POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Degree Thesis

Digital Embedded Scope for Power Electronic Applications

Supervisors

Prof. Francesco MUSOLINO

Dr. Riccardo TINIVELLA

Candidate

Edoardo BENINTENDE

October 2023

Summary

The transaction towards hybrid and electric vehicles has fueled up the demand for advanced power electronic circuits. In this panorama, proper diagnostic instruments for measurements in power electronics fields are essential. This thesis proposes a novel paradigm for developing such instrumentation.

Measuring signals inside power converters is challenging due to noise and safety concerns. The high voltage and switching behavior of these devices make measurement difficult and dangerous for non-isolated instrumentation. Therefore, nowadays it is common to use optic isolated probes or oscilloscopes to deal with this topic. However, such solutions come with limitations due to their fragility and high cost. This work proposes an innovative solution to address all these problematics: a device that uses Wi-Fi as an isolation medium. The choice of Wi-Fi allows the use of microcontrollers, which are cheap and practical to use. The project focuses on the integration of a digital scope device over the on-board charger from BRUSA HyPower. The proposed architecture uses an ESP32 microcontroller to retrieve the signals collected by the ADC embedded in the on-board charger via SPI, and forward them over Wi-Fi to a user-accessible environment using the TCP protocol. This architecture is more resistant and less expensive than fiber optic isolation, while providing the same level of isolation.

The first phase of the project was conducted on a testbench, where the overall communication architecture was developed and validated. Once the communication architecture was working, the device was tested in a real-world scenario with the actual on-board charger, verifying its operation in two different states: standby and charging. The final results showed promising results in using Wi-Fi as an isolation medium, but the current architecture of the device needs some improvements to be fully functional, as it is susceptible to electromagnetic disturbances and not optimized in terms of code complexity.

The conclusion of the thesis outlines the Wi-Fi as a possible new technology for building isolated instrumentation for diagnostic of measurements in the field of power electronics.

Acknowledgements

This thesis is the product of a collective effort of multiple people, and I am deeply grateful to all those who have supported me on this project.

I would like to express my sincere gratitude to Professor Francesco Musolino for his precise guidance and availability throughout the work and the writing of the thesis. He has been an invaluable academic supervisor, and I am grateful for his support.

I am thankful to Dr. Riccardo Tinivella for proposing this project to me and for his supervision throughout its development. I have immensely appreciated the knowledge and help he has shared me during all the phases of this work. He has been a wise and supportive mentor, and I am truly appreciative for his assistance.

I would also like to thank the entire organic at BRUSA HyPower for their support during this project providing me with precious suggestions and hints for the development of my thesis. I am particularly grateful to Andreas, Falk, Raphael, Alessandro, Calogero, Marco, Riccardo and Emiliano, who dispensed me with valuable insights and advices.

Finally, I would like to extend my sincere gratitude to BRUSA HyPower for providing me with the necessary resources and instrumentation to develop this project.

Table of Contents

List of Tables

List of Figures

Acronyms

MCU
: Main Control Unit

LED
: Light Emitting Diode

SNR
: Signal to Noise Ratio

CAN
: Control Area Network

OBC
: On-board Charger

EV
: Electric Vehicle

DSP
: Digital Signal Processor

ADC
: Analog to Digital Converter

SPI
: Serial Peripheral Interface

FPGA
: Field Gate Programmable Array

OSI

Open Systems Interconnection

TCP

Transmission control protocol

UDP

User datagram protocol

MISO

Master In - Slave Out

MOSI

Master Out - Slave In

CLK

Clock Signal

SS

Slave Select Signal

FIFO

First In - First Out

IDE

Integrated Development Environment

CSV

Comma Separated Value

Chapter 1

Introduction

Power electronics is a branch of electrical engineering that focuses on the control and conversion of electrical power. It deals with the processing of electrical energy to achieve specific voltages or currents depending on the load characteristics. Power electronic circuits, such as inverters, converters, and rectifiers, are essential components in various applications, ranging from renewable energy systems and electric vehicles to industrial automation and consumer electronics.

One of the most prominent sectors where power electronics has made a profound impact is the automotive industry. The transition towards electric and hybrid vehicles has fueled the demand for advanced power electronic systems. These systems control the flow of electrical energy, enabling efficient battery management, motor control, and regenerative braking. The constant development of this technology is driven by the need to address one of the main problems facing today's society: an over-reliance on non-renewable resources, being as world population dependent more than 85% on energy coming from fossil fuels [1]. Power electronics contribute significantly to improved energy efficiency, reduced emissions, and enhanced vehicle performance, thereby leading to a more sustainable and responsible approach to energy consumption.

The company BRUSA HyPower AG with the joint effort of the Politecnico di Torino proposed the thesis project described in what follows. BRUSA HyPower has almost forty years of know-how and long-term experience to build complete, high-end quality and reliable electronic systems for e-mobility along with its components. The company is expert in Model Simulation, Software Development, Mechanical Design, PCB Design, Hardware Development and Testing. The developed products are fully qualified and released according to relevant standards to be used in on-highway (e.g. car, trucks, buses which are driven on roads) and off-highway applications (e.g. construction/agriculture vehicles that work on steep or uneven ground). BRUSA HyPower main products are on-board chargers, and bidirectional DC/DC converters for electric motors and fuel cells applications. The company main devices are represented in figure 1.1. The thesis has been carried out in their facility, the company supported the thesis furnishing the instruments and the laboratories in which the project has been executed.

Figure 1.1: BRUSA HyPower main products

1.1 Background about Power Electronics Mearuements

Power electronics systems during their operating life experience several different forms of stresses: thermal, chemical, electrical, mechanical, and degrade gradually, possibly leading to failures. As a consequence, a major concern, both in terms of safety and cost, is their reliability. Among the several sources of stresses, the most common are: high voltage and current levels, switching transients, harmonics and distortions, temperature effects, noise, high bandwidth requirements, voltage and current measurements synchronization, isolation and safety concerns. For these reasons, voltages and currents in different sections of power converters must be sensed for diagnostic purposes so that to check for possible deviations from their nominal behavior. Typically, the signals to be sensed serve a wide array of functions, such as analog signals for control and protection (e.g. reference voltages for active devices), power signals (e.g. inputs and outputs of converters), digital signals (e.g. semiconductor driver signals, peripheral interfaces signals, or converted analog-to-digital signals), or thermal signals for reliability and safety assessments. A power converter encloses within it signals having different electrical characteristics, each demanding distinct measurement characteristics. The figure 1.2 shows how a power electronic system could be organized, depicting the different domains and the different type of signals that are involved.

The biggest issue that result from having multiple electric domains is that they must be separated and isolated each from the others to operate correctly. Without isolation, there can be ground loop current flowing between two units

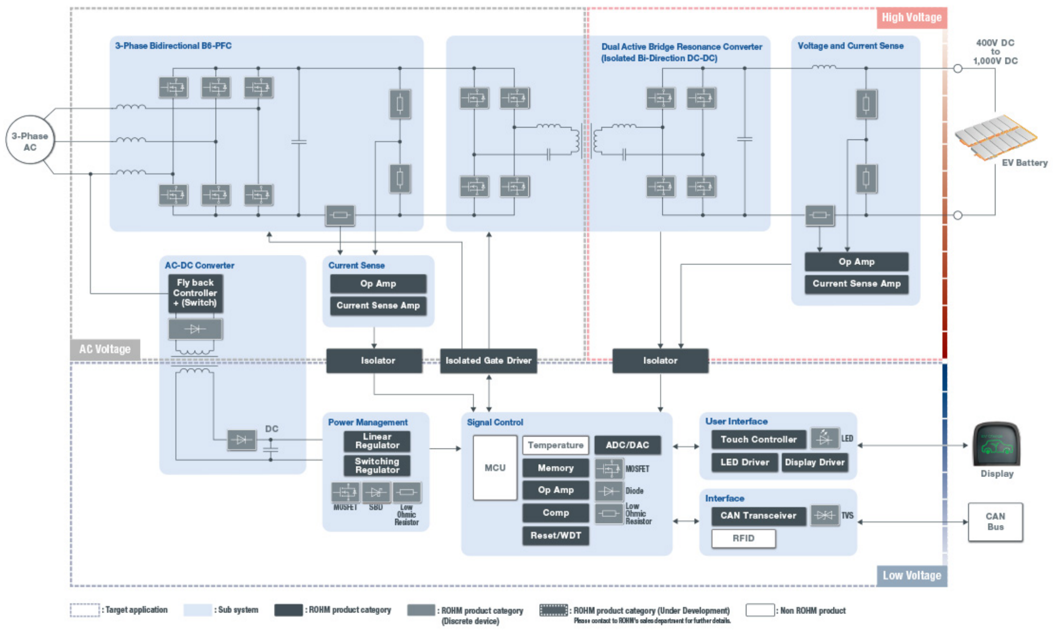


Figure 1.2: On-Board Charger block schematic

sharing a ground conductor, which constitute electrical noise that can interfere with the operations of either circuit. Moreover, if the difference in ground potentials is sufficiently large, the resulting ground-loop current can pose a safety issue. Electric vehicles are characterized by high-voltage circuitry carrying lethal currents. The high-voltage sections are under the control of digital electronics employing milliamp-level currents. Galvanic isolation, compared to other types of isolation, is a more robust way of preventing faults in power stages from damaging the control electronics that operates them [2].

The best scenario would be to control and measure the voltages and currents that are important for the electrical stress of the power converter by exploiting from the main control unit (MCU) processor. This choice has the effect of possibly employing standard oscilloscope so that to increase the safety in taking the measurements. However, practical implementation would require an isolated communication between the low voltage domain and all the others domains. This could be challenging because isolated communication, achieved through devices such as opto-couplers, introduces some issues. Opto-couplers consist of a light emitting diode (LED) on the input side and a photosensitive transistor on the output side. When an electrical signal is applied to the LED, it emits light, which is detected by the photosensitive transistor, and the absence of direct connection between the two devices ensures the electrical isolation. The problem is the introduced propagation delay which reduces the bandwidth of the system, limiting its sampling performances. Lower propagation delays can be obtained using digital isolators but they come at higher costs. The best solution would be to avoid adding complexity and cost to the system by introducing other elements inside the design. An alternative possible solution is to have an external diagnostic probing system which connects in a direct manner to each domain separately.

The most important characteristics of the required instrumentation are: high number of channels, signal isolation and high resolution. A device equipped with numerous channels is necessary to measure simultaneously the multitude of signals characterizing a power electronics system. Indeed, oscilloscopes with eight or more channels are becoming more popular.

Isolation creates an electrical barrier between the input side (where the signal to be measured is present) and the output side (where the measurement instrument or data acquisition system is connected). This barrier ensures that there is no direct electrical connection between the two sides, granting safety for the measurement equipment and operators which are protected from potential electrical shocks or dangerous voltage levels. Isolation helps maintain the integrity of the measured signals by preventing unwanted coupling or interference between channels, and it helps reducing the effect of common-mode voltage. By rejecting common-mode voltage, isolation ensures that the measured values are accurate and not distorted by noise that may be present in the electrical environment. Therefore, each channel must be isolated.

High resolution is another important characteristic, power electronic systems deal with both high and low value signals and it is necessary that the smaller signals are measured properly as much as the large ones. For example, given a high signal-to-noise ratio (SNR) with 1000V voltage range probe, even 1% error would be very high in a 5V domain. There would be 10V of uncertainty which would compromise the measure in terms of noise and voltage level.

The current state of the art in the measurements of electrical stresses in power converters offers different solutions, such as multi-channel oscilloscopes, differential probes, isolated probes, digital oscilloscopes or software-based diagnostic solutions. However, they address only to specific aspects without covering all the needs of the complex landscape of power electronic measurements.

Multi-channel oscilloscopes, are versatile instruments capable of capturing multiple signals simultaneously. They excel in scenarios where the requirement is to monitor various aspects of a system concurrently. However, their channels lack electrical isolation, making them not suitable for direct connection to power electronics systems.

Differential probes are single channel devices used to measure floating signals (like, for example, the gate to source signals in high-side power mosfet). These probes incorporate a differential amplifier that allows to measure a voltage differences between any two points. This topology is characterized by a good common-mode noise rejection, but they are sensible to frequency, therefore it is important to chose the device according to the needed bandwidth. These devices are usually sold for nominally high voltage measurements like 1000V, therefore they are not suited for signals ranging in the low voltage domain. They are ideal for comparing high voltage signals, but lack isolation.

Isolated differential probes are designed to ensure safety by electrically isolating measurement instruments from the circuit under test. The best isolated probes are based on fiber optic technology, the light is used as communication mean and ensure an indirect connection between the oscilloscope and the device under test. Fiber optics provides very fast communication and extremely high common-mode noise rejection. The main drawbacks of these devices are their extreme fragility and high cost.

Digital oscilloscopes convert analog voltage waveforms into digital data, which can be displayed, stored, and analyzed on a digital screen. They offer a range of benefits, including high resolution, waveform storage, and advanced triggering capabilities. However, they do not provide the same degree of electrical isolation as isolated probes, limiting their suitability for high-voltage measurements in safety-critical applications.

Lastly, software-based diagnostic solutions,like PCAN[3] and CANoe[4], usually exploit the existing communication infrastructure found in every vehicle. An automotive system comprises a network of different components interconnected through a Control Area Network (CAN). It serves as a communication bus, facilitating the exchange of control and command messages among various devices inside the vehicle. These messages are used to transmit device states and measurements recorded by their respective sensors. However, their utility is constrained due to the inherent limitations of the CAN infrastructure, which is not natively isolated and operates at a relatively low bandwidth capped at 1 Mbps. This speed is enough for transmitting status updates and basic control commands but, for example, it could be not enough when it comes to capturing rapid variations in measurements to track the causes of faults. In theory, it would be possible to provide CAN with isolation but it is not trivial, limiting the efficiency of CAN.

The current technologies are not sufficient to meet all the multifaceted demands of power electronic measurements. The integration of various solutions in one testing setup becomes necessary, creating a complex hardly-maneuvering system.

In the context of the automotive industry, where there are lot of standards and norms to be respected, this issue is particularly tough. Indeed, when developing a power electronic device for automotive, several tests are required. Each test needs to be performed in a defined environment and mounting or moving such diagnostic systems in each setup is quite time demanding and slows down the development phase. In addition, manufacturers often work closely with clients to test the integration of their devices in the client's systems, and it would be easier to test the functionality of the device directly at the client's facility. This is not possible without a flexible diagnostic system.

Currently, BRUSA adopts an big optic fiber based system where the optical isolated probes are connected through particular transceivers to an oscilloscope. The communication is based on audio optic fiber and it reaches speeds in the range of Gigabit. While the efficiency of the system in terms of bandwidth is over the top, it is excessive for the power electronics where the maximum frequency range of the applications is in the MHz range. The main problems of the system are the fragility of the easy to shatter fiber-glass based probes, its high cost, and it unsuitability for on-field testing. It is a telecommunication designed system and as so it requires a clean and low temperature environment (-20°C to 45°C). On the other hand, on-field applications for power electronics need to sustain temperatures ranging from -40°C to 80°C, and face exposure to dirt and dust. All these characteristics makes it not ideal for the power electronics measurements where lot of times it is necessary to test devices directly to the client.

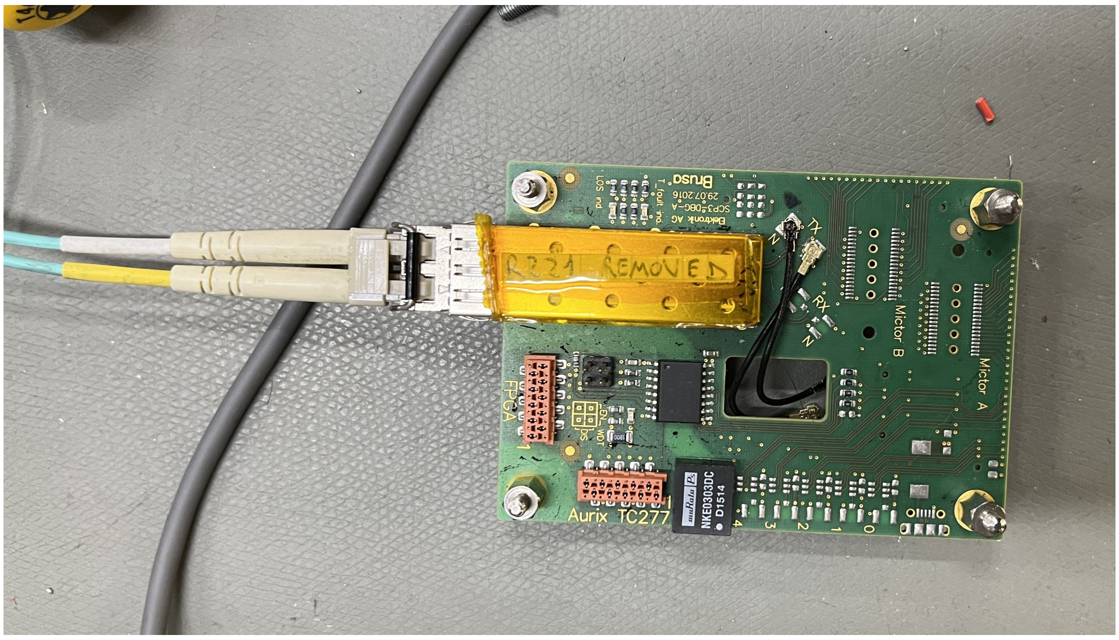


Figure 1.3: Optic fiber transceiver

The fragility of the current instrumentation is shown in the figure 1.3 where it is presented one of the transceiver of the system. It is noticeable how the coaxial cable connection (small black wires) are not meant to be attached and detached often as instead would be needed.

Therefore, the motivation leading this thesis project is the need for a diagnostic system which is able to tackle all these problematic together without being extremely complex. The need is of a single device suited to measure several different signals providing isolation and high resolution within every domain. Such device would be for the company much more flexible and easy to setup.

1.2 Objective of the Work

The goal of the thesis work is to develop a virtual scope device for carrying out diagnostic measurements in the filed of power electronics. The main objective is to provide an easy-to-use tool to debug and monitor power electronics without the need for expensive equipment. The device must respect the requests of high number of channels, isolation and high resolution which are addressed by the power electronics measurements.

The project focuses on BRUSA HyPower on-board chargers (OBCs), which in simple terms are AC-to-DC converters receiving the three-phase voltages and currents from the grid as inputs and converting them into DC quantities at its output for the recharge of the battery of an electric vehicle (EV).

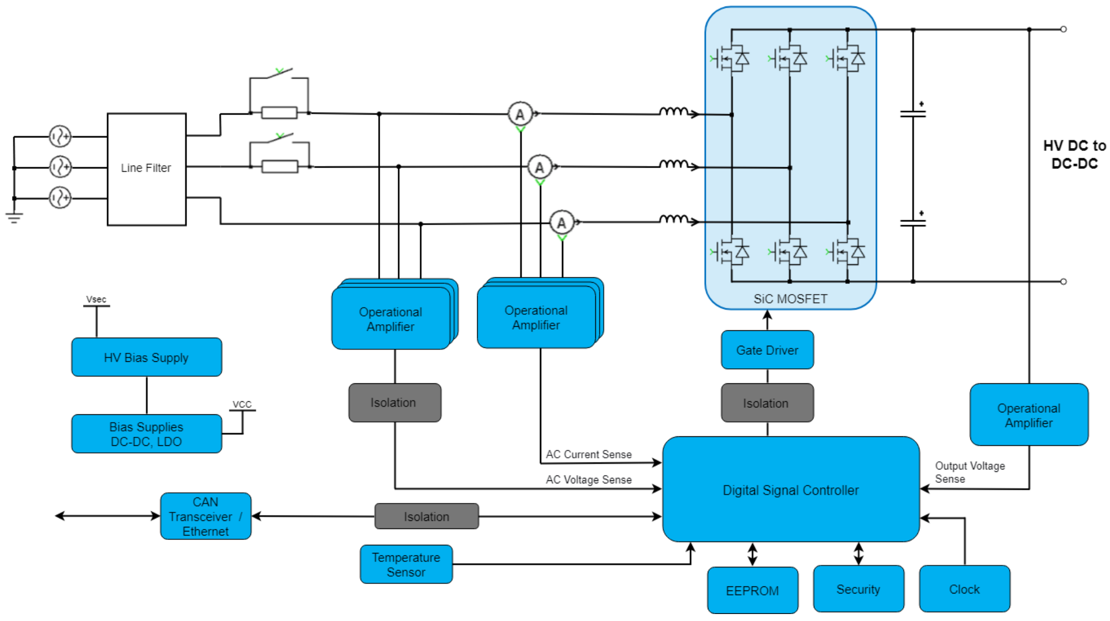


Figure 1.4: On-board charger

In the figure 1.4 a general schematic of the system is shown. Each OBC is coordinated by a digital signal processor (DSP) enrolled of controlling the converter behaviour. It samples the input, output voltages and currents of the system. It measures the temperature of the on-board charger power board. Eventually, all this information is forwarded to the MCU via isolated CAN. The DSP is inserted in the AC high voltage domain inside the converter and has isolated connection to the DC high voltage and low voltage domain. However, it lacks of the isolation with respect to the external world. The objective is then to employ systems with large bandwidths with respect to the isolated CAN, the data captured from the DSP to the external world where they can be studied and analysed.

Currently, the OBC is already a validated device on the market but it requires constant review and improvement to fit the customers requests. Consequently, BRUSA engineers need to tune and monitor the behaviour of the device in all its operating states. At the moment, both procedures are executed using two separate devices. The goal is to merge them into a single instrument, with a particular focus on simplifying the supervision of the converter's behavior, which currently is performed using the optical fiber system. This simplification aims to enable easy monitoring of signal values in case of sporadic faults. It comes to be very useful in a practical situation where, for example, a client is not able to set the device in charging state, and it would be needed to know what are the values of current and voltages inside the system to track perfectly its faulty behaviour, without the need of mounting all the equipment for the optic fiber system.

The most effective approach to achieve this goal is to sample the signals minimizing the information losses of measured data and in this scenario a real time capture of the device is not the best solution. Real-time acquisition implies defining a refresh rate for the device, the refresh rate corresponds to a time-window in which data cannot be acquired because the data is being processed. In real-time communication there is the overhead of the device processing operation to consider. For example, it could be the code speed of the device or the overhead intrinsic of a communication protocol. Therefore, the non-real time is best suited to retrieve all the elements without losing information. Currently, on the market no device is able to provide such functionality, as oscilloscope are not able to save data in this manner. Consequently, the development of a novel virtual scope concept is necessary.

Summarizing, the request is to build a on-field usable virtual scope able to provide a non-real time capture of the input and the output measures of the OBC when a fault occurs. Subsequently, this data should be graphed and stored in a user-accessible file for reference.

1.3 Thesis Description

The virtual scope is a ESP32 microcontroller which connects to the DSP of the power converter. It continuously reads the data sampled by the DSP and saves it when a faulty condition occurs. The collected data is sent to the external world using Wi-Fi communication. The employment of the wireless technology allows to intrinsically isolate the communication with the user.

The proposed solution exploits the ability of the DSP to already sample all the signals characterizing the converter. Therefore, by connecting directly to the DSP there is no need for a direct connection to all the different domains within the system. The DSP analog-to-digital converter (ADC) samples each measure at 45 kHz and provides high resolution 16-bit digital signals. The communication between the DSP and the ESP32 microcontroller is made using the serial peripheral interface (SPI). This communication protocol is very fast and its bandwidth accommodates the retrieval of multiple signals, granting the possibility of a multi-channel capture. The virtual scope is able to capture 20s of samples per four channels. The SPI communication with the DSP occurs continuously in real time. However, when a fault condition is met, the capture process is triggered and after 20s the systems stops communicating with the DSP and sends the data via Wi-Fi to a python server application. The Wi-Fi protocol used is the TCP, it is a reliable communication mean which ensure the retransmission of lost elements over the wireless bus. The python server application retrieves the data received via Wi-Fi, stores it in a user-accessible file and finally plots it. This communication structure is overall non-real time but yields to a 20s error-less capture of data centered on a fault condition.

1.4 Significance of the Study

The work marks a first step towards the development of a novel instrumentation device tailored for power electronic measurements. The absence of such possibility significantly limits the development of such systems in the automotive market. Therefore, companies such BRUSA HyPower eagerly seek for this technology with the objective of simplifying the testing process of their products. Moreover, flexible devices have also the benefit of enhancing the company customer service, enabling for on-field support and improving overall client experience.

The study delves into the potential of Wi-Fi isolation as a means to address challenges associated with diagnostic measurements. The successful establishment of this communication architecture lays the foundation for a potential future realization of a black-box, almost plug-and-play diagnostic device. Such a development has the potential to define a new trend-setting class of devices for power electronic measurements in the automotive field.

1.5 Organization of the Thesis

The thesis is organized in 5 chapters other that this one.

Chapter 2 is dedicated to a review of the literature to highlight the state of the art for diagnostic measurements in the field of power electronics. Chapter 3 presents the methodology followed during the thesis development, highlighting the tools used and the steps that led towards the conclusion of the work. Chapter 4, exploiting a top-down approach, describes the design and the development of the probe. Chapter 5 resumes the testing phase performed on the final implementation of the device showing the obtained results. Chapter 6 is dedicated to the discussion about the results of the testing phase. Here the conclusions are drawn and a reflection about the future possible development of the thesis is expressed.

Chapter 2

Overview of Diagnostic Techniques in Power Electronics Systems

Different solutions to monitor electrical unexpected failures in power electronics systems have been proposed so far. There exists a variety of measurement devices designed for power electronics applications, such as differential probes, isolated probes, digital oscilloscopes and isolated channel oscilloscopes.

Usually, a setup is made of a combination of a probe and an oscilloscope. The most straightforward setup typically would be made by a differential probe in conjunction with a digital oscilloscope, however it cannot accommodate the needs of switching converters. The presence of both high and low voltages, the noisy environment and the safety requirements existing within the power converter typically require an isolated instrumentation.

Nowadays, most efficient power converter in automotive are high frequency switching converters based on Wide-Bandgap semiconductors. The employment of this technology allows to reach high efficiencies and power density. The fast switching transitions demand a significantly increased measurement bandwidth up to hundred MHz to measure the signals of interest. Therefore, high performance measurement tools are required to accurately characterize and verify the correct operation of the circuit. In addition, certain measurements have to be performed on floating reference potential, that is, the reference potential of the voltage to be measured differs from the reference potential of the measurement equipment which is typically referred to protective earth. Furthermore, isolation between different measurement channels is required when various measurements, all with different reference potential, are performed simultaneously[5].

Indeed the preferred setup for measurements is based on isolated differential probes in combination with a digital oscilloscope. This type of technology represents the current state of the art for measurements for diagnostic purposes in power electronic fields. in terms of efficiency and safety. Alternatively, another solution involves the employment of isolated channel oscilloscopes but due to their limited availability their usage is also limited.

2.1 Differential Probes

The most common type of probes are the differential probes. They are employed to measure the voltage difference between any two points without having to connect to ground one of the two tips. Their basic structure is presented in figure 2.1, it is a differential amplifier with buffered inputs. This circuit maintains a constant gain when the ratio between the impedances of the differential amplifier are equal (referring to the schematic 2.1, (R10||C6)/R8 = (R11||C5)/R9). The voltage follower buffers are used to decouple the operational amplifier from the input, thereby reducing the effect of the common mode voltage on the output. The resistor at the input of the circuit are used to adjust the input value to scale down the supply range for the buffer operational amplifiers to a more manageable and readily available value[6].

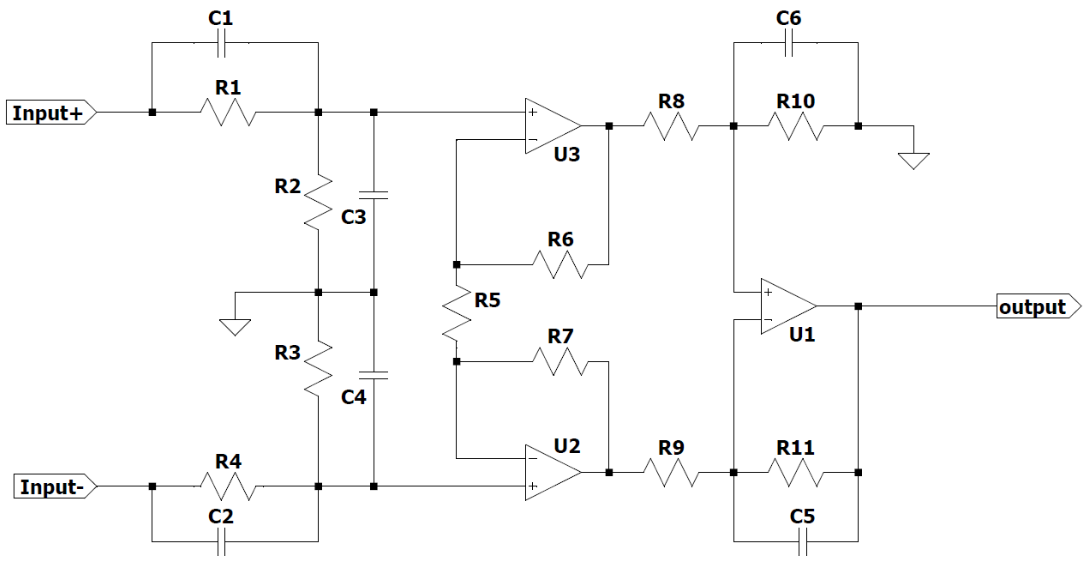


Figure 2.1: Differential Probe schematic

Typically, the differential amplifier gain is configured as attenuation, this allows to measure high voltages by adapting the output to the oscilloscope input. The frequency behaviour is dependant on the parasitic capacitance of the operational amplifier. In particular, the common mode rejection ratio and the input impedance decrease as the frequency grows. Consequently, this impose limitation on its usage at higher frequencies. Another limitation of the differential probes is related to their connection heads which usually present long wires. The wires introduce stray inductance on the circuit which produces unwanted spikes when measuring high switching circuits.

The main characteristics of a differential probe are:

- Attenuation
- Bandwidth
- Maximum differential and common mode voltage
- Noise level
- Input impedance
- DC offset
- Common mode rejection ratio (CMRR)

The choice of the appropriate probe depends on the specific characteristics of the system being measured. For instance, the "Bumblebee" probe[7] from PMK, as described in A.1, is a high voltage differential probe. It offers multiple possible attenuation values, like the 500:1, which enable high voltage measurements rescaling. The maximum voltage ratings for both differential and common mode of the probe determine its measurable limits. In this case, depending on the selected attenuation, the probe can handle voltages up to 2000V for both. The bandwidth is also large, allowing measures of signals up to 300MHz. Additionally, this probe is characterized by a good common mode rejection ratio, reaching -60dB at 1MHz.

Other worth to mention differential probes are Keysight N2792a[8], Teledyne LeCroy AP033[9], PicoTechnology TA044[10] and MicSig DP series[11].

2.2 Isolated Differential Probes

They offer significant advantages over traditional differential probes, especially in terms of handling common mode interference and allowing for higher maximum voltages between probe inputs and ground. In a non isolated differential probe, neither of the two differential inputs is grounded, but differential input circuit of the probe references to a 0V rail and this circuit is connected to ground. The same ground is shared with the instrumentation device. On the other hand, an isolated probe has isolation coupling between its differential input circuit and the test equipment, meaning it is not connected to the ground of the equipment. So, the non-isolated probe is used where the test equipment can share a common ground connection with the circuit/equipment under test. And an isolated probe is used when the two cannot share a common ground[12]. This concept is depicted in the 2.2.

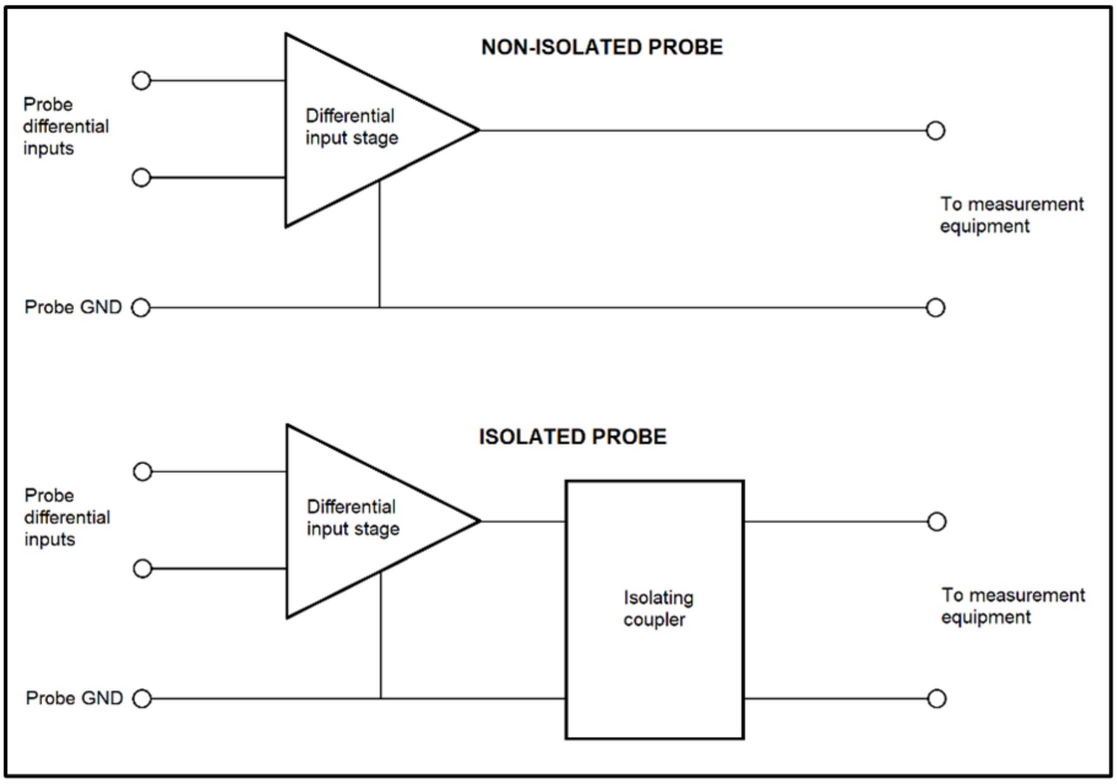


Figure 2.2: Non isolated and isolated probe concept topology

The most effective isolation technology for differential probes is the optical isolation. This probes are characterized by a tip where the differential amplifier circuit is located, the power supply and the signals are then transmitted to and from the probe exploiting the light via an optic fiber. The absence of a direct connection between the probe and the rest of the instrumentation provide isolation. However, it is worth noting that the cost of isolated probes is usually higher due to the fiber optic technology.

The advantages of this technique are the drastic reduction of common mode voltage noise in the measure, an improvement of the measurable bandwidth since the light is very fast, and an overall higher measurable voltage range. Comparing the example provided in the previous paragraph with the TIV probes from Tektronix[13], described in A.2, it is noticeable that optical isolation yields much higher CMRR being -145db at 1MHz, and more possible voltage ranges, allowing to sample with much more precision high and low voltages.

Some other worth to mention optical isolated differential probes are Teledyne LeCroy HVFO108[14], and Micsig MOIP series[15].

Another type of isolated differential probe designed for measurements in power electronics is presented in the article "A Digital Isolated High Voltage Probe for Measurements in Power Electronics" [16]. It combines an analog to digital converter with an FPGA. The ADC is responsible of converting the input signal to digital, after which the isolation is applied to the digital lines connected to the FPGA. Instead of using optical fiber technology, signal isolation is achieved through the use of digital isolators for the line connecting the ADC channels to the FPGA. The results confirm that also this type of probe can be used for high-precision measurements.

2.3 Isolated Channels Oscilloscopes

Isolated channel oscilloscopes are a valid alternative to isolated probes. The current state-of-the-art for isolated oscilloscope is the Cleverscope. It is a PC Mixed Signal 14 bit USB Oscilloscope with four isolated channels, spectrum analyzer, maths functions, streaming and isolated signal generator for frequency response analysis. The system consists of an ADC connected to the analog part within the isolated circuit. The ADC receives in input the probed differential voltage, and communicates via optic fiber with an FPGA responsible for processing and transmitting the information to the user. These isolated oscilloscopes are perfectly suited for measuring high switching power circuits, the bandwidth is of 200MHz and their CMRR is extremely good being -100db at 50MHz for a nominal operating isolated voltage range of 2kV. The only noticeable drawback of the device is it high price[17] [18].

Chapter 3

Virtual Scope Design Methodology

This Chapter introduces the main concepts adopted in this thesis to develop the scope for power electronics applications. In what follows, the methods and the tools employed to reach the objective of the work and the consequent test methodology are presented.

3.1 Design Methodology and Basic Concepts

The article "A Digital Isolated High Voltage Probe for Measurements in Power Electronics"[16] introduced a concept idea that served as inspiration for the development of the proposed project. The core idea revolves around employing the ADC of the DSP to sample the signals of the OBC and then send the digitized data via an isolated communication. This approach avoids the need of sampling the data from an external domain, eliminating the requirement for direct signal transmission through isolation. This solution is applicable because the DSP is already inside the High Voltage domain. Once the DSP retrieves the data, a second microcontroller is used as a Wi-Fi transceiver to retrieve the sampled data from the DSP and forward it wirelessly to the user domain. This secondary transmission benefits from less restrictions due to the inherent isolative properties of Wi-Fi, as it does not require direct contact with the user's instrumentation.

3.1.1 Open System Interconnection Model

To build a proper acquisition system is necessary to plan the correct flow of the data and consequently to define a network structure. It arises the necessity of deepening the knowledge about the communication protocols and the Open Systems

Interconnection Model (OSI Model). The OSI model defines a layered architecture for the logic design of a network. This architecture consists of a stacked pile of communication protocols divided into 7 layers. Together, these execute all the functionalities necessary for the communication of data inside the network, following a hieratical-logic model. The OSI model covers all the aspects of a network, starting from the physical connections to the applications used by the user. The layers are:

1. **Physical Layer:** it is responsible for the physical medium to transmit data over the network.
2. **Datalink Layer:** it encapsulate the data and provides error detection and correction.
3. **Network Layer:** it is responsible for routing data packets between different nodes on the network.
4. **Transport Layer:** it provides end-to-end communication between two applications.
5. **Session Layer:** it manages the communication between two applications, such as opening and closing connections.
6. **Presentation Layer:** it ensures that data is presented in a format that can be understood by the application.
7. **Application Layer:** it is responsible for providing services to the user, such as file transfer or email.

The biggest strength of the OSI model is its modularity because it enables the independent design of communication protocols for each layer without the need to consider the others. As result for example, this structure allows to built on the same physical mean different algorithms of data networking [19].

Given the OSI structure, the analysis of the problem is divided into three key aspects. The first focuses on the physical layer to determine the medium for communication. The second is about the transport layer to define a method for data transfer. The third is related to the application layer which provides the user with an interface. The architecture of the communication structure is then divided in two parts, each characterized by its own set of protocols. The first one is a wired connection based on the serial peripheral interface (SPI), where the data is transmitted bit per bit with a custom transport protocol built over it. The objective is to move the data sampled by the ADC of the DSP to the second microcontroller. The second one is based over Wi-Fi with a transmission control protocol (TCP) as transport layer on top of it. The aim is to communicate from

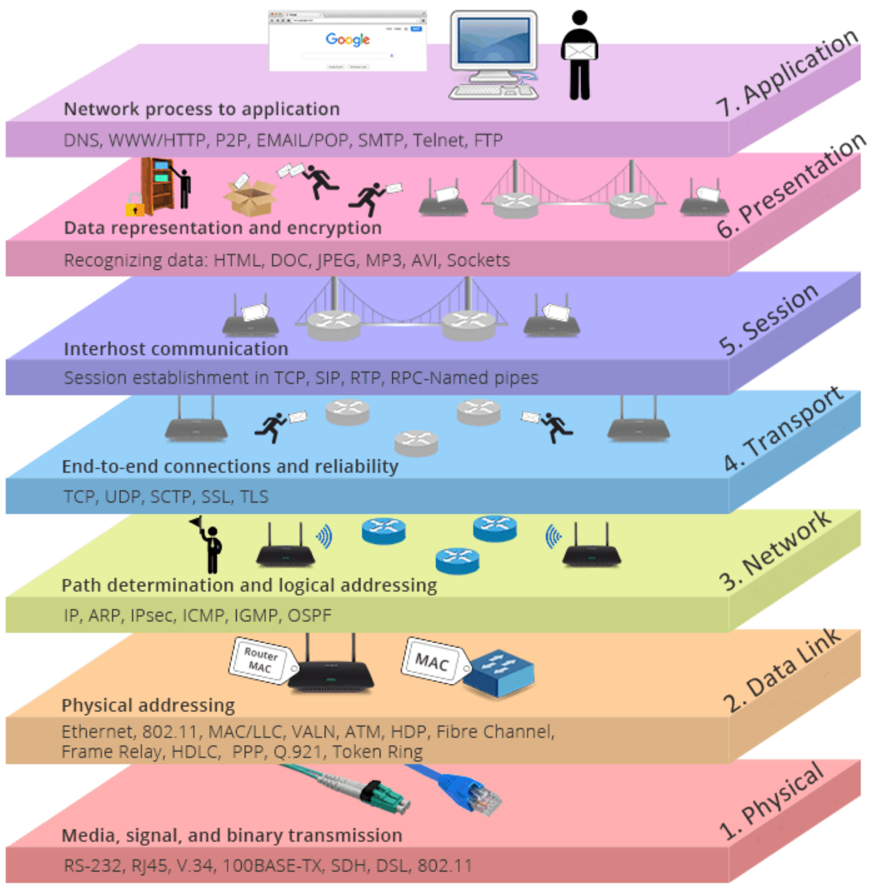

7. Application
Network process to application
DNS, WWW/HTTP, P2P, EMAIL/POP, SMTP, Telnet, FTP
6. Presentation
Data representation and encryption
Recognizing data: HTML, DOC, JPEG, MP3, AVI, Sockets
5. Session
Interhost communication
Session establishment in TCP, SIP, RTP, RPC-Named pipes
4. Transport
End-to-end connections and reliability
TCP, UDP, SCTP, SSL, TLS
3. Network
Path determination and logical addressing
IP, ARP, IPsec, ICMP, IGMP, OSPF
2. Data Link
Router MAC
MAC
Physical addressing
Ethernet, 802.11, MAC/LLC, VALN, ATM, HDP, Fibre Channel, Frame Relay, HDLC, PPP, Q.921, Token Ring
1. Physical
Media, signal, and binary transmission
RS-232, RJ45, V.34, 100BASE-TX, SDH, DSL, 802.11

Figure 3.1: OSI Model

the second microcontroller with a Python server application to move the data there to be accessible for the user. This architecture allows to have isolation between the user and the device under test because of the wireless connection between the two sides.

3.1.2 Serial Peripheral Interface

The SPI is a wired communication protocol operating at the first and second layer of the OSI model. It is characterized by a master-slave relationship and it exploits four wires to work: master in slave out (MISO), master out slave in (MOSI), clock (CLK) and slave select (SS). The master is enrolled of selecting a slave and providing the clock for the communication to occur. The data transmission occurs serially, one bit per clock pulse, and it is full-duplex. The SPI is a widely adopted protocol in microcontrollers, its fundamental operation principle relies on a precise management of the internal registers of the peripheral integrating it.

In a communication, the clock is registered by the slave only when the master is simultaneously transmitting data over the MOSI line. Therefore, the master is always sending information to the slave, depending on the needs this data can either carry meaningful information or be referred to as dummy data. Based on this concept it is possible to build also a half-duplex communication:

- **From Master to Slave:** no need for dummy data from the slave, the master only sends the data from the slave and this data also work as clock for the slave SPI interface.
- **From Slave to Master:** the master must send dummy data to the slave to keep the clock alive, the slave ignores the content of the data and sends its information to the master.

In case of full-duplex communication:

- **Full-Duplex:** the master sends the clock and data to the slave; in this case the dummy data is replaced by meaningful data. The slave then sends data to the master.

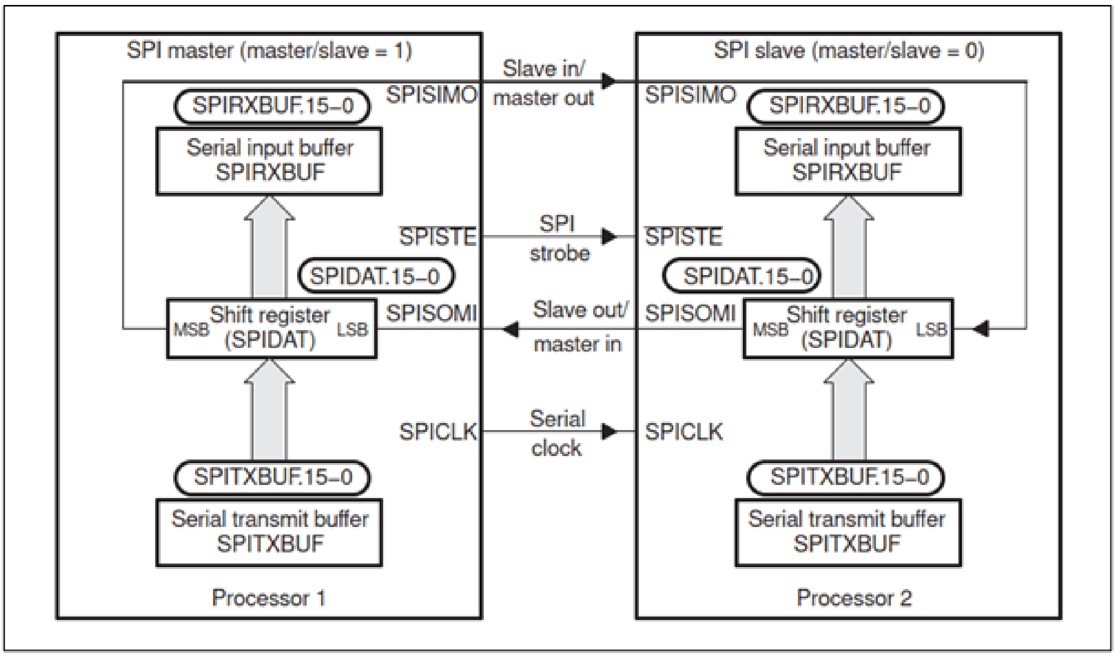

SPI master (master/slave = 1)
SPI slave (master/slave = 0)
Slave in/ master out
SPISIMO
SPIRXBUF.15–0
Serial input buffer SPIRXBUF
SPISTE
SPI strobe
SPIDAT.15–0
SPISOMI
Slave out/ master in
MSB Shift register (SPIDAT) LSB
SPICLK
Serial clock
SPITXBUF.15–0
Serial transmit buffer SPITXBUF
Processor 1
Processor 2

Figure 3.2: SPI communication registers

Referring to the figure 3.2, the communication mechanism is simple: the master sends data from its shift register (SPIDAT), bit per bit, to the shift register of the slave. In this process, the slave simultaneously moves on bit out while receiving one bit in. If the slave is not configured to manage any received data, all the content sent from the master to the slave is mirrored back to the master. Depending on the type of configuration for the communication, the interface can act differently. Considering the most comprehensive scenario of a full duplex communication, all the steps the interface must perform are covered. In this case, the master initiates the process by sending one bit to the slave. Before receiving this bit, the slave must load its shift register with the data it intends to transmit back to the master through its serial transmit buffer. Failing this procedure leads the master to receive back its own sent data. If the shift register is properly configured, the slave sends one bit out and receives one bit in. Once the shift register is full with the received data, the content is copied to the serial input buffer. Subsequently, the shift register is loaded again with new data from the Serial transmit buffer, making it ready for another communication cycle.

Summarizing:

1. Master and Slave must load properly the shift register from the Serial transmit buffer, otherwise if the slave receives the bit before this procedure the master will receive back the data it just sent.

2. The slave and the master send and receive data until their shift registers are filled with new information.

3. Then the master and the slave copy the content of the shift register in the Serial input buffer to be read.

4. Then the content of the shift register is loaded with new data coming from the Serial transmit buffer to be ready for a new communication cycle.

This communication method can be enhanced by incorporating a FIFO to each serial buffer. FIFO stands for First In, First Out. It operates as a data structure that maintains a queue where the first data item added is the first to be removed. It is used to create a queue for the data to transmit to avoid congestion. When a FIFO is employed, the data intended for transmission is placed into the transmission FIFO rather than directly into the serial transmit buffer. Similarly, data meant for reading is stored within the receive FIFO rather than the serial read buffer. From a practical point of view, the communication is organized by words. This means that, although the communication is still serial (bit by bit), from a register programming perspective the data is loaded by group of a user defined dimension of bits.

3.1.3 Transmission Control Protocol

TCP is a transport layer protocol of OSI model, and it is used in combination with the above network layer protocol IP. This protocol primary purpose is to provide reliability over the communication on the underlaying layers by providing control over packet losses, data congestion and data flow. After the establishment of a connection between terminal nodes, this communication channel sets up a bidirectional (full-duplex) byte flow.

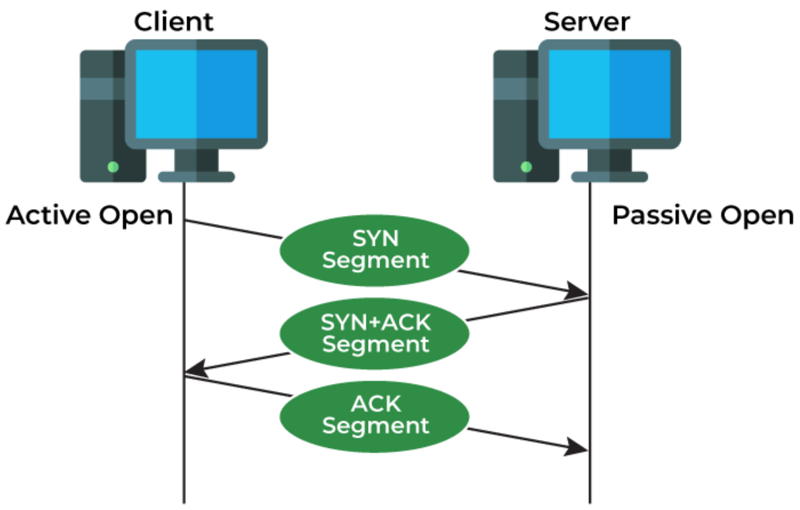


Figure 3.3: TCP three-way handshake procedure

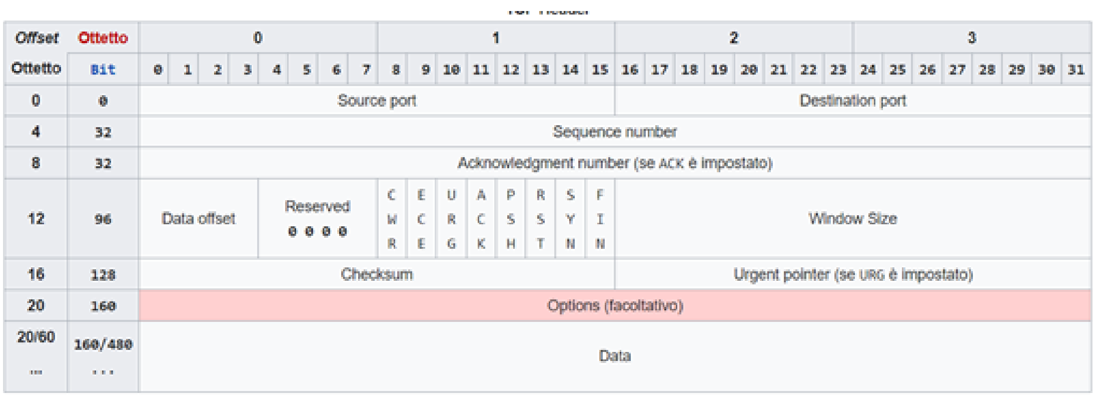
It is a connection oriented protocol, meaning that it requires a connection procedure to be performed between the sender and the receiver to operate a communication. This procedure is a three-way handshake and it is presented in the figure **??**. The steps are:

1. The client initiates the data transfer by sending a sequence number to the server, representing the number that the data packet transfer should begin with.

2. The server acknowledges the client sequence number and responds with its own one.

3. The client checks the received sequence number and then acknowledges the server, completing the connection procedure.

The strenght of this protocol is its reliability in terms of data accuracy. The three-way handshake is employed also in the transmission process, and it ensures to always retransmit the data in case of failed transmission. The protocol data unit is presented in figure 3.4.



Offset	Ottetto	0								1								2								3							
Ottetto	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	32	Acknowledgment number (se ACK è impostato)																															
12	96	Data offset				Reserved 0 0 0 0				CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
16	128	Checksum																Urgent pointer (se URG è impostato)															
20	160	Options (facoltativo)																															
20/60 ...	160/480 ...	Data																															

Figure 3.4: TCP protocol data unit

Each segment encapsulates a TCP header and a TCP payload. The header is used to establish a communication channel between TCP sender and TCP receiver.It contains all the characteristics of the established connection, like the used port to communicate, the sequence number used for the handshake mechanism, and other feature characterizing the defined transport layer. The payload contains the data object of the communication.

This protocol is perfectly suited for non-real time data transfer because. Despite being not very fast due to the overhead introduced by the three-way handshake procedure, it guarantees a lossless transmission of the data[20].

3.2 Hardware and Software Tools

The tools adopted to develop the systems are:

- **Arduino IDE**[21]**:** integrated development environment (IDE) used to program the ESP32 board.
- **Code Compose Studio**[22]**:** IDE used to program the TI DSP of the on-board charger.
- **Visual Studio Code**[23]**:** integrated development environment used to program the server application interacting with the ESP32.
- **Wireshark**[24]**:** it is a free and open-source packet analyzer software that can be used to capture and analyze network traffic. It allows to view the contents of network packets, including the headers, payload, and timestamps. It is exploited to validate the test done over the Wi-Fi connection.

- **PCAN Explorer 6**[3]**:** it is software used to send and read messages on the CAN bus. It is employed to configure and control the OBC during the testing validation of the final integration phase of the virtual scope device.
- **PicoScope**[25]**:** it is a logic analyzer which is an electronic instrument that captures and displays multiple digital signals simultaneously in a digital circuit. It is used to check the test results about the data the ESP32 and the DSP exchange on the SPI bus.

3.3 Integration of the Virtual Scope

The first part of the project has been developed exploiting a testbench to simulate the behaviour of the DSP of the OBC. This development proceeded in three main steps: first, the creation of the SPI communication architecture; next, the establishment of the Wi-Fi communication architecture; and finally, the integration of both communication methods together. Once the full communication architecture was validated on the testbench, the architecture was implemented on an actual on-board charger. The integration into the on-board charger required an initial hardware modifications to align the DSP hardware configuration to the needs of the SPI interface. Subsequently, a firmware update including the new SPI interface and communication protocol was applied to the DSP.

After the setup for the final integration, the communication was tested by capturing signals of the OBC in two different states: standby and charging. These two states represents the two operational conditions of the on-board charger, distinguishing between when the converter is inactive and when it is switching.

3.4 Testing Procedures

To validate the system each step of the development required a test. During the testbench validation, the following critical criteria were assessed:

1. For the SPI communication, it was crucial to ensure that all sampled data was received by the ESP32, and at the correct rate matching the sampling speed of 45kHz of the ADC within the DSP.
2. For the Wi-Fi communication, the aim was to receive on the Python server all data without any losses, and ensuring it fell within acceptable limits for overhead time due to the TCP protocol. The non-real time application does not impose stringent requirements on this communication time, nevertheless it cannot be too high, otherwise the user would wait too much time to receive the data.

After the validation of the testbench, the tests performed on the real on-board charger were consistency test. It was checked that the real system responded like the testbench in its standby and charging state.

Chapter 4

Virtual Scope Design

This chapter is dedicated to the description of the virtual scope design. The goal of the project is to develop a system capable of capturing various signals from BRUSA HyPower's on-board charger, including DC voltage and current, AC phase currents and voltages, and operating temperature. The signals must be easily accessible for monitoring in case of system faults. To achieve this objective, the plan is to exploit the ADC of the digital signal processor operating within the high-voltage domain. A second microcontroller equipped with Wi-Fi transceiver is employed to continuously retrieve real-time data samples from the DSP via SPI and transmit them wirelessly to a user-accessible server application. The communication with the server occurs in non-real-time; in case of a fault detection, the second microcontroller stops the SPI communication with the DSP and forwards the retrieved data to the server application. As a result, on the server it is possible to visualize a snapshot of the analyzed waveforms.

Before to system design, an analysis of the requirements is conducted, taking into consideration the current OBC implementation of BRUSA HyPower and the specific needs of the company. This analysis serves as the foundation for compiling the bill of components necessary for system development

4.1 Requirements for the System

The virtual scope must communicate with the DSP of the BRUSA HyPower's OBC. The requirements are to measure four signals simultaneously respecting the sampling rate of the 16-bit ADC set at 45kHz without losing any information. These requirements define, the number of channels, the bandwidth, the necessary memory of the system, and the communication protocols to use. Furthermore, the virtual scope should be isolated and practical to use. Therefore, it is essential to have a Wi-Fi transceiver module within a small and light form factor device. The given specification requests are:

- $N_{\text{bits}} = 16\text{bits}$ is the resolution of the ADC of the DSP, representing the number of bits composing a measurement.
- $N_{\text{channels}} = 4\text{channels}$ is the number of channels to sample.
- $SamplingRate_{\text{DSP}} = 45\text{kHz}$ is the sampling rate of the ADC of the DSP.
- Wi-Fi module.
- Small form factor.

4.1.1 Bandwidth

The bandwidth refers to the speed of the communication the virtual scope must respect to manage the amount of data to receive without losing information. The calculation to be performed is simple, it is necessary to consider how much bits must be transferred each second:

$$\begin{aligned} Bandwidth &= N_{\text{bits}} * N_{\text{channels}} * SamplingRate_{\text{DSP}} \\ &= 16\text{bits} * 4\text{channels} * 45\text{kHz} = 2.88\text{Mbps} \end{aligned} \tag{4.1}$$

4.1.2 Memory

The memory refers to the RAM needed to save the sampled elements locally on the virtual scope. The memory to store to store N cycles of measurements is the product of the number of cycles per the bandwidth:

$$\begin{aligned} Memory &= N_{\text{cycles}} * N_{\text{bits}} * N_{\text{channels}} * SamplingRate_{\text{DSP}} \\ &= N_{\text{cycles}} * Bandwidth \\ &= N_{\text{cycles}} * 2.88\text{Mb} \\ &= N_{\text{cycles}} * 360\text{KB} \end{aligned} \tag{4.2}$$

Therefore, to save one cycle of measurements it is necessary at least 360 KB of free space in the RAM. This value is useful to understand the maximum feasible duration of the capture time given a certain memory.

4.1.3 Capture Time

The capture time is memory dependent. Given that one cycle correspond to one second of captures, the formula of the capture time is the obtained from (4.2):

$$N_{\text{cycles}} = T_{\text{capture}} = \frac{Memory}{360\text{KB}} \tag{4.3}$$

Therefore, the greater is the memory of the chosen microcontroller the longest the capture can be.

4.1.4 Communication Protocols

The system is organized in a two-step communication: DSP to virtual scope and virtual scope to server application. The chosen protocol are respectively:

- **Serial Peripheral Interface:** The data communication between the DSP and the virtual scope must be real time and must respect the bandwidth requirement. This communication speed is quite high and necessitate of a fast protocol. Moreover, the DSP on the on-board charger is not equipped with a wireless transceiver, therefore the communication must occur over a wired connection. Among the various options provided by the peripherals of the DSP, the only protocol capable of meeting the required communication speed is the Serial Peripheral Interface. The DSP support a SPI peripheral with a maximum bitrate of 25MHz. Other available communication protocol are UART, CAN and I2C but they are too slow for this application.

- **Transmission Control Protocol:** The communication between the virtual scope and the server application does not have a stringent bitrate restriction but it should be sufficiently fast to minimize the user waiting time when accessing captured data. The requirements over this connection are isolation and error correction. Wi-Fi communications are isolated by definition since there is no direct connection between the two communication nodes. Among the possible solutions for Wi-Fi based communication, the chosen protocol is the Transmission Control Protocol because it is robust to faulty transmissions. Another available protocol is the user datagram protocol (UDP), it has much lower communication overhead but it does not ensure retransmission of lost data.

4.2 Selection of Components

The specification defines precise requests for the virtual scope. The components accommodating all the requirements are a Wi-Fi integrated microcontroller combined with a server application as a user interface.

4.2.1 Microcontroller Choice

A microcontroller is a programmable compact system on a chip designed to govern specific operations within an embedded system without a complex front-end operating system. It combines a processor core, memory, input/output peripherals, and interfaces for communication with other devices. From a cost effective and versatility point of view this device category is perfectly suited to deal with communication management between different interfaces.

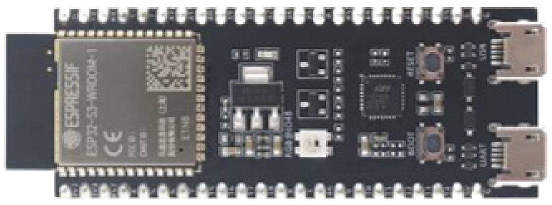


Figure 4.1: Espriff ESP32-S3 DevkitC-1U microcontroller

The chosen microcontroller for this application is the Espriff ESP32-S3-DevKitC-1U (P1R8N8 version)[26]. The benefits the ESP32 family of microcontroller offer are their Wi-Fi integrated interface and large memory. The U indicates that the microcontroller has an external antenna connector. The P1N8R8 refers to the dimensions of external flash and PSRAM that the ESP mounts, in this case 8MB of flash and 8MB of PSRAM which is exploited to expand the available memory. This type of microcontroller is taught especially for IoT and neural network projects where it is necessary to connected items to networks allowing also to have all the enhancements that a generic microcontroller offers (B.2). Moreover, the board is very cheap (10-15 euros), and it has a small form factor, making it suitable to build a practical and light device. The ESP32-S3 offers all the features requested to build the wireless sampling architecture for the virtual scope, emerging as a perfect candidate for the thesis development.

Programming this microcontroller is straightforward, as it is compatible with the Arduino IDE. The provided SPI peripheral ensures compliance with bandwidth and channel number requirements delivering a capture time of:

$$T_{\text{capture}} = \frac{Memory}{360\text{KB}} = \frac{8\text{MB}}{360\text{KB}} \approx 22\text{s} \tag{4.4}$$

Other analyzed options are:

- **Texas Instruments CC3235SF:**[27] This microcontroller is provided of Wi-Fi integrated module but a lower memory space, being limited to 1MB of Flash. Furthermore, its form factor is bigger than the ESP32 and it costs more. The advantage of this microcontroller solution would have been that, being produced by the same manufacturer of the DSP on the BRUSA HyPower's OBC, it would have provided easier interfacing of the two devices. Nevertheless, the above reasons led to choose the ESP32-S3 over the CC3235SF.

- **Realtek AMB01:**[28] This microcontroller presents the same characteristics of the ESP32-S3 but with less memory and at higher price. Therefore, the ESP32-S3 was preferred.

- **Raspberry Pi Zero 2 W:**[29] This microcontroller offers the same characteristics of the ESP32-S with a slightly higher price. However, this type of microcontroller are meant for software based applications and are more complex to program with respect to normal microcontrollers. It is a more structured system including an operative system based on Linux. The simplicity of the ESP32 has played the final role in the microcontroller choice, preferring the ESP32 to the Raspberry Pi.

4.2.2 Server Application

The server application is written in Python, the reason behind this choice is that it is a simple and adaptable language with a vast library support. The only downside of this language is its low computational speed but, in the context of non-real time application, it is not an issue.

For faster efficiency in terms of code performances, it would have been possible to write the server in C. Nevertheless, the complexity of the language with respect to Python led to prefer this last solution due to the absence of restriction with respect to the code efficiency. Furthermore, Python is a language made for application development, therefore the support it has in terms of libraries is much more valuable with respect to the optimization C could provide.

4.2.3 Testbench

The development phase was first carried out over a testbench. The testbench in figure 4.2 simulates the real behaviour of the OBC, providing the update of the signals to capture with a rate of 45kHz. The testbench consists of the ESP32 connected with the LAUNCHXL-F280049C evaluation board[30]. This is a microcontroller board containing the same microcontroller used as DSP in the OBC.

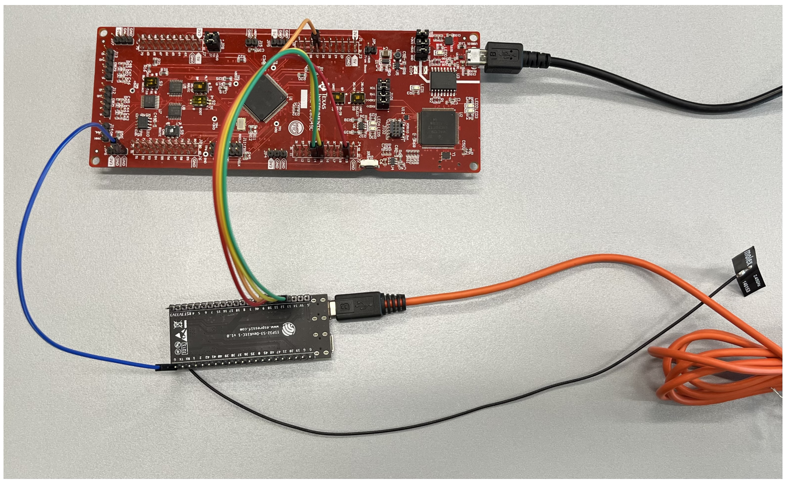


Figure 4.2: Virtual scope testbench: ESP32 and LAUNCHXL-F280049C

4.3 Description of the Virtual Scope Design

The virtual scope is organized as an interconnected system and its structure is based on three main components: the DSP of the OBC, the ESP32-S3, and the Python server application. The block structure of the system is presented in the figure 4.3.

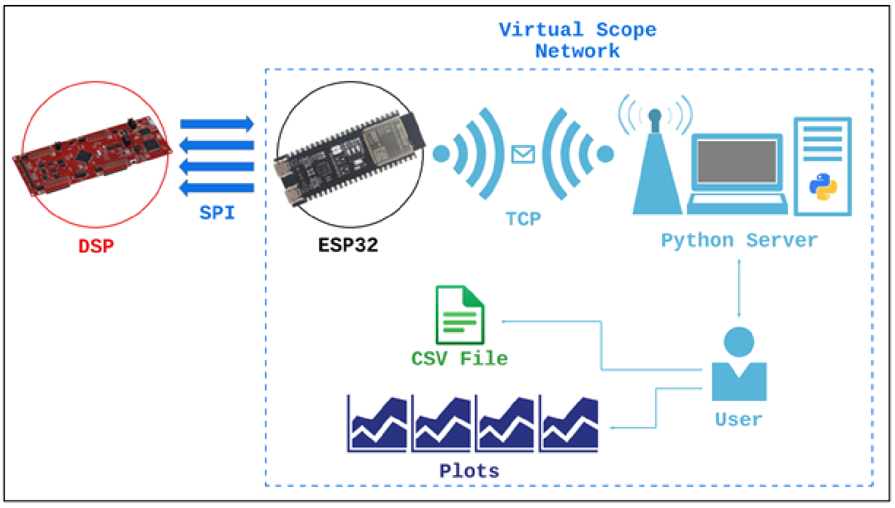


Figure 4.3: Virtual Scope architecture design

- **DSP:**
  The BRUSA HyPower's OBC adopts the Texas Instruments F280049[31]

microcontroller as DSP. This type of microcontroller is quite complete and possesses almost every peripheral it could be useful for an embedded system (reference to B.1).

The DSP functions which interest the project are related only to the ADC sampling. It provides a new value for each of these measures with a frequency of 45kHz. The virtual scope has to retrieve four of these signals via SPI. The virtual scope algorithm for the OBC must integrate the SPI configuration and communication protocol.

For the project, the main components used are the SPI interface peripheral, the GPIOs, CPU Timers and the ePIE module. The SPI is used to send the data from the DSP to the virtual scope. The GPIOs used are four to allow the SPI to properly be configured, and it is necessary to program their functionality to work as SPI pins. The CPU timer is used only in the testbench to generate a routine every 45kHz to generate new values which simulate the sampling of the real measurement performed by the power board. The ePIE module is used because the interrupt routines are exploited in the programming of the microcontroller.

- **Python Server:**
  The python server application must run on the user PC. It allows the user to insert some parameters to configure the measurement capture, and graphically represent the data generated by the DSP. The PC acts like an AP creating a local network via Wi-Fi hotspot exploiting the network it is connected to. The ESP32 then connects to the network as a client and interact with the server via TCP socket.

  The server functions are implemented using the libraries:

  - Socket[32]: it is used for the TCP communication.
  - Pandas[33]: it is used for the CSV file management.
  - Matplotlib[34]: it is used to plot the data.
  - Numpy[35]: it is used to write the data inside the CSV file.

  The server main functions are: reading user-defined capture configuration parameters, receiving measurements from the ESP32, storing data in a comma-separated value (CSV) file, and displaying data through graphical plots.

- **ESP32-S3:**
  This microcontroller is the actual virtual scope. This microcontroller is the node connecting the DSP with the user. Its main functions are: retrieving the data from the DSP via SPI, and sending data to the server application via Wi-Fi with the TCP protocol.

The peripheral used for the project are: the SPI, the GPIOs and the integrated Wi-Fi transciever.

The system achieve isolation with respect to the user through to the Wi-Fi connection between the server application and the virtual scope device, while the latter remains connected to the on-board charger. The Wi-Fi grants isolation because of the absence of a direct physical connection between the user interface and the on-board charger. The advantage of the Wi-Fi over the optic communication is its easier implementation because Wi-Fi transceiver are much less expensive and fragile with respect to optic fibers. They are already available in microcontrollers and easily configurable. The biggest drawback of Wi-Fi with respect to optic fiber technology is the reachable bandwidth which is much smaller. However for this particular application where the bit rate is in the order of the MHz, this limitation holds little significance.

4.4 Working Principle of the Virtual Scope

The working principle of the virtual scope is to continuously read the DSP data and to generate a snapshot of the sampled waveforms when a fault occurs. The snapshot is centered on the time instant the fault happens including data previous and post the detection. Each channel measurement is acquired in real-time by the ESP32 and stored sequentially in a ring buffer. A ring buffer is a circular buffer: when the pointer writes to the last position of the array, its value is reset to the first position. This allows the buffer to continuously store data without ever overflowing. This characteristic is exploited to be able to center the snapshot on the fault. A fault is detected when a defined channel measurement exceeds a defined threshold, this occurrence triggers the beginning of the snapshot capture. After the triggering event, the ring buffer contains all elements of from the four channels preceding the error. Subsequently, the array is written for half of its capacity more. Once this operation is completed the buffer contains half of the measurements recorded before the fault and half of the measurements recorded after the fault. Upon capturing the snapshot, the continuous data acquisition from the DSP is halted, and the content of the ring buffer is sent to the server application for visualization purposes. The data within the buffer is saved on a CSV file and each channel is plotted on a graph.

The configuration of the capture is performed by the user which is asked by the server to select:

- **N_channel**: the number of channels to plot, the capture always samples the 4 channels but then the server plots only the desired number of channels. This value is comprised between 1 and 4.

- **Trigg_channel**: the channel triggering the capture, it decides on which channel the triggering event is checked. This value is comprised between 1 and N_channel.
- **Trigg_type**: the type of triggering condition, once the triggering threshold is defined it is requested to know also if it is a low side boundary or high side boundary. Therefore, there are two types of Trigg_type, either the measured value is greater ('g') or lower ('l') than the threshold.
- **Trigg_value**: the threshold value which triggers the capture. This value is dependent on the measurement to sample.

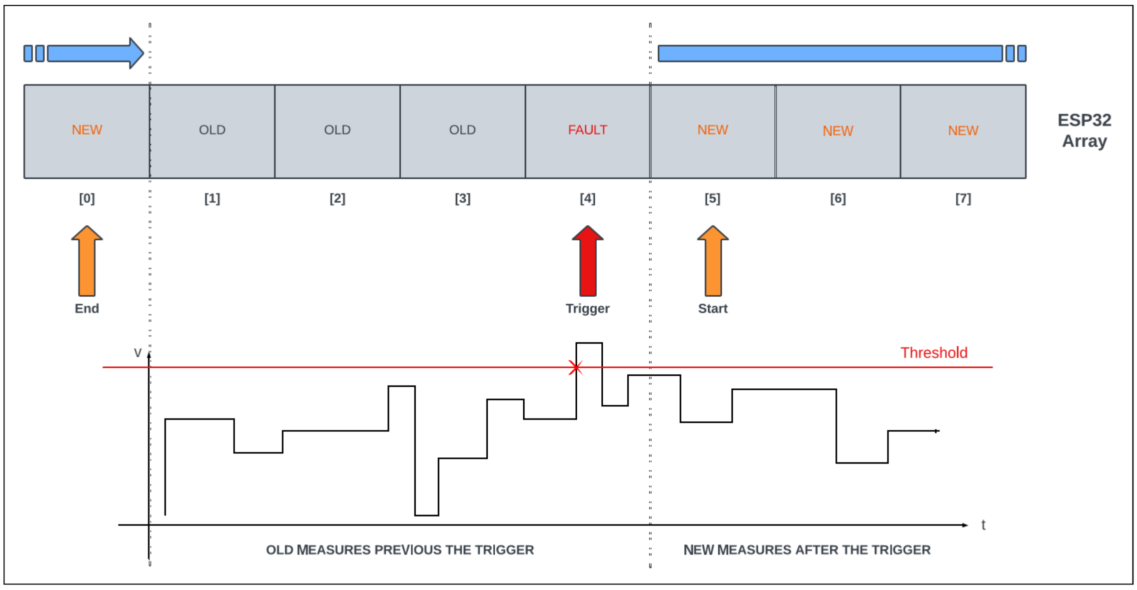


Figure 4.4: Ring buffer feature: how data is acquired when triggering occurs

Figure 4.4 shows an example of capture. For simplicity, a single channel is considered. The capture starts when the trigger condition is met (red cross), the ring buffer is then filled starting from the next position, up to half of its length (four elements from position [5] to [0]). When the ring buffer is sent to the server, the reading operation starts at the end position +1 (position [1]). This ensures that the fault will be at the center of the snapshot of the waveform.

4.4.1 Workflow of the Virtual Scope

The workflow of the virtual scope is described in figure 4.5.

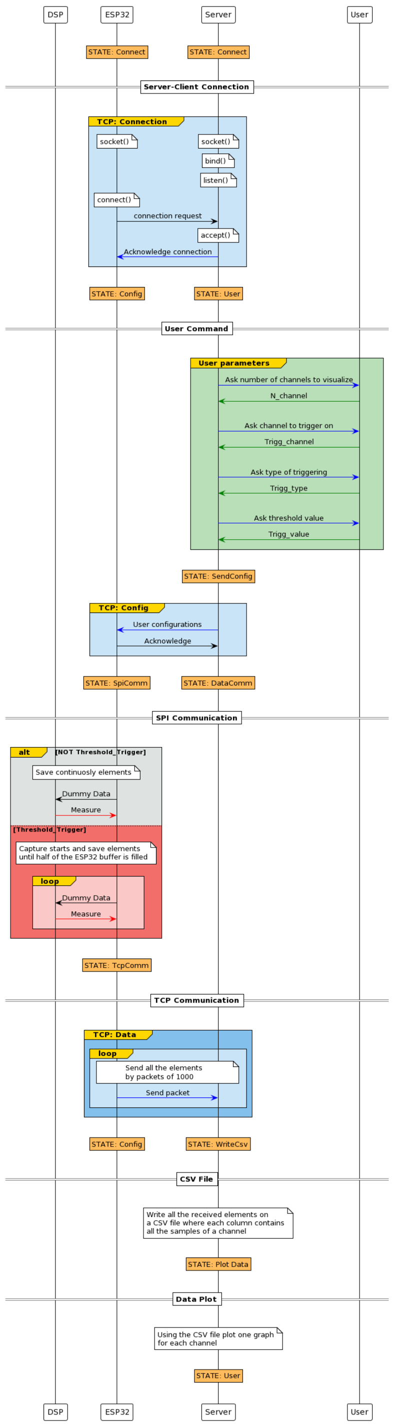


Figure 4.5: Workflow of the virtual scope

1. **TCP Connection**
   In this phase, both the ESP32 and the Server initialize a TCP socket and connect to the Wi-Fi network generated by the user's PC using a Wi-Fi hotspot. Subsequently, the server continuously monitors for connection requests until one is received. After receiving a connection request, the server acknowledges it and establishes a client-server relationship with the requesting device.

2. **User parameters**
   Once the connection is established, the server asks the user to select the parameters for the configuration of the capture.

3. **TCP Config**
   Upon acquiring the user configurations, the server assembles them into a string and it sends it via TCP to the ESP32. The ESP32 processes the received string and acknowledges to the Server.

4. **SPI Communication**
   In this phase, the ESP32 communicate with the DSP through SPI, reading continuously all the measurements until the trigger event occur. Following the triggering event, the ring buffer of the ESP32 is filled for half of its dimension more. At this point, the data is ready to be sent to the Server.

5. **TCP Data**
   During this step, the ESP32 divides the ring buffer content in smaller segments of 1000 elements each and forwards them via TCP to the server. The server reassembles the original array and build it into a matrix such that each column contains one of the channel measurements.

6. **CSV File**
   Once the matrix is created, the server writes the CSV file keeping the same format.

7. **Data Plot**
   After the CSV file is written, the server plots the requested graphs.

4.4.2 State Machines of the Virtual Scope

The state machines of the system are described in the figure 4.5.

1. **ESP32**

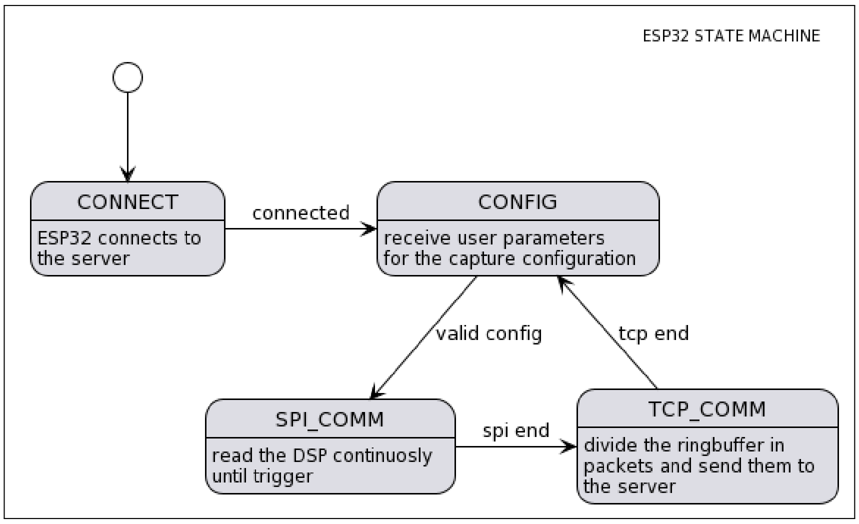
   - *Connect:* the device establishes a connection with the Server.

- *Config:* the device receives the user configuration from the server via TCP. The device processes this information and validates them, if the validation result is positive then it acknowledges the server. Otherwise, it waits for another TCP packet containing a valid configuration.
- *SPI_comm:* the device continuously receives data from the DSP through SPI, if the corresponding element of a channel goes over the threshold trigger the capture starts. Half of the ring buffer is filled with new elements and the device changes state.
- *TCP_comm:* the device divides the ring buffer in several packets of 1000 elements and sends them to the server one by one via TCP. Once the packets are all sent, the device changes state.

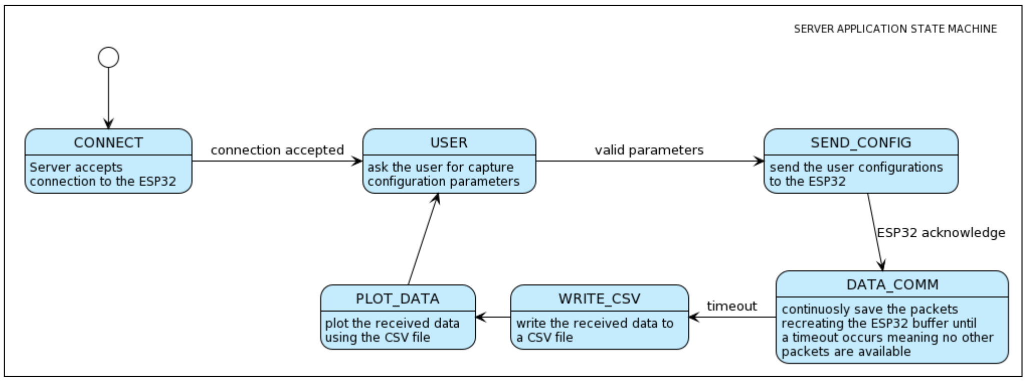
2. **Server**

   - *Connect:* the device listens and establishes a connection with the client.
   - *User:* the device asks the user for the configuration parameters, once valid values are received it changes state.
   - *Send_config:* the device packs the user parameters in a string and forwards it to the client. When the acknowledge is received it changes state.
   - *Data_comm:* the device waits for the client to send TCP packets, when they are received, the server updates a local array with the received content and when no more packets are received (a timeout elapses), the server changes state.
   - *Write_csv:* the server writes the CSV file from the local array it built in the previous state.
   - *Plot_data:* the server plots the number of channels requested by the user exploiting the data stored on the CSV file.

3. **DSP**
   The complete implementation of the DSP on the on-board charger does many functions and has many states. For this application it is not important to detail all the states of the DSP. What matters is to know that it sends 16-bit data via SPI at a 45kHz rate.


ESP32 STATE MACHINE
CONNECT
ESP32 connects to the server
connected
CONFIG
receive user parameters for the capture configuration
valid config
tcp end
SPI_COMM
read the DSP continuosly until trigger
spi end
TCP_COMM
divide the ringbuffer in packets and send them to the server

(a) State Machine of the ESP32


SERVER APPLICATION STATE MACHINE
CONNECT
Server accepts connection to the ESP32
connection accepted
USER
ask the user for capture configuration parameters
valid parameters
SEND_CONFIG
send the user configurations to the ESP32
ESP32 acknowledge
DATA_COMM
continuosly save the packets recreating the ESP32 buffer until a timeout occurs meaning no other packets are available
timeout
WRITE_CSV
write the received data to a CSV file
PLOT_DATA
plot the received data using the CSV file

(b) State Machine of the Python Server

Figure 4.5: State machines of the virtual scope components

4.4.3 SPI Communication Architecture

In this communication, the ESP32 is the master and the DSP is a slave. The transmission is organized in a loop fashion, where each cycle transmits one block comprising one sample for each of the four channels. Therefore, each block is made of four elements of 16 bits each. A visual representation of this process is depicted in figure 4.6.

Within one cycle:

- The slave loads its transmission FIFO with four elements of 16-bits and waits for the dummy data from the master to send them on the serial line.
- The master sends four consecutive dummy data and receives four consecutive

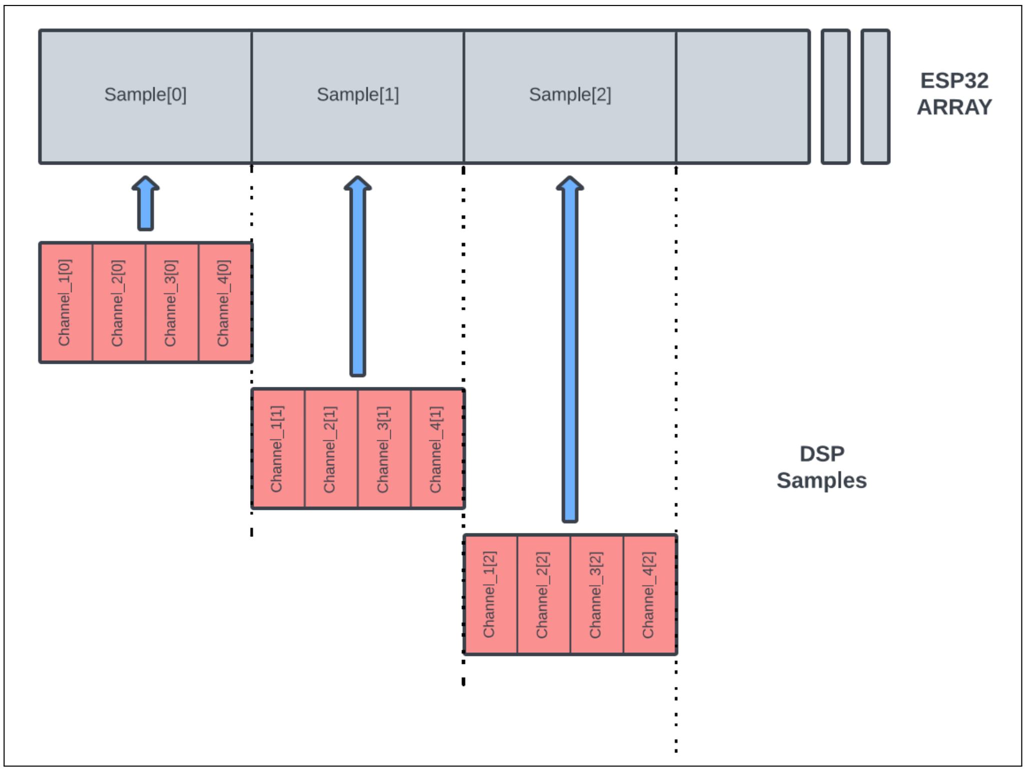


Figure 4.6: Four channel ESP32 data storing

elements from the slave. The received measurements are saved in the ESP32 ring buffer.

The whole SPI communication is a continuous repetition of the cycle depicted in the figure 4.7, the end of the cycle is defined by the the fault detection occurrence.

DSP Algorithm for SPI Communication

The DSP has to communicate continuously with the ESP32 via SPI, the algorithm is organized in 2 parts:

1. *Configuration of the SPI interface*
   The SPI interface must be configured accordingly to the needs of the communication architecture. The DSP is a slave and the parameter to define are: Bit Rate, clock configuration mode, initial bit of the word, polarity of the slave select, FIFO enable, and the pinmux configuration.

2. *Data transmission*
   The data transmission is regulated by periodically adding a block of four

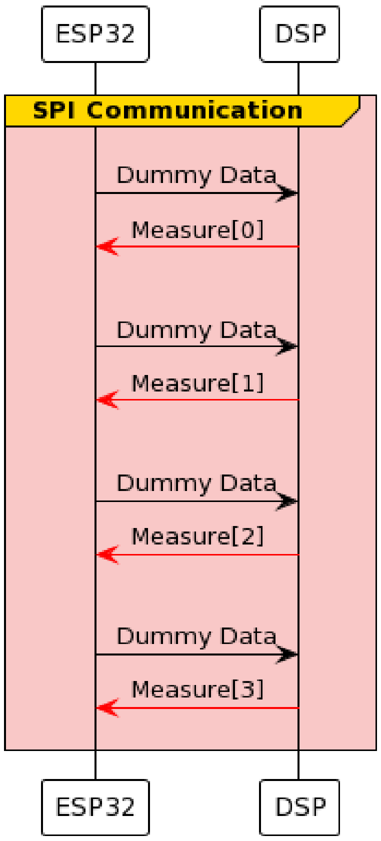


Figure 4.7: Diagram of one cycle of SPI communication

elements to the transmission FIFO of the DSP. This happens at a rate of 45 kHz, which correspond with the ADC sampling rate of the DSP. Therefore, every time the DSP samples new measurements, they are inserted into the FIFO. The SPI then proceeds to transmit these elements transparently to the data transmission algorithm.

The data transmission algorithm operates under the control of two conditions, and transmission occurs only when both of these conditions are met:

- The value of the slave select pin must be active.
- There must be enough space inside the FIFO to add four elements, otherwise the sample is skipped. Ideally, this situation should not happen, nevertheless this condition ensures to always have a chronological update of the measurements avoiding an uncontrolled subscription or overflowing of the FIFO.

Figure 4.8 shows the flowchart describing the algorithm for the DSP.

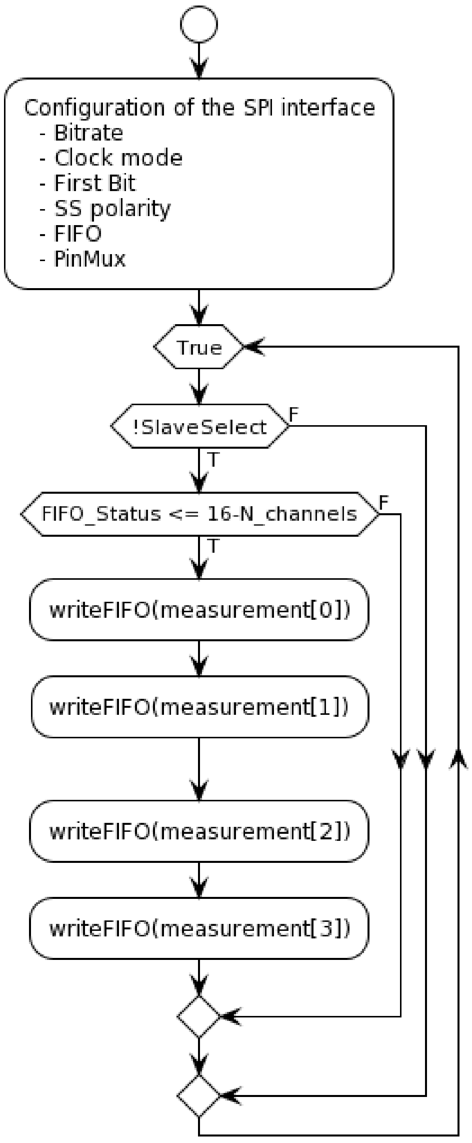

Configuration of the SPI interface
- Bitrate
- Clock mode
- First Bit
- SS polarity
- FIFO
- PinMux
True
!SlaveSelect
F
T
FIFO_Status <= 16-N_channels
F
T
writeFIFO(measurement[0])
writeFIFO(measurement[1])
writeFIFO(measurement[2])
writeFIFO(measurement[3])

Figure 4.8: Flowchart of DSP SPI communication

ESP32 Algorithm for SPI Communication

The ESP32 algorithm for SPI communication is divided in two parts:

1. *Configuration of the SPI interface*
   The configurations to perform are intuitively the same done on the DSP.

2. *Data Reception*
   The read data is consecutively stored in the ring buffer. The algorithm

performs the fault detection check on the received data; if the received data exceeds the threshold, the capture procedure is initiated. The ring buffer is filled for half of its length and the SPI communication ends.

It is important to correctly deal with the indexes and the pointer of the ring buffer to correctly take advantage of its circular buffer features. The following operations are critical:

- Reset the pointer from the last position to the first when the superior limit position of the buffer is reached.
- Properly define the starting point from which the ring buffer must be read once the capture is completed. This value is used in the TCP data communication phase. The starting position is given by equation 4.5.
- Properly define the ending point after which the capturing must stop once the triggering occurs. This position is given by equation 4.6.

Summarizing, the data reception is organized in the following steps:

(a) Pull-down the SS pin to start SPI communication.

(b) Read four by four the data over the MISO and save it continuously in the ring buffer.

(c) Compare the received data with the threshold.

(d) If the check is positive, the indexes are evaluated and then the data is saved in the ring buffer until the end position index is reached.

(e) Pull-up the SS pin to terminate SPI communication.

$$Index_{\text{Start}} = \begin{cases} Index_{\text{Fault}} - \dfrac{bufferDim}{2} & if \geq 0 \\ bufferDim + (Index_{\text{Fault}} - \dfrac{bufferDim}{2}) & if < 0 \end{cases} \tag{4.5}$$

$$Index_{\text{End}} = \begin{cases} Index_{\text{Fault}} + \dfrac{bufferDim}{2} - 1 & if > bufferDim \\ (Index_{\text{Fault}} - \dfrac{bufferDim}{2}) - bufferDim - 1 & if \leq bufferDim \end{cases} \tag{4.6}$$

Figure 4.9 shows the flowchart describing the SPI communication algorithm for the ESP32.

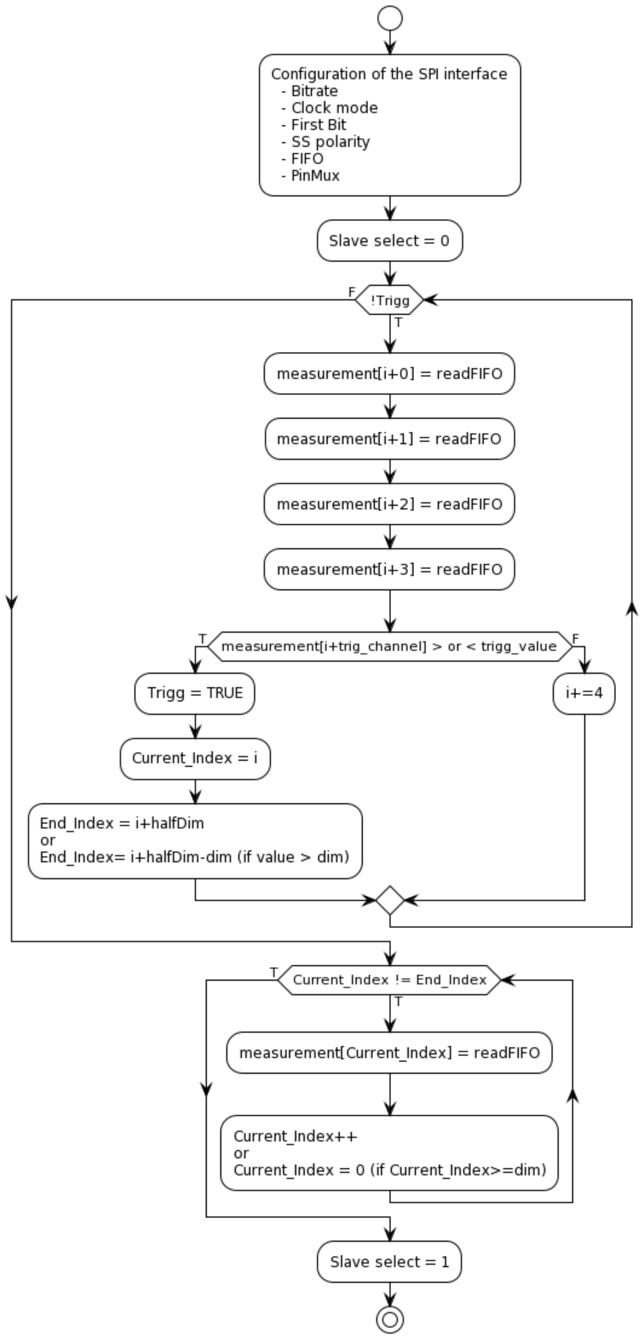

Configuration of the SPI interface
- Bitrate
- Clock mode
- First Bit
- SS polarity
- FIFO
- PinMux
Slave select = 0
!Trigg
F
T
measurement[i+0] = readFIFO
measurement[i+1] = readFIFO
measurement[i+2] = readFIFO
measurement[i+3] = readFIFO
measurement[i+trig_channel] > or < trigg_value
T
F
Trigg = TRUE
i+=4
Current_Index = i
End_Index = i+halfDim
or
End_Index= i+halfDim-dim (if value > dim)
Current_Index != End_Index
T
T
measurement[Current_Index] = readFIFO
Current_Index++
or
Current_Index = 0 (if Current_Index>=dim)
Slave select = 1

Figure 4.9: Flowchart of DSP SPI communication

4.4.4 TCP Communication Architecture

The communication between the ESP32 and the server exploits a local network made via hotspot by the computer which runs the server application. This communication occurs over Wi-Fi using the TCP protocol, with the ESP32 as the client. The TCP communication happens three times within a complete acquisition cycle:

1. *Client-Server connection*
   This communication instance occurs only once when the virtual scope is started. The server is programmed to accept a single connection at the time. While no client is connected the server listens to the port it is bind to. The ESP32, when connected to the Wi-Fi network, forwards to the server a connection request. Upon receiving the request, if no other connection is currently active, the server acknowledges the request and establishes the connection.

2. *User configuration*
   This communication instance occurs when the user inserts the configuration parameters for the acquisition before the virtual scope begins the data acquisition. The server transmits this information as a string to the ESP32 which decodes it. If the information is valid, the ESP32 answers with an acknowledge.

3. *DSP data transmission*
   This communication instance occurs after the ESP32 has completed the capture of the DSP via SPI. The ESP32 data array is segmented into packets, each containing 1000 elements, and these packets are sent sequentially to the server, as in the figure 4.10. Each packet is essentially a string composed of these 1000 elements, with commas separating them. On the server side, these strings are reconstructed into a single long string. Once this string is assembled, it is divided into a python list by dividing each element from the other through the comma.

The figure 4.11 presents a graphical representation of the TCP communication architecture described.

Once all the data is retrieved, it is saved in a CSV file organized such that each column represents a channel, and each row contains one sample for each channel. The CSV file is then exploited to plot the data.

ESP32 Algorithm for TCP Communication

The algorithm the for the TCP communication can be divided in three phases where the ESP32 and the Server communicate: connection, user configuration and data transmission. The ESP32 algorithm is divided in two parts:

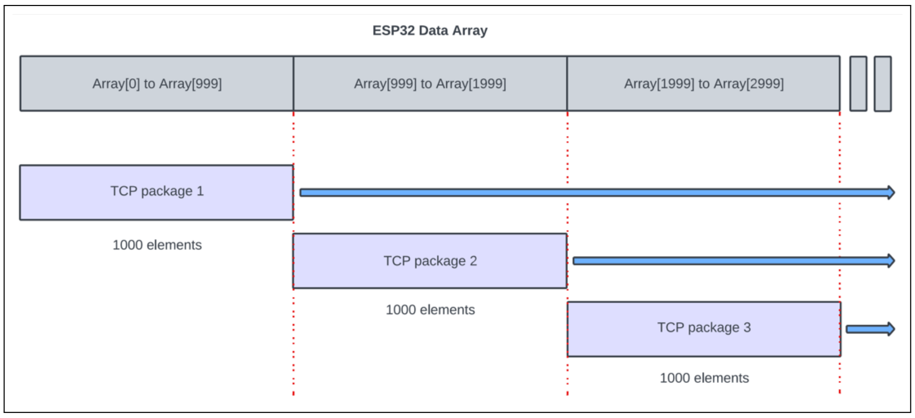


Figure 4.10: How TCP packets are sent from ESP32

1. *Configuration of the Wi-Fi interface*
   The Wi-Fi functions of the ESP32 are managed using the Arduino library "WiFi.h". To connect to the Wi-Fi network hosted by the user PC the ESP32 needs:

   - WIFI_SSID: the name of the Wi-Fi network hosted by the user pc.
   - WIFI_PASSWORD: the password of the Wi-Fi network.

   When the Wi-Fi connection is established, the ESP32 to establish a connection with the server must allocate a TCP socket, which should be configured accordingly to the server's settings. The required parameters are:

   - HOST: the IP address of the network host (the user PC), this address is static.
   - PORT: the defined port dedicated to the server-client communication, it can be almost any number, the chosen one is 8888.

2. *Wireless communication*
   The wireless communication follows the three-step communication architecture:

   (a) TCP: connection
   The WiFi.h library handles the three-way handshake procedure between the client and the server.

   (b) TCP: configuration
   The ESP32 reads continuously the TCP receive buffer and performs a validation check on the received packet. The packet to be valid must contain a string of four elements separated by a comma, and the content must be inside their range of values. For example, the string could

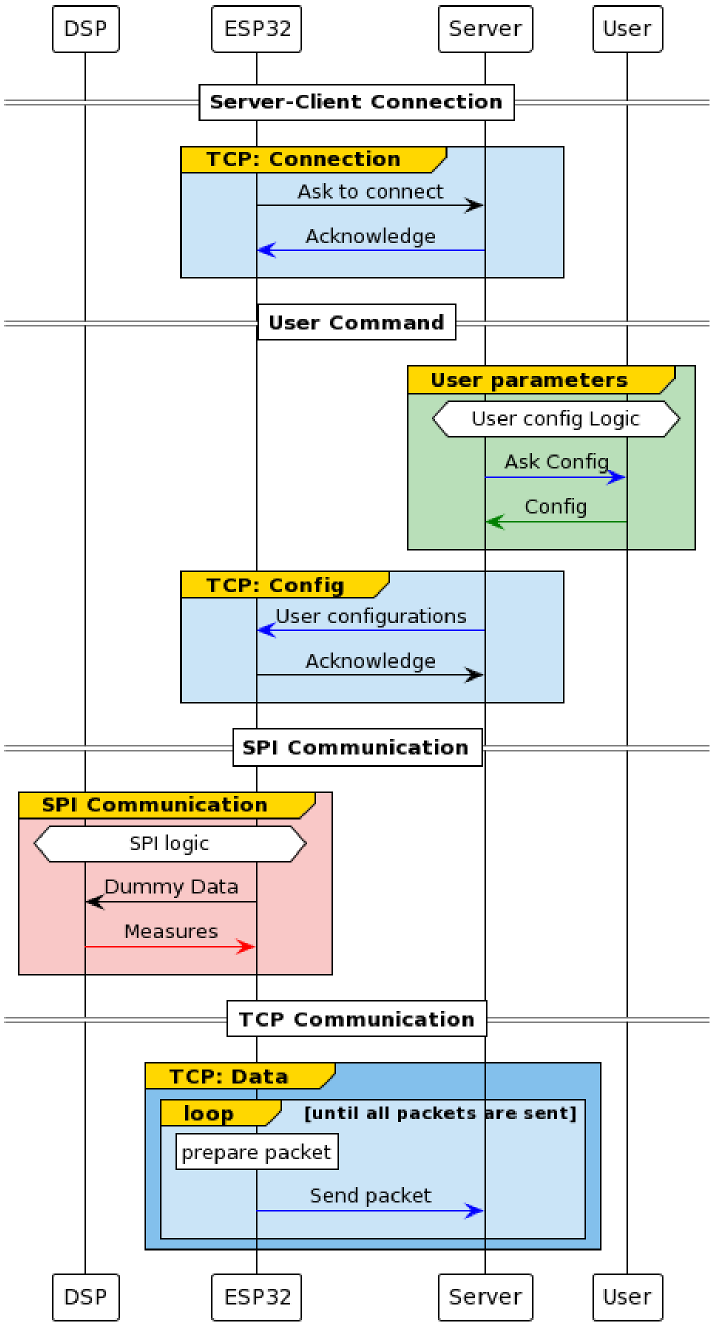


Figure 4.11: One cycle of the TCP communication architecture

be “4,1,g,113”. Each element represents one of the user configuration parameters (respectively N_channel, Trigg_channel, Trigg_type, and Trigg_value). If the packet is valid, the ESP32 acknowledges to the server application.

(c) TCP: data
The ESP32 communicates captured data in a loop fashion. In each loop iteration, the ESP32 creates the TCP packet by reading a block of 1000 elements from the ring buffer, the first iteration starts at the index obtained from 4.5. The ESP32 then updates the index by 1000 every cycle. The ESP32 repeats this loop until it has read the entire ring buffer.

The figure 4.12 shows the flowchart describing the algorithm for the ESP32.

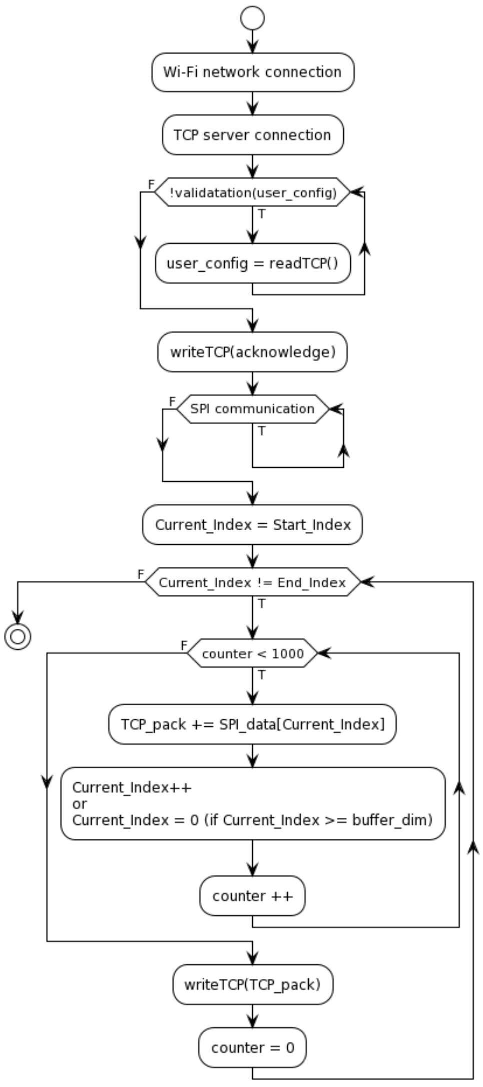


Figure 4.12: Flowchart for ESP32 TCP communication algorithm

4.4.5 Data Visualization

The server application performs the data visualization functionality of the virtual scope. After receiving the data, the server performs two operations:

1. *CSV file writing*
   When the data is received, the server saves it into a CSV file. The measurements are divided into columns, where each column corresponds to one channel. On a column, it is possible to see the entire capture history of the corresponding channel. The data is written to the CSV file by dividing the reconstructed array from the ESP32 TCP communication into a subset of lists containing one measurement for each channel. This means that each sub-list contains four elements and corresponds to a row in the CSV file (figure 4.13). Before the data is written to the CSV file, the file is initialized with a header to define the file formatting. The header defines the names of the columns in the CSV file, it is:"Header = [Channel 1, Channel 2, Channel 3, Channel 4]"

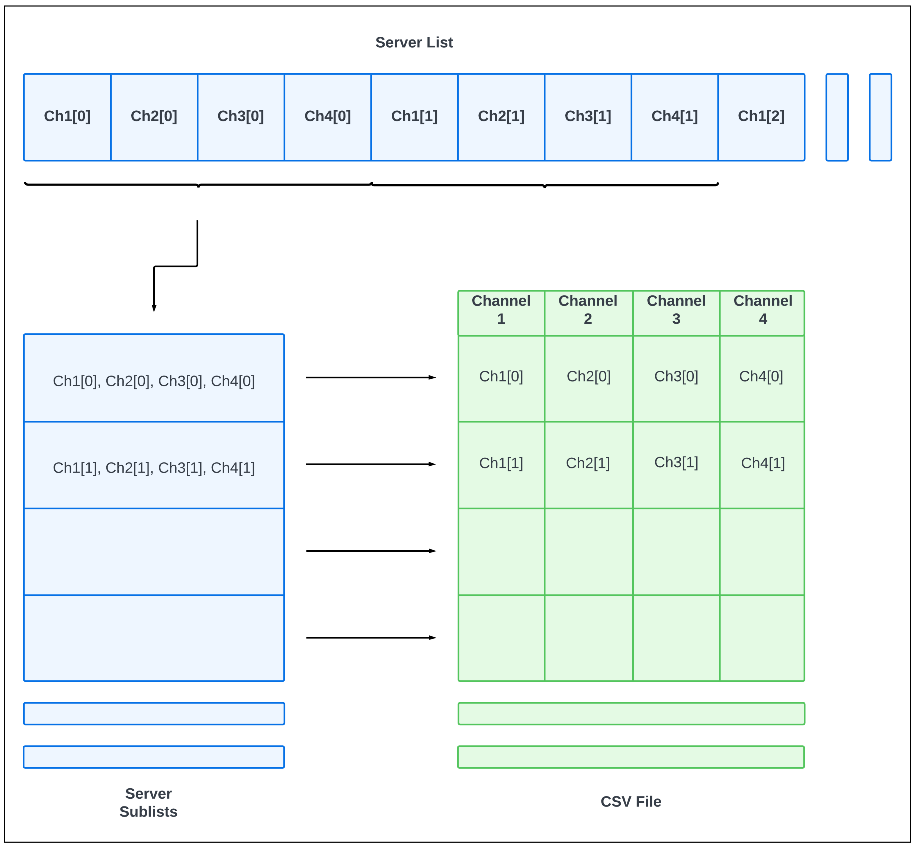


Figure 4.13: How data is saved in the CSV file

2. *Data plot*
   The data is plotted reading the saved elements from the CSV file. The number of plots depends on the user configuration parameter N_channel. The channels are plotted in sequential order, so if N_channel=2, the first two channels will be plotted. If N_channel=3, the first three channels will be plotted, and so on. The x-axis of the plot represents time in milliseconds (ms), and it is the same for all the channels. The y-axis represents the data for the corresponding channel, from the CSV file. To generate the x-axis, it is first calculated the sampling period, which is the inverse of the sampling rate. The sampling rate is 45 kHz, so the sampling period is $1/45000 = 22$ μs. Next, the sampling period is multiplied by the dimensions of the column of CSV file elements obtaining the number of elements of the x axis array. Then, it is created an equally spaced set of points comprised between 0 and the calculated value. This gives the total time range of the plot. The y-axis is simply one of the columns of the CSV file corresponding to the interested channel.

4.5 Integration of the Virtual Scope on the On-Board Charger

After designing the virtual scope, it has been implemented over the real on-board charger of BRUSA HyPower. The integration of the virtual scope required some modification on the hardware side and software side of the product from BRUSA HyPower. The hardware modification was necessary to connect the SPI peripherals of the microcontrollers, and the software update was necessary to add the SPI communication algorithm to the DSP.

4.5.1 On-board Charger

BRUSA HyPower proposes four different type of on-board chargers, the variants are:

- OBC754-1P: this is a one phase 400VDC nominal output voltage converter for the US market.
- OBC758-1P: this is a one phase 800VDC nominal output voltage converter for the US market.
- OBC764-3P: this is a three phase 400VDC nominal output voltage converter for the EU market.
- OBC768-3P: this is a three phase 800VDC nominal output voltage converter for the EU market.

The specific charger into which the designed virtual scope has to be integrated is a OBC754-1P. It is a 400V AC-DC power converter receiving in input a single-phase of 240V AC voltage at 60Hz and delivering an output of 400V DC voltage with a nominal DC current of 2A. Its ratings are of a 19kW converter with 94% efficiency. Providing at the output 400V and a maximum current of 72A.

The operating states of the OBC are reported in figure 4.14. When turned-on the device goes in StartUp, after which it transitions to Standby, where the device is in idle state. Once the charging request arrives, the on-board charger switches to Ready2Charge waiting for the input. If the input is provided, the power converter start its operation and provides the output. When a request to stop charging is received, the device goes in StopCharging and then in Standby. When the device is turned-off, the state transitions to ShutDown and the system shuts down. The SafeState is entered when a fault occurs, either communication errors with the main control unit board or some safety threshold which has been triggered.

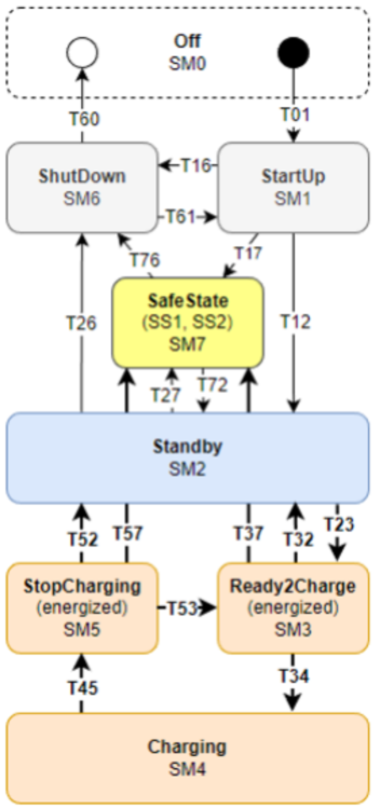


Figure 4.14: OBC state machine

4.5.2 Hardware Modifications

Some modifications of the hardware of the designed virtual scope are required with the aim to connect the SPI pins selected from the DSP to the ESP32. The choice of the pins on the DSP is subjected to the availability of the GPIOs on the microcontroller which is already almost fully employed. The consulted hardware schematics is reported in C.1.

The chosen pins are the ones from the second available SPI interface of the DSP, the SPIB:

- MOSI -> GPIO 30 -> Pin 98
- MISO -> GPIO 31 -> Pin 99
- CLK -> GPIO 58 -> Pin 67
- SS -> GPIO 59 -> Pin 92

Once the pins are chosen, it was necessary to connect them to the SPI interface of the ESP32. Each pin was connected through a resistance either to ground or to supply. The resistances have been de-soldered and the wires have been connected to the ESP32 SPI interface pins. The figure 4.15 shows the modifications to the OBC.

4.5.3 Software Modifications

The software on the DSP of the OBC required a firmware update to enable the virtual scope communication capability. The DSP code has been implemented by adding two custom files:

- VirtualScope.h
- VirtualScope.c

The files are respectively the header and the source code files containing the used functions for the virtual scope capability integration. The defined functions are:

- **SpiInit**
  This function initializes the SPIB peripheral.
- **GpioSpiInit**
  This function initializes the SPIB peripheral GPIO pins to behave as SPI pins.
- **TransmissionSPIblocking**
  This function is employed to add one element to the SPI transmission FIFO. The insertion algorithm requires that there must be at least one available slot in the FIFO to occur.

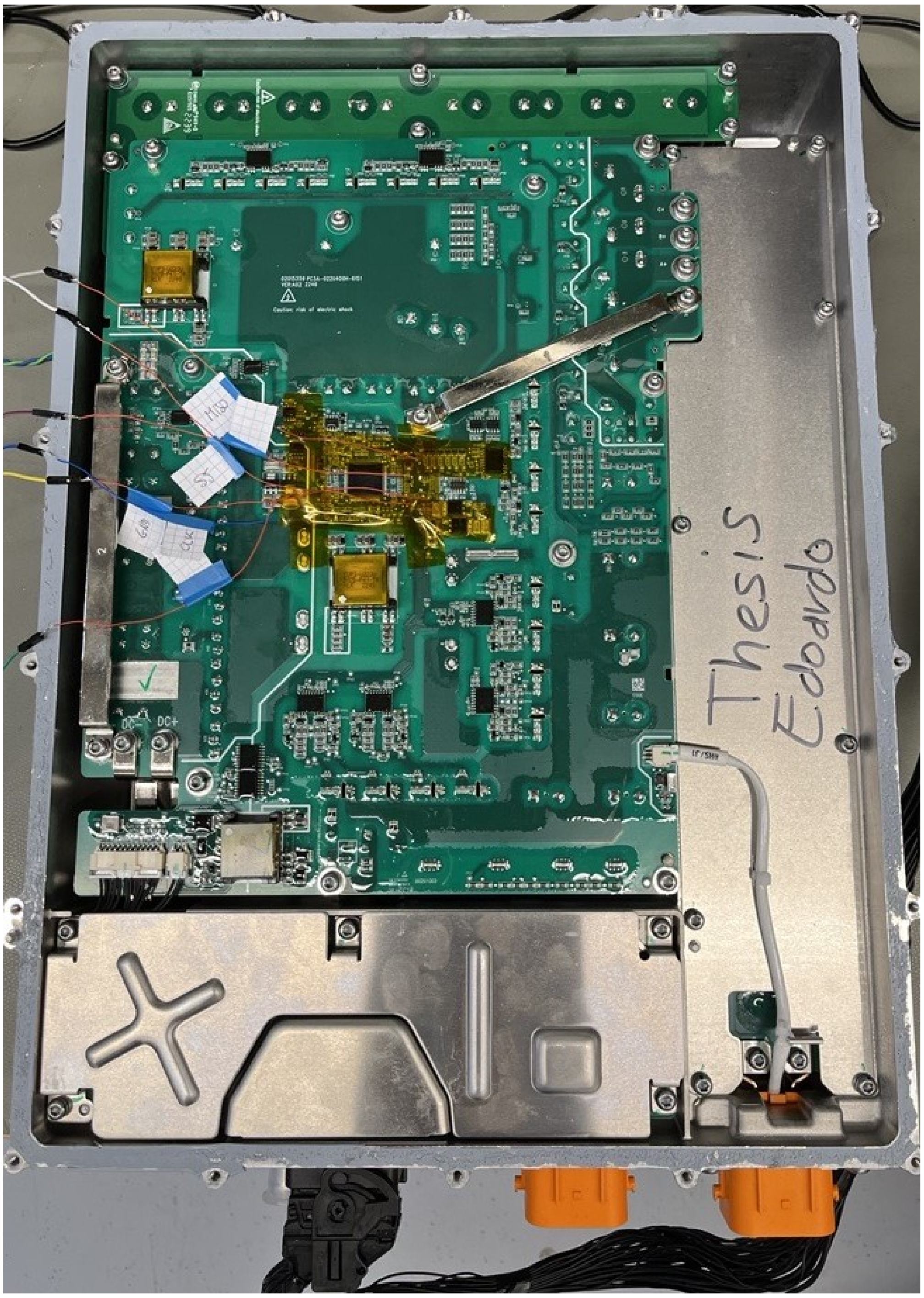


Figure 4.15: BRUSA HyPower's modified OBC7

- **VirtualScope**
  This function contains the SPI communication protocol described in 4.4.3. It implements the virtual scope in the DSP code.

All these functions are programmed in low level language by controlling each bit of the SPI interface and GPIO registers accordingly. Register level programming allows to have a highly optimized code in terms of computational speed because no intermediate function is used. The whole project of the DSP of the OBC is programmed with this approach, therefore also the virtual scope code has been integrated following this idea.

Chapter 5

Experimental Testing and Results

This chapter presents the tests performed to validate the performances of the design of the virtual scope. Initially, the tests have been performed on a testbench setup (reference to 4.2.3) with the intent to verify the correct behaviour of the algorithms of the virtual scope. The second part of the tests has been carried out using the on-board charger to validate the behaviour of the virtual scope on a real-world application scenario.

5.1 Testbench Tests

The testing setup comprises:

- *LAUNCHXL-F280049C*
  It emulates the ADC sampling of the DSP embedded into the OBC. For this purpose, a CPU timer interrupt is triggered at the same rate of the actual ADC (1/45kHz=22.22us). Every time the related ISR (interrupt service routine) is called, the LAUNCHXL updates four values. This data will be transmitted by the SPI interface, representing the four channels to acquire. For testing purposes, a sawtooth wave is generated for each channel. Since there is no need to asses the ability of the microcontroller to generate complicated waveforms in these tests, a very simple wave is sufficient. The sawtooth values are comprised between 0 and 1000. The sawtooth generation consists in incrementing the channel value cyclically by 1. Once the peak value is reached, it is reset to 0, creating the sawtooth pattern.

- *ESP32-S3*
  It represents the virtual scope.

- *User PC*
  The user PC is used as Wi-Fi hotspot to generate the local network which the ESP32 and the python server application are connected to. Moreover, the user PC is used to power the two microcontroller boards through USB, to run the python server application and as input terminal for the user.

- *Python Server Application*
  It is the server application of the virtual scope.

- *PicoScope Logic Analyzer*
  It is a logic analyzer used to test the SPI communication. It is employed to measure the real speed of the interfaces and to check the accuracy of the data over the SPI Bus.

- *Wireshark*
  It is a network traffic analyzer tool exploited to test the TCP communication. It is used to verify that all the packets contain the correct payload and to estimate the communication time.

The test performed regards each part of the communication architecture: SPI communication, TCP communication, and both together.

5.1.1 SPI Communication Tests

The test to be performed are:

1. *Data accuracy test*
   The data received from the DSP must be consistent with the data that was sent. The goal of this test is to ensure that the ESP32 receives the same values that the LAUNCHXL is transmitting via SPI. In this configuration, the LAUNCHXL generates a sawtooth waveform for each channel, with each waveform being offset by one from the others. These elements are then transmitted following the algorithm detailed in 4.4.3. The communication cycles follow the pattern outlined below:

   $$[1,2,3,4]\ [2,3,4,5]\ [3,4,5,6]\ [4,5,6,7]\ \ldots$$

   The expected outcome is to receive the same sequence of numbers. A logic analyzer is used to read the data transmitted over the MISO pin of the ESP32. Subsequently, the data are saved to simplify the analysis into a CSV file, where each column is dedicated to one channel.

2. *Bandwidth test*
   The goal of this test is to verify the SPI communication speed and to confirm that it is consistent with the design specifications. To conduct this test, a GPIO pin is toggled simultaneously to the communication algorithm. A logic analyzer is then used to measure the duration of the resulting signal. The analyzed parameters are:

   - **Clock speed of the SPI**
     It should be comparable with the value set in the configurations of the interface (2.88MHz). It is evaluated by monitoring the CLK pin of the SPI interface.
   - **Update time of the FIFO of the DSP**
     It represents the computational time elapsed to fill the FIFO of the DSP with the four elements to send. It should be greater than one clock but lower than two clock cycles. Otherwise, the FIFO would be filled too fast compared to the its emptying speed, which would result in some data losses. The measurement is carried out by monitoring an additional dedicated GPIO pin on the LAUNCHXL, ensuring it toggles as expected.
   - **Reading time of the ESP32**
     This is the time the ESP32 takes to read the data from the FIFO, essentially indicating the computational time required by the ESP32 to empty the DSP's FIFO. This value should not be lower than the update time for the FIFO and comparable with the SPI clock to keep the communication at the desired speed. It is measured by monitoring an additional dedicated GPIO pin on the ESP32, ensuring it toggles as expected.

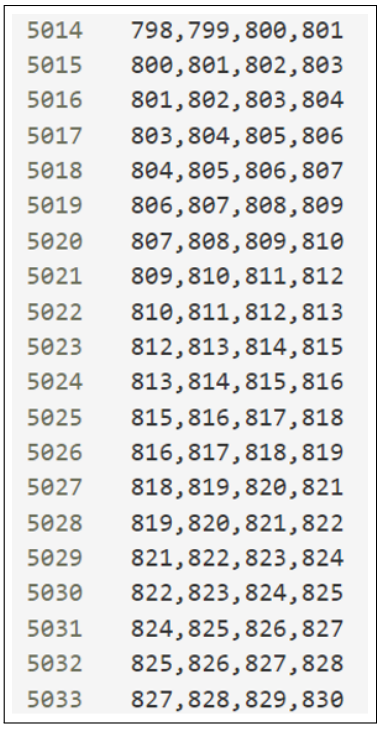
   The desired outcome is to observe a frequency of 2.88MHz for the SPI clock, and ensuring that the update and read times are approximately in the same range.

5.1.2 SPI Communication Tests Results

1. *Data accuracy test*
   The retrieved data are not as the expected ones: almost 1 element every 2 is skipped. In the figure 5.1 a piece of the CSV file where the retrieved data is stored is presented.

2. *Bandwidth test*
   The retrieved values are show in table 5.1.

   The SPI clock is opearing at the correct speed, but the ESP32 code is reading data at a slower pace than the expected interface bitrate. The FIFO is



```
5014   798,799,800,801
5015   800,801,802,803
5016   801,802,803,804
5017   803,804,805,806
5018   804,805,806,807
5019   806,807,808,809
5020   807,808,809,810
5021   809,810,811,812
5022   810,811,812,813
5023   812,813,814,815
5024   813,814,815,816
5025   815,816,817,818
5026   816,817,818,819
5027   818,819,820,821
5028   819,820,821,822
5029   821,822,823,824
5030   822,823,824,825
5031   824,825,826,827
5032   825,826,827,828
5033   827,828,829,830
```

Figure 5.1: Data accuracy test results for 2.88Mbps bit rate

SPI clock	2.9 MHz
DSP FIFO update time for 4 channels	5.7 μs
ESP SPI reading time for 1 channel	7.9 μs

Table 5.1: Bandwidth test results for 2.88Mbps bit rate

updating more quickly than the ESP32 can read from the SPI bus. In figure 5.2, the red waveform shows the time required to add four new elements to the FIFO, it is noticeable that every two occurrences there is a longer gap. This gap precisely matches twice the duration of the DSP CPU timer ISR period, indicating that while the LAUNCHXL updates the measured data, it is not being inserted into the FIFO. As a result, one element per channel is lost every two communication cycles due to the computational overhead of the reading algorithm of the ESP32.

The actual SPI bit rate is evaluated in equation 5.1.

$$\frac{ESPreadingtime}{16\text{bits}} = \frac{7.9\mu\text{s}}{16bits} = 2.025\text{Mbps} \tag{5.1}$$

The deviation from the actual bit rate is approximately 40% when compared

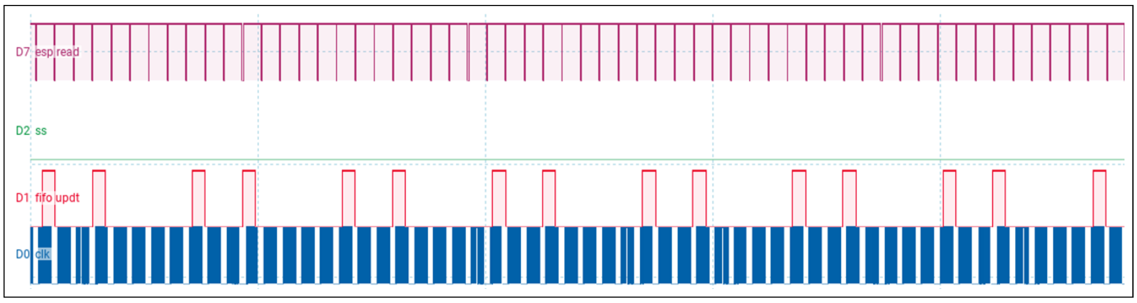


Figure 5.2: Time diagram of the SPI communication at 2.88Mbps

to the required bandwidth of 2.88 Mbps. To improve the result, one potential solution is to adjust the bit rate using a coefficient of 1.4. This modification would bring the actual SPI bit rate of the system closer to the desired ideal rate. After conducting several tests, it was determined that the optimal transmission rate is achieved with a coefficient of 1.5. Equation 5.2 illustrates the formula used for the adjusted bit rate.

$$Bitrate = 45\text{kHz} * 16\text{bits} * 4\text{ch} * 1.5 = 4.32\text{Mbps} \tag{5.2}$$

With this coefficient one element is skipped around every seven. The results are show in table 5.2.

SPI clock	4.35 MHz
DSP FIFO update time for 4 channels	5.7 μs
ESP SPI reading time for 1 channel	5.8 μs

Table 5.2: Bandwidth test results for 4.32Mbps bit rate

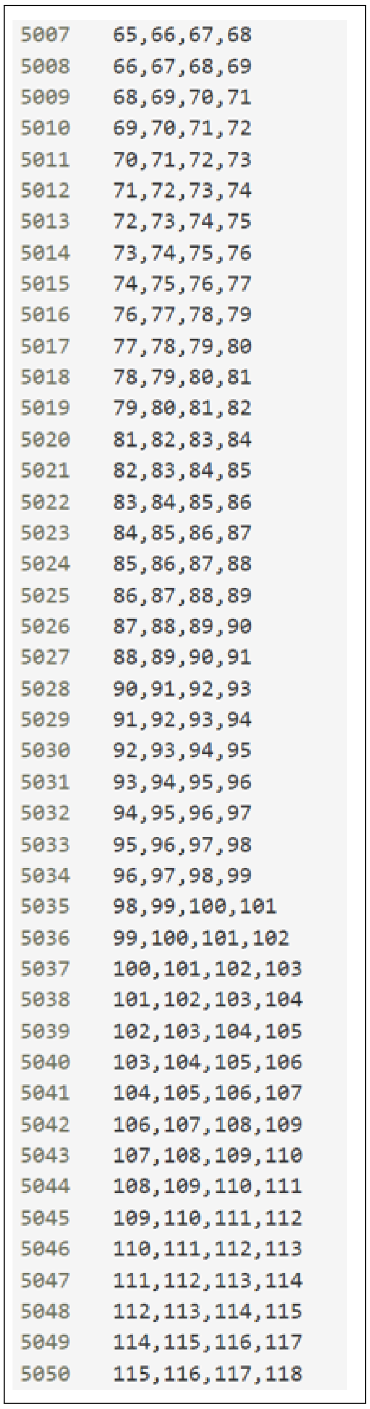
Figure 5.3 shows the stored data in the CSV file and the waveforms of the control signals, highlighting the described behavior.

5.1.3 TCP Communication Tests

The test to perform are:

- *Data accuracy test*
  The data sent from the ESP32 to the server application must be consistent. The goal of the test is to verify that the server receives the exact values that the ESP32 is sending via TCP communication.

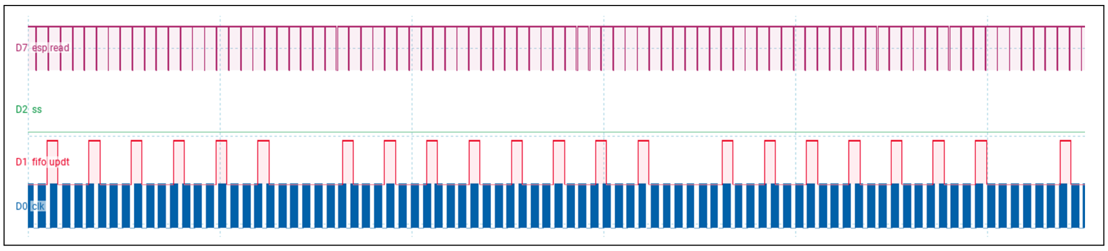
  To ensure that all the data is correctly received, the communication is tested by sending an array with known values from the ESP32 to the server. The



```
5007    65,66,67,68
5008    66,67,68,69
5009    68,69,70,71
5010    69,70,71,72
5011    70,71,72,73
5012    71,72,73,74
5013    72,73,74,75
5014    73,74,75,76
5015    74,75,76,77
5016    76,77,78,79
5017    77,78,79,80
5018    78,79,80,81
5019    79,80,81,82
5020    81,82,83,84
5021    82,83,84,85
5022    83,84,85,86
5023    84,85,86,87
5024    85,86,87,88
5025    86,87,88,89
5026    87,88,89,90
5027    88,89,90,91
5028    90,91,92,93
5029    91,92,93,94
5030    92,93,94,95
5031    93,94,95,96
5032    94,95,96,97
5033    95,96,97,98
5034    96,97,98,99
5035    98,99,100,101
5036    99,100,101,102
5037    100,101,102,103
5038    101,102,103,104
5039    102,103,104,105
5040    103,104,105,106
5041    104,105,106,107
5042    106,107,108,109
5043    107,108,109,110
5044    108,109,110,111
5045    109,110,111,112
5046    110,111,112,113
5047    111,112,113,114
5048    112,113,114,115
5049    114,115,116,117
5050    115,116,117,118
```

(a) CSV File

Figure 5.3: Data accuracy test of SPI communication result for 4.32Mbps bit rate



(b) Signal waveforms

Figure 5.3: Data accuracy test of SPI communication result for 4.32Mbps bit rate

server saves the data into a CSV file, and to assess the correct behaviour of the system, a comparison with the know values is done.

The test has been performed sending for each channel the value of a sawtooth waveform (like in the 5.1.1), so the expected result is a sequentially increasing sequence, incrementing by one for each column in the CSV file.

- *Communication speed test*
  The goal of the test is to understand the impact of the TCP protocol's overhead on the TCP communication architecture. Evaluating the time taken to transmit data is important as it determines the user's waiting time to access the data. While there are no strict constraints on this specification, it is desirable for this time to be relatively short. Two variables that influence communication speed are:

  – **TCP packet dimensions**
    This represents the packet size into which the ring buffer is divided when the ESP32 sends captured measurements to the server application. The test conducted for this variable involves measuring an entire communication period while varying the packet size. The objective is to determine the value that results in the fastest communication.

    The maximum dimension for a TCP packet is 65535 bytes, as per the protocol standard. The size of a string can be calculated using equation 5.3.

$$\begin{aligned} dimension_{\text{string}} &= dimension_{\text{char}} * N_{\text{characters}} + dimension_{\text{terminator}} \\ &= 1\text{byte} * N_{\text{characters}} + 1\text{byte} \end{aligned} \tag{5.3}$$

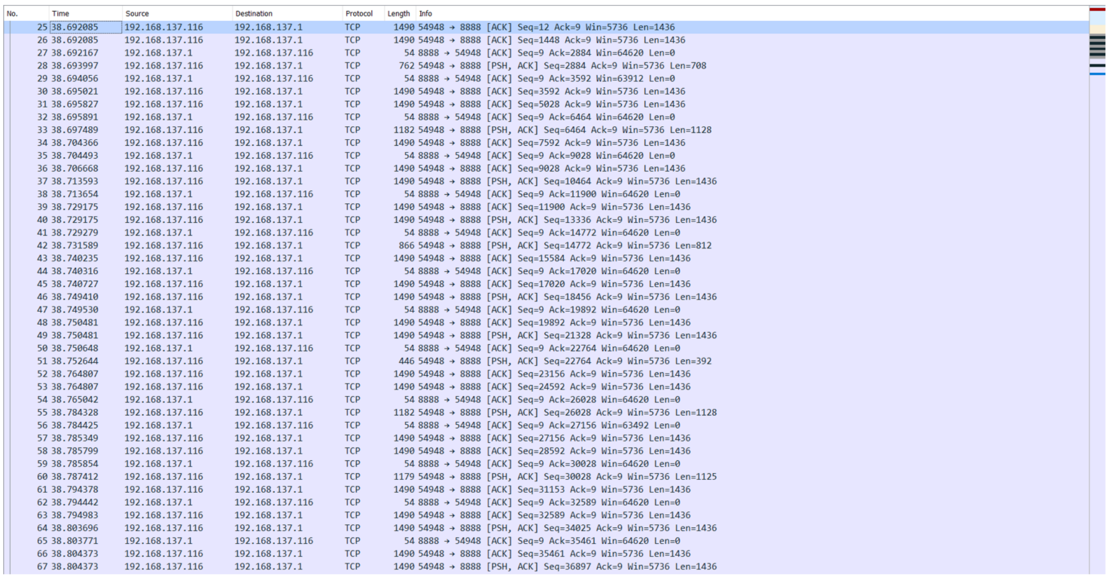
The equation (5.4) presents the maximum dimension for a TCP packet considering $N_{\text{characters}} = 4$.

$$\begin{aligned} dimension_{\text{pack}}^{\text{Max}} &= \frac{dimension_{\text{TCP}}^{\text{Max}}}{dimenison_{\text{terminator}}} + N_{\text{characters}} \\ &= \frac{65535\text{bytes}}{(1+4)\text{bytes}} = 13107\text{elements} \end{aligned} \tag{5.4}$$

Subsequently, the test is executed with a snapshot dimension equal to 45000 * 4 = 180,000 elements to be sent. This corresponds to a one-second sampling of the DSP for four channels.

– **Snapshot dimensions**
This is the number of samples retrieved for a snapshot, which corresponds to the duration of the captured snapshot. The test conducted for this variable involves measuring an entire communication period while varying the snapshot dimensions. The goal is to understand the entity of TCP overhead for different communication durations. For this test, the snapshot dimension was set to 1000.

The timing performance of the communication has been assessed reviewing the Wireshark network analysis report, an example is shown in picture 5.4.



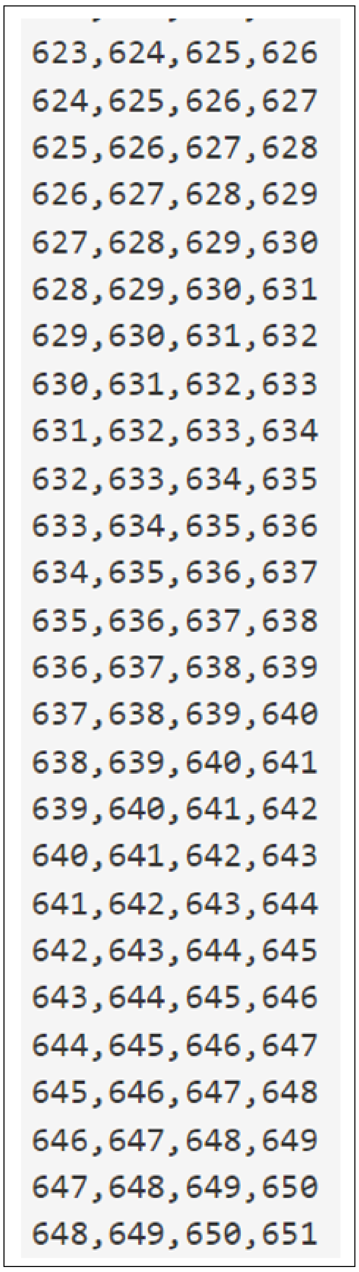
```
No.     Time        Source            Destination       Protocol Length Info
     25 38.692085   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=12 Ack=9 Win=5736 Len=1436
     26 38.692085   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=1448 Ack=9 Win=5736 Len=1436
     27 38.692167   192.168.137.1     192.168.137.116   TCP        54 8888 → 54948 [ACK] Seq=9 Ack=2884 Win=64620 Len=0
     28 38.693997   192.168.137.116   192.168.137.1     TCP       762 54948 → 8888 [PSH, ACK] Seq=2884 Ack=9 Win=5736 Len=708
     29 38.694056   192.168.137.1     192.168.137.116   TCP        54 8888 → 54948 [ACK] Seq=9 Ack=3592 Win=63912 Len=0
     30 38.695021   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=3592 Ack=9 Win=5736 Len=1436
     31 38.695827   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=5028 Ack=9 Win=5736 Len=1436
     32 38.695891   192.168.137.1     192.168.137.116   TCP        54 8888 → 54948 [ACK] Seq=9 Ack=6464 Win=64620 Len=0
     33 38.697489   192.168.137.116   192.168.137.1     TCP      1182 54948 → 8888 [PSH, ACK] Seq=6464 Ack=9 Win=5736 Len=1128
     34 38.704366   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=7592 Ack=9 Win=5736 Len=1436
     35 38.704493   192.168.137.1     192.168.137.116   TCP        54 8888 → 54948 [ACK] Seq=9 Ack=9028 Win=64620 Len=0
     36 38.706668   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=9028 Ack=9 Win=5736 Len=1436
     37 38.713593   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [PSH, ACK] Seq=10464 Ack=9 Win=5736 Len=1436
     38 38.713654   192.168.137.1     192.168.137.116   TCP        54 8888 → 54948 [ACK] Seq=9 Ack=11900 Win=64620 Len=0
     39 38.729175   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=11900 Ack=9 Win=5736 Len=1436
     40 38.729175   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [PSH, ACK] Seq=13336 Ack=9 Win=5736 Len=1436
     41 38.729279   192.168.137.1     192.168.137.116   TCP        54 8888 → 54948 [ACK] Seq=9 Ack=14772 Win=64620 Len=0
     42 38.731589   192.168.137.116   192.168.137.1     TCP       866 54948 → 8888 [PSH, ACK] Seq=14772 Ack=9 Win=5736 Len=812
     43 38.740235   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=15584 Ack=9 Win=5736 Len=1436
     44 38.740316   192.168.137.1     192.168.137.116   TCP        54 8888 → 54948 [ACK] Seq=9 Ack=17020 Win=64620 Len=0
     45 38.740727   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=17020 Ack=9 Win=5736 Len=1436
     46 38.749410   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [PSH, ACK] Seq=18456 Ack=9 Win=5736 Len=1436
     47 38.749530   192.168.137.1     192.168.137.116   TCP        54 8888 → 54948 [ACK] Seq=9 Ack=19892 Win=64620 Len=0
     48 38.750481   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=19892 Ack=9 Win=5736 Len=1436
     49 38.750481   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [PSH, ACK] Seq=21328 Ack=9 Win=5736 Len=1436
     50 38.750648   192.168.137.1     192.168.137.116   TCP        54 8888 → 54948 [ACK] Seq=9 Ack=22764 Win=64620 Len=0
     51 38.752644   192.168.137.116   192.168.137.1     TCP       446 54948 → 8888 [PSH, ACK] Seq=22764 Ack=9 Win=5736 Len=392
     52 38.764807   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=23156 Ack=9 Win=5736 Len=1436
     53 38.764807   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=24592 Ack=9 Win=5736 Len=1436
     54 38.765042   192.168.137.1     192.168.137.116   TCP        54 8888 → 54948 [ACK] Seq=9 Ack=26028 Win=64620 Len=0
     55 38.784328   192.168.137.116   192.168.137.1     TCP      1182 54948 → 8888 [PSH, ACK] Seq=26028 Ack=9 Win=5736 Len=1128
     56 38.784425   192.168.137.1     192.168.137.116   TCP        54 8888 → 54948 [ACK] Seq=9 Ack=27156 Win=63492 Len=0
     57 38.785349   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=27156 Ack=9 Win=5736 Len=1436
     58 38.785799   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=28592 Ack=9 Win=5736 Len=1436
     59 38.785854   192.168.137.1     192.168.137.116   TCP        54 8888 → 54948 [ACK] Seq=9 Ack=30028 Win=64620 Len=0
     60 38.787412   192.168.137.116   192.168.137.1     TCP      1179 54948 → 8888 [PSH, ACK] Seq=30028 Ack=9 Win=5736 Len=1125
     61 38.794378   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=31153 Ack=9 Win=5736 Len=1436
     62 38.794442   192.168.137.1     192.168.137.116   TCP        54 8888 → 54948 [ACK] Seq=9 Ack=32589 Win=64620 Len=0
     63 38.794983   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=32589 Ack=9 Win=5736 Len=1436
     64 38.803696   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [PSH, ACK] Seq=34025 Ack=9 Win=5736 Len=1436
     65 38.803771   192.168.137.1     192.168.137.116   TCP        54 8888 → 54948 [ACK] Seq=9 Ack=35461 Win=64620 Len=0
     66 38.804373   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [ACK] Seq=35461 Ack=9 Win=5736 Len=1436
     67 38.804373   192.168.137.116   192.168.137.1     TCP      1490 54948 → 8888 [PSH, ACK] Seq=36897 Ack=9 Win=5736 Len=1436
```

Figure 5.4: Wireshark report example

5.1.4 TCP Communication Tests Results

1. *Data accuracy test*
The data is received correctly, the figure 5.5 shows a section of the CSV file

with the results.



```
623,624,625,626
624,625,626,627
625,626,627,628
626,627,628,629
627,628,629,630
628,629,630,631
629,630,631,632
630,631,632,633
631,632,633,634
632,633,634,635
633,634,635,636
634,635,636,637
635,636,637,638
636,637,638,639
637,638,639,640
638,639,640,641
639,640,641,642
640,641,642,643
641,642,643,644
642,643,644,645
643,644,645,646
644,645,646,647
645,646,647,648
646,647,648,649
647,648,649,650
648,649,650,651
```

Figure 5.5: Data accuracy test for TCP communication

2. *Communication speed test*
   The best result in terms of overhead has been obtained for a packet dimension of 1000 elements, as shown in table 5.3.

TCP Pack dimension	**Time to send all the samples**
100	3 s
1000	2.4 s
2500	3 s
10000	3.5 s
13107	3.5 s

Table 5.3: TCP pack dimension test results

Nevertheless, the general overhead of the protocol over the architecture is around the 250%, being quite high. The table 5.4 shows the results of the overhead over different communication durations.

Number of samples (16 bits)	**Time required to receive all samples**	**Ideal time required to sample at 45KHz**
40000	550 ms	220 ms
180000	2.5 s	1 s
1800000	25 s	10 s
4000000	50 s	22 s

Table 5.4: TCP capture dimension test results

5.1.5 Complete Communication Tests

According to the results of the previous test, it is necessary to check that the whole system behaves like expected. The goal of this test is to plot and visualize the sawtooth waveform generated by the launchpad on the user interface of the server application. The expected result is the combination of the results of the previous tests. Therefore, the expected data precision is the one from the SPI communication and the expected speed of communication is mostly related to the TCP overhead.

Based on the outcomes of the previous tests, it is necessary to verify that the entire system operates as expected. The objective of this test is to plot and visualize the sawtooth waveforms generated by the LAUNCHXL on the user interface of the server application. The expected result is a combination of the accuracy observed in the SPI communication results and the communication speed, which is primarily influenced by the TCP overhead. The user configurations used for this test are:

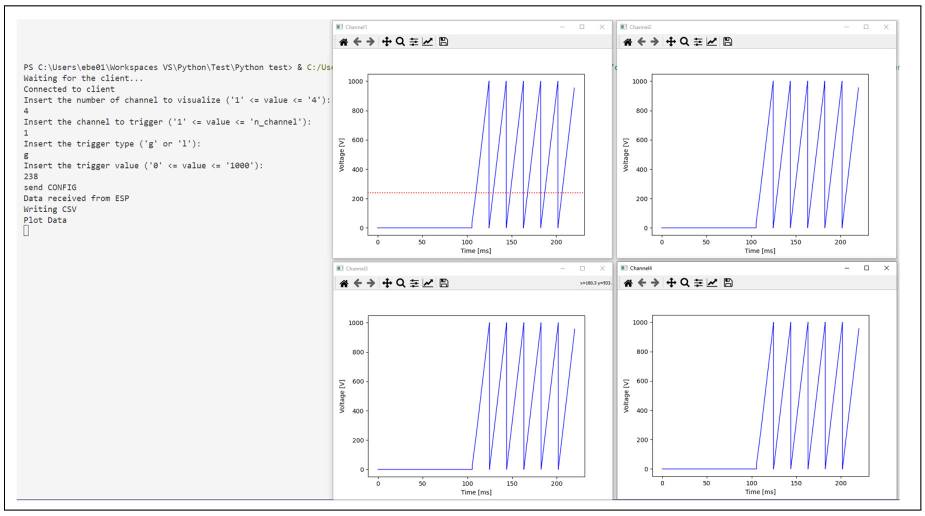
- N_channel = 4
- Trigg_channel = 1

- Trigg_type = greater
- Trigg_value = 238

The ring buffer's dimension has been configured to 40000 elements (10000 per channel) for simplicity. The capture period for this buffer dimension is evaluated in equation 5.5.

$$\begin{aligned} T_{\text{capture}} &= \frac{dimension_{\text{ring buffer}}}{4} * T^{\text{ADC}}_{\text{sampling}} \\ &= \frac{40000\text{elements}}{4} * \frac{1\text{s}}{45000\text{elements}} = 220\text{ms} \end{aligned} \tag{5.5}$$

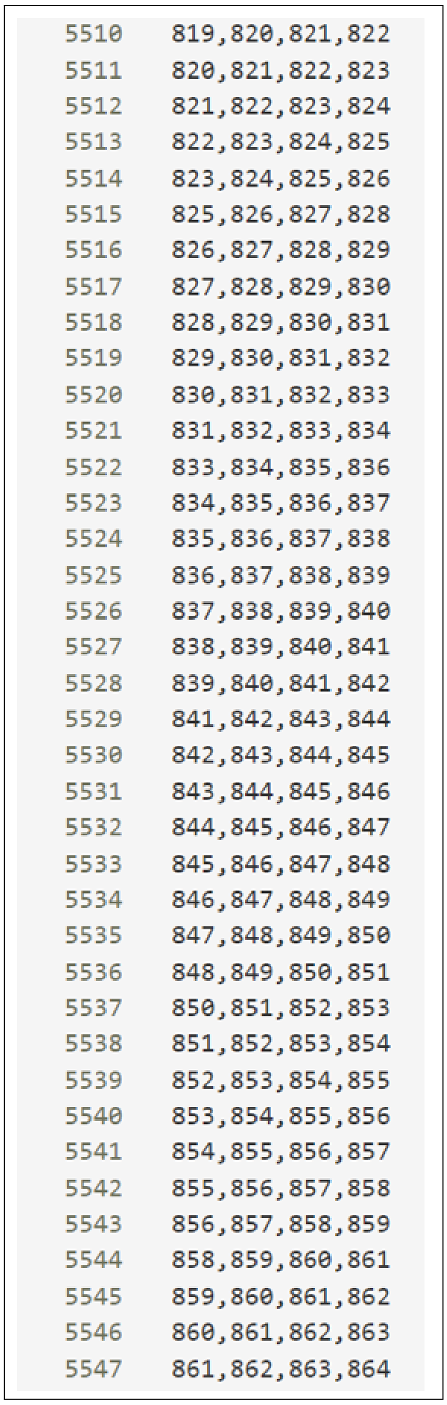
A fault will be detected within one period of the sawtooth waveform since the trigger value falls between the minimum (0) and maximum (1000) values of the wave. Consequently, the expected plot will display approximately half of the sawtooth waves and half of 0s, which represents the initialization value of the ring buffer. This is because there will not be any real data history before the fault trigger, as the trigger event occurs within one cycle of the DSP sampling. Figure 5.6 shows the result of the test.


```
PS C:\Users\ebe01\Workspaces VS\Python\Test\Python test> & C:/Use
Waiting for the client...
Connected to client
Insert the number of channel to visualize ('1' <= value <= '4'):
4
Insert the channel to trigger ('1' <= value <= 'n_channel'):
1
Insert the trigger type ('g' or 'l'):
g
Insert the trigger value ('0' <= value <= '1000'):
238
send CONFIG
Data received from ESP
Writing CSV
Plot Data
```

(a) CSV File

Figure 5.6: Complete communication test on testbench

The communication work as expected. The received data reflects the pattern of one element skipped every seven, and due to the TCP overhead, the user receives the data in approximately 500ms.



```
819,820,821,822
820,821,822,823
821,822,823,824
822,823,824,825
823,824,825,826
825,826,827,828
826,827,828,829
827,828,829,830
828,829,830,831
829,830,831,832
830,831,832,833
831,832,833,834
833,834,835,836
834,835,836,837
835,836,837,838
836,837,838,839
837,838,839,840
838,839,840,841
839,840,841,842
841,842,843,844
842,843,844,845
843,844,845,846
844,845,846,847
845,846,847,848
846,847,848,849
847,848,849,850
848,849,850,851
850,851,852,853
851,852,853,854
852,853,854,855
853,854,855,856
854,855,856,857
855,856,857,858
856,857,858,859
858,859,860,861
859,860,861,862
860,861,862,863
861,862,863,864
```

(b) Signal waveforms

Figure 5.6: Complete communication test on testbench

5.2 On-Board Charger Tests

The objective of this testing phase is to validate the functionality of the system over a real device in real working conditions. The tests are conducted on one of the on-board chargers from BRUSA HyPower. The on-board charger, described in **??** is tested in two different states:

- Standby: in this state the device is in the idle state, the converter is not switching and the current and voltage in output are almost null.
- Charging: in this state the device is active and provide at the output the desired output power.

The testing setup includes the virtual scope connected to the DSP of the OBC, which is also connected to a power grid simulator and a load. The power grid simulator generates a three-phase 230V AC voltage at 50Hz, while the load is a voltage generator set to 400V. Figure 5.7 shows a picture of the setup.

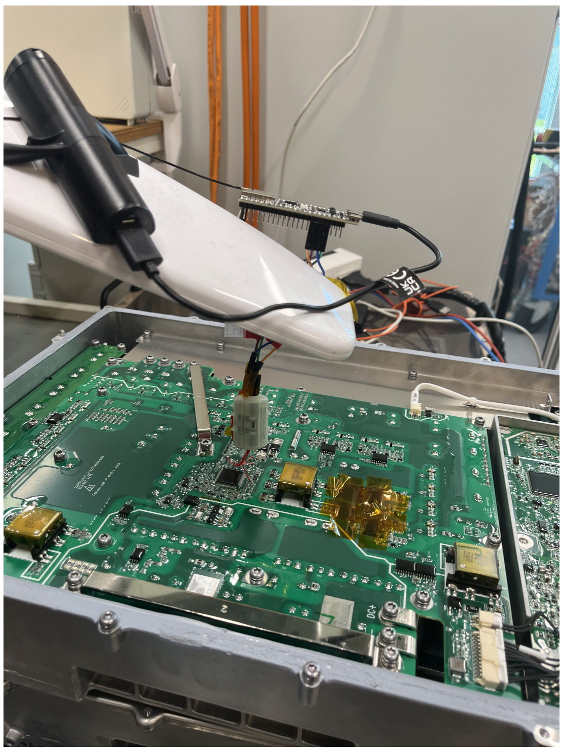


Figure 5.7: Test setup with on-board charger

The performed tests consists in monitoring the output DC voltage and current during the standby and charging phase with a load. The conducted tests are:

1. *Standby state test*
   In this test the on-board charger is in standby phase. The output of the converter is tracked while the load is modified from 0V to 400V. The goal of the test is to verify that the virtual scope is able to accurately track the changes in the output voltage corresponding to a load variation. This behaviour would assess the proper operation of the communication architecture of the virtual scope. The expected behaviour consists in observing the output voltage of the converter transitioning from 0V to 400V with a ramp-like trend, while the current remains close to 0A.

2. *Charging state transition test*
   In this test, the on-board charger transitions from the standby phase to the charging phase. The procedure involves changing the load from 0V to 400V and subsequently initiating the charging process. This test serves to evaluate the correct functioning of the communication architecture within the virtual scope when the power converter undergoes switching behavior. The expected behavior consists in observing in output a voltage ramp followed by a stable voltage of 400V, along with a constant output current of 2A after the system enters the charging state.

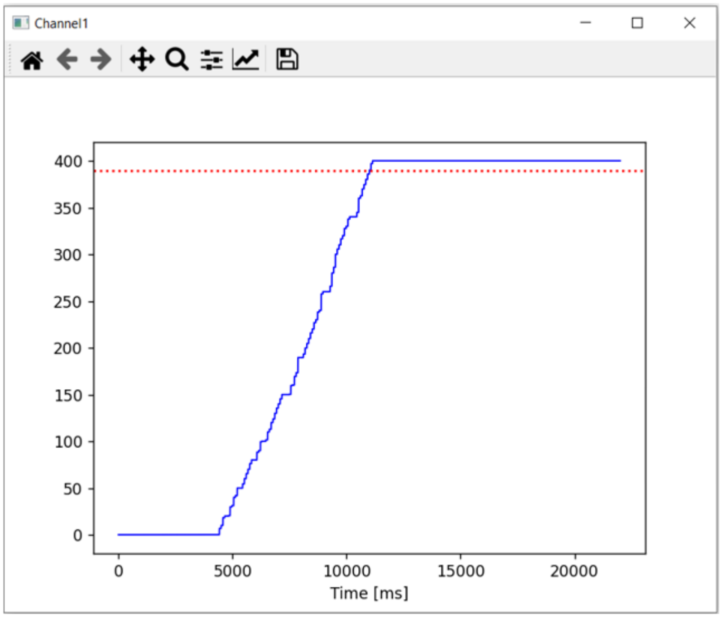
5.3 On-Board Charger Tests Tesults

1. *Standby state test*
   The test are successful, the result are shown in figure 5.8. It is observable that the voltage is properly measured as it increases from 0V to 400V, while the current exhibits oscillations around a value of approximately 350mA.

2. *Charging state transition test*
   The test results are not the expected ones. When the device transitions into the charging state, the observed behavior of the output is disrupted by noise which comes from the switching behavior of the converter and the wires connecting the SPI peripherals of the two microcontrollers. This noise impacts the SPI physical communication medium. Consequently, the retrieved data is compromised and cannot be properly acquired when the converter is in the charging state

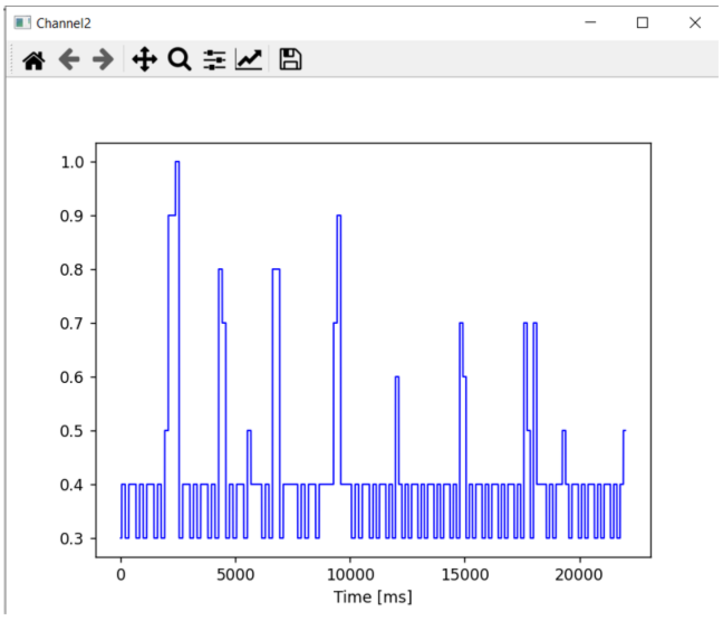
   Figure 5.9 shows the obtained results. In particular, figure 5.9c shows how the received data is corrupted by noise, because the expected value of the voltage is registered but mixed with noise.

To try to improve the results during the charging state transition test, some ferrite beads are added in series to the SPI wires. The effect of the ferrite beads is



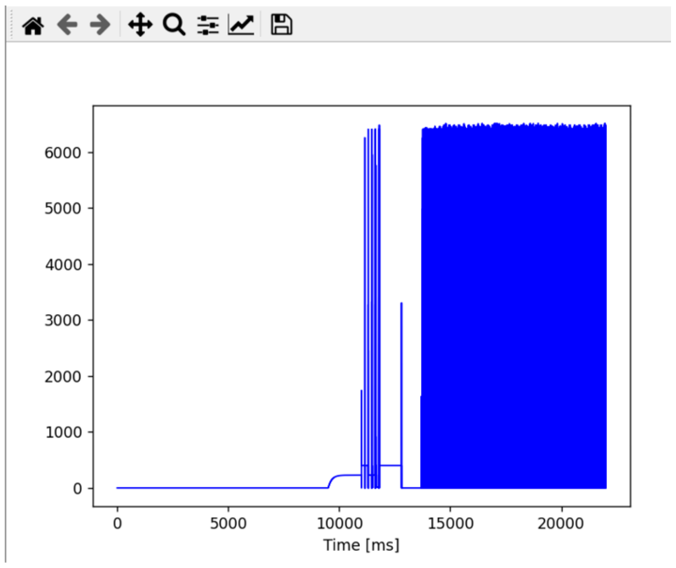
(a) DC output voltage

Figure 5.8: Standby state test results



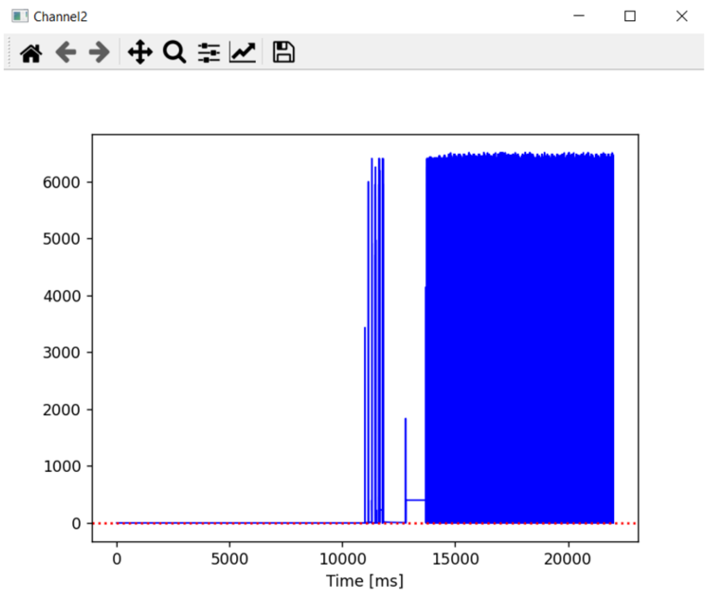
(b) DC output current

Figure 5.8: Standby state test results



(a) DC output voltage

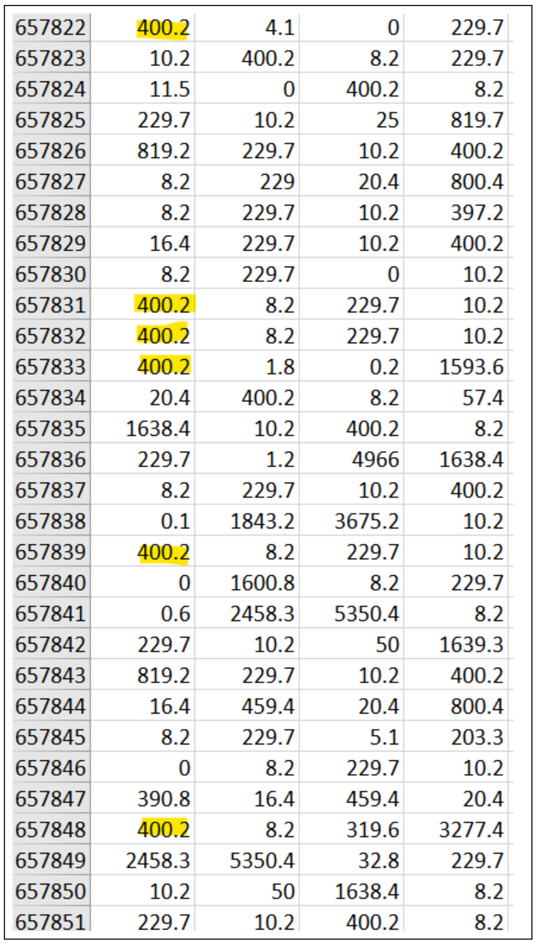
Figure 5.9: Charging state transition test results



(b) DC output current

Figure 5.9: Charging state transition test results

to reduce the effect of disturbances over some communication cables. They provide attenuation in certain narrow frequency bands. For these test three different ferrite



657822	400.2	4.1	0	229.7
657823	10.2	400.2	8.2	229.7
657824	11.5	0	400.2	8.2
657825	229.7	10.2	25	819.7
657826	819.2	229.7	10.2	400.2
657827	8.2	229	20.4	800.4
657828	8.2	229.7	10.2	397.2
657829	16.4	229.7	10.2	400.2
657830	8.2	229.7	0	10.2
657831	400.2	8.2	229.7	10.2
657832	400.2	8.2	229.7	10.2
657833	400.2	1.8	0.2	1593.6
657834	20.4	400.2	8.2	57.4
657835	1638.4	10.2	400.2	8.2
657836	229.7	1.2	4966	1638.4
657837	8.2	229.7	10.2	400.2
657838	0.1	1843.2	3675.2	10.2
657839	400.2	8.2	229.7	10.2
657840	0	1600.8	8.2	229.7
657841	0.6	2458.3	5350.4	8.2
657842	229.7	10.2	50	1639.3
657843	819.2	229.7	10.2	400.2
657844	16.4	459.4	20.4	800.4
657845	8.2	229.7	5.1	203.3
657846	0	8.2	229.7	10.2
657847	390.8	16.4	459.4	20.4
657848	400.2	8.2	319.6	3277.4
657849	2458.3	5350.4	32.8	229.7
657850	10.2	50	1638.4	8.2
657851	229.7	10.2	400.2	8.2

(c) CSV file

Figure 5.9: Charging state transition test results

beads are inserted in series to the device, each one of them is attenuating a different frequency range with the intent of finding the frequency range of the disturbance. Figure 5.10 shows the employed setup.

Figure 5.11 shows that this setup did not improve the results, the data is still noisy around the required voltage level. This means that the problem must be tackled differently.

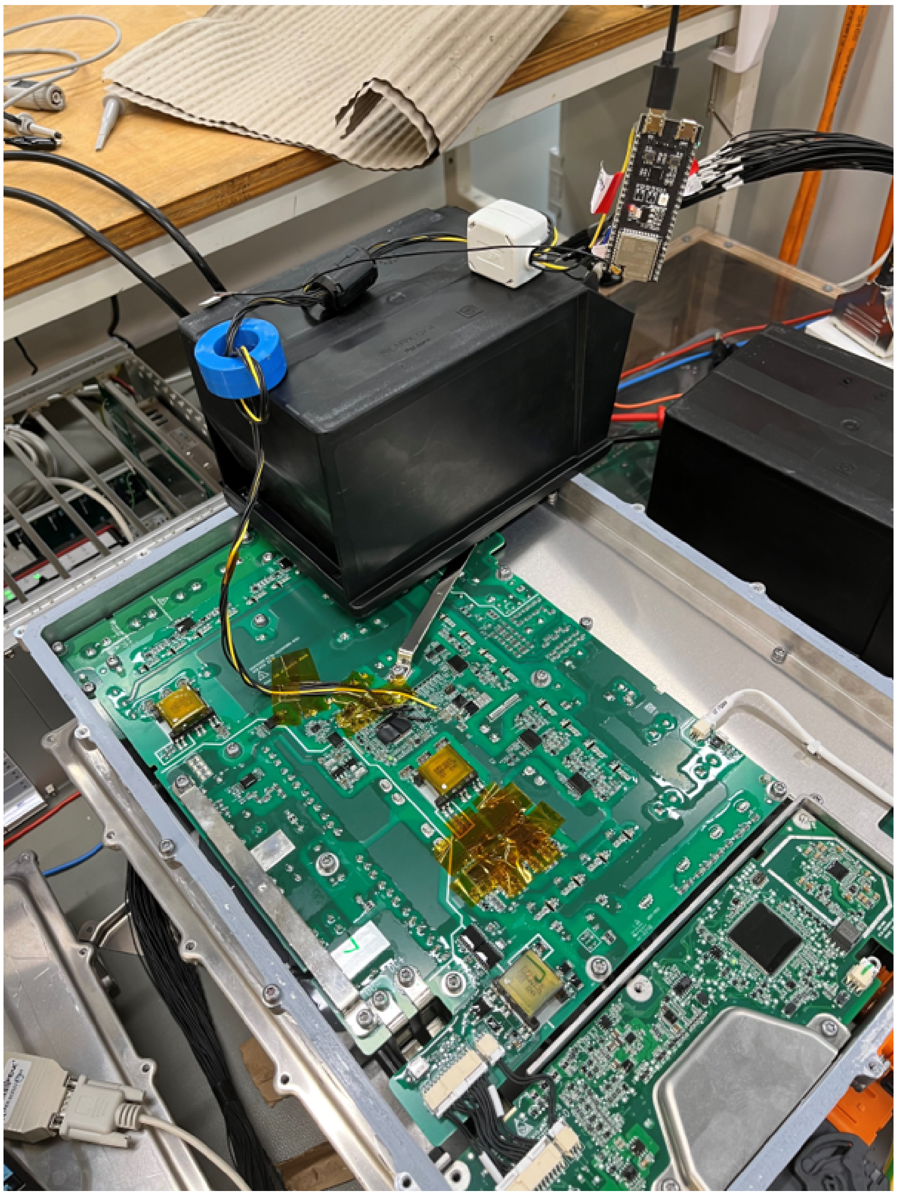
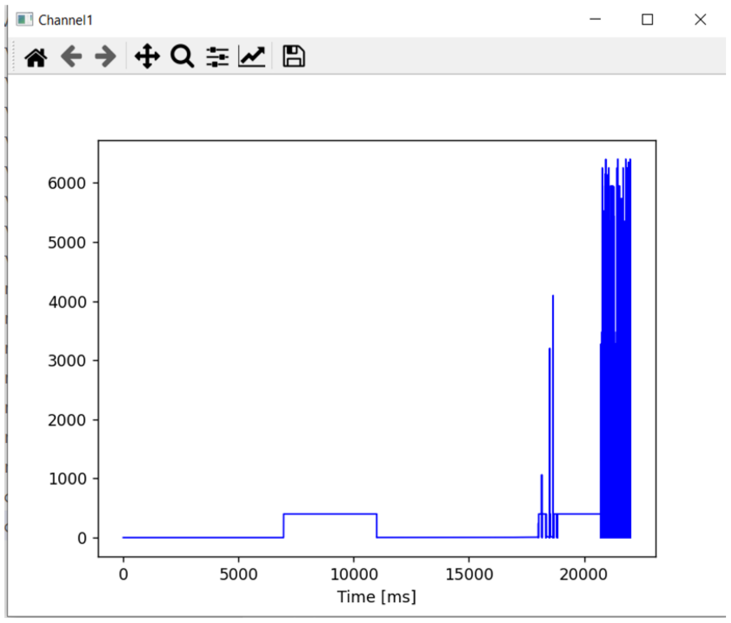
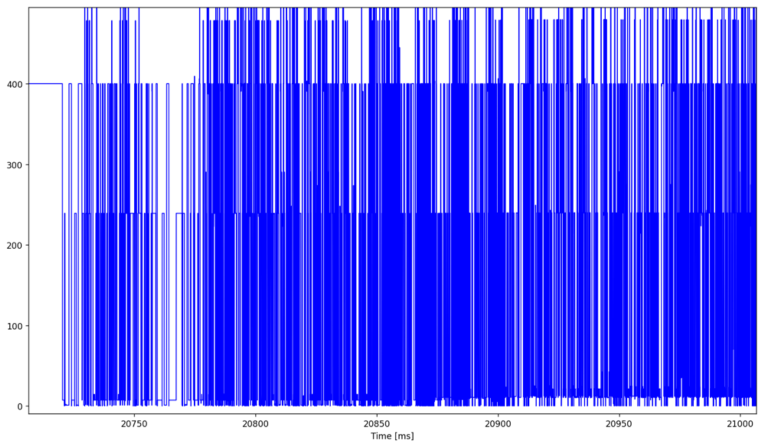


Figure 5.10: Test setup with on-board charger and ferrite beads



(a) DC output voltage

Figure 5.11: Charging state transition test results with ferrite beads



(b) DC output voltage zoom

Figure 5.11: Charging state transition test results with ferrite beads

Chapter 6

Conclusions and Future Developments

This chapter presents the conclusions of the thesis work, including interpretations of the test results and proposals for future developments of the project. The first part of the chapter describes how the results from the tests are interpreted, summarizing the main problems of the proposed system. The resolution of these issues poses the basis of a possible evolution of the presented architecture. The second part of the chapter is dedicated to conclusive remarks.

6.1 Interpretation of the Results

The results of the tests showed that the communication architecture of the virtual scope works as intended but noise susceptible. When the converter does not exhibit switching behavior and does not introduce disturbances, the communication architecture functions mostly as intended, despite some minor issues. The user is able to access the data captured from the DSP on an isolated environment with respect to the power converter. However, the absence of a robust noise-resistant design makes the device unsuitable for use in all operational states of the converter.

Summarizing, the problems of the system are:

- **Limited bandwidth**
  The results described in 5.1.2 showed how the ESP32 computational speed is affecting the final capturing algorithm, thus constraining the real bandwidth of the system.

- **TCP overhead**
  The results described in 5.1.4 showed the overhead imposed on the communication of the TCP protocol. Given the volume of data being transmitted, it

considerably slows down the communication. It extends the time needed for the user to visualize the data to more than the double of the acquisition time needed by the ESP32 to retrieve all the capture from the DSP.

- **Noise susceptibility**
  The most significant problem of the system is its susceptibility to disturbances over the on-board charger. The radiated noise in the measurements is caused by the switching behaviour of the power converter. Fast current transient causes voltage spike and ringing at the output due to parasitic inductance and capacity in the circuits. The spike and ringing noise will increase with higher current load. In this scenario, connecting the power converter DSP with the ESP32 through relatively long wires is a problem. Long wires introduce several issues over a SPI protocol because of possible propagation delays which de-synchronizes the master and the slave, reduced noise immunity due to long-distance and unbalanced signal paths, damaged transceivers due to large ground-potential differences, and data transmission errors due to unterminated data lines[36].

6.2 Potential Improvements and Future Research Directions

The system can be improved to address the issues presented in 6.1. Analyzing the problems:

- **Limited bandwidth**
  The bandwidth bottleneck is the computational speed of the ESP32 code. Currently, the program is written in high level language exploiting some libraries that Arduino provides. While high-level programming is generally efficient, the results of the tests indicate that for applications where the speed of the system is meaningful, it may not suffice. A potential solution could be to rewrite the code in a low-level language, allowing for direct manipulation of the registers of the device. Low-level programming is highly optimized but considerably more complex. To correctly write into the registers of a microcontroller, a deep understanding of the architecture of the device and precise documentation is essential. Furthermore, in the case of the ESP32, utilizing the Espriff programming environment is necessary. However, this environment lacks extensive documentation, making the process of transitioning to low-level programming a long and challenging research endeavor.

  Another strategy which should be adopter is to sample the data at a significantly higher frequency than the ADC bitrate used by the DSP. By subsequently applying a suitable post-processing filtering procedure, it would be possible to eliminate the noise introduced by the SPI interface. This approach could lead to a overall smoother data acquisition, improving the overall performance of the system.

- **TCP overhead**
  The TCP overhead is a known feature of the protocol, consequently it is something which cannot be eliminated. A possible solution to mitigate it would be to use a more efficient language for the server like C. This modification, combined with register-level programming for communication on the ESP32 side, could potentially yield improved results. Moreover, another solution could involve reducing the volume of data transmitted over TCP by implementing a proper data filtering on the ESP32 side during the capture process.

- **Noise susceptibility**
  The noise generated by the switching behaviour of the converter is not a trivial topic. This problem must be tackled in the design phase by modelling a proper communication connection for the two microcontrollers. Wires are not a suitable solution for SPI communication within the on-board charger, a better implementation to address this issue would be a modification of the PCB of the power board. A proper connector should be addedd, this would grant a shorter connection between the devices. Furthermore, a proper filter could be added to properly deal with interferences. Nevertheless, this work requires lot of research and design testing, which, for example, could be carried out in another thesis project.

There are numerous potential future developments for this project, for instance, the system can be significantly enhanced by implementing the ideas mentioned above. In addition, the system architecture could be reconfigured to provide direct access to the memory locations of the DSP within the on-board charger. This modification would enable users to select the signals they wish to measure without the need to modify the DSP code each time, leading to a more easy-to-use acquisition system.

A further evolution of the project could be a real-time Wi-Fi virtual scope. However, in this scenario, careful consideration would be necessary to designing the refresh rate of the device. This would require the use of register-level programming to ensure an optimized algorithm in terms of computational speed. Another consideration to take into account is the replacement of the TCP protocol with the UDP protocol. UDP is known for its speed despite offering less control over data transmission errors. This modification could enhance the device real-time capabilities while acknowledging potential trade-offs in data accuracy.

As a final objective this device could be standardized, implementing one or more communication protocols common to different power converter devices and become a market solution for measurements for diagnostic purposes in power electronic fields.

6.3 Conclusion Remarks

This project proposes a novel solution to address the problems of the measurements for diagnostic purposes in power electronic fields, offering an innovative approach based on the use of Wi-Fi as isolation mechanism. The problem of power converters, operating in diverse domains, is the abundance of noise and dangerous voltages and currents populating these domains. To deal with this matter, when monitoring the behaviour of the converter, it is necessary to employ instrumentation with robust isolative properties.

This project proposes the adoption of Wi-Fi as an isolation medium, which differentiate from traditional approaches, such as optic fibers. Optic fiber systems are often characterized by fragility and limited adaptability for on-field applications. On the other hand, Wi-Fi enables versatility and practicality. The Wi-Fi-based isolation architecture enables the use of simple microcontrollers as diagnostic devices, yielding practical tools for on-field applications for power electronics diagnostic purposes. Microcontrollers offer a variety of peripheral to use, like SPI communication protocol which grants high bandwidth and facilitates multichannel communication. Furthermore, microcontrollers comes at a relatively low price, improving the cost-efficiency balance of the device.

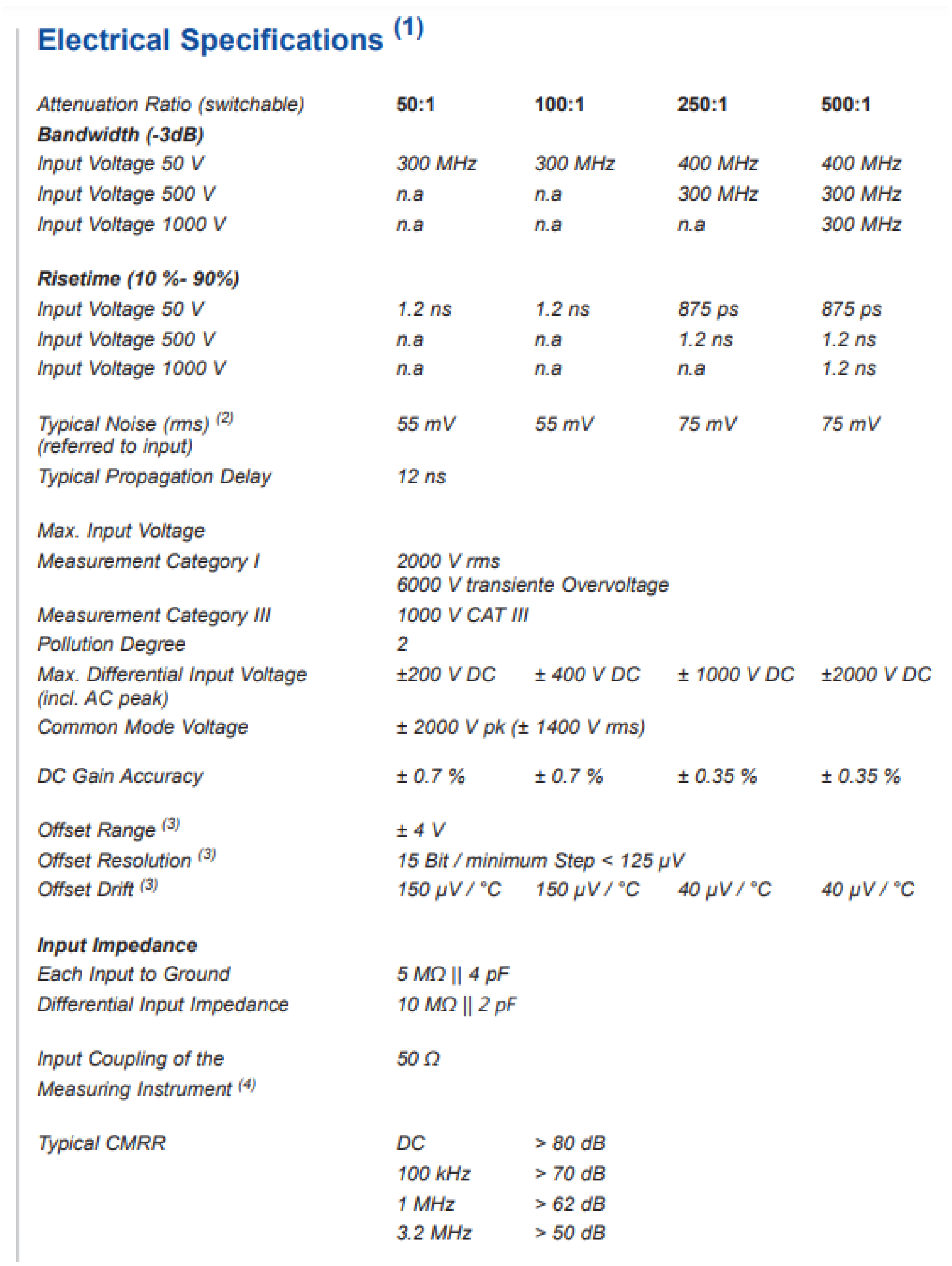
The proposed communication architecture is from an algorithmic point of view is properly functional. Nonetheless, this project faced several problems, including code computational speed issues, and communication disturbances. Looking ahead, future developments of the project will involve addressing these challenges. To improve the device bandwidth is necessary to optimize the code computational speed through register-level programming. To properly mitigate noise interference in the communication, the implementation of a another connection mean between the DSP and the ESP32 is needed. By properly including a connector for the SPI over the PCB of the on-board charger which would substitute the present wiring connection, it would be possible to enhance the rejection of disturbances over the SPI bus. To reduce the TCP protocol overhead, the optimization of the server application is mandatory, this would lead to a faster response of the user interface for visualization purposes. Lastly, to improve the accuracy of the communication a process of oversampling and filtering of data needs to be implemented.

In conclusion, this project underscores the promising potential of utilizing Wi-Fi-based virtual scopes for diagnostic of measurements in power electronics fields. With further development and refinement, this architecture could pave the way for significant advancements in this field.

Appendix A

Differential Probes

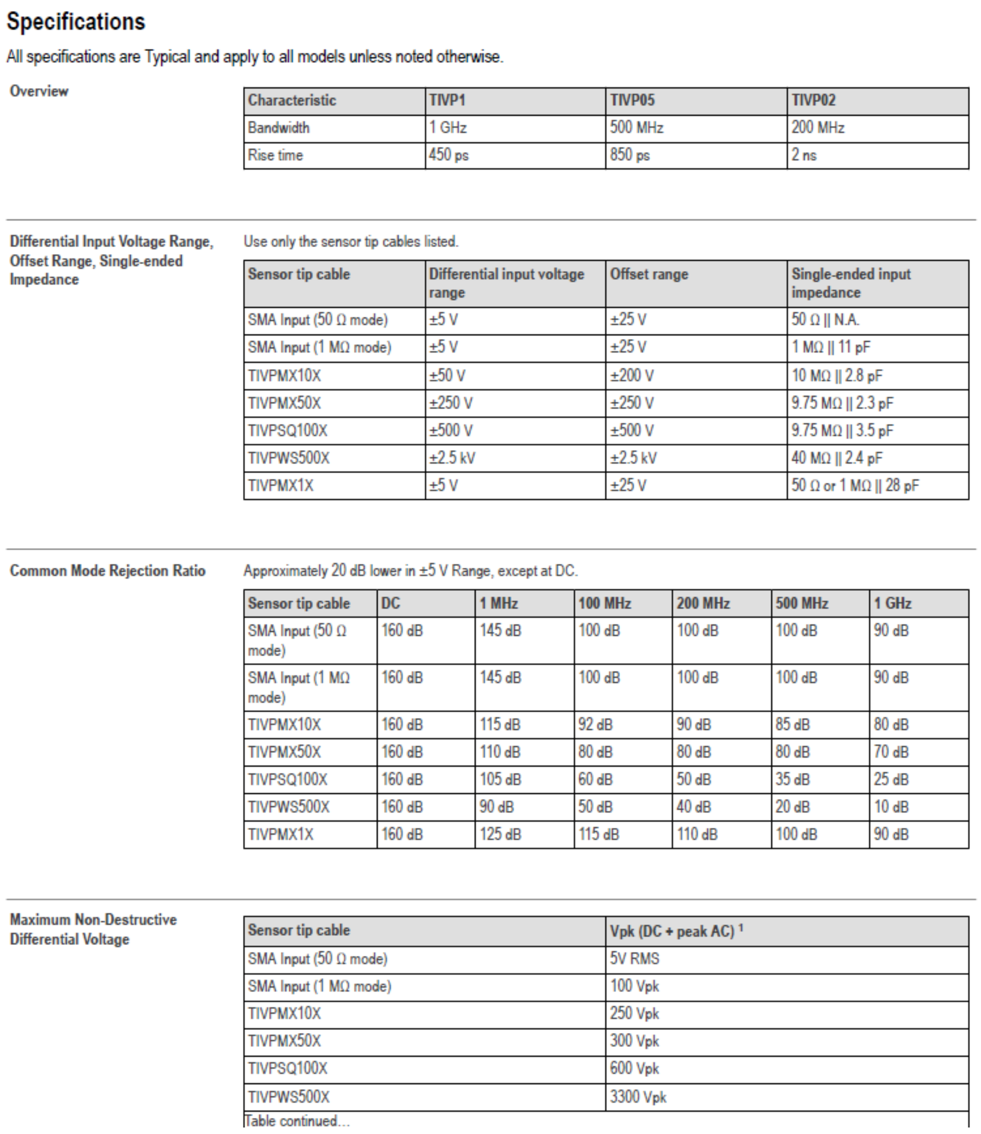
The appendix presents the datasheet of some of the probes analysed in the review of the state-of-the-art technologies for the diagnostic in the field of power electronics. The devices are the PMK Bumblebee non isolated differential probe and the Tektronix family of TIV isolated differential probes. The datasheet shows the characterizing parameters of the probes, it is noticeable their frequency behaviour and nominal ratings. Each datasheet can be used as a reference to decide which probe is more suited to the application under test.



Electrical Specifications [1]

Attenuation Ratio (switchable)	**50:1**	**100:1**	**250:1**	**500:1**
Bandwidth (-3dB)				
Input Voltage 50 V	300 MHz	300 MHz	400 MHz	400 MHz
Input Voltage 500 V	n.a	n.a	300 MHz	300 MHz
Input Voltage 1000 V	n.a	n.a	n.a	300 MHz
Risetime (10 %- 90%)				
Input Voltage 50 V	1.2 ns	1.2 ns	875 ps	875 ps
Input Voltage 500 V	n.a	n.a	1.2 ns	1.2 ns
Input Voltage 1000 V	n.a	n.a	n.a	1.2 ns
Typical Noise (rms) [2] (referred to input)	55 mV	55 mV	75 mV	75 mV
Typical Propagation Delay	12 ns			
Max. Input Voltage				
Measurement Category I	2000 V rms 6000 V transiente Overvoltage			
Measurement Category III	1000 V CAT III			
Pollution Degree	2			
Max. Differential Input Voltage (incl. AC peak)	±200 V DC	± 400 V DC	± 1000 V DC	±2000 V DC
Common Mode Voltage	± 2000 V pk (± 1400 V rms)			
DC Gain Accuracy	± 0.7 %	± 0.7 %	± 0.35 %	± 0.35 %
Offset Range [3]	± 4 V			
Offset Resolution [3]	15 Bit / minimum Step < 125 µV			
Offset Drift [3]	150 µV / °C	150 µV / °C	40 µV / °C	40 µV / °C
Input Impedance				
Each Input to Ground	5 MΩ || 4 pF			
Differential Input Impedance	10 MΩ || 2 pF			
Input Coupling of the Measuring Instrument [4]	50 Ω			
Typical CMRR	DC	> 80 dB		
	100 kHz	> 70 dB		
	1 MHz	> 62 dB		
	3.2 MHz	> 50 dB		

Figure A.1: PMK Bumblebee differential probe datasheet



Specifications

All specifications are Typical and apply to all models unless noted otherwise.

Overview

Characteristic	TIVP1	TIVP05	TIVP02
Bandwidth	1 GHz	500 MHz	200 MHz
Rise time	450 ps	850 ps	2 ns

Differential Input Voltage Range, Offset Range, Single-ended Impedance

Use only the sensor tip cables listed.

Sensor tip cable	Differential input voltage range	Offset range	Single-ended input impedance
SMA Input (50 Ω mode)	±5 V	±25 V	50 Ω || N.A.
SMA Input (1 MΩ mode)	±5 V	±25 V	1 MΩ || 11 pF
TIVPMX10X	±50 V	±200 V	10 MΩ || 2.8 pF
TIVPMX50X	±250 V	±250 V	9.75 MΩ || 2.3 pF
TIVPSQ100X	±500 V	±500 V	9.75 MΩ || 3.5 pF
TIVPWS500X	±2.5 kV	±2.5 kV	40 MΩ || 2.4 pF
TIVPMX1X	±5 V	±25 V	50 Ω or 1 MΩ || 28 pF

Common Mode Rejection Ratio

Approximately 20 dB lower in ±5 V Range, except at DC.

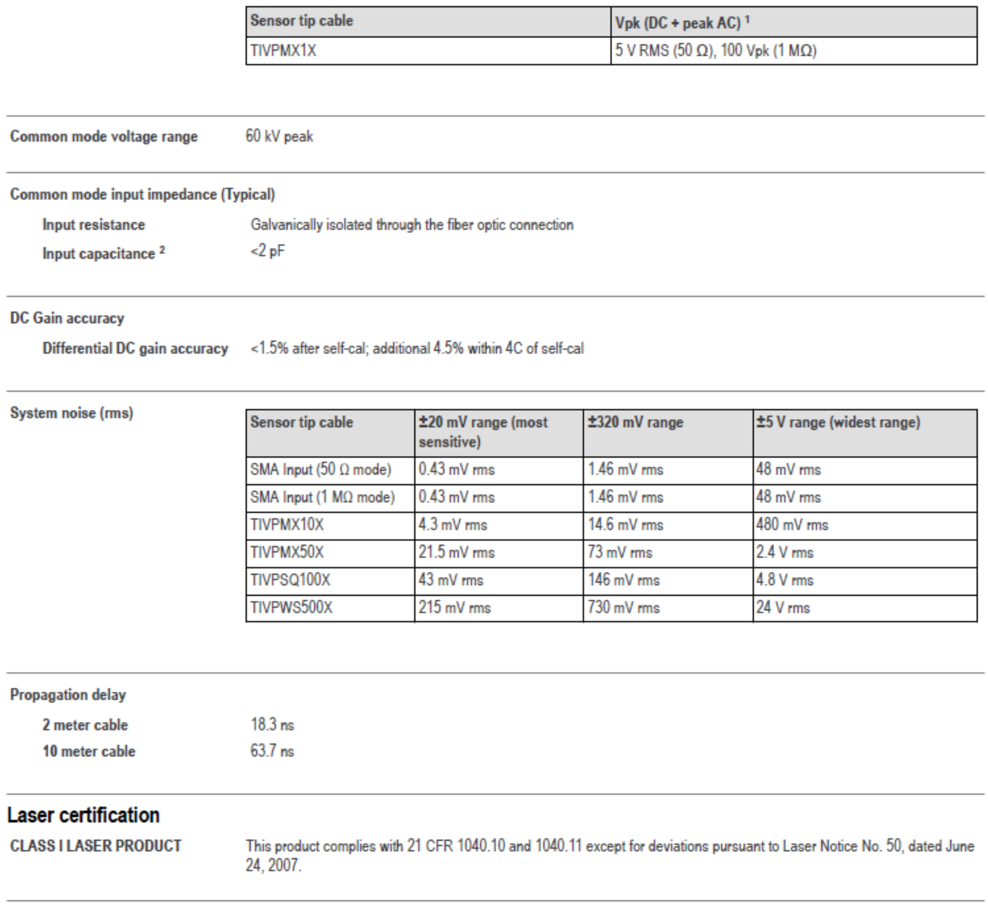
Sensor tip cable	DC	1 MHz	100 MHz	200 MHz	500 MHz	1 GHz
SMA Input (50 Ω mode)	160 dB	145 dB	100 dB	100 dB	100 dB	90 dB
SMA Input (1 MΩ mode)	160 dB	145 dB	100 dB	100 dB	100 dB	90 dB
TIVPMX10X	160 dB	115 dB	92 dB	90 dB	85 dB	80 dB
TIVPMX50X	160 dB	110 dB	80 dB	80 dB	80 dB	70 dB
TIVPSQ100X	160 dB	105 dB	60 dB	50 dB	35 dB	25 dB
TIVPWS500X	160 dB	90 dB	50 dB	40 dB	20 dB	10 dB
TIVPMX1X	160 dB	125 dB	115 dB	110 dB	100 dB	90 dB

Maximum Non-Destructive Differential Voltage

Sensor tip cable	Vpk (DC + peak AC) [1]
SMA Input (50 Ω mode)	5V RMS
SMA Input (1 MΩ mode)	100 Vpk
TIVPMX10X	250 Vpk
TIVPMX50X	300 Vpk
TIVPSQ100X	600 Vpk
TIVPWS500X	3300 Vpk

Table continued…

(a) Page 1



Sensor tip cable	Vpk (DC + peak AC) [1]
TIVPMX1X	5 V RMS (50 Ω), 100 Vpk (1 MΩ)

Common mode voltage range 60 kV peak

Common mode input impedance (Typical)

Input resistance	Galvanically isolated through the fiber optic connection
Input capacitance [2]	<2 pF

DC Gain accuracy

Differential DC gain accuracy	<1.5% after self-cal; additional 4.5% within 4C of self-cal

System noise (rms)

Sensor tip cable	±20 mV range (most sensitive)	±320 mV range	±5 V range (widest range)
SMA Input (50 Ω mode)	0.43 mV rms	1.46 mV rms	48 mV rms
SMA Input (1 MΩ mode)	0.43 mV rms	1.46 mV rms	48 mV rms
TIVPMX10X	4.3 mV rms	14.6 mV rms	480 mV rms
TIVPMX50X	21.5 mV rms	73 mV rms	2.4 V rms
TIVPSQ100X	43 mV rms	146 mV rms	4.8 V rms
TIVPWS500X	215 mV rms	730 mV rms	24 V rms

Propagation delay

2 meter cable	18.3 ns
10 meter cable	63.7 ns

Laser certification

CLASS I LASER PRODUCT This product complies with 21 CFR 1040.10 and 1040.11 except for deviations pursuant to Laser Notice No. 50, dated June 24, 2007.

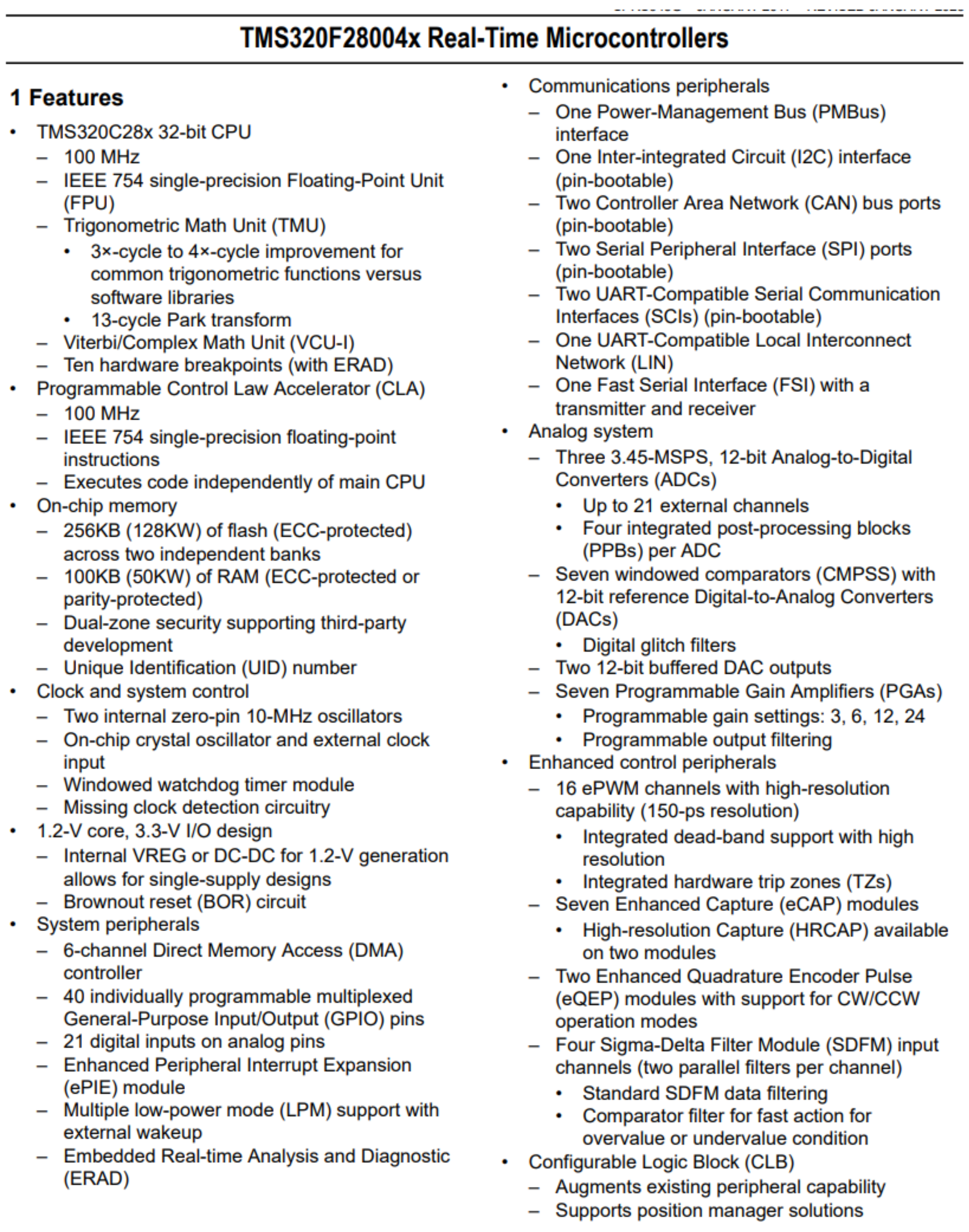
(a) Page 2

Figure A.2: Tektronix TIVP1, TIVP05, TIVP02 Datasheet

Appendix B

Microcontrollers

This appendix is dedicated to the datasheet of the microcontroller used in the development of the project: the TMSF280049C from Texas Instruments and the ESP32-S3 Devkit-1U from Espriff. The datasheet describes all the implemented interfaces and modules, giving a general idea of the capabilities of each device.

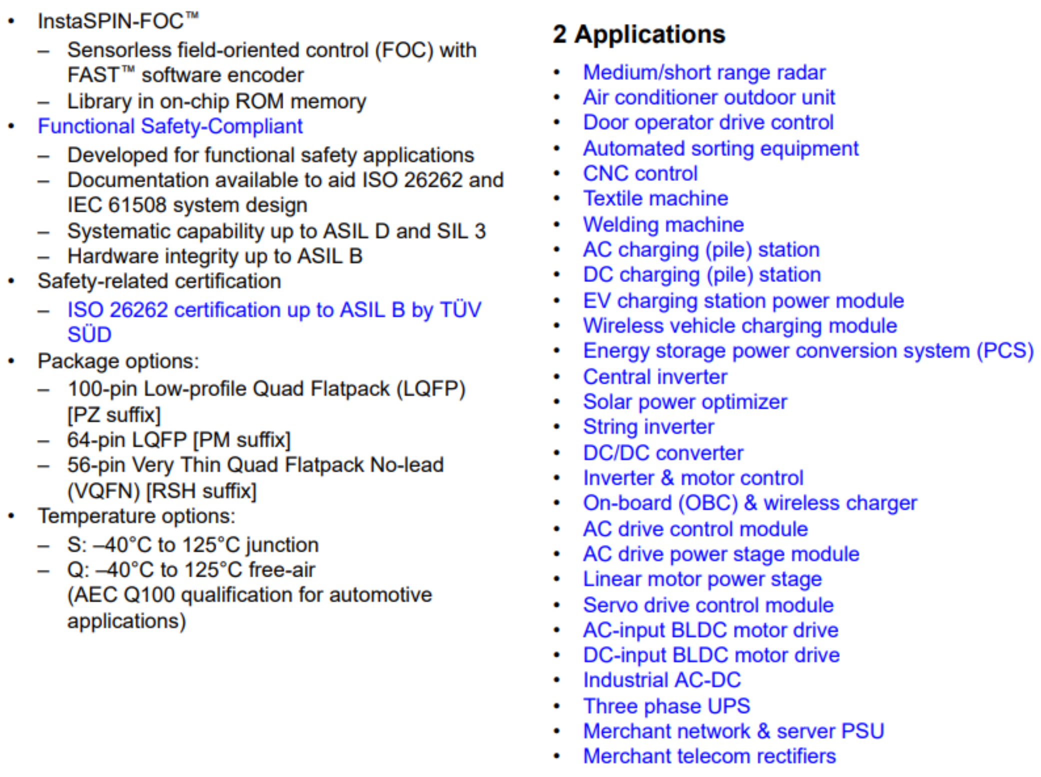


TMS320F28004x Real-Time Microcontrollers

1 Features

- TMS320C28x 32-bit CPU
  - 100 MHz
  - IEEE 754 single-precision Floating-Point Unit (FPU)
  - Trigonometric Math Unit (TMU)
    - 3×-cycle to 4×-cycle improvement for common trigonometric functions versus software libraries
    - 13-cycle Park transform
  - Viterbi/Complex Math Unit (VCU-I)
  - Ten hardware breakpoints (with ERAD)
- Programmable Control Law Accelerator (CLA)
  - 100 MHz
  - IEEE 754 single-precision floating-point instructions
  - Executes code independently of main CPU
- On-chip memory
  - 256KB (128KW) of flash (ECC-protected) across two independent banks
  - 100KB (50KW) of RAM (ECC-protected or parity-protected)
  - Dual-zone security supporting third-party development
  - Unique Identification (UID) number
- Clock and system control
  - Two internal zero-pin 10-MHz oscillators
  - On-chip crystal oscillator and external clock input
  - Windowed watchdog timer module
  - Missing clock detection circuitry
- 1.2-V core, 3.3-V I/O design
  - Internal VREG or DC-DC for 1.2-V generation allows for single-supply designs
  - Brownout reset (BOR) circuit
- System peripherals
  - 6-channel Direct Memory Access (DMA) controller
  - 40 individually programmable multiplexed General-Purpose Input/Output (GPIO) pins
  - 21 digital inputs on analog pins
  - Enhanced Peripheral Interrupt Expansion (ePIE) module
  - Multiple low-power mode (LPM) support with external wakeup
  - Embedded Real-time Analysis and Diagnostic (ERAD)
- Communications peripherals
  - One Power-Management Bus (PMBus) interface
  - One Inter-integrated Circuit (I2C) interface (pin-bootable)
  - Two Controller Area Network (CAN) bus ports (pin-bootable)
  - Two Serial Peripheral Interface (SPI) ports (pin-bootable)
  - Two UART-Compatible Serial Communication Interfaces (SCIs) (pin-bootable)
  - One UART-Compatible Local Interconnect Network (LIN)
  - One Fast Serial Interface (FSI) with a transmitter and receiver
- Analog system
  - Three 3.45-MSPS, 12-bit Analog-to-Digital Converters (ADCs)
    - Up to 21 external channels
    - Four integrated post-processing blocks (PPBs) per ADC
  - Seven windowed comparators (CMPSS) with 12-bit reference Digital-to-Analog Converters (DACs)
    - Digital glitch filters
  - Two 12-bit buffered DAC outputs
  - Seven Programmable Gain Amplifiers (PGAs)
    - Programmable gain settings: 3, 6, 12, 24
    - Programmable output filtering
- Enhanced control peripherals
  - 16 ePWM channels with high-resolution capability (150-ps resolution)
    - Integrated dead-band support with high resolution
    - Integrated hardware trip zones (TZs)
  - Seven Enhanced Capture (eCAP) modules
    - High-resolution Capture (HRCAP) available on two modules
  - Two Enhanced Quadrature Encoder Pulse (eQEP) modules with support for CW/CCW operation modes
  - Four Sigma-Delta Filter Module (SDFM) input channels (two parallel filters per channel)
    - Standard SDFM data filtering
    - Comparator filter for fast action for overvalue or undervalue condition
- Configurable Logic Block (CLB)
  - Augments existing peripheral capability
  - Supports position manager solutions

(a) Page 1



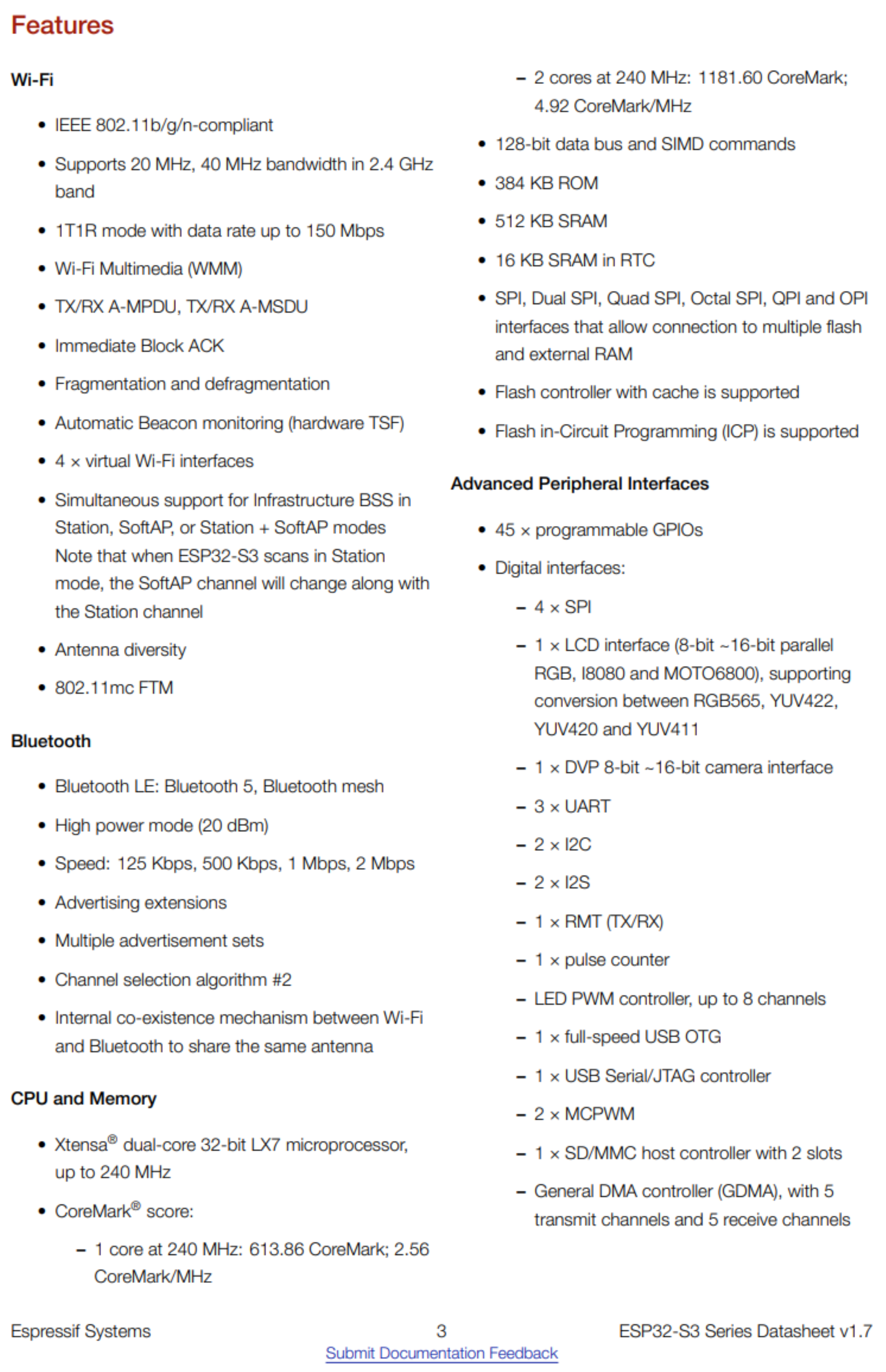
- InstaSPIN-FOC™
  - Sensorless field-oriented control (FOC) with FAST™ software encoder
  - Library in on-chip ROM memory
- Functional Safety-Compliant
  - Developed for functional safety applications
  - Documentation available to aid ISO 26262 and IEC 61508 system design
  - Systematic capability up to ASIL D and SIL 3
  - Hardware integrity up to ASIL B
- Safety-related certification
  - ISO 26262 certification up to ASIL B by TÜV SÜD
- Package options:
  - 100-pin Low-profile Quad Flatpack (LQFP) [PZ suffix]
  - 64-pin LQFP [PM suffix]
  - 56-pin Very Thin Quad Flatpack No-lead (VQFN) [RSH suffix]
- Temperature options:
  - S: –40°C to 125°C junction
  - Q: –40°C to 125°C free-air (AEC Q100 qualification for automotive applications)

2 Applications

- Medium/short range radar
- Air conditioner outdoor unit
- Door operator drive control
- Automated sorting equipment
- CNC control
- Textile machine
- Welding machine
- AC charging (pile) station
- DC charging (pile) station
- EV charging station power module
- Wireless vehicle charging module
- Energy storage power conversion system (PCS)
- Central inverter
- Solar power optimizer
- String inverter
- DC/DC converter
- Inverter & motor control
- On-board (OBC) & wireless charger
- AC drive control module
- AC drive power stage module
- Linear motor power stage
- Servo drive control module
- AC-input BLDC motor drive
- DC-input BLDC motor drive
- Industrial AC-DC
- Three phase UPS
- Merchant network & server PSU
- Merchant telecom rectifiers

(a) Page 2

Figure B.1: Texas Instruments TMS320F280049x Microcontrollers Datasheet



Features

Wi-Fi

- IEEE 802.11b/g/n-compliant
- Supports 20 MHz, 40 MHz bandwidth in 2.4 GHz band
- 1T1R mode with data rate up to 150 Mbps
- Wi-Fi Multimedia (WMM)
- TX/RX A-MPDU, TX/RX A-MSDU
- Immediate Block ACK
- Fragmentation and defragmentation
- Automatic Beacon monitoring (hardware TSF)
- 4 × virtual Wi-Fi interfaces
- Simultaneous support for Infrastructure BSS in Station, SoftAP, or Station + SoftAP modes Note that when ESP32-S3 scans in Station mode, the SoftAP channel will change along with the Station channel
- Antenna diversity
- 802.11mc FTM

Bluetooth

- Bluetooth LE: Bluetooth 5, Bluetooth mesh
- High power mode (20 dBm)
- Speed: 125 Kbps, 500 Kbps, 1 Mbps, 2 Mbps
- Advertising extensions
- Multiple advertisement sets
- Channel selection algorithm #2
- Internal co-existence mechanism between Wi-Fi and Bluetooth to share the same antenna

CPU and Memory

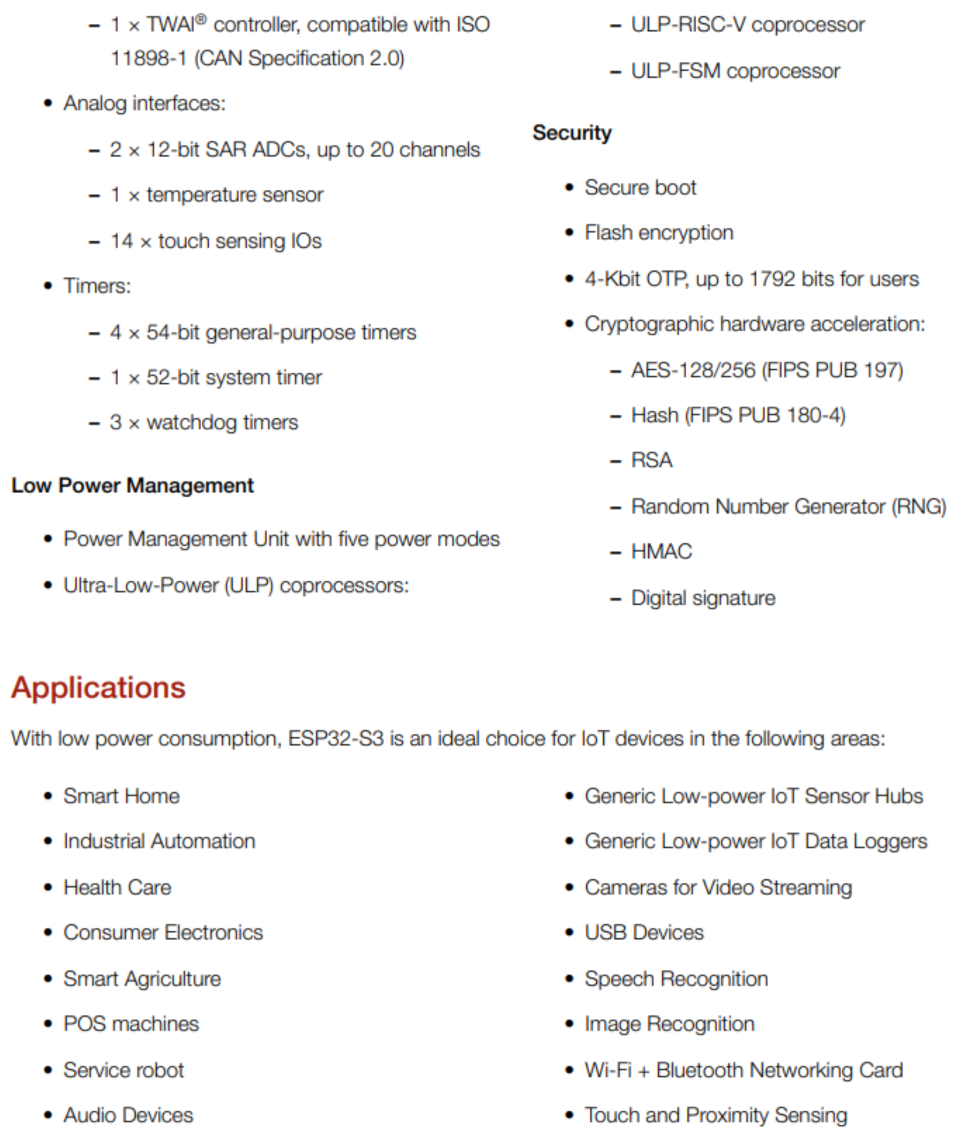
- Xtensa® dual-core 32-bit LX7 microprocessor, up to 240 MHz
- CoreMark® score:
  - 1 core at 240 MHz: 613.86 CoreMark; 2.56 CoreMark/MHz
  - 2 cores at 240 MHz: 1181.60 CoreMark; 4.92 CoreMark/MHz
- 128-bit data bus and SIMD commands
- 384 KB ROM
- 512 KB SRAM
- 16 KB SRAM in RTC
- SPI, Dual SPI, Quad SPI, Octal SPI, QPI and OPI interfaces that allow connection to multiple flash and external RAM
- Flash controller with cache is supported
- Flash in-Circuit Programming (ICP) is supported

Advanced Peripheral Interfaces

- 45 × programmable GPIOs
- Digital interfaces:
  - 4 × SPI
  - 1 × LCD interface (8-bit ~16-bit parallel RGB, I8080 and MOTO6800), supporting conversion between RGB565, YUV422, YUV420 and YUV411
  - 1 × DVP 8-bit ~16-bit camera interface
  - 3 × UART
  - 2 × I2C
  - 2 × I2S
  - 1 × RMT (TX/RX)
  - 1 × pulse counter
  - LED PWM controller, up to 8 channels
  - 1 × full-speed USB OTG
  - 1 × USB Serial/JTAG controller
  - 2 × MCPWM
  - 1 × SD/MMC host controller with 2 slots
  - General DMA controller (GDMA), with 5 transmit channels and 5 receive channels

Espressif Systems 3 ESP32-S3 Series Datasheet v1.7
Submit Documentation Feedback

(a) Page 1



- 1 × TWAI® controller, compatible with ISO 11898-1 (CAN Specification 2.0)

- Analog interfaces:
  - 2 × 12-bit SAR ADCs, up to 20 channels
  - 1 × temperature sensor
  - 14 × touch sensing IOs
- Timers:
  - 4 × 54-bit general-purpose timers
  - 1 × 52-bit system timer
  - 3 × watchdog timers

Low Power Management

- Power Management Unit with five power modes
- Ultra-Low-Power (ULP) coprocessors:
  - ULP-RISC-V coprocessor
  - ULP-FSM coprocessor

Security

- Secure boot
- Flash encryption
- 4-Kbit OTP, up to 1792 bits for users
- Cryptographic hardware acceleration:
  - AES-128/256 (FIPS PUB 197)
  - Hash (FIPS PUB 180-4)
  - RSA
  - Random Number Generator (RNG)
  - HMAC
  - Digital signature

Applications

With low power consumption, ESP32-S3 is an ideal choice for IoT devices in the following areas:

- Smart Home
- Industrial Automation
- Health Care
- Consumer Electronics
- Smart Agriculture
- POS machines
- Service robot
- Audio Devices
- Generic Low-power IoT Sensor Hubs
- Generic Low-power IoT Data Loggers
- Cameras for Video Streaming
- USB Devices
- Speech Recognition
- Image Recognition
- Wi-Fi + Bluetooth Networking Card
- Touch and Proximity Sensing

(a) Page 2

Figure B.2: Espriff ESP32-S3 Datasheet

Appendix C

On-Board Charger Schematics

In the following it is presented a piece of the hardware schematics of the power board of the on-board charger of BURSA HyPower. The only shown piece regards the DSP integration over the device, it depicts the allocation of the GPIO pins of the microcontroller.

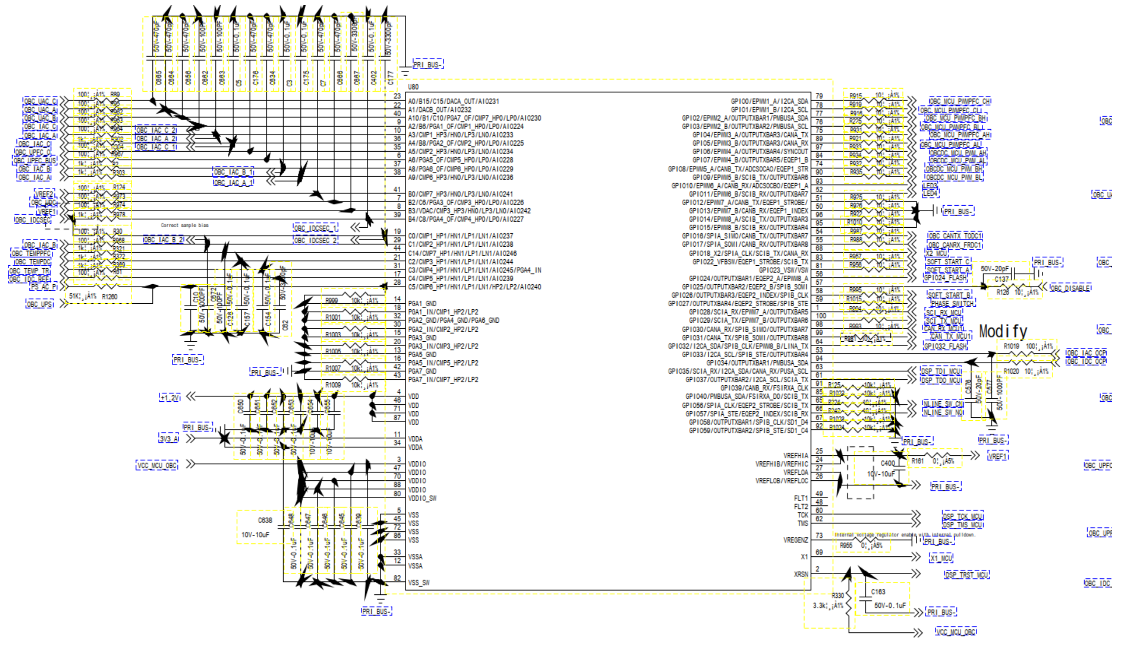


Figure C.1: BRUSA HyPower OBC Power Board schematics: DSP

Bibliography

[1] "World Energy Usage", Accessed Spet. 2023, [ONLINE]: https://www.texasgateway.org/resource/79-world-energy-use (September 2023) (cit. on p. 1).

[2] "Galvanic isolation for electric vehicle systems", "Power Electronic Tips, an EE world online resource", Accessed Sept. 2023, [ONLINE]: https://www.powerelectronictips.com/galvanic-isolation-for-electric-vehicle-systems/ (cit. on p. 3).

[3] "PCAN Explorer 6", Accessed Sept. 2023, [ONLINE]: https://www.peak-system.com/PCAN-Explorer-6.415.0.html?&L=1 (cit. on pp. 5, 23).

[4] "CANoe software", Accessed Spet. 2023, [ONLINE]:: https://www.vector.com/int/en/products/products-a-z/software/canoe/ (cit. on p. 5).

[5] Pascal S. Niklaus, Reto Bonetti, Christof Stäger, Johann W. Kolar, and Domink Bortis. «High-Bandwidth Isolated Voltage Measurements With Very High Common Mode Rejection Ratio for WBG Power Converters». In: *IEEE Open Journal of Power Electronics* (2022) (cit. on p. 11).

[6] Andrew Levido. «High-Voltage Differential Probe». In: *Circuit Cellar Magazine* (2019). Accessed Sept. 2023, [ONLINE]: https://circuitcellar.com/research-design-hub/high-voltage-differential-probe/ (cit. on p. 12).

[7] "Bumblebee High Voltage Differential Probe", PMK, Accessed Sept. 2023, [ONLINE]: https://www.batronix.com/files/PMK/Datenblaetter/880-102-501_Datasheet-BumbleBee_en.pdf (cit. on p. 13).

[8] "N2792a Low Voltage Differential Probe", Keysight, Accessed Sept. 2023, [ONLINE]: https://www.keysight.com/us/en/assets/7018-02322/data-sheets/5990-4753.pdf (cit. on p. 13).

[9] "AP033 High Voltage Differential Probe", Teledyne LeCroy, Accessed Sept. 2023, [ONLINE]: https://cdn.teledynelecroy.com/files/manuals/ap033-active-differential-probe-operators-manual.pdf (cit. on p. 13).

[10] "TA044 High Voltage Differential Probe", Pico Technology, Accessed Sept. 2023, [ONLINE]: `https://my.element14.com/pico-technology/ta044/probe-active-differential-hi-volt/dp/1667349` (cit. on p. 13).

[11] "DP Series Voltage Differential Probe", Micsig, Accessed Sept. 2023, [ONLINE]: `https://www.micsig.com/uploads/HighVoltageDifferentialProbeUserManual_1656560834.pdf` (cit. on p. 13).

[12] "What is the difference between an isolated differential probe and a non isolated differential probe?", Accessed Sept. 2023, [ONLINE]: `https://electronics.stackexchange.com/questions/657084/what-is-the-difference-between-a-differential-probe-isolated-and-a-differential` (cit. on p. 14).

[13] "TIV Series Isolated Voltage Differential Probe", Tektronix, Accessed Sept. 2023, [ONLINE]: `https://www.tek.com/en/datasheet/isolated-measurement-systems-tivp1-tivp05-tivp02-datasheet` (cit. on p. 14).

[14] "HVF108 Isolated Voltage Differential Probe", Teledyne LeCroy, Accessed Sept. 2023, [ONLINE]: `https://cdn.teledynelecroy.com/files/pdf/hvfo-probes-datasheet.pdf` (cit. on p. 15).

[15] "MOIP Series Isolated Voltage Differential Probe", Micsig, Accessed Sept. 2023, [ONLINE]: `https://www.micsig.com/uploads/SigOFITProbeDatasheetV1.5_1687829714.pdf` (cit. on p. 15).

[16] Michael Grubmüller, Bernhard Schweighofer, and Hannes Wegleiter. «A Digital Isolated High Voltage Probe for Measurements in Power Electronics». In: *IEEE International Symposium on Industrial Electronics (ISIE)* (2018) (cit. on pp. 15, 16).

[17] "Cleverscope C448 datasheet", Accessed Sept. 2023, [ONLINE]: `https://www.meilhaus.de/cosmoshop/default/articleMedia/cs448/en/Datasheet_Cleverscope-CS448_en.pdf` (cit. on p. 15).

[18] Bart Schroder, "Cleverscope Development Report", Accessed Sept. 2023, [ONLINE]: `https://hackaday.com/2021/11/18/isolated-oscilloscope-design-process-shows-how-its-done/` (cit. on p. 15).

[19] "What is OSI Model? – Layers of OSI Model", Accessed Sept. 2023, [ONLINE]: `https://www.geeksforgeeks.org/open-systems-interconnection-model-osi/` (cit. on p. 17).

[20] Wikipedia, "Transmission Control Protocol - TCP", Accessed Sept. 2023, [ONLINE]: `https://en.wikipedia.org/wiki/Transmission_Control_Protocol` (cit. on p. 22).

[21] "Arduino IDE", Accessed Sept. 2023, [ONLINE]: `https://www.arduino.cc/en/software` (cit. on p. 22).

[22] "Code Composer Studio - CCS", Accessed Sept. 2023, [ONLINE]: `https://www.ti.com/tool/CCSTUDIO` (cit. on p. 22).

[23] "Visual Studio Code", Accessed Sept. 2023, [ONLINE]: `https://code.visualstudio.com/` (cit. on p. 22).

[24] "Wireshark", Accessed Sept. 2023, [ONLINE]: `https://www.wireshark.org/` (cit. on p. 22).

[25] "PicoScope 5000 series Datasheet", Accessed Sept. 2023, [ONLINE]: `https://www.picotech.com/download/manuals/picoscope-5000d-series-users-guide.pdf` (cit. on p. 23).

[26] "ESP32", Espriff, Accessed Sept. 2023, [ONLINE]: `https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf` (cit. on p. 28).

[27] "LAUNCHXL-CC3235SF", Texas Instruments, Accessed Sept. 2023, [ONLINE]: `https://www.digikey.it/it/products/detail/texas-instruments/LAUNCHXL-CC3235SF/10434597` (cit. on p. 28).

[28] "AMB01", Realtek, Accessed Sept. 2023, [ONLINE]: `https://www.amebaiot.com/en/ameba1/` (cit. on p. 29).

[29] "Zero 2 W", Raspberry Pi , Accessed Sept. 2023, [ONLINE]: `https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/` (cit. on p. 29).

[30] "LAUNCHXL-F280049C", Texas Instruments, Accessed Sept. 2023, [ONLINE]: `https://www.digikey.com/en/products/detail/texas-instruments/LAUNCHXL-F280049C/9860033` (cit. on p. 29).

[31] "TMS320F28004x Datasheet", Texas Instruments, Accessed Sept. 2023, [ONLINE]: `https://www.ti.com/product/TMS320F280049C?keyMatch=&tisearch=search-everything&usecase=partmatches` (cit. on p. 30).

[32] "socket library documentation", Python, Accessed Sept. 2023, [ONLINE]: `https://docs.python.org/3/library/socket.html` (cit. on p. 31).

[33] "pandas library documentation", Python, Accessed Sept. 2023, [ONLINE]: `https://pandas.pydata.org/` (cit. on p. 31).

[34] "Matplotlib library documentation", Python, Accessed Sept. 2023, [ONLINE]: `https://matplotlib.org/` (cit. on p. 31).

[35] "NumPy library documentation", Python, Accessed Sept. 2023, [ONLINE]: `https://numpy.org/` (cit. on p. 31).

[36] Thomas Kugelstadt. «Extending the SPI bus for long-distance communication». In: (2011). `https://www.ti.com/lit/an/slyt441/slyt441.pdf` (cit. on p. 73).