measurement Control Application

The Measurement Control application receives data from the PXI, stores data to a hard disk drive (HDD) in MATLAB's MAT file format, shows real-time waveforms of sensed signals, and shows data calculated locally from received data (ex.: rms values of current and voltage). All system states are displayed on a GUI. The NI LabWindows/CVI provides functionalities for fast GUI design [40]. The application, too, is designed using a multithread technique to improve the user interface response speed. In this case, locking a thread to a specific CPU is unnecessary, and so has not been done. Each task that requires a fast response has its own separate thread. Tasks that are always executed in a sequence are combined into one thread.

Figure 18 shows the main panel of the Measurement Control application. The window is divided into three general areas:

- PIU control (used for perturbation setup),
- Real-Time waveform plots (used for real-time visualization of voltage and current waveforms from all sensors), and
- different real-time data display (shows various information calculated out of data received from the PXI and PIU.

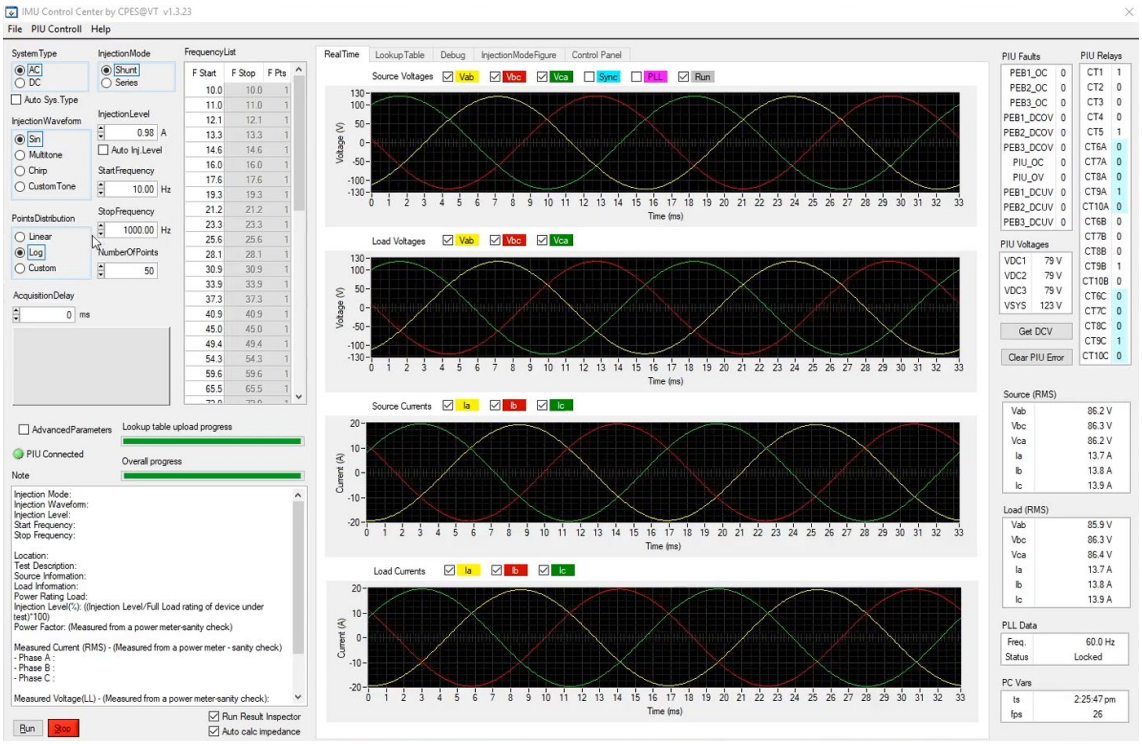

Figure 18: GUI of the Measurement Control application

3.3.2 PIU Control

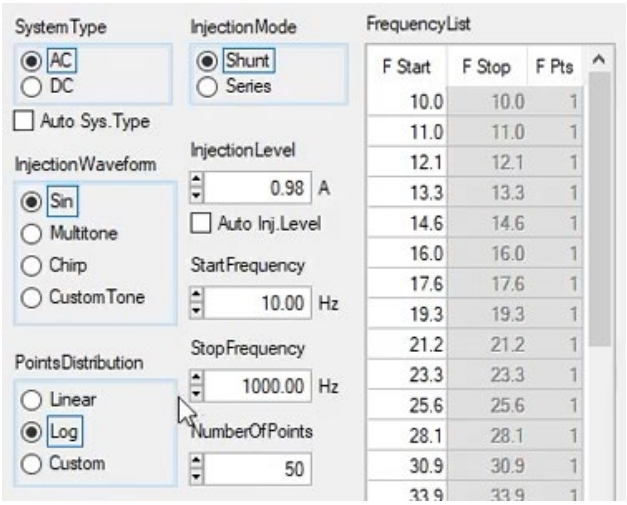

Figure 19: PIU control – detail

The PIU is used to perturb the system. Impedance is calculated out of the measured system response. The PIU receives commands from the Measurement Control application and sends back data, warnings, or error messages. Communication with the PIU is realized as block-based communication on top of the TCP/IP socket, since its footprint is small enough to fit the embedded PIU controller. The LabWindows/CVI provides an easy-to-use TCP support library, which involves a client and a server for each connection. In the case with this project, the PIU acts as a server and waits for the PC, as a client, to connect. A TCP client can send and request data to and from a server application [41].

The fundamental piece of the block is one word, which is 2 bytes long. Every block has a beginning word (BOB) and an ending word (EOB). The word following BOB is the command code. Parameters of the command follow the command code and end with the above-mentioned ending word. The size of the block is not fixed and can be determined by counting words between the beginning and the end of the block.

The PC invokes communication by sending the command to the PIU. For the PIU to read the command, the command has to have at least three words: the beginning of the block, the command code, and the end of the block. If the PIU receives a valid block and understands the command, it will adequately respond. It will respond with an error if a block or a command is not valid.

The example of pseudo-code in Figure 20 shows the basic block. Notice that this command does not have parameters.

```
1.  unsigned short sendBuf[3];
2.
3.  // prepare the block
4.  sendBuf[0]=BOB;  // beginning of the block
5.  sendBuf[1]=COMMAND;  // command
6.  sendBuf[2]=EOB;  // end of the block
7.
8.  // send block that is three words long
9.  ClientTCPWrite (&sendBuf, 3);
```

Figure 20: Pseudo-code that describes the PIU communication block

Some of the parameters that can be adjusted and sent to the PIU using the Measurement Control applications are:

- **System Type** – The PIU can perturb AC or DC systems. If "Auto Sys. Type" is selected, the application will automatically determine the system type based on the measured frequency received from the PXI. Otherwise, the user can manually determine the system type. If the system type is not chosen properly, the PIU will not proceed with perturbations, and the application will show an error message.
- **Injection Mode** – The injection mode can be either shunt or series. In shunt mode (Figure 21 (a)), the PIU will perturb between two phases (A and B). In series mode (Figure 21 (b)), the PIU will connect itself in series between the source and load.

By choosing one of the modes, the PC tells the PIU how to reorganize built-in relays to achieve the desired configuration. The PIU has feedback from these relays, knowing its configuration, and sending feedback back to the PC. If the desired mode is not achievable, the PIU will send a warning, and the PC will, through the GUI, inform the user of the new condition.

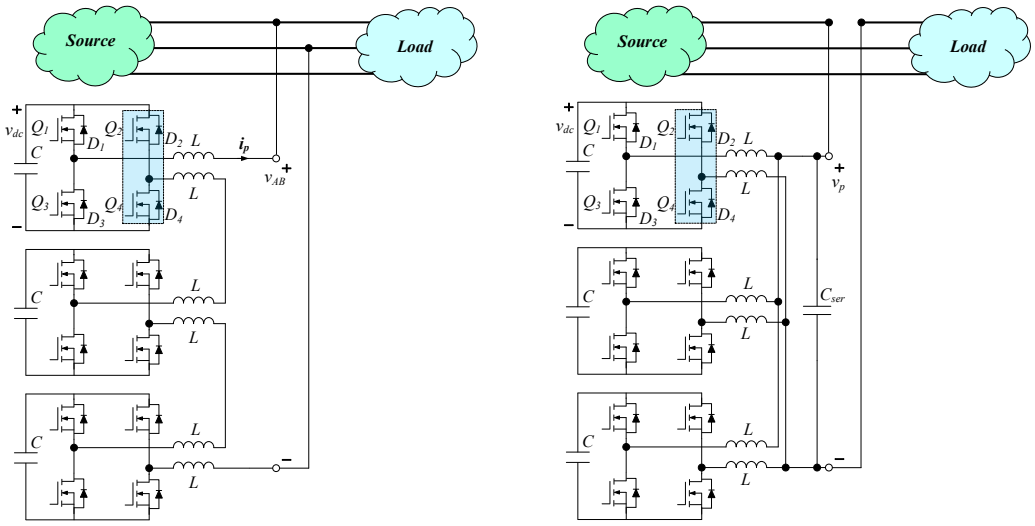

Figure 21: Perturbation mode: (a) AC shunt (current) mode, and (b) AC series (voltage) mode

- **Injection Waveform** – Different waveform types (sine, multi-tone, chirp, or even custom/user-defined tone) can be used for perturbation. Sine, multi-tone, and chirp signals are designed to calculate impedance from the measured voltage and current responses caused by the perturbations. Additionally, the custom tone can be used for further research. There is no additional tool for designing custom tones. The system accepts a CSV (Comma-Separated Values) file with custom tone time-domain data. CSV files can be easily generated using MATLAB or EXCEL.

The sine is a simple sinewave, and from the frequency domain perspective, it is a single frequency signal. To measure impedance, the system has to run perturbation for the whole spectrum (frequency sweep) from “Start Frequency” to “Stop Frequency.” The user can choose the start and stop frequency of the sweep, the number of frequencies, and the distribution of points. A higher number of frequencies means better measurement resolution but a longer measurement time.

For online impedance measurement in AC systems, the widely used sine (or frequency sweep) method takes a long time and may not be practical in systems where the operating point cannot be maintained for a long time. A wide bandwidth chirp signal can significantly reduce the measurement time. The chirp signal is going to be split into chunks. The point where the chirp signal will be split is determined by system frequency and harmonics [42]. For example, if the system frequency is 60 Hz, the chirp signal is split into chunks from 10 Hz to < 60 Hz, then > 60 Hz to < 120 Hz, then > 120 Hz to < 180 Hz, etc. So, using the chirp signal for perturbation, the 1 kHz spectrum is divided into ~17 chunks, and one chunk is one measurement; this means that using the chirp signal, 17 measurements will achieve a resolution similar to a sine sweep of 990 measurements.

A similar case exists with the multi-tone signal. Technically, a chirp signal has an infinite number of frequencies, but the multi-tone signal has a finite number of frequencies. For example, suppose the perturbation has been done with a multi-tone signal consisting of three frequencies. In that case, the measurement will be three times faster, or the measurement resolution will increase three times.

- **Injection Level** – The magnitude of the perturbation is automatically set to 5% of the measured system voltage or current but can't be set higher than 10% of the system voltage or current, depending on the mode in which the PIU is operating.
- **Frequency List** – The list of frequencies of all perturbation signals that will be injected into the system.

  The first frequency is determined by the "Start Frequency" field. The smallest start frequency for this IMU's rated condition is 10 Hz. The "Stop Frequency" field shows the frequency of the last signal that will be injected. The largest frequency for this IMU's rated condition is 1 kHz. The "Number of Points" field is used to determine the number of signals injected into the system. A larger number of points yields better frequency resolution but a longer measurement procedure. The "Points Distribution" field allows the user to choose between a linear or logarithmic distribution of points. If the "Points Distribution" is set to "Custom," the user is allowed to change the frequencies in the frequency list.

3.3.3 Receiving and Storing the Data

The central area of the GUI (Figure 22) shows real-time waveform plots. Voltage and current waveforms from all sensors are shown in real time, achieved using the LabWindows/CVI graph control library. Graph controls are fast and contain various plot types representing different types of data [43].

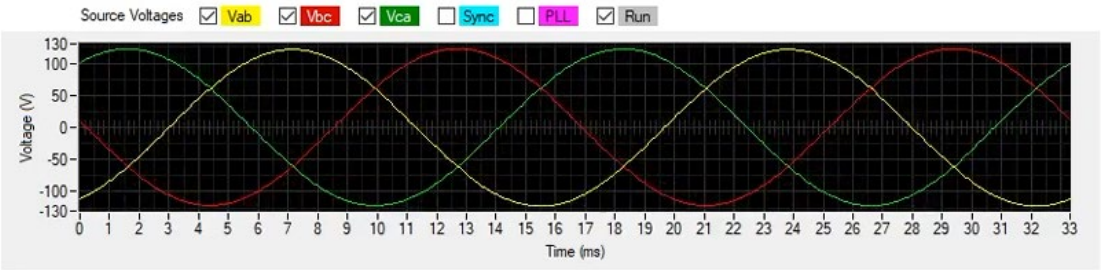

Figure 22: Real-time waveform plot (source voltage waveform)

There is a group of checkboxes above each waveform plot. The first three checkboxes (yellow, red, and green) are used to turn the plot of each phase on or off. All

other checkboxes are unique only for the first graph. If the PLL checkbox is checked, the phase plot will be added as the fourth plot. The same is true for the Sync signal, which is used to send a synchronizing signal from the PIU to the PXI each time a perturbation is started.

The far-right partition of the Measurement Control GUI (Figure 23) shows different useful real-time data from the PIU and PXI, such as RMS voltages and currents of each phase, DC-link voltages, system frequency, PIU relay status, error status, etc.

The **PIU Faults** table shows different faults received from the PIU. If the value in some cells is 1, there is an error. A pop-up window will appear to inform the user whenever an error occurs, and the error will be logged. The **PIU Relays** table shows the status of different contactors and relays controlled by the PIU. The configuration of these relays is determined by the chosen Injection Mode and the System Type. The **PIU Voltages** table displays DC voltages for each capacitor bank (VDC1, VDC2, VDC3) and system voltage (VSYS) measured by the PIU. Data in these three tables will be automatically refreshed if the PC and PIU establish communication. If no communication is established, all numerical values will show 0 (zero), and all descriptive (textual) values will show N/A.

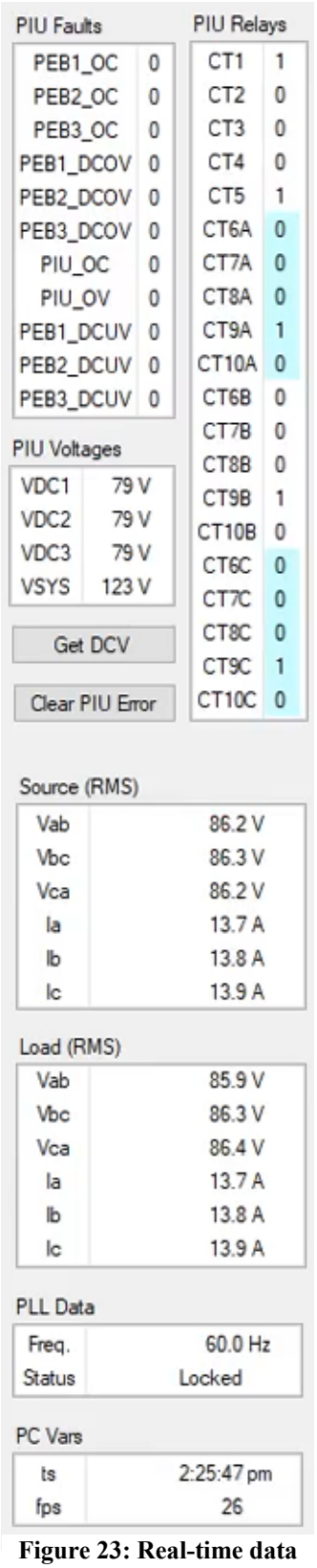

Figure 23: Real-time data display

The "**Get DCV**" button is used to manually force the PIU to send data and error codes to the PC's standard input output log.

System data is received from the PXI. This is data from 12 sensors that measure voltages and currents on the load and the source side. Establishing communication between the PC and PXI is automatic, and data will be shown if communication has been established. The **Source (RMS)** table shows voltages and currents on the source. Voltages are measured between two phases, and currents are measured at each phase. The **Load (RMS)** table similarly shows voltages and currents on the load side. The **PLL Data** table shows the status and frequency calculated by the PXI. In normal operating conditions, the PLL should lock the phase in a couple of seconds.

DAQ is the most time-sensitive task on a PC. To avoid losing sampled data, before new data arrives and overwrites it, old data has to be saved to the HDD. This must be done promptly without distraction from any other working process. A dedicated thread is assigned for this purpose to guarantee no data loss. It receives sampled data from the PXI using the NI network streams protocol. This thread has the highest priority, so any other running thread has to stop execution when this thread is running. Data is stored to a reserved memory area (buffer). When a certain amount of data is collected, it is passed directly to the GUI and displayed. For communication between threads, the TSQ is used. The same technique used in the PXI for thread communication is used for the PC's thread communication. As shown in Figure 18, the system voltage and current can be observed directly on the GUI.

Another thread is responsible for writing data to the disk. This thread waits for a certain amount of data, and when data is delivered, the thread will write it to the disk. To

guarantee uninterrupted data collection, storing the data is done in a separate thread because disk access is a time-consuming job. There are two ways of writing data to the disk: direct and sequential. Disk access is very efficient in sequence mode. That's why a batch of the data is stored in the buffer and is then written to the disk all at once. This is another reason to pause writing to the disk until all data is collected.

The data format for storage is selected to be MATLAB's 'MAT' format, which will later be used as an interface for impedance calculation.

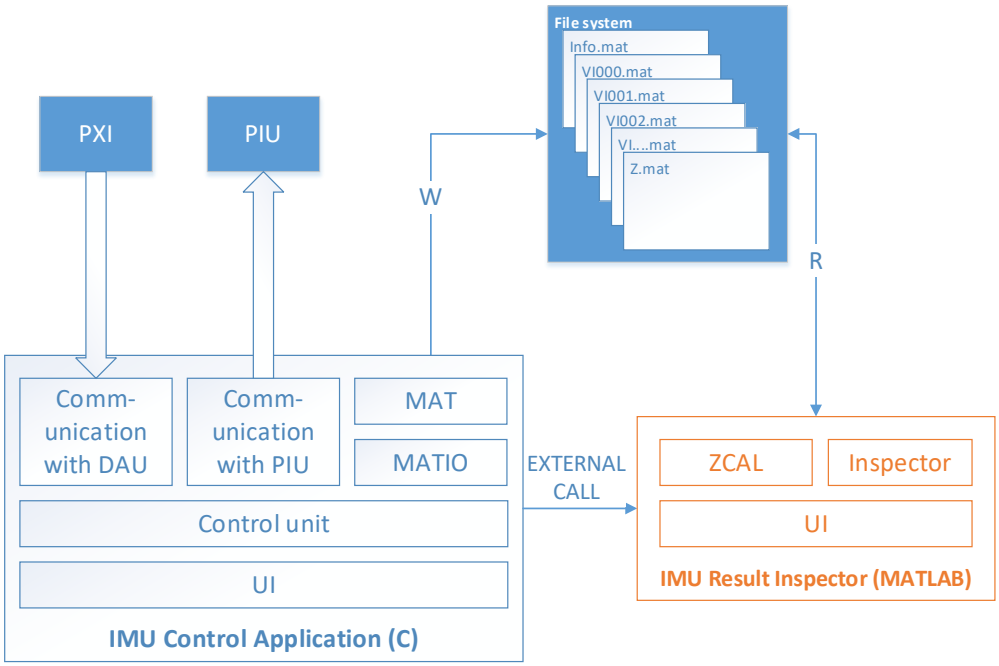

Figure 24: Relationship between C and MATLAB application

There are several ways of calling MATLAB functions using a C application. MathWorks provides two interfaces for this purpose. The first interface is MATLAB Coder. It generates C code that implements the same functions as the MATLAB scripts. The other interface for C code to access MATLAB functions is through the MATLAB

Compiler. It generates a dynamic link library (DLL) file that encapsulates all the MATLAB scripts and provides several interfaces for the C file to call the functions. Both described methods work, but in this case, a different approach was used. The MATLAB code was decoupled from the C code. To achieve that, the C application (Measurement Control) was designed only for the purpose of inspecting results. DAQ, data storage, and the MATLAB application (Result Inspector) were designed for impedance calculation and data presentation. So, two applications, one written in C and one written in MATLAB, work together (Figure 24). In this approach, the C application does not use MATLAB libraries. When the Measurement Control application reads data from the PXI, it stores the data to a MAT file using the Matio [44] external library. When the measurement is finished, the control unit begins (external/system call) the Result Inspector application.

The MATLAB file format has been used to make data available for future research or later improvement of impedance calculations. So the decoupling of the code followed the same thought: one application for DAQ and another for data processing. To write MAT files from C code, this project uses the library called Matio, which is an open-source C library for reading and writing binary MATLAB MAT files [44] . This library is designed for use by programs/libraries that do not have access to or do not want to rely on MATLAB's shared libraries.

The collected data is saved to disk as soon as the measurement of one frequency chunk is finished. The data storage path is passed to the Result Inspector application as information where are data is for processing. This approach is suitable for offline data processing, which is the case in this project.

3.4 Impedance Calculation

After the data-logging unit collects the data, digital signal processing must be applied to extract the impedance information [45] [46]. MATLAB is well suited for data processing, so MATLAB is used for this purpose instead of using the C programming language used to develop all other parts of the software. MATLAB is known and used by lots of researchers, and since the data format for storage was selected to be the MATLAB standard 'MAT' format, this can be easily imported back into MATLAB for further processing.

The Result Inspector application calculates and shows the measurement results. It is called automatically after the measurement procedure is done. Since Result Inspector was designed to be a standalone program, it can be executed independently from the Measurement Control application.

When the program is started, an empty window will show. By choosing File -> Open, the dialog box for project/folder selection will appear. After choosing a folder with measurement data, the application will show time-domain data, load and source impedance data, and stability plots.

3.4.1 Result Inspector Main Menu

- File
    - Open – Open the IMU project folder
    - Scan Workspace – Scan folder for IMU projects
    - Merge Impedances – Merge data from different measurements
    - Exit – Exit application
- Settings
    - Curve Fitting – Valid only for impedance plots
        - Linear – Linear curve fitting
        - Pchirp – Pchirp interpolation
        - Spline – Spline Interpolation
    - Show Measurements – Turn on or off the measurement's point mark.
- Help
    - About

The s**can workspace** option is used for scanning one folder or even the entire HDD for IMU projects. After choosing a folder, scanning will start. When finished, a table with basic information for all the projects found will be shown (Figure 25). The project can be opened directly from this table.

To merge data from two or more projects, the user should choose the option **Merge Impedances**. The application requires that all projects for merging are in one common folder. After choosing a folder with multiple projects, the application will look for

measurement data and will calculate impedances out of all data found in the main folder and all subfolders. So, before calculating impedance out of multiple projects, the user must ensure that only the projects to be merged are in the folder.

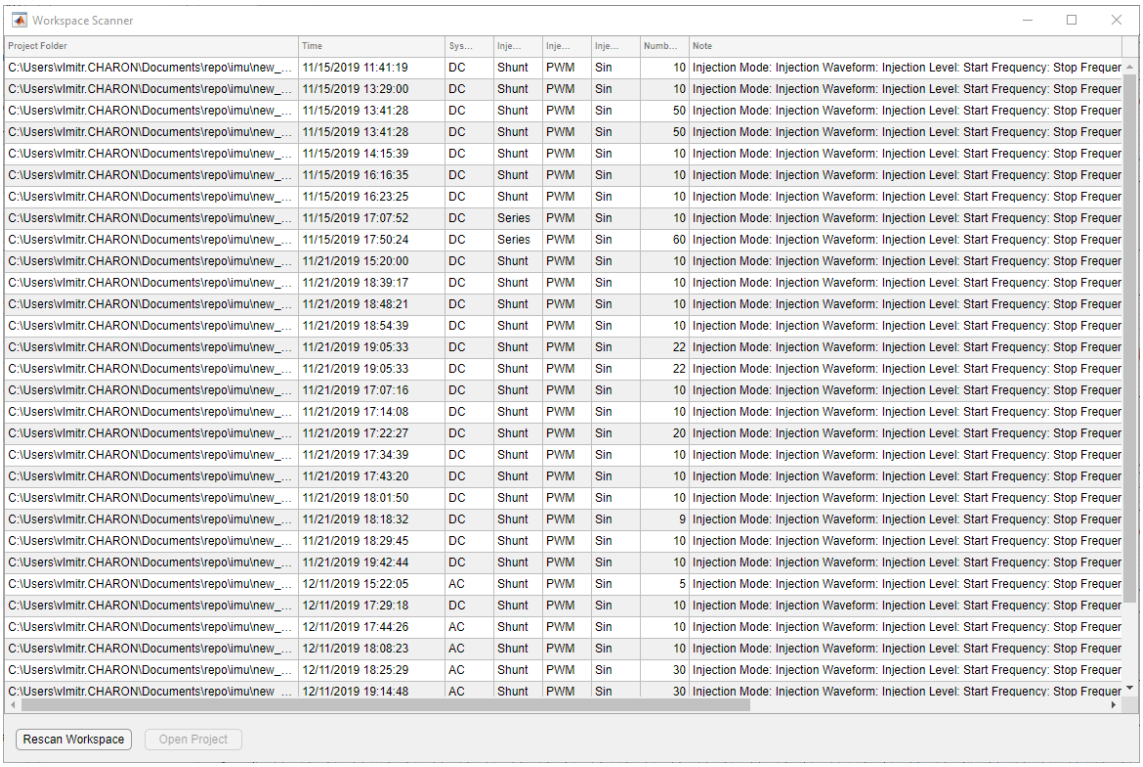

Figure 25: Workspace scanner.

For each perturbation, the user can analyze time-domain data. Data is shown in the Perturbations tab (Figure 26). On the far left-hand side is the Overall Info area, which shows data related to the project. Data can't be altered in this application. It is used only for impedance calculation and data presentation.

The Perturbations tab shows:

1. List of perturbations (perturbations table). By selecting different perturbations, data will be shown for that selected perturbation.
2. Information about each perturbation (segment info) and
3. Plots of:
    a. Source voltage and current,
    b. Load voltage and current,
    c. Frequency, and
    d. Phase.

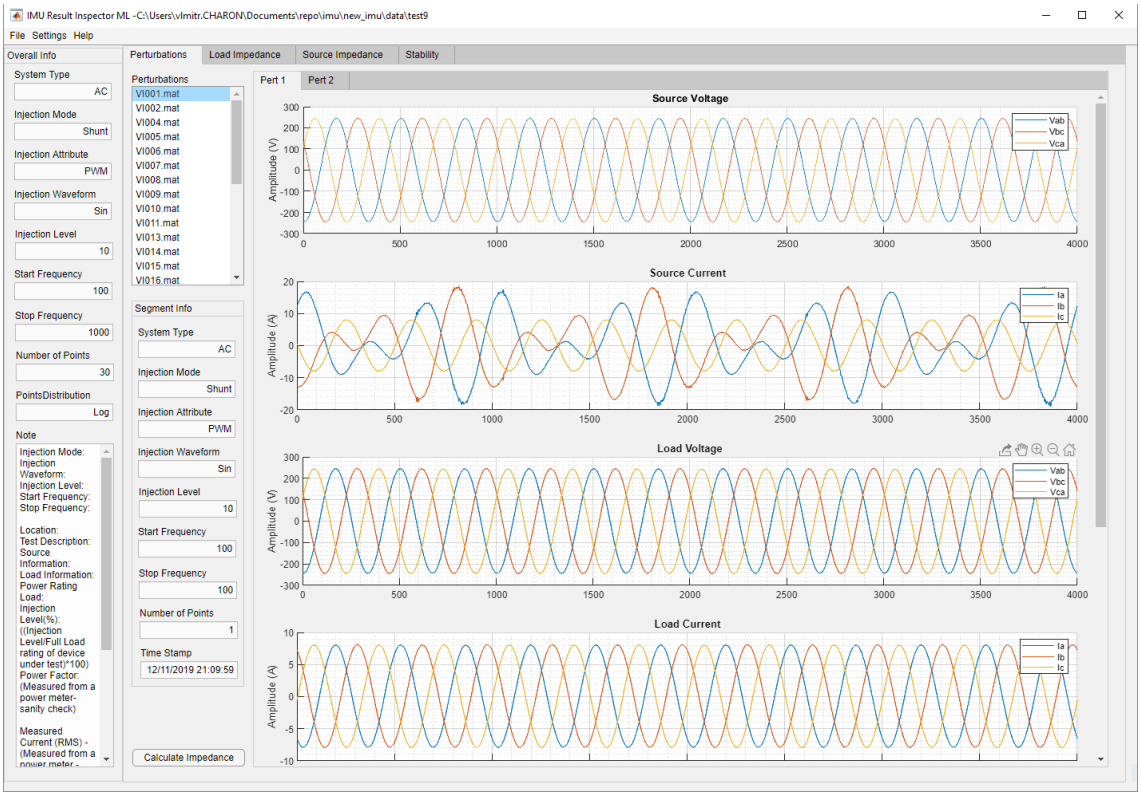

Figure 26: Time-domain data.

Each plot can be zoomed in and out, paned, and exported as jpeg, png, tiff, pdf, or MATLAB/Simulink *.fig file. Also, data can be copied to the clipboard as a raster or vector image.

If the Impedance and Stability tabs are disabled, there is no impedance data. To force impedance calculation, the user should click on the button Calculate Impedance. After successful impedance calculation, the application will show those tabs.

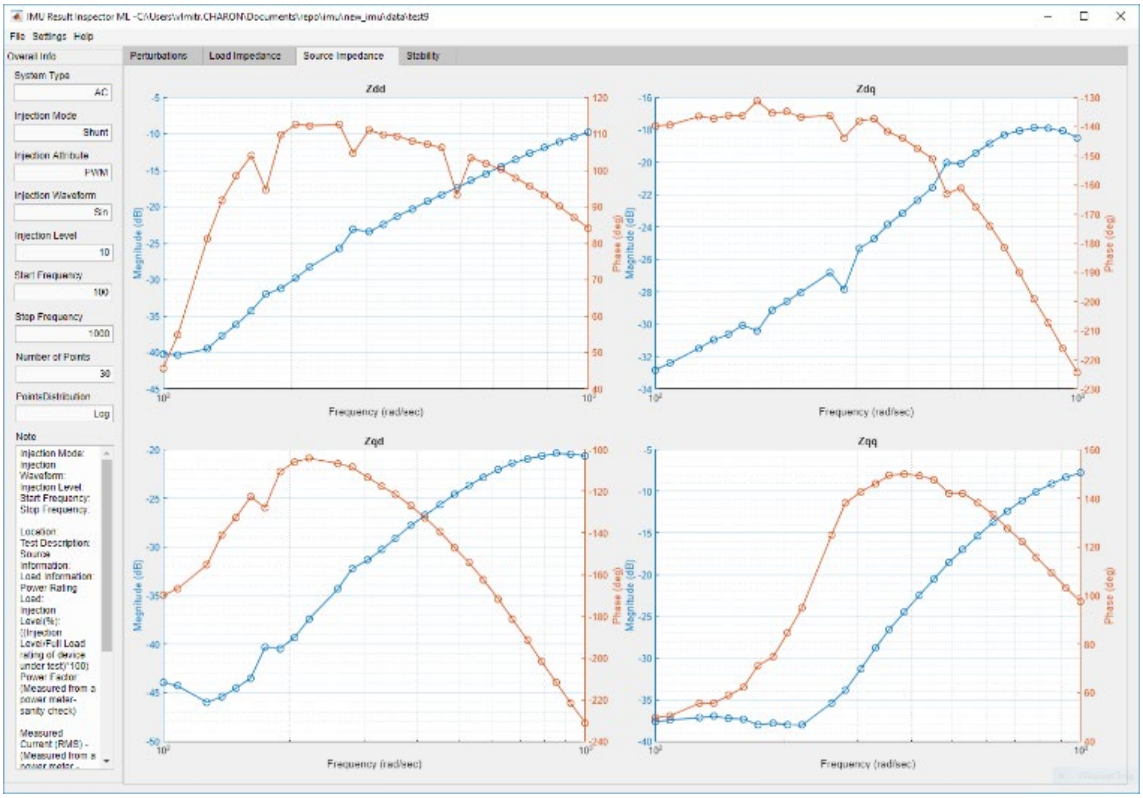

Figure 27: Impedance plot.

When calculated, the impedances of the system can be seen from the Load Impedance and Source Impedance tabs. The AC system impedance is shown in the DQ

frame (Figure 27). Nyquist and Bode plots of the eigenvalues are shown in the Stability tab (Figure 28).

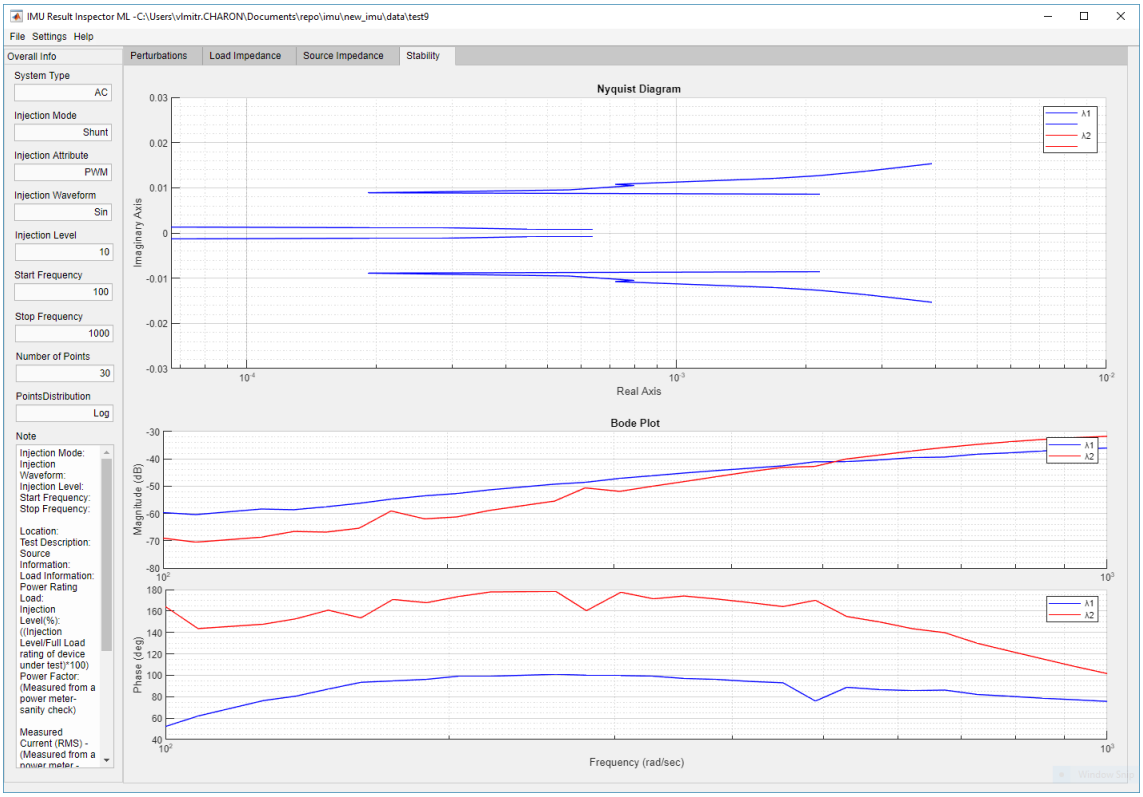

Figure 28: Stability plots.

3.5 Conclusion

Different software and communication techniques for data acquisition and streaming are presented in this chapter. Most of them are based on high-level software libraries like NI-DAQmx for sampling data (digital communication between DAQ card and PXI CPU), TSQ for inter-process communication, or Network Streams for peer-to-peer network communication. Also, useful signal processing techniques were mentioned to help reduce noise, mitigate aliasing issues, or even reduce the amount of bandwidth

needed for transferring data among different threads in a real-time computer (PXI). The mechanism of locking threads in PXI's real-operating system brings more control over the execution of different processes. Organizing the code in logical blocks gives more flexibility when developing the application for data acquisition and data processing. Overall, all these techniques helped develop data acquisition software for power system impedance measurements, although these techniques can be applied in any other kind of real-time data acquisition.

Chapter 4 Communication Inside the PEBB

4.1 Introduction

The most common approach for controlling gate drivers (GD) in a power electronics system is to send a pulse width modulated (PWM) signal from the controller to the GDs (green lines shown in Figure 29) and to do ADC conversion of signals from sensors as feedback to apply control (yellow lines shown in Figure 29). The GD fault is signaled using separate chanel (red lines shown in Figure 29).

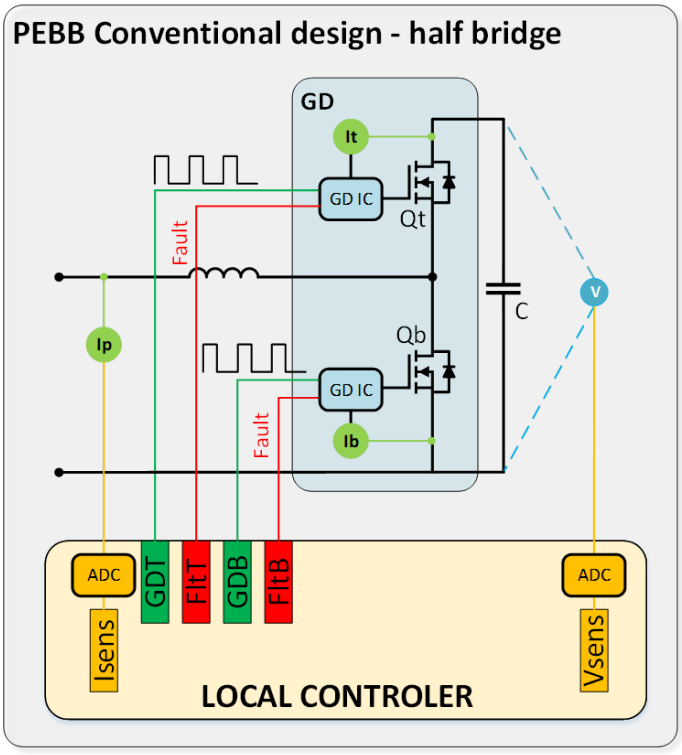

Figure 29: Conventional design of the half bridge converter

The problem with this approach is that it is not modular nor flexible, since the controller usually has a limited number of inputs for sensors and a limited number of outputs for GDs, which is especially problematic with modular multilevel converters (MMC) when the number of GDs dramatically increases. Additionally, modern GD modules have voltage, current, and temperature sensors for protection purposes, but that information cannot be used for anything else with the traditional approach.

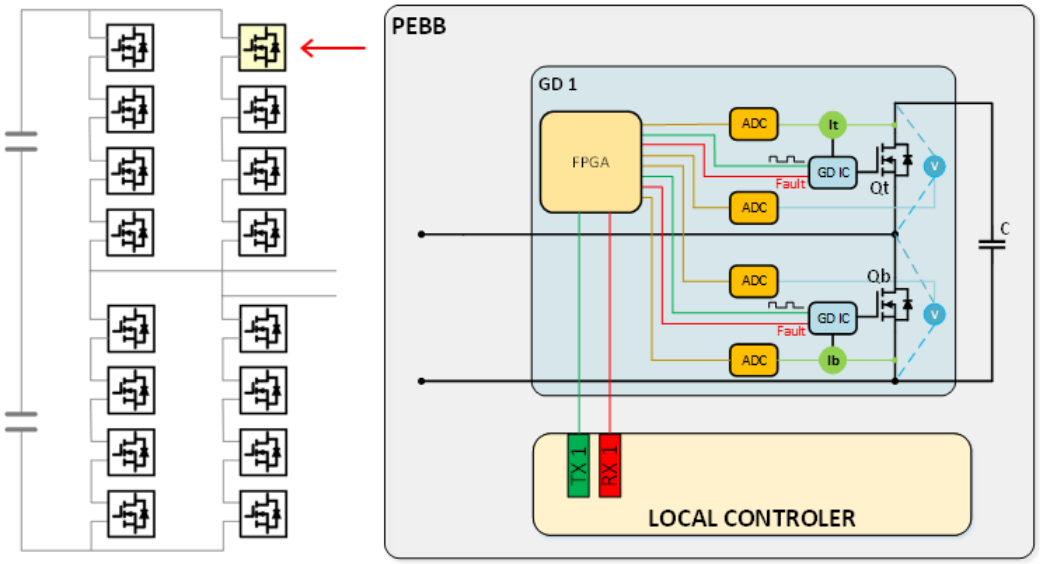

Figure 30: Communication system inside power cell.

In the communication system inside the power cell shown in Figure 30, the digital controller communicates with GDs (on one side; TX1 and RX1 lines) and another power cell (on the other side), forming a distributed control and communication network.

Communication inside the power cell brings:

- **Flexibility** - Compared to the most common approach where modulation signal is sent to the GD directly from the controller, in this case different data could be exchanged with a GD. For example, the turn on and turn off time or I_{ds} and V_{ds_on}

so that R_{ds_on}, could be calculated for estimation of the remaining useful lifetime of the device.

- **Distributed control** – The GD could be equipped with a powerful enough FPGA (Field Programmable Gate Array) that a part of the control algorithm can be executed in the GD.
- **Hardware abstraction** - Provides abstraction of the power stage's terminal behavior independent from hardware design.
- **Hardware simplification** - Since enhanced GDs are equipped with current, voltage, and temperature sensors, that information could be reused.

4.1.1 Overview of the Most Popular Industrial Real-Time Communication Protocols

Before developing a new communication protocol, several real-time industrial protocols were compared in hopes of choosing and adopting one.

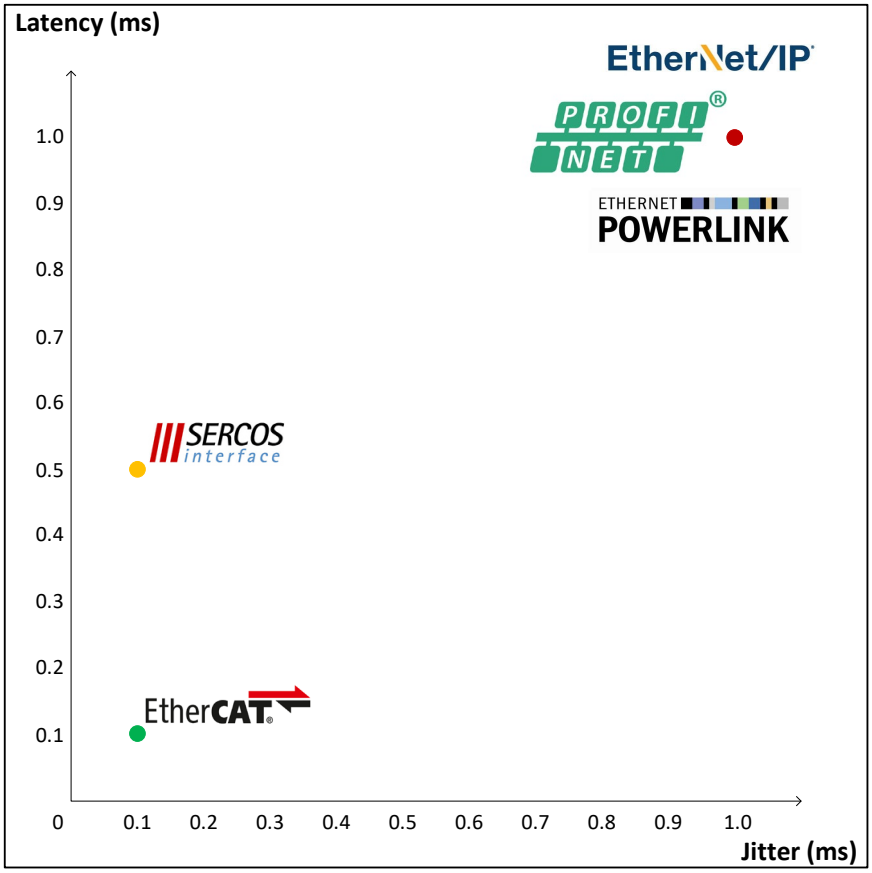

Figure 31: Real-time industrial protocols comparison

Figure 31 compares a jitter (the variation in response time) and latency (response time/cycle time) of several industrial real-time communication protocols in an application. In this case 100 nodes are controlled synchronously [47]. Ethernet is the foundation of the existing real-time communication protocols. The communication speed is 100 Mbps or

faster. The latest version of EtherCAT (Ethernet for Control Automation Technology) supports speeds as high as 10 Gbps [48].

A communication is used in MMCs, because of the way how MMCs operates. RapidIO and CAN, as a high-speed serial communication protocols are often used in MMCs [49]. Problem with those communication methods it that they require extensive wiring and higher-cost devices. On the other hand, the distributed control using CAN-bus can be implemented for the MMC with a few submodules, but the problem is that it is not scalable and cannot be used to control a high number of submodules.

Some research [50] describes a 31-level BTB (Back-to-Back) MMC system developed using an EtherCAT communication protocol. To evaluate the performance of both the communication and the control algorithm in the developed hardware prototype, two algorithms (directly modulated control and a group sorting) are implemented. The experimental results show that EtherCAT communication can be used to accomplish the master-slave data exchange without a need for complex wiring. More important is that, extending the MMC system to a higher level can be accomplished easily. Additional modules has to be connected to the ethernet line and EtherCAT data frame has to modified in the EtherCAT master controller code.

A hierarchical control algorithm for MMC is presented in [51]. The main goal of the algorithm was on reducing the amount of data exchanged inside the MMC during a sampling period. The grid-connected low-voltage MMC setup is used for testing the method. The EtherCAT is used for internal communication. The experimental results show good performance.

Three high-speed (100 Mbps) control networks (EtherCAT, MACRO and PROFINET IRT) for use in a three-phase MMC are analyzed and reviewed in [52].

MACRO is developed by “Delta Tau Data Systems” to connect amplifiers, multi-axis motion controllers, and I/O. The researchers from Virginia Tech used MACRO as foundation to develop Power Electronics System Network (PESNet) protocol in 1999 [52]. PESNet is a custom-made communication solution for power electronics (PE) converter systems used fot data exchange control between a digital controller and power stage [53] [54].

Siemens developed PROFINET IRT. One of it’s special function is isochronous data transmission used in high-accuracy closed-loop control tasks. It has redundancy which is very usefull feature in the PE converter system. Special hardware support is required on both master-slave devices to support PROFINET IRT [55].

EtherCAT is an ethernet-based fieldbus communication protocol. It is developed by Beckhoff Automation. The protocol is standardized in IEC 61158. It is used for automation technologes where real-time computing is required. Main goal of EtherCAT was to apply ethernet protocol for automation applications and to have as short as possible data update times (≤ 100 μs) with as low as possoble jitter (≤ 1 μs). [56]

EtherCAT, does not receive, interprete, and copy ethernet packet as a whole in every node. Instead, processing starts as soon as receiver starts receiving data. The EtherCAT is processing data "on the fly". The node read the data addressed to it while the telegram passes through the device. Input data are inserted while the packet passes through. This way, one frame could be used to address the whole network.

The relevant data is extracted from incoming telegrams directly, and new data is inserted at the place of extracted data and transmitted to the next slave. The last node will forward the full processed telegram to the master. EtherCAT provides ethernet checksum , faults detection, and distributed clocks with jitter of less than 1 μs [57].

Different network topologies could be achieved with EtherCAT. Some of them are: line, tree, ring, star, or any combination. Research [58] shows that EtherCAT outperforms PROFINET IRT and EtherNet/IP from a control performance perspective. However, even though EtherCAT has the best jitter performance compared to other available RT protocols, it could be a limiting factor in the converter design [59] [60].

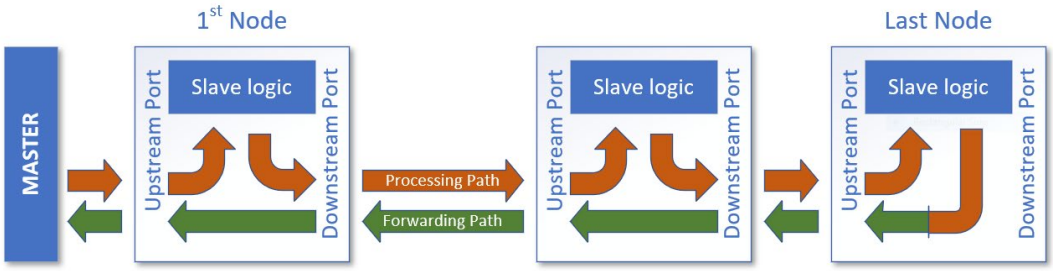

Figure 32: EtherCAT working principle

Fault-tolerant control for MMC based on EtherCAT is presented in [60]. Because of the high redundancy design of the MMC the converter will operate even after a sub-module failed. The EtherCAT has been successfully implemented. Two main features that enables this was its fault-tolerant operation and the good synchronization of nodes in network. The system has a central and a local controller. The central controller is responsible for the current and averaging control. Local controllers are doing the

modulation and capacitor voltage balancing. Experimental data show that the converter tolerates hardware and communication faults.

4.1.2 Synchronization Accuracy

Synchronization between nodes in the communication network is important part of real-time control. Synchronization protocol called “Distributed Clock” is inbuilt to EtherCAT and used to minimize latency and jitter. Research [61] has described the development of an MMC with EtherCAT communication. The amount of synchronization jitter claimed in the paper is ± 20 ns. Described modulation technique, the synchronization protocol and comunication path made decentralization of control tasks possible. Reducing the processing power of the main controller is one of the benefits of decentralization.

Similar results are achieved using the custom communication protocol [54]. The three nodes network is arranged in a ring topology. The the jitters of first, second and third nodes were 15, 20 and 25 ns, respectively.

Using a tree network topology based control system for MMCs while implementing a customized protocol designed with a specific clock multiplier chip is a way to achieve ±1.5 ns synchronization accuracy [62]. The paper states that tree-shaped topologies are superior to conventional daisy chains and ring-type networks. It was also shown that the high-speed clocks that are inherent to such links could be used to synchronize network nodes with an accuracy of ± 1.5 ns, leading to a total timing accuracy superior to ±4.3 ns between any two PWM signals across the network.

Other research [63] shows a high-performance distributed power electronics communication network design with a 5 Gbps data rate and sub-nanosecond synchronization accuracy. The synchronization method is based on PTP (Precision Time Protocol) and enhanced with phase detection. That eliminates the synchronization accuracy limitation of having only one clock cycle. Compared with the original White Rabbit (WR) protocol [64], this paper simplifies the offset clock regulation by adopting a PLL IP core instead of a closed-loop PLL [65]. Sub-ns synchronization accuracy is experimentally validated in a four-cell 6kV SiC-based modular converter.

WR is a high-precision time-frequency synchronization technology based on synchronous Ethernet (SyncE), PTPv2, and digital dual-mixer time difference (DDMTD). Using a DDMTD phase detector enhances the accuracy of time synchronization to less than nanoseconds. Further possible improvements to the WR synchronization network are presented in [66]. The clock offset and frequency offset are estimated and applied to the correction of the slave clock. Experimental results show that this method can significantly improve the synchronization accuracy of the master-slave clock, reduce the frequency deviation of the master-slave clock, and reduce the hardware implementation complexity of the WR network while achieving nanosecond-level high-precision synchronization.

Ethernet standards and the EtherCAT concept heavily influenced the development of custom communication protocols. Also, communication and synchronization heavily rely on experiences learned from [67].

4.2 Communication System Inside the Power Cell

Figure 33 shows the internal structure of one power cell. As seen, it consists of two GDs for top and bottom devices, voltage, temperature sensors, and the local controller. Power cells could be combined to achieve the desired topology. A clock for synchronization purposes is not regenerated from communication, but it is physically distributed between the local controller and the GDs. This approach offers good synchronization using relatively inexpensive communication hardware.

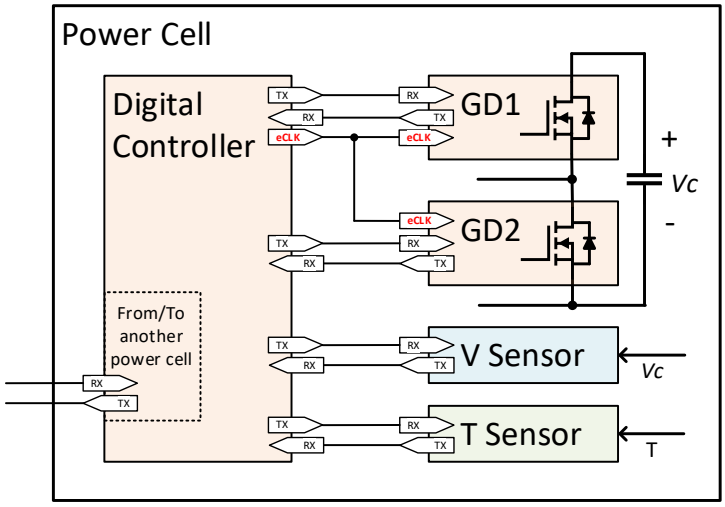

Figure 33: Communication system inside a power cell

The goal was sub-nanosecond accuracy using slow (50Mbps) transceivers and a decent FPGA. a new communication protocol has been developed. The development of such a protocol brought different challenges, the biggest of which was synchronization. To achieve very good synchronization, all elements in the signal chain related to communication and synchronization must have known and predictable latency. For

example, some FIFO buffers in line with TX (transmit) and RX (receive) blocks had unknown variable latency and caused unacceptable amounts of jitter. A carefully planned signal path and choosing or rewriting needed IPs for FPGA led to success. The clock signal is physically shared from the main controller to the GD to achieve a sub-nanosecond synchronization jitter. This way, good syntonization has been achieved, which leads to good synchronization.

4.2.1 Features of Custom Communication Protocol

The main features of the custom communication protocol are:

- **Real-time** - Data must be exchanged in the specified time frame (one control cycle).
- **As fast as possible communication speed -** In the case with this project, the communication speed is 25 Mbps, and it is heavily constrained by used transceivers (PoF) and FPGA (MAX10). Speed is very important in the realization of real-time protocol.
- **Shortest possible package length** - For that purpose, the 10BASE-T ethernet protocol has been adopted. A short package has more chances to be received at least once in one control cycle.
- **Free of electromagnetic interference (EMI)** – To reduce the influence of EMI on data integrity, plastic optical fibers (POFs) have been used.
- **The communication protocol used for control purposes must provide good synchronization between nodes** - In this case, it is ~1 ns. This means that the GD

can execute commands received from the controller at the exact moment with a variation of ~1 ns.

- **Fast and asynchronous fault broadcasting** - Fault information should not be packed in the communication packet. Thus, the fault information will be delayed at least one control cycle. The idea of fast fault broadcasting is to use the same communication line differently to broadcast fault as quickly as possible. This way, the fault information will be received within one control cycle.

The communication is used for sending information to the GD on how to behave and receive information about the state and measured device current (Ids) and voltage (V_{dson}) from the GD. For example, in simple hysteresis control, the GD will receive reference current, and it will internally control switching devices without intervention from the controller.

4.2.2 Physical Layer – POF Transceivers

To reduce the influence of EMI and provide a high level of voltage isolation, the physical layer for communication is fiber optic. 50 MBd (megabaud) POF transmitter (T-1624Z) [68] and receiver (R-2624Z) [69] are used in this case. The POF transceivers utilized in this work allow a maximum of 50 MBd. Since one bit per baud will be transmitted, the maximum possible communication speed is 50 Mb/s. Due to the chosen communication protocol and internal clock arrangement, the real communication speed is 25 Mbps.

4.2.3 Encoding/Decoding Data

Following the idea of using a well-defined standard communication protocol and with the given transceiver constraints, a custom communication protocol based on the ethernet 10BASE-T standard for 10 Mbps ethernet has been created. Manchester code, also known as phase encoding (PE), is used for encoding data. It is a line code in which the encoding of each data bit is either low then high or high then low for equal time. Manchester coding is a special case of binary phase-shift keying (BPSK), where the data controls the phase of a square wave carrier whose frequency is the data rate [70]. Manchester code ensures frequent line voltage transitions directly proportional to the clock rate (Figure 34); this helps clock recovery. The clock is embedded in data, but it is not used since a more accurate clock is needed. Manchester is used in the 10BASE-T standard for 10Mbps ethernet. Encoder realization is a simple XOR circuit in which serial data is XORed with the clock. The decoder could be realized using PLL or by oversampling the input signal, as is the case here.

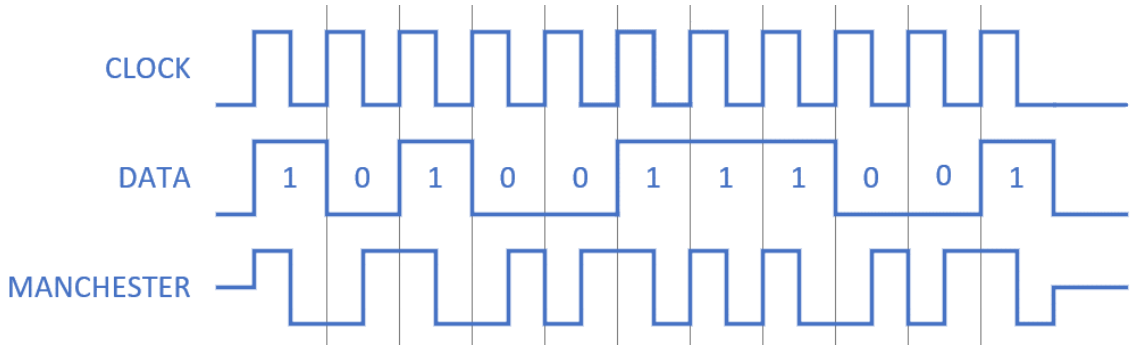

Figure 34: An example of Manchester encoding [70]

The layer 2 ethernet frame has been used as a reference. Real-time communication means that it must happen in a given time frame. In the case with this work, all communication should be completed before the next switching cycle. If communication is fast enough and the packet is short enough, there will be more time for calculation. To achieve real-time communication, the standard 10BASE-T packet is rearranged. The preamble is shortened to 3+1 octets, and the payload is fixed to 12 octets. Differences are shown in

Table 1. The speed is increased from 10 Mbps to 25 Mbps. This way, an exchange of ~15 packets per 10 kHz switching cycle is achieved. The communication protocol is tested on switching cycle control (SSC) and integrated capacitor blocked transistor (ICBT) controls, and all needed data are transmitted in time. The first packet is always dedicated to synchronization.

Table 1: Standard 10BASE-T and custom communication protocol comparison

	Standard 10BASE-T	Custom Protocol
Transmission speed	10 Mbps	25 Mbps
Preamble + SOF	7+1 Bytes (Octets)	3+1 Bytes (Octets)
Payload	64 – 1552 Octets (Var. Size)	12 Octets (Fixed Size)
Frame check seq.	CRC32	CRC32
Encoding/Decoding	Manchester	Manchester

The package carries a payload between the GD and the controller. The payload is divided into three subpackets, as shown in Figure 35:

- **PTP** – precision time protocol,
- **Timestamp** – used for carrying a timestamp for a message, and
- **Message** – the combination of function and data that is going to be executed at the GD. The Modbus [71] protocol inspires this message format.

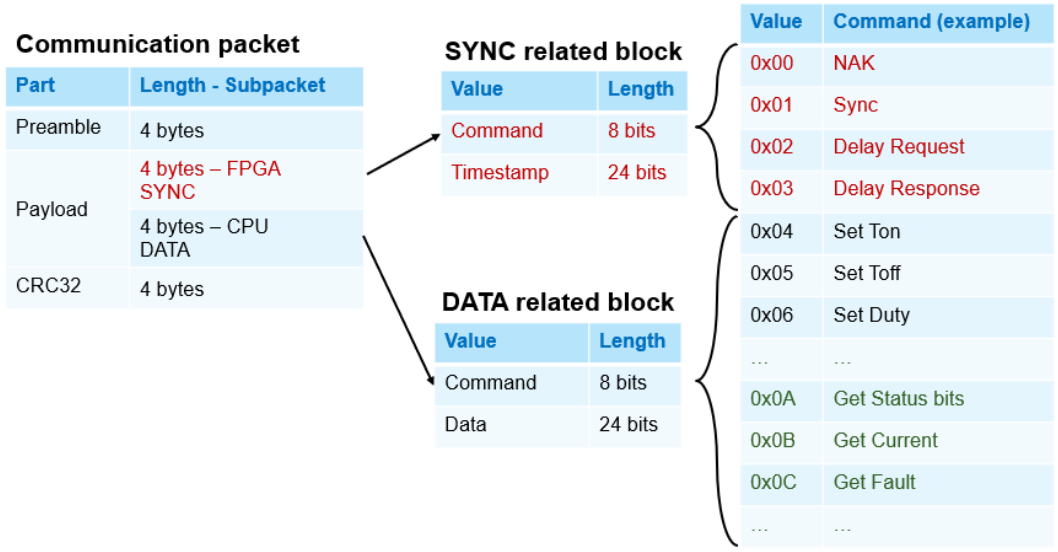

Figure 35: Example of the packet structure

Figure 36 shows the communication packets and how long the overall communication lasts. There is enough time between the last packet and the first packet in the next sequence.

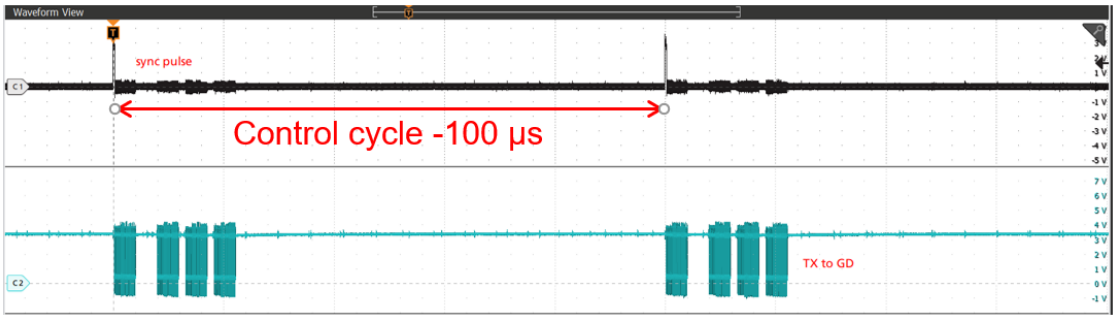

Figure 36: Communication packets in time

4.2.4 Synchronization - PTP

To achieve control, the nodes must be synchronized, which means that the controller and GDs must have synchronous timers. The clock signal is physically distributed using an additional POF transceiver, as shown in Figure 37. This way, syntonization is achieved between the controller and the GDs.

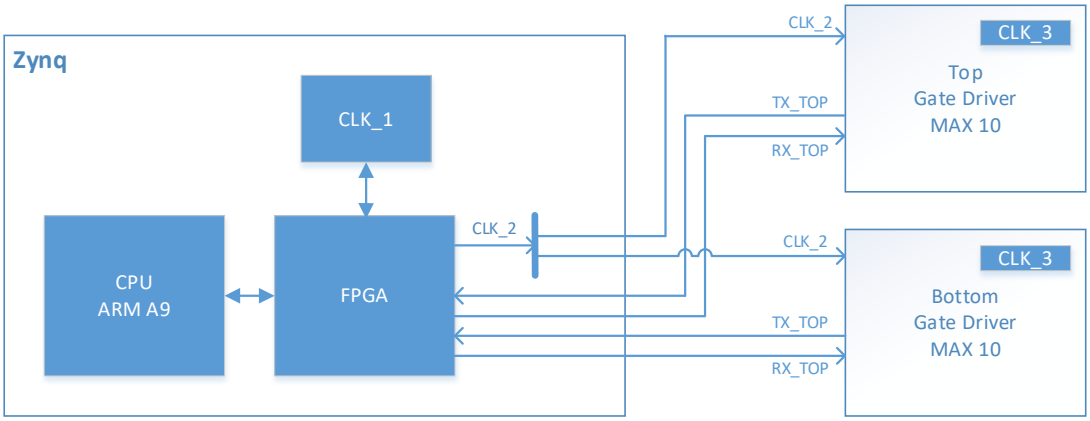

Figure 37: Clock sharing

To achieve synchronization, the PTP-based protocol is used. The PTP (IEEE 1588 standard) synchronizes timers throughout a computer network. It achieves clock accuracy in the sub-microsecond range on a LAN, making it suitable for measurement and control systems. In this case, the time accuracy is in the nanosecond range, since the clock is distributed/shared. For the same reason, the timer's jitter and latency are also in the nanosecond range.

To synchronize timers of the controller and GDs, according to the PTP, they must exchange four messages [72]:

1. The local controller sends time (sync message) to the GD, followed by a follow-up message.
2. After the Sync has been received, the GD is aware of the time at the local controller but still does not know the network delay.
3. After the exchange of Delay Request and Response messages, the GD is aware of the transmission delay, and both timers are in sync.

4.2.5 Fault Broadcasting

Fast and asynchronous fault recognition is very important. It is crucial to inform all devices in the system about one device's fault as soon as possible. If fault information is packed in a communication packet, the time needed for that information to travel from one device to another equals the sum of time needed to pack data, send it, receive it, and unpack it. That is, the interval is measured in microseconds. The goal is to broadcast a general fault message in intervals shorter than a microsecond without packing that information in the communication packet but using the same communication line. The fault is sensed by sensing the line break, and if there is a fault, the line should be broken intentionally. If there is no communication, no matter why, all devices in the network should stop with the operation and go to a safe state. Later, when the GD reaches the safe state, it should turn on communication to send fault and status data back to the controller. Since the EOP (or interpacket gap) is 100 ns long, the fault signal is an intentional drop of a line longer than 120 ns (Figure 38).

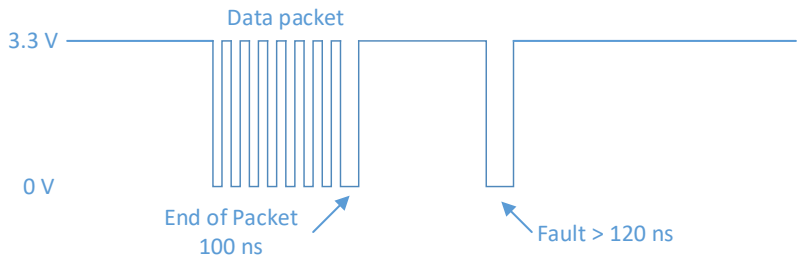

Figure 38: Fault and EOP sensing

Figure 39 and Figure 40 show an example of the sequence when a fault occurs. Communication with the controller will be intentionally broken if the fault occurs at GD #1. The controller would sense the fault after 120 ns + 60 ns. Then, it would intentionally break the communication line with GD #2 to signalize fault. After the other 180 ns, GD #2 will be aware of the fault in the system. The benefits of this approach are obvious, speed-wise. The fault transmission is done asynchronously, so it is not related to packet transmission.

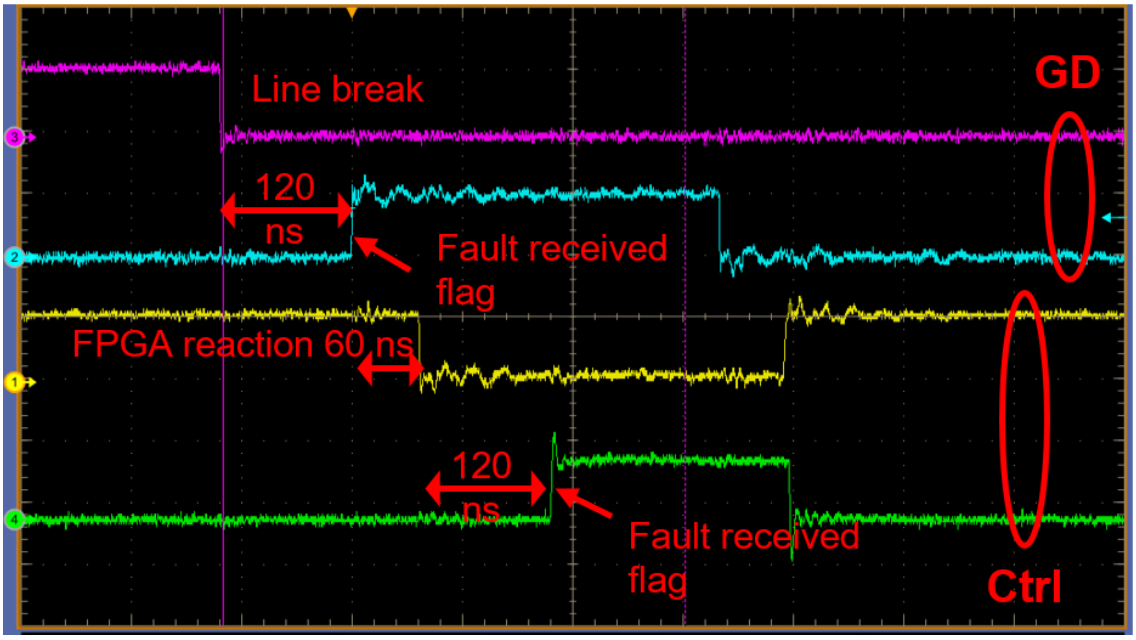

Figure 39: Fault recognition timing waveforms

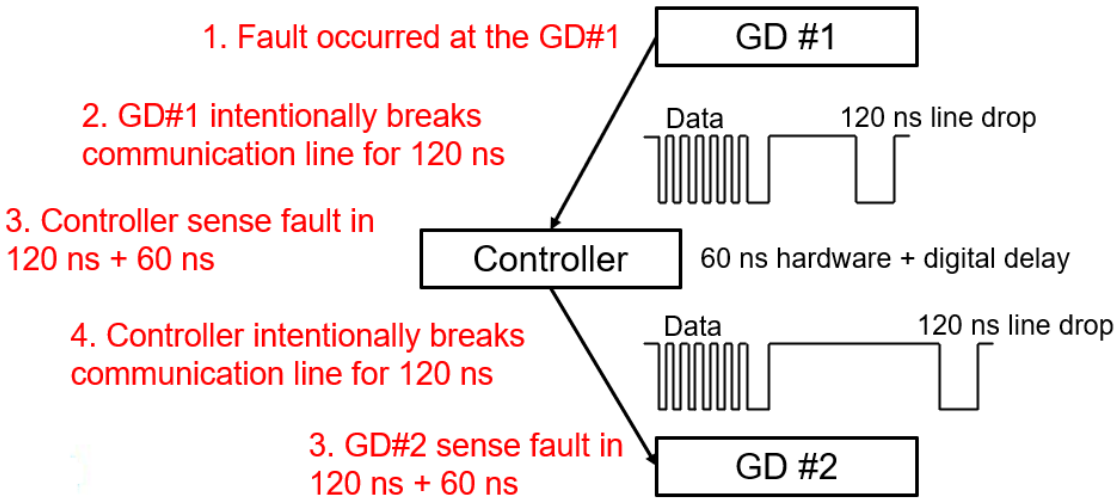

Figure 40: Fault recognition sequence

4.3 Experimental Results

4.3.1 Testbed

The testbed consists of a main controller connected to two local controllers, as shown in Figure 41. Each local controller is connected to two GDs.

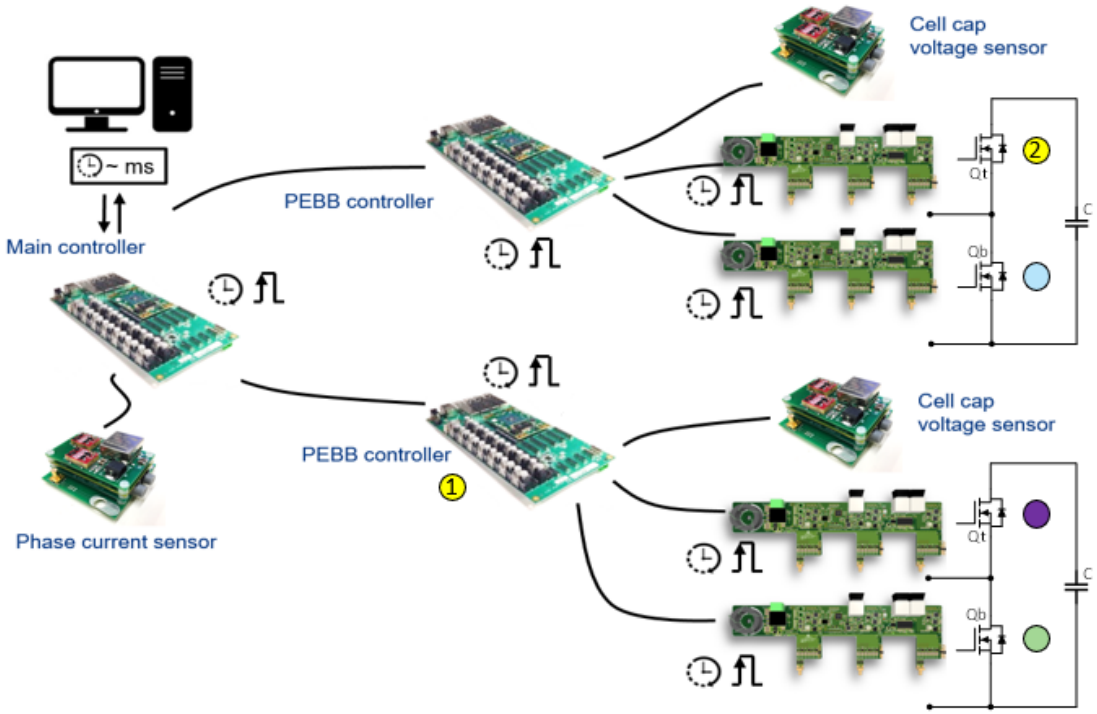

Figure 41: Inner pebb communication testbed

4.3.2 Synchronization Results

Figure 42 shows the synchronization of timers of the local PEBB controller (channel 1) and three GDs (channels 2-4). There is a very small latency and jitter (in this range) along an extensive time period.

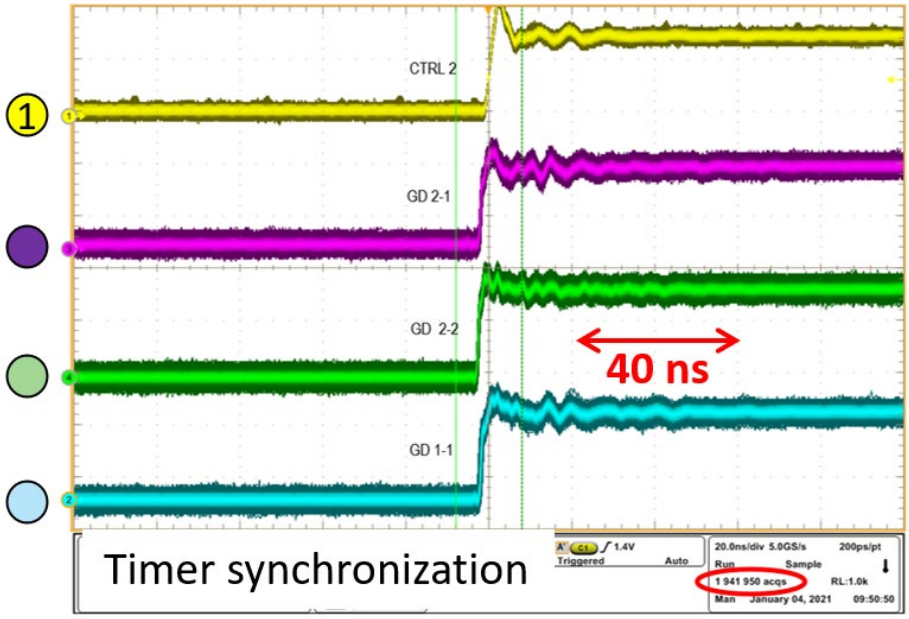

Figure 42: Timer synchronization

Synchronization pulses shown in Figure 43 are interesting in that they rely heavily on communication and synchronization. The local controller calculates the exact time when the synchronization pulse should occur. It sends that information to all GDs, which will then trigger the pulse at that exact moment calculated by the local controller. Since triggering the sync pulse is an operation/action that happens in the GD at a defining moment, any other actions can be carried out at any moment synchronously across all GDs in the network/power cell, relative to the syncronization pulses.

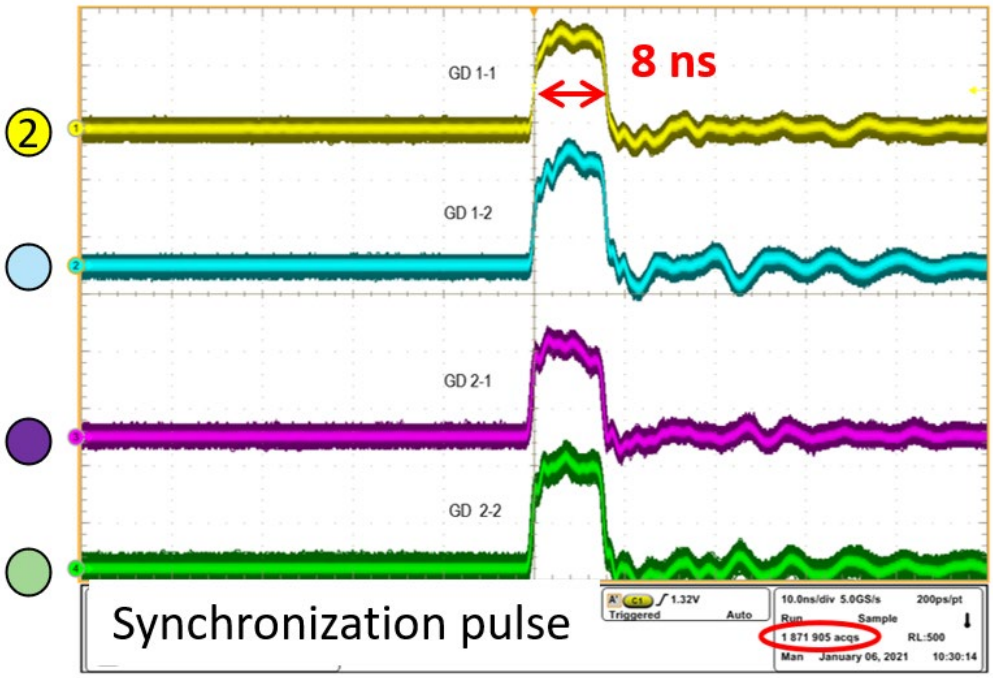

Figure 43: Synchronization pulse

An example of such a feature is a start operation flag (on the GD), as shown in Figure 43. Since pulse signals are synchronized between GDs, all GDs in the network/power cell will simultaneously start operating synchronously.

4.3.3 Peak Current Mode Control

Peak current-mode control (PCMC) is a good example of the benefits of communication. In the case shown here (Figure 44), the controller generates a current reference and sends it to the GD while the GD runs PCMC. This is also a very good example of distributed control. The green waveform shown in Figure 44 is the output of the DAC (current reference) used for PCMC.

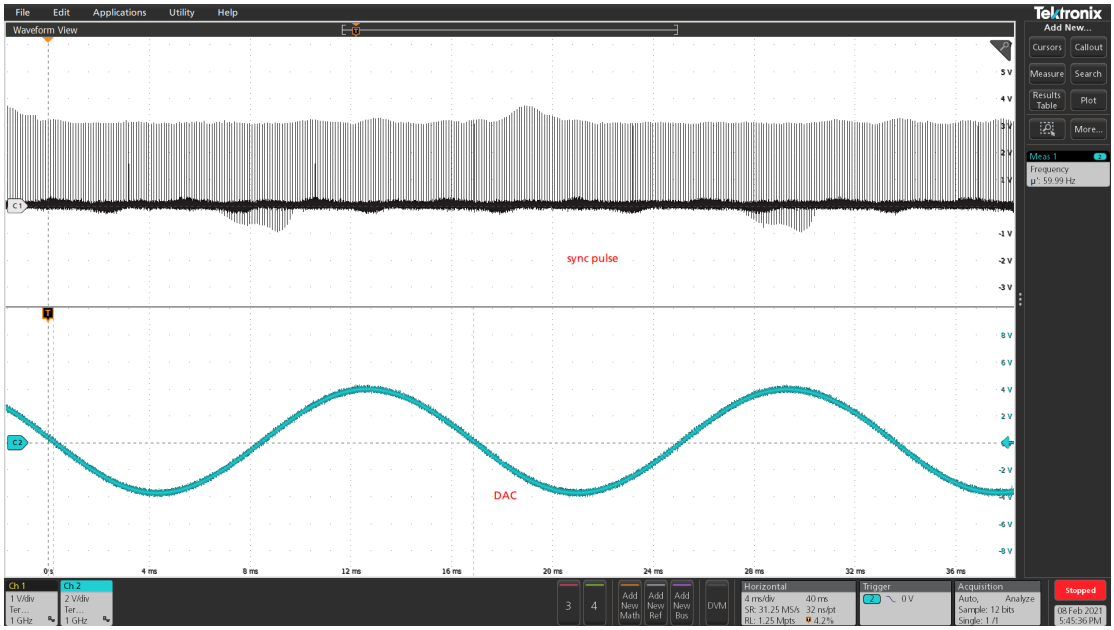

Figure 44: PCMC reference

4.4 Conclusion

The described communication network is designed and developed at The Center for Power Electronics Systems (CPES) and is tested in a medium-voltage modular multilevel converter (MV MMC). The communication between the controller and the GDs brings flexibility and the possibility of distributed control as well as hardware abstraction and simplification. The concept, network architecture, and components are described. Results of three key features of real-time communication protocol -- fast communication speed, packet size that fits in the control cycle, and good synchronization -- are presented. Also, a way of using communication channels for fast and reliable fault broadcasting has

been shown. These results show that this communication could be used for GD control instead of the traditional approach of using a PWM signal.

The described communication network is a good starting point for implementation of more advanced and distributed controll systems but further research should be done to prove it’s advantage over traditional approach.

Chapter 5 Summary and Future Work

5.1 Summary

Three different levels of communication techniques and protocols used in real-life power electronics projects, with all the challenges they bring, are presented in this thesis.

The custom controller was developed as a communication interface between different power electronics devices from different vendors in the smart house project. The additional capability of the developed device was to provide a platform for developing an energy management algorithm used to make a house grid-zero, if not grid-positive. Results of that approach to energy management are presented.

Communication protocols and techniques used in the IMU are described. This complex measurement device consists of different power electronics devices that need to communicate to provide valuable and accurate impedance measurements. Results of impedance measurement are presented.

Finally, a PEBB level, real-time communication protocol with all its challenges is developed and described. It provides communication and synchronization between different nodes, such as GDs, sensors, and actuators inside the PEBB. This intra-PEBB communication and synchronization combined with inter-PEBB communication and synchronization provide the foundation for the development of truly distributed event (time) driven control as well as hardware and software abstraction. It is an excellent starting point for the future development of a more robust real-time communication protocol, too.

5.2 Conclusion

As presented in this thesis, communication between different levels and nodes in a power electronics system brings the possibility of implementing distributed control. An excellent example of that is switching cycle control, where the main assumption is that the GD can run PCMC autonomously, and the only information needed for running it is the current reference.

Flexibility is another important benefit of communication. If, for example, compared with the traditional approach of running power electronics systems in which a PWM signal is sent from the controller to the GD, the communication allows different data to be exchanged between GD and controller.

Communication brings hardware simplification since, for example, information collected from sensors used for protection could be implemented for control. Therefore, fewer sensors are needed for the system.

Hardware and software abstraction can also be achieved using communication; for example, an abstraction of the power stage terminal behavior that is independent from hardware design (HAL – hardware abstraction layer).

Synchronization is the key aspect of reliable control, and can be achieved in different ways. One way is a shared clock signal. But communication allows different synchronization protocols over communication lines such as SyncE, PTP, or state-of-the-art White Rabbit.

On the other hand, if looked at from the perspective of complex power electronics devices such as IMUs or complex power electronics systems such as a smart house, communication is needed for the harmonious work of all power electronics devices.

5.3 Future Work

Future work will focus on inner-PEBB communication and distributed control. The first task should be increasing the communication speed ten times, up to 250 Mbps, using the same POF but different, much faster transceivers. That will allow a shorter control cycle, so faster switching frequencies could be achieved. Faster speed will allow compatibility with the 100BASE-T ethernet protocol. Compatibility with well-known communication protocols could expose GDs, sensors, and actuators to devices on the network other than dedicated controllers.

Communication should be compatible with different network topologies such as star and ring. The ring topology is very challenging. It will declutter the PEBB and simplify the hardware, but will also introduce additional levels of jitter and latency, impacting synchronization. So, the impact of different network topologies and communication protocols on synchronization jitter and latency should be further investigated.

A fast network opens the possibility for more complex distributed control, which opens the possibility of future research and implementation of control algorithms in distributed environments, especially in MMCs .

Bibliography

[1] SmartGrid.gov, "The Smart Grid," SmartGrid.gov, [Online]. Available: https://www.smartgrid.gov/the_smart_grid/smart_grid.html. [Accessed 20 11 2021].

[2] FORD, "2022 FORD F-150 LIGHTNING," FORD, [Online]. Available: https://www.ford.com/trucks/f150/f150-lightning/2022/. [Accessed 28 11 2021].

[3] C. R. Andre Groll, "Secure and authentic communication on existing in-vehicle networks," in *2009 IEEE Intelligent Vehicles Symposium*, 2009.

[4] H. S. Z. W. Z. F. Qixun Zhang, "Sensing and Communication Integrated System for Autonomous Driving Vehicles," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020.

[5] Y. J. L. K. R. Ali Emadi, "Power Electronics and Motor Drives in Electric, Hybrid Electric, and Plug-In Hybrid Electric Vehicles," *IEEE Transactions on Industrial Electronics,* vol. 55, no. 6, pp. 2237-2245, 2008.

[6] N. H. Y. K. Terry Ericsen, "PEBB - Power Electronics Building Blocks from Concept to Reality," in *2006 Record of Conference Papers - IEEE Industry Applications Society 53rd Annual Petroleum and Chemical Industry Conference*, 2006.

[7] I. M. D. B. R. C. J. G. I. Celanovic, "A new distributed digital controller for the next generation of power electronics building blocks," in *APEC 2000. Fifteenth Annual IEEE Applied Power Electronics Conference and Exposition (Cat. No.00CH37058)*, 2000.

[8] V. Tech, "Future HAUS," [Online]. Available: https://www.futurehaus.tech/.

[9] D. D. W. Z. L. J. D. B. F. C. L. P. M. Igor Cvetkovic, "A testbed for experimental validation of a low-voltage DC nanogrid for buildings," in *2012 15th International Power Electronics and Motion Control Conference (EPE/PEMC)*, 2012.

[10] I. C. R. B. D. D. Dushan Boroyevich, "Intergrid: A Future Electronic Energy Network?," *IEEE Journal of Emerging and Selected Topics in Power Electronics,* vol. 1, no. 3, pp. 127-138, 2013.

[11] LG. [Online]. Available: https://www.lg.com/us/business/solar/neon-r-solar-panels.

[12] Schneider. [Online]. Available: https://solar.se.com/us/en/product/conext-mppt-80-600-solar-pv-charge-controller/.

[13] "PHI 3.2™ HIGH OUTPUT BATTERY," Simpliphi, [Online]. Available: https://simpliphipower.com/product/phi-3-2-high-output-battery/. [Accessed 5 12 21].

[14] O. Power. [Online]. Available: https://www.outbackpower.com/products/inverter-chargers/radian-series.

[15] ACROMAG INCORPORATED, "WHITE PAPER: INTRODUCTION TO MODBUS TCP/IP," 2020/20201. [Online]. Available:

https://www.acromag.com/wp-content/uploads/2019/08/White-Paper-Introduction-to-ModbusTCP_765B-.pdf. [Accessed 10 11 2021].

[16] Schneider Electric, "Modbus TCP Communication Device Type Manager," 2 2019. [Online]. Available: https://download.schneider-electric.com/files?p_enDocType=User+guide&p_File_Name=EIO0000000365.06.pdf&p_Doc_Ref=EIO0000000365. [Accessed 20 11 2021].

[17] M. M. Vladimir Mitrovic, "Power quality data logger with internet access," in *2017 International Symposium on Power Electronics (Ee)*, Novi Sad, 2017.

[18] N.-E. GMBH. [Online]. Available: https://www.enerserve.eu/en/smartpi.html. [Accessed 27 10 2021].

[19] R. P. Foundation. [Online]. Available: https://www.raspberrypi.org/. [Accessed 20 10 2021].

[20] A. D. S. B. Ho Ming-Tzu, "Control system design using low order controllers: constant gain, PI and PID," in *Proceedings of the 1997 American Control Conference*, 1997.

[21] nodered.org, "Node-RED," [Online]. Available: https://nodered.org/. [Accessed 28 11 2021].

[22] [Online]. Available: https://www.solardecathlonme.com/. [Accessed 21 10 21].

[23] D. B. R. B. P. M. Z. S. Bo Wen, "Small-Signal Stability Analysis of Three-Phase AC Systems in the Presence of Constant Power Loads Based on Measured d-q Frame Impedances," *IEEE Transactions on Power Electronics,* vol. 30, no. 10, pp. 5952-5963, 2015.

[24] D. B. R. B. P. M. Z. S. Bo Wen, "Analysis of D-Q Small-Signal Impedance of Grid-Tied Inverters," *IEEE Transactions on Power Electronics,* vol. 31, no. 1, pp. 675-687, 2016.

[25] R. B. D. B. F. W. K. Gerald Francis, "An algorithm and implementation system for measuring impedance in the D-Q domain," in *2011 IEEE Energy Conversion Congress and Exposition*, 2011.

[26] M. J. P. M. D. B. J. V. M. B. Zhiyu Shen, "Design and implementation of three-phase AC impedance measurement unit (IMU) with series and shunt injection," in *2013 Twenty-Eighth Annual IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2013.

[27] Z. S. I. C. D. B. R. B. P. M. Marko Jaksic, "Multi-level single-phase shunt current injection converter used in small-signal dq impedance identification," in *2014 IEEE Applied Power Electronics Conference and Exposition - APEC 2014*, 2014.

[28] D. B. R. B. Z. S. I. C. P. M. Marko Jaksic, "Modular interleaved single-phase series voltage injection converter used in small-signal dq impedance identification," in *2014 IEEE Energy Conversion Congress and Exposition (ECCE)*, 2014.

[29] N. Instruments. [Online]. Available: https://www.ni.com/en-us/innovations/white-papers/10/lossless-communication-with-network-streams--components--archite.html.

[30] Plexim, "RT Box 3," Plexim, [Online]. Available: https://www.plexim.com/products/rt_box/rt_box_3. [Accessed 28 11 2021].

[31] N. Instruments, "Multithreading in LabWindows™/CVI," Instruments, National, [Online]. Available: https://www.ni.com/en-us/support/documentation/supplemental/06/multithreading-in-labwindows--cvi.html. [Accessed 26 10 2021].

[32] N. Instruments, "Using Thread Safe Queues to Protect Data," National Instruments, [Online]. Available: https://zone.ni.com/reference/en-XX/help/370051AG-01/cvi/programmerref/threadsafequeue/. [Accessed 26 10 2021].

[33] A. R. H. S. Pethuru Raj, Architectural Patterns, Packt Publishing, 2017.

[34] N. Instruments, "Thread Safe Queue Class Help," National Instruments, May 2017. [Online]. Available: https://zone.ni.com/reference/en-XX/help/370051AG-01/cvi/libref/thread_safe_queue_class_utility/. [Accessed 26 10 2021].

[35] N. Instruments, "What is NI-DAQmx?," National Instruments, 18 9 2019. [Online]. Available: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z000000P8baSAC&l=en-US. [Accessed 29 10 2021].

[36] N. Instruments, "Analysis Library Overview," National Instruments, May 2017. [Online]. Available: https://zone.ni.com/reference/en-XX/help/370051AG-01/cvi/libref/cvianalysis_library/. [Accessed 27 10 2021].

[37] N. Instruments, "Mean," National Instruments, May 2017. [Online]. Available: https://zone.ni.com/reference/en-XX/help/370051AG-01/cvi/libref/cvimean/. [Accessed 26 10 2021].

[38] M. J. B. Z. P. M. D. B. J. V. M. B. Zhiyu Shen, "Analysis of Phase Locked Loop (PLL) influence on DQ impedance measurement in three-phase AC systems," in *2013 Twenty-Eighth Annual IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2013.

[39] N. Instruments, "Lossless Communication with Network Streams: Components, Architecture, and Performance," National Instruments, 3 Aug 2020. [Online]. Available: https://www.ni.com/en-us/innovations/white-papers/10/lossless-communication-with-network-streams--components--archite.html. [Accessed 26 10 2021].

[40] N. Instruments, "LabWindowsTM/CVITM," National Instruments, October 2010. [Online]. Available: https://www.ni.com/pdf/manuals/373552g.pdf. [Accessed 28 10 2021].

[41] N. Instruments, "Building Networked Applications with the LabWindows/CVI TCP Support Library," National Instruments, May 2017. [Online]. Available: https://zone.ni.com/reference/en-XX/help/370051AG-01/cvi/libref/cvitcp_clients_and_servers/. [Accessed 28 10 2021].

[42] M. J. P. M. D. B. J. V. M. B. Zhiyu Shen, "Three-phase AC system impedance measurement unit (IMU) using chirp signal injection," in *2013 Twenty-Eighth Annual IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2013.

[43] National Instruments, "Programming with Graph Controls," National Instruments, May 2017. [Online]. Available: https://zone.ni.com/reference/en-

XX/help/370051AG-01/cvi/uiref/cviprogramming_with_graph_controls/. [Accessed 28 10 2021].

[44] C. C. Hulbert, "MATIO," 25 Mar 2021. [Online]. Available: https://github.com/tbeu/matio. [Accessed 28 10 2021].

[45] R. B. B. W. D. B. J. V. D. V. M. B. Ye Tang, "A Novel DQ Impedance Measurement Method in Three-Phase Balanced Systems," in *2019 20th Workshop on Control and Modeling for Power Electronics (COMPEL)*, 2019.

[46] D. B. R. B. P. M. Bo Wen, "Modeling the output impedance of three-phase uninterruptible power supply in D-Q frame," in *2014 IEEE Energy Conversion Congress and Exposition (ECCE)*, 2014.

[47] J. Hibbard, "5 Real-Time, Ethernet-Based Fieldbuses Compared," KingStar, 17 5 2016. [Online]. Available: https://www.manufacturingtomorrow.com/article/2016/05/5-real-time-ethernet-based-fieldbuses-compared/8044/. [Accessed 28 10 2021].

[48] BERCKHOF, "Beckhoff Introduces EtherCAT G to Provide Next-Level Gigabit Performance," BERCKHOF, Nov 2018. [Online]. Available: https://www.beckhoff.com/en-us/company/press/pressemitteilungen_77898.html. [Accessed 28 10 2021].

[49] C. B. a. S. D. A. The, "CAN-based distributed control of a MMC optimized for low number of submodules," in *2015 IEEE Energy Convers. Congr. Expo. ECCE 2015*, 2015.

[50] B. B. N. J. M. K. a. C. K. K. C. H. Park, "Back-to-back 31 level modular multilevel converter with EtherCAT communication," in *2019 IEEE Energy Convers. Congr. Expo. ECCE 2019*, Sep. 2019.

[51] P. D. B. a. R. T. L. Mathe, "Control of a Modular Multilevel Converter with Reduced Internal Data Exchange," *IEEE Trans. Ind. Informatics,* vol. 13, no. 1, p. 248–257, Feb. 2017.

[52] C. L. T. a. L. E. Norum, "A performance analysis of three potential control network for monitoring and control in Power Electronics converter," in *2012 IEEE Int. Conf. Ind. Technol. ICIT*, 2012.

[53] J. W. Z. S. R. B. D. B. a. S. Z. Y. Rong, "Distributed Control and Communication System for PEBB-based Modular Power Converters," in *2019 IEEE Electr. Sh. Technol. Symp. ESTS*, 2019.

[54] Y. R. e. al., "A Synchronous Distributed Control and Communication Network for SiC-Based Scalable Impedance Measurement Unit," in *ECCE 2020 - IEEE Energy Convers. Congr. Expo.*, Oct. 2020.

[55] P. N. e. V. (PNO), "PROFINET System Description," [Online]. Available: http://us.profinet.com/wp-content/uploads/2012/11/PI_PROFINET_SystemDescription_EN_2014_01.pdf. [Accessed 2022].

[56] "EtherCAT Technology Group | EtherCAT.," [Online]. Available: https://www.ethercat.org/en/technology.html. [Accessed 2022].

[57] I. C. B. A. V. a. C. Z. G. Cena, "A high-performance CAN-like arbitration scheme for EtherCAT," in *ETFA 2009 - 2009 IEEE Conf. Emerg. Technol. Fact. Autom.*, 2009.

[58] L. X. Xuepei Wu, "Performance evaluation of industrial Ethernet protocols for networked control application," *Control Engineering Practice,* vol. 84, pp. 208-217, 2019.

[59] L. E. N. C. L. Toh, "A high speed control network synchronization jitter evaluation for embedded monitoring and control in modular multilevel converter," in *2013 IEEE Grenoble Conference*, 2013.

[60] L. M. M. R. H. P. A. S. R. T. Paul Dan Burlacu, "Implementation of fault tolerant control for modular multilevel converter using EtherCAT communication," in *2015 IEEE International Conference on Industrial Technology (ICIT)*, 2015.

[61] F. H. G. C. a. R. M. J. H. Fey, "Development of a modular multilevel converter demonstrator with EtherCAT communication," in *Proc. - 2019 IEEE 13th Int. Conf. Compat. Power Electron. Power Eng. CPE-POWERENG*, Apr. 2019.

[62] G. F. a. N. C. B. Steinmann, "Tree-shaped networked control system for modular power converters with sub-µs latency and ns-scale synchronization accuracy," in *2019 IEEE Energy Convers. Congr. Expo. ECCE 2019*, 2019.

[63] Y. R. e. al., "High-performance Distributed Power Electronics Communication Network Design with 5 Gbps Data Rate and Sub-nanosecond Synchronization Accuracy," in *2021 IEEE Energy Convers. Congr. Expo. ECCE 2021 - Proc.*, 2021.

[64] Y. X. A. S. J. L. a. J. L. G. R. E. F. Dierikx, "White Rabbit Multi-Point Time Distribution Network," in *2021 Jt. Conf. Eur. Freq. Time Forum IEEE Int. Freq. Control Symp. EFTF/IFCS 2021*, 2021.

[65] J. W. Z. S. B. F. D. B. a. R. B. Y. Rong, "PLL-based Clock Recovery Stability Analysis for Distributed Power Electronics Communication Networks with Sub-nanosecond Synchronization Accuracy," in *IECON Proc. - Industrial Electron. Conf.*, 2021.

[66] H. G. J. P. a. X. Z. P. Li, "Time synchronization of white rabbit network based on kalman filter," in *2019 IEEE 3rd Int. Conf. Electron. Inf. Technol. Comput. Eng. EITCE*, 2019.

[67] Y. Rong, "A Synchronous Distributed Control and Communication Network for High-Frequency SiC-Based Modular Power Converters - Master Thesis," Virginia Tech, Blacksburg, VA, 2019.

[68] Broadcom, "AFBR-16xxZ and AFBR-26x4Z/25x9Z Data Sheet," [Online]. Available: https://docs.broadcom.com/doc/AV02-4369EN. [Accessed 5 12 2021].

[69] Broadcom. [Online]. Available: https://www.broadcom.com/products/fiber-optic-modules-components/industrial/industrial-control-general-purpose/650nm/afbr-2624z.

[70] Wikipedia, "Manchester code," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Manchester_code. [Accessed 29 10 2021].

[71] Wikipedia, "Modbus," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Modbus. [Accessed 29 10 2021].

[72] PERLE, "PTP - Precision Time Protocol," PERLE, [Online]. Available: https://www.perle.com/supportfiles/precision-time-protocol.shtml. [Accessed 29 10 2021].

[73] Beckhoff, "EtherCAT - the Ethernet Fieldbus," Beckhoff, [Online]. Available: https://www.ethercat.org/en/technology.html. [Accessed 28 10 2021].

[74] J. W. Z. S. S. Z. B. W. R. B. D. B. Yu Rong, "A Synchronous Distributed Control and Communication Network for SiC-Based Scalable Impedance Measurement Unit," in *2020 IEEE Energy Conversion Congress and Exposition (ECCE)*, 2020.

[75] J. W. Z. S. R. B. D. B. S. Z. Yu Rong, "Distributed Control and Communication System for PEBB-based Modular Power Converters," in *2019 IEEE Electric Ship Technologies Symposium (ESTS)*, 2019.

[76] L. M. M. R. H. P. A. S. a. R. T. P. Dan Burlacu, "Implementation of fault tolerant control for modular multilevel converter using EtherCAT communication," in *Proc. IEEE Int. Conf. Ind. Technol.*, 2015.